Automatic Data Augmentation by Learning the Deterministic Policy

Yinghuan Shi\textsuperscript{1}, Tiexin Qin\textsuperscript{1}, Yong Liu\textsuperscript{1}, Jiwen Lu\textsuperscript{2}, Yang Gao\textsuperscript{1}, Dinggang Shen\textsuperscript{3}

\textsuperscript{1}Nanjing University, China \textsuperscript{2}Tsinghua University, China \textsuperscript{3}UNC Chapel Hill, USA

Abstract

Aiming to produce sufficient and diverse training samples, data augmentation has been demonstrated for its effectiveness in training deep models. Regarding that the criterion of the best augmentation is challenging to define, we in this paper present a novel learning-based augmentation method termed as DeepAugNet, which formulates the final augmented data as a collection of several sequentially augmented subsets. Specifically, the current augmented subset is required to maximize the performance improvement compared with the last augmented subset by learning the deterministic augmentation policy using deep reinforcement learning. By introducing an unified optimization goal, DeepAugNet intends to combine the data augmentation and the deep model training in an end-to-end training manner which is realized by simultaneously training a hybrid architecture of dueling deep Q-learning algorithm and a surrogate deep model. We extensively evaluated our proposed DeepAugNet on various benchmark datasets including Fashion MNIST, CUB, CIFAR-100 and WebCaricature. Compared with the current state-of-the-arts, our method can achieve a significant improvement in small-scale datasets, and a comparable performance in large-scale datasets. Code will be available soon.

1. Introduction

Recent progress in deep learning has substantially improved the results of image classification or recognition tasks. With the goal of increasing the amount and diversity of the available training samples, data augmentation has shown its advantage for boosting the classification accuracy \cite{16, 23}. By training the deep learning models on the original training images along with the obtained augmented images, a good generalization performance in the testing stage can be expected, especially for the small-scale datasets.

The most simple yet popular way of data augmentation is to perform the basic image preprocessing operators, e.g., flipping, translation, rotation and color jittering, on the original images to generate the augmented images. This traditional setting could be regarded as an independent image preprocessing step before training the deep model, since the subsequent classification does not influence these pre-generated augmented images (i.e., data augmentation does not get any feedback from the subsequent model training). Despite its success in some tasks, this kind of traditional augmentation still suffer from the major limitation of: 1) separation – the data augmentation and model training are separated without a consistent objective, 2) redundancy – the augmentation is usually random which might produce a large number of redundant samples, and 3) harmfulness – some augmented images might be harmful for the model training.

Being aware of these limitations, the researchers are recently paying attention to the learning-based augmentation \cite{1, 10, 7, 32, 3, 37}. According to our analysis, we notice that the major bottleneck of the previous learning-based augmentation methods is that the criterion of the best augmentation is challenging to define. Thus, these methods usually require the pre-defined assumption of data distribution before augmentation. For examples, the generative adversarial network (GAN) based augmentation methods assume the augmented and original images should belong to
a same data distribution [39, 14, 5]. The Mixup method [37] mainly focuses on the marginal region between different classes during augmentation. However, the augmentation results are largely influenced by these assumptions.

To relax these assumptions, we innovatively formulates the final augmented data as a collection of several sequentially augmented subsets, where the consecutive augmented two subsets should maximize the classification performance improvement. Thus, we present a novel method DeepAugNet, which directly optimizes the classification performance improvement in deep model training by automatically learning the best data augmentation policy in a deep reinforcement learning environment. Specifically, regarding that the original objective of augmentation is difficult to define and hard to optimize, we formulate it as a truncated objective where a trial-and-error strategy can be employed to combine the data augmentation and model training into a consistent goal. More precisely, the final augmented set in our method can be considered as a generalized union on several sequentially augmented subsets where each subset is required to maximize the performance improvement compared with the latest augmented subset on an additional surrogate model. To achieve this goal, we model our problem using a hybrid architecture of the dueling architectures-based deep Q-learning algorithm (Dueling DQN) which could be trained with the later surrogate model. To achieve this goal, we developed a surrogate model for automatic data augmentation using DRL. Our method is the first attempt to model automatic data augmentation as a deep reinforcement learning problem by learning a deterministic policy.

Our method is the first attempt to model automatic data augmentation as a deep reinforcement learning problem by learning a deterministic policy.

The solution of final augmented set is formalized as to solve a generalized union on several sequentially augmented subsets that satisfy the maximum performance improvement.

A joint scheme by integrating a hybrid architecture of Dueling DQN and a surrogate model is developed for effective policy learning.

Extensive evaluation on various benchmark datasets including Fashion MNIST, CUB, CIFAR-100 and WebCaricature validates the advantage of our method.

2. Related Work

Data Augmentation. Basically, the common goal of different data augmentation methods is to generate the sufficient and various samples to augment the original training samples. According to previous studies, an appropriate data augmentation benefits the recognition performance compared with that of only using the original training samples. An initiatory yet frequently-used data augmentation method is to directly perform some ordinary operators (e.g., flipping, rotation, translation) on the original images to produce the augmented images [16, 23].

Recently, there have been a few attempts of learning-based augmentation and most of them use the GAN-based architecture which requires the augmented and original samples belong to a same distribution. Wang et al. [32] presented a meta learning-based GAN to produce additional training samples. Gurumurthy et al. [7] developed a model to adopt a mixture model to parameterize the latent generative space and meanwhile jointly learn the parameters of this mixture model and GANs. Gatys et al. [4] modeled the content and style loss separately which was helpful to generate new training images. Huang et al. [10] proposed a structure-aware image-to-image translation network and Antreas et al. [1] addressed the problem of predicting the label for the novel unseen classes according to the generated samples. Volpi et al. [30] also introduced a worst-case formulation over data distribution to tackle the augmentation on unseen domains. Recently, there are a very few attempts of nonGAN-based augmentation: Mixup [37] and AutoAugment [3]. In Mixup [37], a simple yet effective linear combination of in-between training samples was learned for augmentation. In AutoAugment [3], the random augmentation policy on dataset-level was exploited for effective augmentation.

Deep Reinforcement Learning. Deep reinforcement learning (DRL) wishes to learn a policy for an agent by a deep model in order to make a sequential decision for maximizing an accumulative reward [19, 20]. DRL has received considerable attention recently for its effectiveness of dealing with the high dimensional data in computer vision tasks. For example, Zhong et al. [38] presented a block-wise network generation pipeline to automatically establish the optimal network structure by sequentially choosing the component layers. Tang et al. [28] introduced a method of selecting the most beneficial frames from video to boost the performance of action recognition. Song et al. [26] employed DRL method to generate a sequence of artificial user input for interactive image segmentation. Also, Han et al. [8] adjusted the location of context box and object box to maximize the segmentation performance. Park et al. [21] modeled the optimal global enhancement in a DRL manner. For object localization and tracking, Caicedo et al. [2] trained an agent learning algorithm to deform a bounding box with some simple transformation actions in order to correctly localize a target object. Ren et al. [22] and Guo et al. [6] introduced the DRL-based methods for object tracking. Kong et al. [15] proposed a collaborative algorithm with multi-agents to localize different objects. Although DRL methods are playing a more important role in various vision tasks, to exploit the automatic data augmentation using DRL is still in its early stage.

Remark. Despite the success of previous learning-based augmentation methods in various tasks, most of them
require a predefined assumption for the data distribution before augmentation (e.g., training and testing samples belonging to a same distribution, the augmented samples distributed in marginal region between different classes, etc.). Compared with them, our method directly optimizes the performance improvement without any assumptions on the original distribution.

In addition, although AutoAugment [3] also employs DRL to learn the policies, our method is largely different from AutoAugment: 1) our method is sample-wise (i.e., learning the best policy for the samples in a batch) while AutoAugment is dataset-wise (i.e., learning the uniform action for the entire dataset), 2) our method learns the deterministic policy while AutoAugment learns the random policy, where the deterministic policy could guide more robust results, and 3) our method introduces the terminal action while AutoAugment only learns the combination of two actions.

3. Our Method: DeepAugNet

3.1. Problem Formulation

Analysis. Our DeepAugNet intends to learn a best way of data augmentation to benefit the subsequent deep learning models. Basically, jointly training the data augmentation and classification models in an end-to-end manner is extremely difficult to realize due to the following key challenges: 1) the searching space of the data augmentation on the available training images is very huge which might bring into a severe overfitting problem. 2) The best augmentation is hard to quantize as a traditional supervised learning problem. In particular, given the training set $X$, we wish to jointly learn an augmented set $X'$ and a prediction function (i.e., deep model) $f$ parameterized by $\theta$ to minimize the following objective as follows:

$$
\min_{\theta, X'} \sum_{x_i \in X \cup X'} L\left(f(x_i; \theta_{X \cup X'}), y_i\right),
$$

where $x_i$ and $y_i$ ($1 \leq i \leq |X \cup X'|$) are the $i$-th training sample and its ground truth label, respectively. $L(\cdot)$ is a loss function (i.e., we adopt the popularly-used cross entropy function) to measure the difference between the prediction $f(x_i; \theta_{X \cup X'})$ and ground truth $y_i$. $\theta_B$ indicates that the model parameter $\theta$ is trained on the set $B$. However, we notice that Eqn. (1) is difficult to solve since 1) the augmentation set $X'$ is hard to represent, 2) $X'$ and $\theta$ are interacted which is difficult to optimize, and 3) the overfitting issue might happen in training because $X'$ is actually generated from $X$ without introducing any additional data.

Therefore, to overcome these limitation, we present a truncated two-layer objective of Eqn. (1) as below:

$$
\min_{\theta} \sum_{x_i \in X \cup g^*(X)} L\left(f(x_i; \theta_{X \cup g^*(X)}), y_i\right)
$$

s.t. $g^* = \arg \min_\gamma \sum_{x_j \in X^*} L\left(f(x_j; \theta_{X \cup g(X)}), y_j\right),$

where $X^*$ is an additional validation set (i.e., $X^* \cap X = \emptyset$) playing a supervision role which does not involve into the training process to avoid overfitting. $X^*$ can be either sampled from the training set $X$ or an existing validation set\(^1\). $g(\cdot)$ indicates that we wish to learn a transformation to generate the augmented data from the observed training data. The way of introducing $g(\cdot)$ is more feasible than that of directly learning the augmentation data from a random initialization or a Gaussian noise. In this meaning, the input of $g(X)$ is the original training set and the output is the augmentation set.

Considering the fact that $g(X)$ is non-trivial to define, we innovatively model the estimation of $g(X)$ as a generalized union of different subsets as follows:

$$
g(X) = \bigcup_{i=1}^N g^T(x_i),
$$

\(^1\)In practice, we split the dataset into training, validation and testing set. The validation set is firstly used to adjust the hyper-parameters, and then acts as the environment in reinforcement learning to return the reward.
where $T$ is the maximum number of steps (i.e., terminal step) and $N$ is the cardinality of original training set (i.e., $N = |X|$). $g^t(x)$ is the generated augmentation subset at the terminal step. To establish the connection between the different subsets, we approximate this estimation of $g(X)$ as an MDP (Markov decision process) task [27] in a deep reinforcement learning framework. Specifically, we first define several basic actions (e.g., rotation, flip and crop) and then obtain the sequential combination of these defined actions for t-th step subset as follows:

$$g^t(X) = \begin{cases} g^{t-1}(X) \otimes a^t, & t \geq 1 \\ X, & t = 0, \end{cases} \quad (4)$$

where $g^{t-1}(X) \otimes a^t$ indicates that the images in $g^{t-1}(X)$ perform the t-th step action $a^t$ to form the generated images. Finally, a series of sets $g^0(X), g^1(X), \ldots, g^T(X)$ are obtained with their generalized union as the final data augmentation set.

As analyzed, our DeepAugNet actually contains two major stages: the action learning stage and the surrogate training stage. We visually illustrate the pipeline of our proposed DeepAugNet in Fig. 2.

**Action Learning.** By receiving the delayed reward and current input set, the action learning stage tries to learn a best action by maximizing the curriculum reward. Typically, we train a state extraction network to learn the current state according to the inputs. To avoid the possible overfitting caused by repeating performing a same action during the training procedure, we punish this action-repeating strategy by a typical penalty for better generalization. To achieve this, we proposed a hybrid architecture for Dueling DQN [33] by introducing the one-hot code of latest two actions as an additional parameter. The output of this one-hot code is combined with the output of the state extraction network by three layer fully connected network to output the learned action. Finally, this action is used to generate the new images as a guidance.

**Surrogate Training.** In the surrogate training stage, we introduce the surrogate model trained on the separate validation set which does not involve in the training to reduce the computational burden and prevent the overfitting. Typically, the input of the surrogate training stage is the current input set, the action learning stage tries to learn a strategy by a typical penalty for better generalization. To prevent the overfitting caused by repeating performing a same action during the training stage, we train a state extraction network to learn the current input set. We visually illustrate the pipeline of our proposed DeepAugNet in Fig. 2.

**3.2. Network Components**

**State.** Given a current image $x_i$ as input, the state $s^t$ can be defined as the output of the state extraction network $\phi$ as follows:

$$s^t = \phi(x_i). \quad (5)$$

Note that, we adopt different state extraction networks $\phi$ according to the property of different datasets. For example, for small-scale images with size smaller than $50 \times 50$ (e.g., Fashion MNIST), we directly use the raw images as the state since its corresponding state space is relatively not too huge but an agent with only a few layers is able to learn. While for large-size images, aiming to prevent from the huge state space, we employ the pre-trained convnet (e.g., VGG) as the state extraction network with the corresponding output (i.e., feature vector) as the state.

**Action.** During the training procedure, each agent outputs an action $a^t$ according to the current state $s^t$. We formally introduce ten types of basic actions, which contains: FP (flip the image), RT (rotate the image), AN (add noise), WP (warp the image), CL (crop from left side), CR (crop from right side), CT (crop from top side), CB (crop from bottom side), ZM (zoom in the image into $1.1x$) and TM (terminate the processing). Specifically, for RT, we perform the rotation on the current image by $30^\circ$ clockwise. For AN, the Gaussian noise is added to the normalized image to form a new image. For WP, the current image is distorted by the PiecewiseAffine operator in imgaug library.

For four different crop action (i.e., CL, CR, CT and CB), the respective 10-20% boundary regions will be cropped from the current image according to different datasets.

**Reward.** The reward directly influences the final classification performance. In each step, the agent is required to choose an action $a^t$ by receiving a reward $r^t$. With the goal of learning the best data augmentation policy, we maximize the improvement on the validation set $X^*$ by measuring the difference between the prediction $f(x; \theta)$ and the ground truth $y_j$ using the loss function $L(\cdot)$. Specifically, the t-th step reward $r^t$ is formally written as follows:

$$r^t = \begin{cases} J^t - J^t-1, & a^t \neq \text{TM} \\ 10(J^t - J^t), & a^t = \text{TM}, \end{cases} \quad (6)$$

where $J^t$ indicates the t-th step loss in the surrogate model:

$$J^t = \sum_{x_j \in X^*} L(f(x_j; \theta_{X^* \cup \{x_i\}}), y_j), \quad (7)$$

where $x_i$ denotes the result of the t-th step on the i-th training sample as $x_i = g^t(x_i^{t-1})$. In Eqn. (7), $f(x_j; \theta_{X^* \cup \{x_i\}})$ indicates the network trained by fine-tuning $f(x; \theta_X)$ with $x_i$.

---

2https://imgaug.readthedocs.io/en/latest/
3.3. Training

To efficiently train our DeepAugNet, we propose to extend the Dueling DQN [33] with a hybrid loss to prevent the overfitting for training our augmentation policy. The Dueling DQN was proven to be powerful in estimating the Q-function by integrating two functions: the state value function (parameterized by $\beta$) and the state-dependent action advantage function (parameterized by $\sigma$). Specifically, during the training procedure, we wish to learn the current estimation of Q-function $Q(s^t, a^t; \gamma, \omega, \beta, \sigma)$ to guide the next action selection for an agent, where $\gamma$ and $\omega$ are the parameters of the state extraction network and one-hot code, respectively. Formally, Q-learning iteratively adopts the Bellman equation to update the selection policy in a recursive manner as follow:

$$Q(s^t, a^t; \gamma, \omega, \beta, \sigma) = r^t + \rho \max_{a_{t+1}} Q(s^{t+1}, a^{t+1}; \gamma, \omega, \beta, \sigma),$$  

(8)

where $\rho$ is the discount factor. Also, inspired by Dueling DQN, we defined the hybrid Q-function as follows:

$$Q(s^t, a^t; \gamma, \omega, \beta, \sigma) = V(s^t; \gamma, \omega, \sigma) + A(s^t, a^t; \gamma, \omega, \beta) - \frac{1}{|A|} \sum_{a_{t+1}} A(s^{t+1}, a_{t+1}; \gamma, \omega, \beta),$$

where $V(s^t; \gamma, \omega, \sigma)$ and $A(s^t, a^t; \gamma, \omega, \beta)$ are the state-value and advantage function [33], respectively. More details could be referred to [33]. We summarize the training procedure in Algorithm 1.

**Algorithm 1:** DeepAugNet

| Data: Training set $X$, validation set $X^*$ | Result: Augmentation set $X'$ |
|---------------------------------------------|--------------------------------|
| for epoch $= 1, \ldots, n$ do | |
| Initialize $X' = \emptyset$; | |
| for $i = 1, \ldots, |X|$ do | |
| $t \leftarrow 1$; | |
| repeat | |
| $s^t \leftarrow$ Compute the state by Eqn. (5) ; | |
| $a^t \leftarrow$ Compute the action by Eqn. (8) ; | |
| $x_i' \leftarrow$ Performing action $a^t$ on $x_i$ ; | |
| $\theta \leftarrow$ Optimize the model parameter by | |
| fine-tuning with $x_i'$ ; | |
| $r^t \leftarrow$ Compute the reward by Eqn. (7) ; | |
| $x_i \leftarrow x_i'$ ; | |
| $t \leftarrow t + 1$ ; | |
| until $a^t = TM$ or $t = T$; | |
| $X' = X' \cup \{x_i\}$; | |
| Compute the TD error as [33]; | |
| end | |
| end | |

full investigation, we evaluate our method on four different datasets, including Fashion MNIST, CUB (Caltech-UCSD Birds), WebCaricature and CIFAR-100. The selected datasets vary from different sizes, resolution and classes as illustrated in Table 1.

Note that, for fair comparison, all the listed baselines used the same base network for surrogate model training. Also, the hyper parameters in the respective models of all these methods are kept to be same.

| Table 1. The details of different datasets. |
|-------------------------------------------|
| **Datasets** | **# Samples** | **# Class** | **Resolution** |
|---------------|---------------|-------------|----------------|
| CUB-20        | 1,153         | 20          | $224 \times 224 \times 3$ |
| WebCaricature | 1,414         | 20          | $224 \times 224 \times 3$ |
| Fashion MNIST | 60,000        | 10          | $28 \times 28$ |
| CIFAR-100     | 60,000        | 100         | $32 \times 32 \times 3$ |

4.1. Results on CUB-20

**Setting.** To investigate the performance of our method on the small-sized but large-resolution dataset, we extract a sub-set from the CUB dataset [34] with 20 classes including 1153 images. We split the training, validation and testing data with the ratio 0.5, 0.2 and 0.3. Each image is with RGB channels of size $224 \times 224 \times 3$. We use the pre-trained VGG [25] as the state extraction network to produce a 4,096-dimensional state vector for each image. The base network for surrogate model is also VGG with kernel size of 12. The learning rates in 1-150, 150-250 and 250+ epoches are set to 0.1, 0.01 and 0.001, respectively. When training the VGG, we set the number of epoches to 300 and the batch size to 32. Also, the learning rate in fine-tuning is set to

4. Experiments

We present the qualitative and quantitative results of the proposed DeepAugNet, and compare it with state-of-the-art methods developed for image recognition task. For
AN (adding noise) is usually selected to add the difficulty for the images with clear background which are easy to classify. ZM (zoom in) is frequently selected for the images with relative clutter background since the foreground birds are hard to be observed. Also, crop operators are useful to increase the size of foreground birds for small-size targets.

Figure 3. The visual examples of automatic data augmentation steps on CUB-20. AN (adding noise) is usually selected to add the difficulty for the images with clear background which are easy to classify. ZM (zoom in) is frequently selected for the images with relative clutter background since the foreground birds are hard to be observed. Also, crop operators are useful to increase the size of foreground birds for small-size targets.

Figure 4. The visual examples of automatic data augmentation steps on WebCaricature. WP (warp) is the most frequently used action. We could notice that the specific properties are enhanced, e.g., eyes and mouths with the bounding boxes of a same color between the original image and final augmented image.

0.001. The exploration rate for Dueling DQN decays from 1 and stops at 0.01. The size of each experience buffer is set as 10,000 transitions.

We introduce the VGG without any data augmentation (termed VGG-WA) and the VGG with traditional augmentation (termed VGG-TA) as the two preliminary baselines since our DeepAugNet is also trained on the same VGG. The traditional augmentation in VGG-TA includes the operators as flipping, rotation and warp etc. We maintain the same number of augmentation samples for both VGG-TA and DeepAugNet. Directly comparing these three methods (i.e., VGG-WA, VGG-TA and DeepAugNet-VGG) is helpful to reveal the role of data augmentation in the recognition task. Besides, for comparison with the current state-of-the-arts, we introduced the three recent published learning-based augmentation methods, i.e., Mixup [37], AutoAugment [37] and Neural + No Loss [31]. For the AutoAugment and Mixup, we directly used their released implementation. For Neural + No Loss, we re-implement it by ourselves.

Results. We present the test accuracy of these baselines in Table 2. Our result is better than the methods with-
out augmentation, with traditional augmentation, and with learning-based augmentation. Moreover, we illustrate several visual examples of our automatic augmentation steps on CUB-20 in Fig. 3. From Fig. 3, we observe that: (1) **AN (adding noise)** is usually selected to add the difficulty for the images with clear background which are easy to classify. (2) **ZM (zoom in)** is frequently chosen for the images with relative clutter background since the foreground birds are hard to be observed and (3) crop operators are useful to increase the size of foreground birds for small-size targets.

### 4.2. Results on Caricature Recognition

**Setting.** The WebCaricature [11, 12] is a photograph-caricature dataset with the large intra-personal variations of caricatures. We employ the same ratios for training, validation and testing as that in CUB-20. The number of classes is 20 and the total number of images is 1,414. To eliminate the factors that are not relevant to the recognition in caricatures, we generate a bounding box to capture human face according to the facial landmarks. We adopt the pre-trained VGG-Face [25] as the state extraction network. For the architecture of surrogate model, we delete the last linear layer of VGG-Face and meanwhile add two new fully connected layers of (2622, 1024) and (1024, 20). During training, we load the pre-trained VGG-Face parameters and fine-tune the network globally, which is demonstrated to be more promising. The learning rate of our model is initialized as 0.1 with a decay of 0.98. The parameters in deep reinforcement learning are maintained to be same as that in CUB-20. We did not include AutoAugment here since it does not contain the warp etc.

**Results.** We report the test accuracy of these baselines in Table 3. Our result outperforms these baselines with a large margin. According to our observation in Fig. 4, **WP (warp)** is the most frequently used action. We could notice that the specific properties are enhanced, e.g., eyes and mouths with the bounding boxes of a same color between the original image and final augmented image.

### 4.3. Results on Fashion MNIST

**Setting.** Fashion MNIST dataset consists of 40,000 training, 10,000 validation and 10,000 testing samples. There are in total 10 classes: T-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot. Regarding the images in Fashion MNIST are relatively small sized, we directly use the raw images as the states for training our DeepAugNet. In our implementation, we use the learning rate decay: the learning rates in 1-150, 150-250 and 250+ epoches are set to 0.1, 0.01 and 0.001, respectively. For the base network of surrogate model, we adopt the AlexNet [16] which is good enough to obtain promising results in Fashion MNIST. Typically, we introduce the AlexNet without any data augmentation (termed AlexNet-WA) and the AlexNet with traditional augmentation (termed AlexNet-TA) as the two preliminary baselines since our DeepAugNet is also trained on the same AlexNet. The traditional augmentation in AlexNet-TA includes the operators as flipping, rotation and warp etc. Besides, we also introduce SqueezeNet [13] and Yu et al.’s method [36] to investigate the current level on the shallow networks (i.e., less than 10 layers). We did not include AutoAugment since most of its actions are specifically designed for color images.

**Results.** The test accuracy of these baselines is reported in Table 4. Our result outperforms all the baselines except

3. https://github.com/WonderSeven/DeepAugNet/blob/master/demo.mp4
4. https://www.kaggle.com/zalando-research/fashionmnist
the Mixup. We illustrate several visual examples of our automatic augmentation steps on Fashion MNIST in Fig. 5, where the actions RT (rotation) and FP (flip) are the most frequently selected to produce the sufficient samples with different orientations.

To exploit the efficacy of policy learned in different sizes of set, we investigated the policy transfer in Fashion MNIST, showing the learned policies are actually helpful to improve the recognition results in Table 5. Specifically, taking the first line of Table 5 as an example, we first train an agent on the training set only including 500 samples. Then, we directly use the learned policy to augment the training set with another 1,000 samples. The testing accuracy for this strategy is 80.86 % which outperforms the baseline (traditional augmentation) trained on 1,000 samples with accuracy of 80.09 %. The results demonstrate the policies learned by our method on a smaller size set (Source) work well on another bigger set (Target).

4.4. Results on CIFAR-100

Setting. CIFAR-100 dataset\(^5\) consists of 50,000 training and 10,000 testing samples with in total 100 classes. To accelerate the training procedure, we randomly sample 4,000 images as the reduced training set and 1,000 images as the validation set from training set. There does not exist any overlap between these two sets. Besides, we use a small network (i.e., DenseNet-BC [9] with 50 layers\(^6\)) as our surrogate model to get the augmentation policy. For dealing with large-scale dataset, we increase the batch size during learning the policy.

Like Fashion MNIST, we directly use the raw images as the states for training our DeepAugNet. In our implementation, we use the cosine annealing learning rate decay [18]. For the final classification model, we adopt the DenseNet-BC with 100 layers. Typically, we introduce the DenseNet-BC without any data augmentation (termed DenseNet-WA) and the DenseNet-BC with traditional augmentation (termed DenseNet-TA) as the two preliminary baselines. Also, two learning-based methods: Mixup and AutoAugment are introduced as the additional baselines.

Table 6. Test accuracy of different baselines on CIFAR-100.

| Method               | Accuracy (%) | Venue       |
|----------------------|--------------|-------------|
| DenseNet-WA [9]      | 73.42        | CVPR 2017   |
| DenseNet-TA [9]      | 74.41        | CVPR 2017   |
| Mixup [37]           | 75.28        | ICLR 2018   |
| AutoAugment [3]      | 74.96        | Arxiv 2018  |
| DeepAugNet-DenseNet  | 75.11        | Proposed    |

Results. The test accuracy of these baselines is reported in Table 6. Our result ranks the 2\(^{nd}\) place in all the listed baselines. Also, we show several visual examples of our automatic augmentation steps on CIFAR-100 in Fig. 6. Intuitively, the actions ZM (zoom in), WP (warp) and AN (adding noise) are the most frequently used.

Discussion. Compared with the current state-of-the-arts leaning-based augmentation methods, we can infer that our method can achieve a significant improvement in small-scale datasets, and a comparable performance in large-scale datasets. This is because that, when the limited samples in small-scale datasets are not sufficient to train a promising model, the predefined assumption of data distribution might not guarantee an effective augmentation. Also, thanks to the deterministic policy, our method can obtain the robust results in both small- and large-scale datasets.

5. Conclusion

Regarding that the criterion of best augmentation is difficult to define, we innovatively attempt to model the automatic augmentation as a deep reinforcement learning problem by learning a deterministic augmentation policy. To achieve this goal, a joint learning scheme by integrating a hybrid architecture of Dueling DQN and a surrogate model is developed where the learning policy guides the augmentation by directly optimizing the performance improvement. Extensive experiments validated the effectiveness of our

---

\(^5\)https://www.cs.toronto.edu/~kriz/cifar.html

\(^6\)https://github.com/bamos/densenet.pytorch
DeepAugNet. Our future directions include the extension to other vision tasks (e.g., segmentation) and the combination to other DRL architectures [24, 17, 29, 35]. Also, we will exploit the different action selection preferences in different datasets to guide better action design and initialization.

References

[1] A. Antoniou, A. Storkey, and H. Edwards. Data augmentation generative adversarial networks. arXiv:1711.04340, 2018. 1, 2, 3

[2] J. C. Caicedo and S. Lazebnik. Practical block-wise neural network architecture generation. In ICCV, 2015. 2

[3] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. arxiv.org/abs/1805.09501, 2018. 1, 2, 3, 7, 8

[4] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In CVPR, 2016. 2

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In NIPS, 2014. 2

[6] M. Guo, J. Lu, and J. Zhou. Dual-agent deep reinforcement learning for deformable face tracking. In ECCV, 2018. 2

[7] S. Gurumurthy, R. K. Sarvadevabhatla, and R. V. Babu. Deligan: Generative adversarial networks for diverse and limited data. In CVPR, 2017. 1, 2, 3

[8] J. Han, L. Yang, D. Zhang, X. Chang, and X. Liang. Reinforcement learning-based learning for video object segmentation. In CVPR, 2018. 2

[9] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In CVPR, 2017. 8

[10] S.-W. Huang, C.-T. Lin, S.-P. Chen, Y.-Y. Wu, P.-H. Hsu, and S.-H. Lai. Auggan: Cross domain adaptation with gan-based data augmentation. In ECCV, 2018. 1, 2, 3

[11] J. Huo, Y. Gao, Y. Shi, and H. Yin. Variation robust cross-modal metric learning for caricature recognition. In ACM Multimedia Thematic Workshops, 2017. 7

[12] J. Huo, W. Li, Y. Shi, Y. Gao, and H. Yin. Webcaricature:a benchmark for caricature recognition. In BMVC, 2018. 7

[13] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. arxiv:1602.07360, 2016. 7

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, 2017. 2

[15] X. Kong, B. Xin, Y. Wang, and G. Hua. Collaborative deep reinforcement learning for joint object search. In CVPR, 2017. 2

[16] A. Krizhevsky and I. S. and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. NIPS, 2012. 1, 2, 7

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. arxiv.org/abs/1509.02971, 2016. 9

[18] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In ICLR, 2017. 8

[19] V. Mnih, K. Kavukcuoglu, D. Silver, and et al. Playing atari with deep reinforcement learning. NIPS Deep Learning workshop, 2013. 2

[20] V. Mnih, K. Kavukcuoglu, D. Silver, and et al. Human-level control through deep reinforcement learning. Nature, 2015. 2

[21] J. Park, J.-Y. Lee, D. Yoo, and I. S. Kweon. Distort-and-recover: Color enhancement using deep reinforcement learning. In CVPR, 2018. 2

[22] L. Ren, X. Yuan, J. Lu, M. Yang, and J. Zhou. Deep reinforcement learning with iterative shift for visual tracking. In CVPR, 2017. 2

[23] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, 2015. 1, 2

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. arxiv.org/abs/1707.06347, 2017. 9

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arxiv:1409.1556, 2014. 5, 7

[26] G. Song, H. Myeong, and K. M. Lee. Seednet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation. In CVPR, 2018. 2

[27] R. S. Sutton and A. G. Barto. Introduction to reinforcement learning. In 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[28] Y. Tang, Y. Tian, J. Lu, P. Li, and J. Zhou. Deep progressive reinforcement learning for skeleton-based action recognition. In CVPR, 2018. 2

[29] A. S. Vezhnelets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. arxiv.org/abs/1703.01161, 2017. 9

[30] R. Volpi, H. Namkoong, O. Sener, J. Duchi, V. Murino, and S. Savarese. Generalizing to unseen domains via adversarial data augmentation. In NIPS, 2018. 2

[31] J. Wang and L. Perez. The effectiveness of data augmentation in image classification using deep learning. In CVPR, 2018. 6, 7

[32] Y.-X. Wang, R. Girshick, M. Herbert, and B. Hariharan. Low-shot learning from imaginary data. In CVPR, 2018. 1, 2, 3

[33] Z. Wang, T. Schaul, M. Hasselt, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling network architectures for deep reinforcement learning. In ICML, 2016. 4, 5

[34] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. BeJongie, and P. Perona. Caltech-ucsd birds 200. In CNS-TR, 2009. 2

[35] M. Wulfmeier, P. Ondruska, and I. Posner. Maximum entropy deep inverse reinforcement learning. arxiv.org/abs/1507.04888, 2015.

[36] B. Yu, T. Liu, M. Gong, C. Ding, and D. Tao. Correcting the triplet selection bias for triplet loss. In ECCV, 2018. 7

[37] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. Mixup: Beyond empirical risk minimization. In ICLR, 2018. 1, 2, 3, 6, 7, 8
[38] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu. Practical block-wise neural network architecture generation. In CVPR, 2018.

[39] Zhu, Jun-Yan, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Computer Vision (ICCV), 2017 IEEE International Conference on, 2017.