16th International Conference on Machine Vision Applications (MVA)
National Olympics Memorial Youth Center, Tokyo, Japan, May 27-31, 2019.

Domain Adaptation using a Gradient Reversal Layer with Instance Weighting

Kosuke Osumi, Takayoshi Yamashita, Hironobu Fujiyoshi
Chubu University
1200 Matsumoto-cho, Kasugai, Aichi 487-8501, Japan
{osmksk05@mprg.cs,takayoshi@isc,fujiyoshi@isc}.chubu.ac.jp

Abstract

We propose a new method for domain adaptation that uses a gradient reversal layer (GRL) with instance weighting. Domain adaptation methods that use GRL have the latent problem of learning data that do not contribute to improving the accuracy with which the target domain is recognized. The proposed method weights each source domain sample. This enables us to control the gradients of training samples that do not contribute to improving accuracy. In an evaluation experiment using computer graphics and real image data, accuracy in recognizing the target domain improved by 5.3% compared with existing domain adaptation methods.

1 Introduction

Deep convolutional neural network (DCNN) [1] have the potential to achieve high recognition accuracy by using a larger amount of training data [2]. However, collecting many training data increases costs because those data require corresponding labels, which are annotated manually. A dataset generated by computer graphics (CG) data can be a practical alternative to a real image dataset. The CG data can reduce the costs of collecting images and preparing annotated data associated with real images, so research that uses CG data has been actively pursued in recent years [3, 4, 5]. However, when evaluating real images with models that were trained with CG data, the recognition accuracy decreases due to the difference in the domain distributions between real images and CG data.

This problem can be solved using *domain adaptation*. Domain adaption improves the accuracy by using both a *target domain*, with an insufficient amount of data for the domain to be recognized, and a *source domain*, with a sufficient amount of data for a domain that will not be recognized. Several methods for domain adaptation have been proposed [6, 7, 8]. Among them, one major method is to use domain adversarial neural network (DANN) [9], which introduce adversarial learning with a negative gradient. A DANN aims to obtain features that are domain invariant by reversing a gradient calculated from a domain-classification error so that it is hard for a domain classifier to classify samples as source or target domains. However, DANN suffer from susceptibility to the influence of source domain samples that do not contribute to improving the accuracy with which the target domain is recognized (i.e. the recognition accuracy).

In this paper, we propose instance weighting, which controls the gradients of training samples that do not contribute to improving recognition accuracy by weighting the source domain samples. Instance weight is calculated using a pre-classifier trained with source domain samples and a target classifier trained with source and target domain samples. The calculated instance weight is used to select useful source domain samples. Consequently, the proposed method can improve the recognition accuracy without training labels of the target domain. Experimental results with two patch-image datasets created from real images and CG data demonstrate that the proposed method can improve classification accuracy.

2 Domain adversarial neural network

DANN is a domain adaptation method that uses adversarial learning. DANN is composed of a feature extractor, label predictor, and domain classifier, as shown in Figure 1. DANN also introduces a *gradient reversal layer* (GRL) between the feature extractor and domain classifier. In forward propagation, the GRL performs as an identity-mapping function for the output of the feature extractor. The output of the feature extractor is input to the domain classifier. In back-propagation processing, the GRL multiplies a gradient calculated from the domain-classification error by a negative scaler, $-\lambda$, and propagates the negative gradient to the feature extractor. Let $\mathbf{x}$ be an input vector, $\mathbf{I}$ be an identity matrix, and R_λ be the GRL. The forward propagation and the back-propagation are defined as

$$R_\lambda(\mathbf{x}) = \mathbf{x}, \tag{1}$$

$$\frac{dR_\lambda}{d\mathbf{x}} = -\lambda\mathbf{I}. \tag{2}$$

Here, in order to make the feature extractor robust for the domain shift, the influence of the negative gradient must be decreased with GRL at the

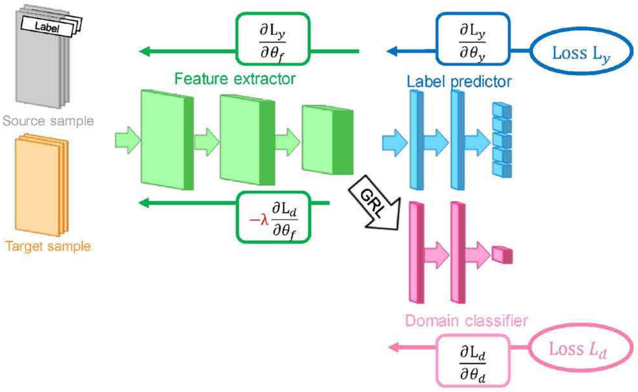

Label
Source sample
Target sample
$\frac{\partial L_y}{\partial \theta_f}$
Feature extractor
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
Loss L_y
Label predictor
GRL
Domain classifier
$\frac{\partial L_d}{\partial \theta_d}$
Loss L_d

Figure 1. Network architecture of DANN.

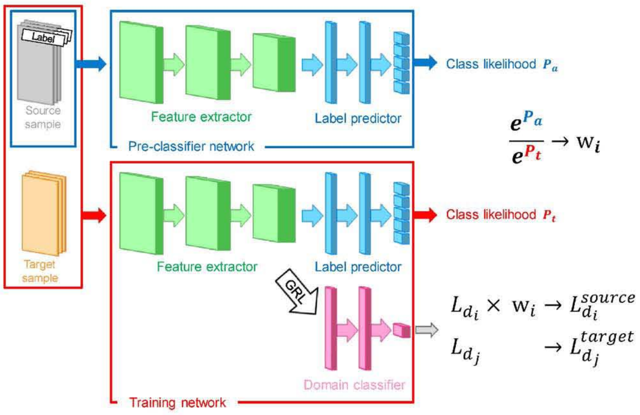


Figure 2. Network architecture of proposed method.

early stage of learning. This is because the domain-classification error is large due to insufficient learning, which impedes the acquisition of the ability to predict labels. The reason for this is that a domain classifier at the early stage of learning outputs larger domain-classification error, which disrupts the improvement of label-prediction performance. This problem can be solved by changing the value of λ from 0 to 1 on the basis of the number of network updates p and a coefficient of λ determination γ. λ is calculated by

$$\lambda = \frac{2}{1+e^{-\gamma p}} - 1. \tag{3}$$

Meanwhile, because the label predictor learns with only the labeled source domain samples, target domain samples in mini-batch data are excluded from training the label predictor. The loss function of DANN can be formulated as follows:

$$\begin{aligned} E &= \sum_{i=1..N} L_y(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y),y_i) \\ &+ \sum_{i=1..N} L_d(G_d(R_\lambda(G_f(\mathbf{x}_i;\theta_f));\theta_d),d_i), \end{aligned} \tag{4}$$

where labeled and unlabeled input images $\mathbf{x}_i$ are denoted as source and target data, respectively. G_f, G_y, and G_d are the feature extractor, label predictor, and domain classifier, respectively. Now, let parameters of each classifier be θ_f, θ_y and θ_d. When labels for the output of G_y and G_d are assumed to be y_i and d_i, network parameters are updated using the loss functions L_y and L_d. λ is determined by the update counts, so it is not affected by back propagation.

If we focus only on the label-prediction performance, the parameters of the feature extractor and label predictor are suitable only for the source domain. However, the recognition accuracy does not improve. Therefore, it is necessary to train the network so that the domain distribution between the source and target domains becomes domain invariant. Domain-invariant features can be obtained by estimating θ_f, which maximizes domain-classification loss, and θ_d, which minimizes domain-classification loss. In order to enable both optimal label-prediction performance and domain-invariant features, $-\lambda$, which adjusts the gradient of the domain classifier, is used.

3 Proposed method

The DANN inputs source and target domain samples at the same time and performs domain classification with the feature extractor and domain classifier. When the domain classifier classifies domain, the DANN also uses source domain samples that do not contribute to improving the recognition accuracy. The proposed method introduces instance weighting in the DANN to efficiently learn source domain samples. Figure 2 shows the network architecture of the proposed method. Our method uses a DANN that must be trained and a pre-classifier network that estimates instance weight.

3.1 Instance weighting

Instance weighting adjusts the influence that source domain samples have on domain-classification error by weighting to improve target-classification performance. Instance weight is calculated from a pre-classifier network trained with only source domain samples in advance and a DANN that must be trained to classify target domain samples correctly. The pre-classifier network consists of a feature extractor and label predictor. Using the pre-classifier network, the class likelihood P_a for each source domain sample is computed.

Next, we prepare a training network, i.e. a DANN. The training network has a structure in which the GRL and domain classifier are added to the pre-classifier network. The training network calculates a class likelihood P_t for the source domain samples. P_t is obtained using the feature extractor and label predictor when a source domain sample is input. Instance weight w_i is computed as follows:

$$w_i = \frac{e^{P_a(y_i|\mathbf{x}_i)}}{e^{P_t(y_i|\mathbf{x}_i)}}, \tag{5}$$

where $P_a(y_j|\mathbf{x}_i)$ is the likelihood of class y_j, and $P_t(y_i|\mathbf{x}_i)$ is the likelihood of class y_i, when a source domain sample $\mathbf{x}_i$ is input to the pre-classifier network.

3.2 Domain-classification loss

The instance weight w_i is used to weight domain-classification loss. Let $L_{d_i}^{source}$ be a domain-classification loss for a source domain sample. Instance weighting multiplies $L_{d_i}^{source}$ by w_i. The domain classifier decreases loss of the source domain samples that do not contribute to improving the recognition accuracy and increases the domain-classification loss of the source domain samples that improve it. If $\mathbf{x}_i$ is a source domain sample, domain-classification loss $L_{d_i}^{source}$ is defined as Eq. (6).

$$L_{d_i}^{source} = L_{d_i}(G_d(R_\lambda(G_f(\mathbf{x}_i;\theta_f));\theta_d), d_i)w_i. \tag{6}$$

If $\mathbf{x}_i$ is a target domain sample, domain-classification loss $L_{d_i}^{target}$ is defined as Eq. (7).

$$L_{d_i}^{target} = L_{d_i}(G_d(R_\lambda(G_f(\mathbf{x}_i;\theta_f));\theta_d), d_i). \tag{7}$$

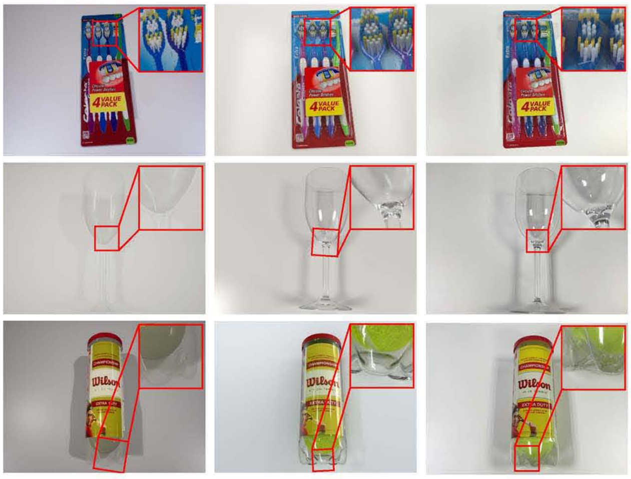
Let N_{source} be the number of source domain samples and N_{target} be the number of target domain samples in a mini-batch. The domain-classification loss L_d of the mini-batch is formulated as

$$L_d = \frac{1}{M}\left(\sum_{i=1}^{N_{source}} L_{d_i}^{source} + \sum_{j=1}^{N_{target}} L_{d_j}^{target}\right), \tag{8}$$

where M is mini-batch size.

3.3 Training procedure

The DANN [9] does not learn using source domain data beforehand. At the early stage of training, the effect of the GRL is rather small, caused by λ in Eq. (3). Consequently, the DANN requires a large number of network updates to acquire domain invariance. However, the proposed method learns the parameters of the pre-classifier network with source domain data in advance. Then, the learned parameters are fine-tuned in the training network. It can be assumed that the training network is pre-trained by source domain data. The proposed method does not need to update the value of λ such as in Eq. (3). The mini-batch in the training network consists of source and target domain samples. The training network has the same architecture as the DANN, so the source and target domain samples can be input to the network. Instance weight is calculated using the ratio of the class-likelihood output from the label predictor of the pre-classifier network to that from the class predictor of the learning network. The calculated instance weight is adapted for the domain-classification loss of the source domain samples of the training network.



(a) Real-time rendering image (b) Pre-rendering image (c) Real image

Figure 3. Example of patch dataset.

4 Experiments

We experimented with a patch-image dataset composed of three domains, including two types of CG data and one type of real image (see figure 3). The two types of CG data were real time-rendering and pre-rendering images. The former is generated by a game engine called Unreal Engine 4. Since the quality of the generated image is low in this CG data, generating time is fast. Thus, it is easy to generate in large quantities. the latter is generated by NVIDIA's physically based rendering technology IRay. This CG data takes time than generating real-time rendering image, but it can generate photorealistic and high-quality images. These domain images are split into patch data, so that the number of data is increased, and the local area can be caught using patch data including the domain-specific texture. Domain adaptation is performed between the two types of CG images, and the model is evaluated with real images.

4.1 Testing proposed method

Our proposed network is composed of a pre-classifier and training network, based on the VGG-16 model [10]. The GRL and domain classifier in the training network are connected for the output of the last convolution

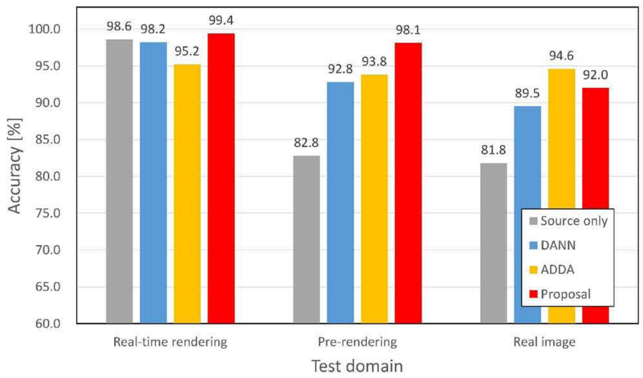
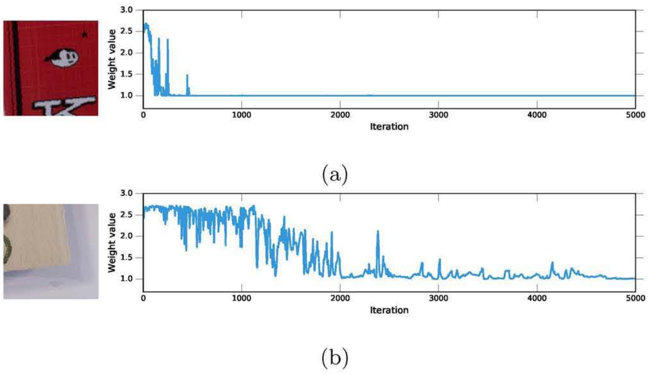


Figure 4. Recognition accuracy in each domain.

layer of the feature extractor. For parameter optimization, we used stochastic gradient descent with momentum. The input-image size is 224 × 224 [pixels]. Each domain sample consists of 25 classes. Each source domain contains 1000 images per class, and each target domain contains 500 images per class. For comparison methods, VGG-16 trained only with a source domain, a DANN [9] in which instance weighting was not applied, and adversarial discriminative domain adaptation (ADDA) [12] are used.

Results For accuracy comparison, we prepared 100 images per class from unlearned data of each domain and used them as evaluation data. Then, the accuracy with which the target domain was recognized was compared using the real-time rendering, pre-rendering, and real images. The results are shown in Figure 4. Compared to the DANN, the proposed method achieved a 5.3% improvement in recognizing the pre-rendering image. The accuracy with which the real-time rendering image was recognized improved by 1.2%. Further, the accuracy of recognizing the real image, the unlearned domain, improved by 2.5%. Comparing the accuracy of the proposed method and that of ADDA, which is a method with more accurate recognition than DANN, accuracy with which pre-rendering images were recognized improved by 4.3%.

Discussion Figure 5 shows the transition of the instance weight for the training samples. Figure 5 (a) seems to reduce the instance weight with a small number of updates because the difference between domains is small, making it is unnecessary to consider the difference between domains. In Figure 5 (b), at the beginning of learning, P_t becomes smaller due to the effect of the target domain on the feature extractor. Hence, the model learns to consider the difference between domains, which would cause the large number of updates until convergence.



(a)

(b)

Figure 5. Two examples of transition of instance weight. (a) Instance weight quickly converges. (b) Instance weight slowly converges.

Table 1. Accuracy with different source domains.

Real-time [%]	Pre-rendering[%]	Real-time + Pre-rendering[%]
98.6	82.8	81.8

4.2 Testing accuracy with different source domains

We tested the effect of changing the source domain on recognition accuracy using the proposed method. To measure the performance of the proposed method, we used a single domain for source and target domains. In this section, we use real images as the target domain and change the source domain to the following three settings: real-time rendering, pre-rendering, and real-time rendering + pre-rendering images. In each source domain setting, we prepare 25,000 training images. As target domain data, we prepare 12,500 training real images. We evaluated with 2,500 real images.

Results As shown in Table 1, using pre-rendering images (high-quality CG) as the source domain enabled better recognition performance than using real-time rendering images (low-quality CG). In addition, using both real-time rendering and pre-rendering images as source domains further improves the accuracy. These results show similarities with the characteristics of domain randomization [11], which improves recognition and detection performance by learning various domains simultaneously.

5 Conclusion

We have proposed unsupervised domain adaptation using a gradient reversal layer with instance weighting. We control the effect of samples that do not contribute to improving recognition accuracy with instance weighting. As a result of our evaluation experiment, which consisted of training with 25,000 images of real-time rendering CG and 12,500 images of pre-rendering CG, the accuracy with which the actual images were recognized improved. We also found that using multiple CG domains as source domains improves the accuracy. In future work, we will examine the adjustment method of the GRL and its application to other tasks such as segmentation.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, pp. 22782324, 1998.

[2] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era," International Conference on Computer Vision, pp. 843852, 2017.

[3] B. Sun and K. Saenko, "From virtual to reality: Fast adaptation of virtual object detectors to real domains," British Machine Vision Conference, 2014.

[4] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On Pre-Trained Image Features and Synthetic Images for Deep Learning," arXiv preprint arXiv:1710.10710v2, 2017.

[5] V.S.R. Veeravasarapu, C. Rothkopf, and R. Visvanathan, "Adversarially Tuned Scene Generation," Computer Vision and Pattern Recognition, pp. 25872595, 2017.

[6] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," International Conference on Computer Vision, pp. 29602967, 2013.

[7] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep Domain Confusion: Maximizing for Domain Invariance," arXiv preprint arXiv:1412.3474v1, 2014.

[8] R. Gopalan, R. Li, and R. Chellappa, "Domain Adaptation for Object Recognition: An Unsupervised Approach," International Conference on Computer Vision, pp. 9991006, 2011.

[9] Y. Ganin and V. Lempitsky, "Unsupervised Domain Adaptation by Backpropagation," International Conference on Machine Learning, pp. 11801189, 2015.

[10] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," International Conference on Learning Representations, 2015.

[11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," International Conference on Intelligent Robots and Systems, 2017.

[12] E. Tzeng, J. Hoffman, K. Saenko, T.Darrell, "Adversarial Discriminative Domain Adaptation," Computer Vision and Pattern Recognition, pp. 71677176, 2017.