HETEROGENEOUS DOMAIN GENERALIZATION VIA DOMAIN MIXUP

Yufei Wang\textsuperscript{1,2}, Haoliang Li\textsuperscript{2}, and Alex C. Kot\textsuperscript{2}

\textsuperscript{1} University of Electronic Science and Technology of China, China
\textsuperscript{2} Nanyang Technological University, Singapore

ABSTRACT

One of the main drawbacks of deep Convolutional Neural Networks (DCNN) is that they lack generalization capability. In this work, we focus on the problem of heterogeneous domain generalization which aims to improve the generalization capability across different tasks, which is, how to learn a DCNN model with multiple domain data such that the trained feature extractor can be generalized to supporting recognition of novel categories in a novel target domain. To solve this problem, we propose a novel heterogeneous domain generalization method by mixing up samples across multiple source domains with two different sampling strategies. Our experimental results based on the Visual Decathlon benchmark demonstrates the effectiveness of our proposed method. The code is released in \url{https://github.com/wyf0912/MIXALL}.

Index Terms— Heterogeneous domain generalization, mixup, generalization capability

1. INTRODUCTION

Deep Convolutional Neural Networks (DCNN) are widely used in different computer vision tasks (e.g. object recognition/detection, semantic segmentation). However, they are also known to lack generalization capability, especially under the scenario that large-scale training data cannot be obtained. To mitigate such a problem, one can use a DCNN pre-trained by large-scale publicly available data (e.g. ImageNet \cite{1}), and further finetune the network based on another classification task with few training samples. Another strategy is to directly extract feature representation from a pre-trained DCNN and further use another classifier (e.g. support vector machine, shallow multi-layer network) for recognition purpose. However, directly finetuning the DCNN or extracting features from a pre-trained DCNN may not be able to achieve desired performance mainly due to the domain gap between training and testing data, which further leads to poor generalization capability of trained classifier and brings negative impact on the practical application of DCNN.

Many efforts have been done to improve the generalization capability of DCNN. Domain adaptation and domain generalization are related research directions, which aim to mitigate the influence of domain shift between training and testing data. Domain adaptation (DA) \cite{2,3,4,5} assumes that we can obtain the samples from the testing data (a.k.a. target domain) during the training stage, but have no label or few labels. Universal domain adaptation (UDA) \cite{4} is a relatively new research branch of DA. In UDA, labels are divided into two parts, common label set and private label set. You et al. \cite{4} adopt the sample-level transferability to develop a weighting mechanism. Another related research direction is partial domain adaptation (PDA) \cite{6}, which assumes that the target label space is a subset of the source label space. Compared with the aforementioned two tasks, open set domain adaptation (OSDA) \cite{7} is more challenging as it sets private labels as an “unknown” class. These tasks all assume that there is an intersection between the source and target domain.

Domain generalization (DG) \cite{8,9,10,11,12,13} aims to address the harder setting where the target domain is not available during training stage. The general idea of DG is to train the model which is expected to generalize better by only using the data from the source domain(s) and then evaluate based on the “unseen” target domain. While both DA and DG require that the label space of source and target to be the same, heterogeneous domain generalization (HDG) \cite{14} assumes that different domains share different label space. In the case of HDG, the label space between source and target are disjoint. Such setting is widely encountered in the field of computer vision where the pre-trained model can be re-used for different types of applications.

This research was carried out at the Rapid-Rich Object Search (ROSE) Lab, Nanyang Technological University (NTU), Singapore and supported by grants from NTU’s College of Engineering (M4081746.D90). This research was also supported by SSIRI (Project No.206-A018001). H. Li thanks the Wallenburg-NTU Presidential Postdoc Fellowship grant.
where the \( \hat{MIXUP} \) distribution is first proposed by \[2.3. \mathbf{MIXUP} \text{ for HDG} \]

test the classification accuracy of \( y \) from the original label \( l \).

We first consider a simple baseline model called aggregation (AGG) for HDG, which minimize the following loss:

\[
\begin{aligned}
&\min_{\theta, \alpha} \sum_{D_j \in \mathcal{D}_s} \sum_{(x_i^j, y_i^j) \in D_j} l(g_\alpha(f_\theta(x_i^j)), \hat{y}_i^j) \\
\end{aligned}
\tag{1}
\]

Here \( l \) is the cross-entropy loss function. \( x_i^j \) is the \( i \)th image in domain \( D_j \) and \( \hat{y}_i^j \) is the one-hot augmented label transferred from the original label \( y_i^j \). We can get the length of one-hot vector \( \hat{y}_i^j \) by

\[
|\hat{y}_i^j| = \sum_{D_j \in \mathcal{D}_s} k_j
\tag{2}
\]

Specifically, the relationship between \( \hat{y}_i^j \) and \( y_i^j \) can be given as

\[
\hat{y}_i^j(n) = \begin{cases} 
1 & \text{if } n > k_0 \text{ and } y_i^j + \sum_{t=1}^{j-1} k_t = n \\
1 & \text{if } n <= k_0 \text{ and } y_i^j = n \\
0 & \text{otherwise}
\end{cases}
\tag{3}
\]

where \( \hat{y}_i^j(n) \) is the \( n \)th element in one-hot label \( \hat{y}_i^j \), \( k_t \) is the number of pairs in \( D_t \). Then we fix the module \( f_\theta \) and use it to extract the features from \( D_t \). We can train a set of classifiers \( G = \{g_{\alpha_s}\}_{s=1}^{|\mathcal{D}_t|} \) using the training set of datasets \( \mathcal{D}_t \). Finally, we test the classification accuracy of \( G \) using the testing set of target domain.

2.3. \textbf{MIXUP} for HDG

\( \text{MIXUP} \) distribution is first proposed by \[2.3. \mathbf{MIXUP} \text{ for HDG} \]

which aims to alleviate the behaviors of DCNN such as memorization and sensitivity to adversarial examples. The original \text{mixup} distribution is defined as follow

\[
\mu(\tilde{x}, \tilde{y}|x_i^q, \hat{y}_i^q) = \frac{1}{n} \sum_{j=1}^{n} \mathbb{E}_{\lambda}[\delta(\tilde{x} = \lambda x_i^q + (1 - \lambda)x_j^q, \tilde{y} = \lambda \hat{y}_i^q + (1 - \lambda)\hat{y}_j^q)]
\tag{4}
\]
where $\lambda \in (0, 1)$, $E$ denotes expectation operation, $\delta$ is Dirac delta function [15]. Here, the data $\{(x_i^q, y_i^q)\}_{i=1}^n$ come from the same distribution $q$.

However, in the task of HDG, the training data come from different distributions and disjoint label space. In order to let the mixup distribution fit the HDG task, we proposed a new distribution named HDG mixup distribution which can integrate information from different domains

$$
\mu_{H DG}(\bar{x}, \bar{y}|x_i^q, y_i^q) = \frac{1}{n_{sum}} \sum_{j=1}^{\vert D_s \vert} \sum_{k=1}^{\vert D_s \vert} E(\delta(\bar{x} = \lambda \cdot x_j^q + (1 - \lambda) \cdot x_k^q, \bar{y} = \lambda \cdot y_j^q + (1 - \lambda) \cdot y_k^q))
$$

where $n_{sum}$ is the total number of samples in $D_s$, $(x_i^q, y_i^q)$ is a feature-target pair from an arbitrary domain $D_q$ in $D_s$.

After encoding the label to a one-hot vector using the method in Sec.2.2, we can sample mixed feature-target pairs from the HDG mixup distribution.

**MIXUP from two domains** The first sampling strategy we proposed is to mix up from two arbitrary domains in one iteration, abbreviated as MIXUP. Specifically, we can sample two data pairs from two arbitrary domains in one iteration and generate a new mixed data pair $(\bar{x}, \bar{y})$

$$
\bar{x} = \lambda x_j^q + (1 - \lambda) x_k^q \quad \bar{y} = \lambda y_j^q + (1 - \lambda) y_k^q
$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$ for $\alpha \in (0, \infty)$. $p, q$ are discrete random variables draw from $\{1, 2, \ldots, \vert D_s \vert\}$. $(x_i^p, y_i^p)$ and $(x_i^q, y_i^q)$ are two feature-target pairs drawn at random from two arbitrary domains, and $\lambda \in [0, 1]$ which is controlled by the hyper-parameter $\alpha$.

**MIXUP from all domains** We can also mixup from all domains(MIX-ALL), i.e., mixup all samples from $\vert D_s \vert$ source domains in one iteration.

$$
\bar{x} = \sum_{q=1}^{\vert D_s \vert} \lambda_q x_i^q \quad \bar{y} = \sum_{q=1}^{\vert D_s \vert} \lambda_q y_i^q
$$

Here $(x_i^q, y_i^q)$ is an arbitrary data pair from domain $D_q$, $\bar{x}$ is the generated sample and $\bar{y}$ is the corresponding label. $\lambda_q$ is a random variable to control the degree of mixing which is generated and normalized as below

$$
\lambda_q = \frac{e^{u_q}}{\sum_{q=1}^{\vert D_s \vert} e^{u_q}}
$$

where random variable $u_q \sim U(0, \beta)$. We introduce a hyper-parameter $\beta$ to control the generation of random variable vector $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_{\vert D_s \vert}]$. The relationship between $\beta$ and the expectation of $max(\lambda)$ and $\sigma(\lambda)^2$ can be found in Fig.1, which shows that with the increase of $\beta$, generated samples become more definiteness and less randomness.

**Fig. 1.** The relationship of hyper-parameter $\beta$ and the expectation of maximum value and variance of random variable vector $\lambda$ under the condition of $\vert D_s \vert = 6$.

### 3. EXPERIMENT

We evaluate our model on Visual Domain Decathlon (VD) benchmark [16] by following [14].
Table 1. classification accuracy (%) and VD scores on target datasets in VD using SVM.

| Target          | Im.N. PT | CrossGrad [10] | MR [11] | MR-FL [14] | Reptile [12] | AGG [14] | FC [14] | MIXUP | MIX-ALL |
|-----------------|----------|----------------|---------|------------|--------------|-----------|---------|-------|---------|
| Aircraft        | 16.62    | 19.92          | 20.91   | 18.18      | 19.62        | 19.56     | 20.94   | 21.00 | 20.61   |
| D. Textures     | 41.70    | 36.54          | 32.34   | 35.69      | 37.39        | 36.49     | 38.88   | 39.36 | 39.04   |
| VGG-Flowers     | 51.57    | 57.84          | 34.92   | 53.04      | 58.26        | 58.04     | 58.53   | 59.71 | 61.37   |
| UCF101          | 44.93    | 45.80          | 47.34   | 48.10      | 49.85        | 46.98     | 50.82   | 51.74 | 55.99   |
| Ave.            | 38.71    | 40.03          | 34.02   | 38.75      | 41.28        | 40.27     | 42.92   | 44.25 |         |
| VD-Score        | 308      | 289            | 296     | 324        | 390          | 344       | 357     | 400   |         |

Table 2. classification accuracy (%) and VD scores on target datasets in VD using KNN.

| Target          | Im.N. PT | CrossGrad [10] | MR [11] | MR-FL [14] | Reptile [12] | AGG [14] | FC [14] | MIXUP | MIX-ALL |
|-----------------|----------|----------------|---------|------------|--------------|-----------|---------|-------|---------|
| Aircraft        | 11.46    | 15.93          | 12.03   | 11.46      | 13.27        | 14.03     | 16.01   | 14.52 | 14.85   |
| D. Textures     | 39.52    | 31.98          | 27.93   | 39.41      | 32.80        | 32.02     | 34.92   | 34.26 | 33.67   |
| VGG-Flowers     | 41.08    | 48.00          | 23.63   | 39.51      | 45.80        | 45.98     | 47.04   | 48.24 | 49.71   |
| UCF101          | 35.25    | 37.95          | 34.43   | 35.25      | 39.06        | 38.04     | 41.87   | 45.03 | 46.41   |
| Ave.            | 31.83    | 33.47          | 24.51   | 32.73      | 32.52        | 34.96     | 35.51   |       |         |
| VD-Score        | 215      | 188            | 144     | 215        | 201          | 189       | 236     | 259   | 268     |

Dataset. There are ten heterogeneous domains in VD, including ImageNet [1], CIFAR-100 [17], Aircraft [18], Describable textures [19], German traffic signs [20], etc. Different domains contain different image categories as well as a different number of images, e.g., SVHN (Street View House Number) [21] Dataset has 10 classes and 47217 training samples, UCF101 Dynamic Images, a compact representation of videos proposed by [22], has 101 classes and 7629 training samples, etc. We regard the label space of them disjoint, which is suitable for HDG task. All images in VD have been resized to have a shorter size of 72 pixels.

By following [14], we split VD into two parts: source domain $D_s$ and target domain $D_t$. In particular, $D_s$ includes six larger domains, i.e., CIFAR-100 [17], Daimler Ped [23], GTSRB [20], Omniglot [24], SVHN [21] and ImageNet [1]. $D_t$ includes four smaller domains, i.e., Aircraft [18], D. Textures [19], VGG-Flowers [25] and UCF101 [22].

Networks and Training Details. We use ResNet-18 [26] as the backbone network for all experiments. For fair comparison, we use the same network architecture as [14] and also freeze the first four blocks of ResNet-18. And we use the weight pre-trained from aggregation network (AGG) [14] for parameter initialization. Noted that there are some differences between the architecture [14] with our work. In AGG, each domain has a independent classifier which is an one-layer fully connected layer $g_{\alpha_i}$, where $\alpha_i$ is the parameters of the classifier of domain $i$. The operation of a fully connected layer $g_{\alpha_i}(\mathbf{x})$ can be written in a matrix form given as $g_{\alpha_i}(\mathbf{x}) = \sigma(\mathbf{M}_i \mathbf{x} + \mathbf{b}_i)$ where $\mathbf{M}_i$ is the mapping matrix of fully connected layer with the shape of $(output_{dim}, input_{dim})$, $\mathbf{b}_i$ is the bias, which is a column vector of size output_{dim}, $\mathbf{x}$ is a column input vector, and $\sigma$ is the activation function (e.g. ReLU). To this end, we can adapt AGG in [14] into our formulation as

$$M = cat(\{M_i\}_{i=0}^{n}) \quad b = cat(\{b_i\}_{i=0}^{n})$$

(9)

Here $n$ is the number of training domains, $\text{cat}$ is the column-level concatenation function. Thus, the classifier weight can be formulated by $M$ and $b$. For evaluation, we use the feature extracted by feature extractor to train an SVM and KNN based on the training set of target domain data, as suggested in [14]. We report the classification accuracy and VD-Score [16] for comparison.

Regarding parameter tuning for SVM and KNN, we use a grid-search strategy. We split each target domain into the training part and valid part, then use the cross validation in training set. Finally, we select the parameter group which has the best performance in the training process and use it to validate the quality of the extracted feature.

For implementation, we train the network by using AMSGrad [27] in an end-to-end manner. We choose the size of minibatch as 64 for each domain. We use an initial learning rate of 0.0005 and a weight decay of 0.0001 for 20k iterations. The learning rate decrease by a factor of 10, 10 after 8k, 16k iterations. We use the mixup strategy during the training phase by choosing $\alpha = 0.4$ (which is the hyperparameter of Eq. to control the strength of mixup) in the first strategy and $\beta = 6$ in the second sampling strategy.

Results. We evaluate our proposed method by using the VD benchmark and compare against different state-of-the-art DG
baselines, including MetaReg(MR) [11], CrossGrad [10], and Reptile [12]. We also compare with Feature-Critic Networks(FC) [14] as another baseline, which was the first work considering HDG problem.

We first evaluate by considering SVM for classification purpose. The results are shown in Table 1. Im.N.PT in Table 1 denotes the result by extracting features with the ImageNet pre-trained resnet18 model. Based on the results, we find that our proposed method can significantly outperform all other baselines based on classification accuracy as well as VD Score [16], which reflects whether the method can achieve stable performance or not.

Then we evaluate our proposed method by considering KNN as classifier. The results are shown in Table 2. We find that we can also achieve much better performance compared with all other baseline methods.

Ablation Study. To evaluate the influence of parameter setting, we further conduct experiments with different hyperparameter in two sampling strategies proposed in 2.3. The results are shown in Fig. 2 and Fig.3 respectively.

![Fig. 2. The performance of MIXUP with different $\alpha$ which controls the strength of mixup.](image)

Based on Fig.2, we observe several interesting findings. First, when the hyperparameter $\alpha$ is close to zero, the model degenerates into the baseline model we proposed as $\lambda$ centers on either 0 or 1. Second, we find a trend in both accuracy and VD-score that the performance of model improves first, and then drops. We conjecture the reason why performance rises may profit from the enhanced robustness and drop due to underfitting.

![Fig. 3. The performance of MIX-ALL with different hyperparameter $\beta$ which controls the strength of mixup. In this figure, the dotted line represents the baseline in Sec. 2.2](image)

We also conduct experiments to study the impact of hyper-parameter $\beta$ which influence the strength of mixup in Eq.(8). The results are shown in Fig. 3. We can find that average accuracy has a drop when $\beta$ is equal to 0 or 1. Such result is consistent to our argument that when we choose a small $\beta$ value, the expectation of $\max(\lambda)$ and $\sigma(\lambda)^2$ tend to be small, which may lead to too much noise. When $\beta$ is very large, the model will degenerate into the baseline model as shown in Fig.1. We can find that the performance continues to drop when $n$ get larger.

4. CONCLUSION

We proposed a novel heterogeneous domain generalization method by considering data augmentation through domain mixup. In particular, we proposed two strategies, mixup between two domains and across multiple domains. By sampling the augmented training data from the heterogeneous mixup distribution, we got a more robustness feature extractor, which can be greatly helpful when used off-the-shelf. The experimental results demonstrate that our proposed methods can outperform all existing state-of-the-art baselines in terms of classification accuracy and VD-score.
5. REFERENCES

[1] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, “ImageNet large scale visual recognition challenge,” International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, Dec 2015.

[2] Lixin Duan, Ivor W. Tsang, Dong Xu, and Tat-Seng Chua, “Domain adaptation from multiple sources via auxiliary classifiers,” in Proceedings of the 26th Annual International Conference on Machine Learning, New York, NY, USA, 2009, ICML ’09, pp. 289–296, ACM.

[3] L. Duan, I. W. Tsang, and D. Xu, “Domain transfer multiple kernel learning,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 3, pp. 465–479, March 2012.

[4] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan, “Universal domain adaptation,” in CVPR, 2019.

[5] Haoliang Li, Sinno Jialin Pan, Renjie Wan, and Alex C Kot, “Heterogeneous transfer learning via deep matrix completion with adversarial kernel embedding,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2019.

[6] Zhangjie Cao, Kaichao You, Mingsheng Long, Jianmin Wang, and Qiang Yang, “Learning to transfer examples for partial domain adaptation,” in CVPR, June 2019.

[7] Pau Panareda Busto and Juergen Gall, “Open set domain adaptation,” in ICCV, 2017, pp. 754–763.

[8] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot, “Domain generalization with adversarial feature learning,” in CVPR, June 2018.

[9] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool, “Dlow: Domain flow for adaptation and generalization,” in CVPR, June 2019.

[10] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi, “Generalizing across domains via cross-gradient training,” arXiv preprint arXiv:1804.10745, 2018.

[11] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa, “Metareg: Towards domain generalization using meta-regularization,” in NIPS, 2018.

[12] Alex Nichol, Joshua Achiam, and John Schulman, “On first-order meta-learning algorithms,” arXiv preprint arXiv:1803.02999, 2018.

[13] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf, “Domain generalization via invariant feature representation,” in International Conference on Machine Learning, 2013, pp. 10–18.

[14] Yining Li, Yongxin Yang, Wei Zhou, and Timothy Hospedales, “Feature-critic networks for heterogeneous domain generalisation,” in The Thirty-sixth International Conference on Machine Learning, 2019.

[15] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz, “mixup: Beyond empirical risk minimization,” CoRR, vol. abs/1710.09412, 2017.

[16] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi, “Learning multiple visual domains with residual adapters,” in NIPS, 2017, pp. 506–516.

[17] Alex Krizhevsky, Geoffrey Hinton, et al., “Learning multiple layers of features from tiny images,” Tech. Rep., Citeseer, 2009.

[18] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi, “Fine-grained visual classification of aircraft,” arXiv preprint arXiv:1306.5151, 2013.

[19] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi, “Describing textures in the wild,” in CVPR, 2014, pp. 3606–3613.

[20] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel, “The german traffic sign recognition benchmark: A multi-class classification competition,” in IJCNN, 2011, vol. 6, p. 7.

[21] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng, “Reading digits in natural images with unsupervised feature learning,” 2011.

[22] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould, “Dynamic image networks for action recognition,” in CVPR, 2016, pp. 3034–3042.

[23] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi, “Fine-grained visual classification of aircraft,” arXiv preprint arXiv:1306.5151, 2013.

[24] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum, “Human-level concept learning through probabilistic program induction,” Science, vol. 350, no. 6266, pp. 1332–1338, 2015.

[25] Maria-Elena Nilsback and Andrew Zisserman, “Automated flower classification over a large number of classes,” in 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing. IEEE, 2008, pp. 722–729.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[27] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar, “On the convergence of adam and beyond,” arXiv preprint arXiv:1904.09237, 2019.