A Hierarchical-Tree-Based Method for Generative Zero-Shot Learning

Xizhao Wang, Zhongwu Xie, Weipeng Cao(✉), and Zhong Ming

College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
caoweipeng@szu.edu.cn

Abstract. It is currently a popular practice to use the class semantic information and the conditional generative adversarial network (CGAN) technique to generate visual features for the unseen classes in zero-shot learning (ZSL). However, there is currently no good ways to ensure that the generated visual features can always be beneficial to the prediction of the unseen classes. To alleviate this problem, we propose a hierarchical-tree-based method for constraining the generation process of CGAN, which can tune the generated visual features based on the multi-level class information. Moreover, to enhance the mapping ability of the model from the visual space to the semantic space, we add a multi-expert module to the traditional single mapping channel, which helps the model to mine the mapping relationship between the visual space and the semantic space. Extensive experimental results on five benchmark data sets show that our method can achieve better generalization ability than other existing generative ZSL algorithms.

Keywords: Zero-shot learning · Hierarchical tree · Generative adversarial networks

1 Introduction

In real-world applications, especially in medical image recognition and wild animal recognition scenarios, sometimes there are no training samples for some classes due to the difficulty of obtaining data or the high cost of labeling them [13]. We call these classes unseen classes, and the corresponding is seen classes, which refer to the classes that have corresponding labeled samples and can be directly used for model training. To enable the model to accurately predict unseen classes, zero-shot learning (ZSL) [22,31] was proposed, which aims to recognize these classes without any training samples with the help of their side information (i.e., the description information of the classes). Generally, the

This work was supported by National Natural Science Foundation of China (61836005, 61976141, 61732011), and the Opening Project of Shanghai Trusted Industrial Control Platform (TICPSH202003008-ZC).

X. Wang and Z. Xie—Joint first authors.

M. Qiu (Ed.): ICA3PP 2020, LNCS 12453, pp. 352–364, 2020.
https://doi.org/10.1007/978-3-030-60239-0_24

side information corresponding to these classes is encoded into semantic vectors of the same dimension, and related techniques include: attribute [7], gaze [14], word2vec [28], etc.

According to the different testing settings, the existing ZSL algorithms can be divided into two categories: conventional zero-shot learning (CZSL) and generalized zero-shot learning (GZSL). For CZSL, testing samples only contain the unseen classes samples; for GZSL, the testing samples include both the seen and unseen classes samples. It is clear that the testing setting of GZSL is closer to the real world than that of CZSL [30]. At present, there are two main strategies to realize ZSL: embedding strategy and generative strategy. The embedding strategy follows the assumption that the distributions of the seen classes and the unseen classes in both the visual space and the semantic space are similar [13], and the main implementation is to transfer the mapping relationship learned from the seen classes to the recognition of the unseen classes. The advantage of this strategy is that it is easy to implement, but its disadvantage is that it may cause the model to suffer from the hubness problem [24] and bias problem [33].

Inspired by generative adversarial networks (GAN) [10], many GAN-based generative ZSL algorithms have been proposed in recent years [8,17,31]. In general, these algorithms aim to use random noises and the semantic vectors to generate pseudo samples and then transform the ZSL task into the general supervised learning task. This generative strategy is straightforward but suffers from the following weakness: the generators of these algorithms are easy to generate unrepresentative visual features, especially for the unseen classes. To alleviate this problem, in [17], the authors proposed to use a regressor (the same to the module R in Fig. 1), which can enhance the correlation between the generated features and the known semantic vectors to a certain extent. But it can not guarantee the quality of these features [9], especially for the unseen classes, because the information used to constrain the generator is still too scarce.

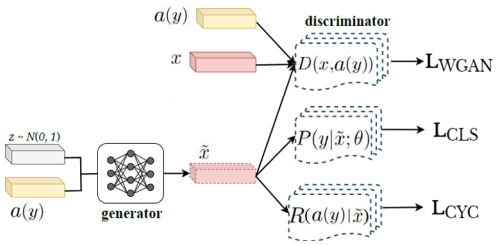


Fig. 1. The framework of our method

To solve the above problems, we propose a novel constraint method based on the class hierarchy tree to guide the generator to generate more useful visual features for the current ZSL task. Specifically, we first use the K-means technique to build a class hierarchy tree based on the semantic vectors corresponding to the classes and then use the multi-level class information to evaluate the quality of

the generated features. In other words, the generated features can be evaluated from multiple dimensions and the corresponding evaluation loss can be used to guide the update of the generator. Moreover, to enhance the mapping ability of the model from the visual space to the semantic space, we add a multi-expert module, which can embed a variety of different structures or algorithms.

The main contributions of this paper can be summarized as follows: (1) We propose to use a class hierarchy tree to constrain the generator of the generative ZSL algorithms for the first time, which can make it generate more representative visual features for model training; (2) We propose to add a multi-expert module, which is beneficial to mine the relationship between the visual space and the semantic space; (3) Extensive experiments on five benchmarks verify that our method outperforms the previous state-of-the-art approaches.

2 Preliminaries

2.1 Generative Adversarial Networks (GAN)

GAN [10] can be used to generate images (or text) from random noise, and its architecture usually contains two modules: generator and discriminator. The input of the generator is random noise z that follows a specific distribution (e.g., $z \sim N(0,1)$). During training, the generator and the discriminator are trained in an adversarial manner, that is, the generator tries its best to generate fake samples to fool the discriminator, while the discriminator does its best to distinguish samples as whether they come from the generator or the original training data set. When the discriminator cannot distinguish the real samples from the generated samples, the model training is completed. At this point, we think that the generator has learned the distribution of the original samples.

In recent years, GAN has received extensive attention, and some notable work includes: in [19], the authors proposed that the label information can be added to the input of the generator and the discriminator, so that the model can better capture the data distribution under the specific class constraints. Different from [19,26] and [25] use sentence descriptions as the auxiliary constraints. In [3], the authors proposed to use the Wasserstein distance to alleviate the problem of unstable model training and model collapse in GAN. In [11], the authors proposed an improved WGAN, which can accelerate its training speed. At present, it has become a popular practice to integrate Wasserstein distance into conditional GAN (CGAN), and various CWGAN algorithms and applications have been proposed [6,32].

2.2 Generative Methods for Zero-Shot Learning

In recent years, the generative strategy has been widely concerned in the field of ZSL, because once one can generate training samples for unseen classes in some way, then the ZSL tasks become the general supervised learning tasks. Related representative work includes: in [20], the authors used the conditional

variable auto-encoder to generate the unseen classes samples. Using CGAN to generate training samples for the unseen classes is currently a hot research topic. In this learning paradigm, the input of the generator is usually the semantic vectors corresponding to the unseen classes and the random noise following a specific distribution. Here the semantic vectors play the role of constraining the output generated by CGAN. In [31], the authors pointed out that it is easier to generate low-dimension visual features through the generator than directly generating pixel-level images, and their experimental results showed that using the generated features can also improve the generalization ability of the model. Similar work also includes [8,17], etc. Note that the generator in our method also generates visual features rather than the pixel-level images.

2.3 Constraints are Necessary

In [31], the authors demonstrated that it is difficult for CWGAN to ensure that the generated features are beneficial to the final model decision. To optimize this problem, they proposed f-CLSWGAN, which uses a classifier trained based on the seen classes samples to evaluate the quality of the generated features and uses the corresponding evaluation loss to constrain the update of the generator. Their experimental results show that even if only additional constraints are imposed on the seen classes, this method can still effectively improve the generalization ability of the final model. In [8], the authors pointed out that if we do not constrain the generated features for the unseen classes during the update of the generator, it will cause the distribution of the generated unseen classes features to be far from their true distribution. To solve this problem, they proposed Cycle-CLSWGAN, which can alleviate the above problem by reconstructing the generated features back to their corresponding semantic vectors. Similar work also includes AFC-GAN [17]. The difference between AFC-GAN and Cycle-CLSWGAN is that the regressor used in the former is pre-trained, and its internal parameters will not be updated throughout the training process, while the latter will update these parameters.

3 The Details of Our Method

3.1 Notations

In this study, $\mathcal{X}$ represents the visual space, $\mathcal{A}$ represents the visual space, and $\mathcal{Y}$ represents the label space. In ZSL, generally the original data set ($\mathcal{D}$) will be divided into the seen classes set $\mathcal{D}_S$ and the unseen classes set $\mathcal{D}_U$, where $\mathcal{D}_S = \{x, y, a(y) | x \in \mathcal{X}_S, y \in \mathcal{Y}_S, a(y) \in \mathcal{A}_S\}$, $\mathcal{D}_U = \{x, y, a(y) | x \in \mathcal{X}_U, y \in \mathcal{Y}_U, a(y) \in \mathcal{A}_U\}$, and $\mathcal{Y} = \mathcal{Y}_S \cup \mathcal{Y}_U$, $\mathcal{Y}_S \cap \mathcal{Y}_U = \emptyset$.

In the experiment, we used the testing setup of GZSL, that is, the testing samples can come from both the unseen and seen classes. Therefore, $\mathcal{D}_S$ will also be divided into two subsets: the subset for training $\mathcal{D}_S^{Tr}$ and the subset for testing $\mathcal{D}_S^{Te}$. So the original data set was finally divided into three parts: $\mathcal{D}_S^{Tr}$, $\mathcal{D}_S^{Te}$, and $\mathcal{D}_U^{Te}$.

3.2 The Basic Idea of Our Method

The basic idea of our method is to optimize the constraints of CWGAN to make the generated features more conducive to the final model decision. The difficulty here is how to evaluate the quality of the generated features during the training process, and then tune the generator based on the evaluation feedback. In [17], the authors pointed out that the generated features should be able to be reconstructed into their corresponding semantic vectors. However, the method they proposed mainly helped the prediction of the seen classes [9], but the constraints on the generated unseen classes features are insufficient.

Inspired by their work, we propose a hierarchical-tree-based constraint method to provide more information about the unseen classes for the generator in CWGAN. Specifically, first, we use the K-means technique to cluster the semantic vectors corresponding to the seen and unseen classes to form a class hierarchy tree (as shown in Fig. 2). Note that the higher-level ancestor nodes are the high-level abstraction of the concepts of the child nodes.

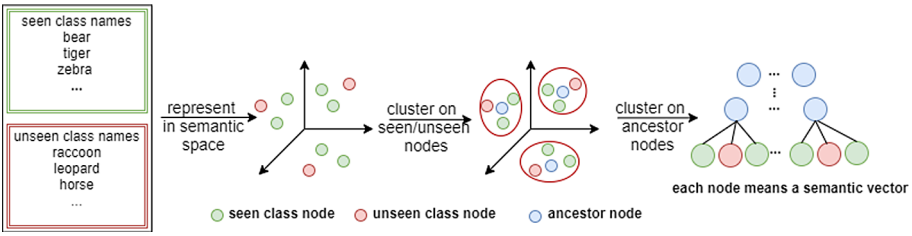


Fig. 2. Tree-building process.

During training, the model first maps the generated features to the semantic space and then uses the above class hierarchy tree to evaluate their quality and obtain the corresponding loss. This loss will provide the generator with additional constraint information. The class hierarchy tree can evaluate whether the generated features conform to the semantic features of the classes from multiple dimensions, thereby guiding the generator to generate better visual features. Moreover, the multi-level tree structure also gives us the opportunity to use multi-expert modules (i.e., the mapping channels) to enhance the feature extraction ability of the model (as shown in Fig. 3), which helps the model to better mine the relationship between the visual space and the semantic space.

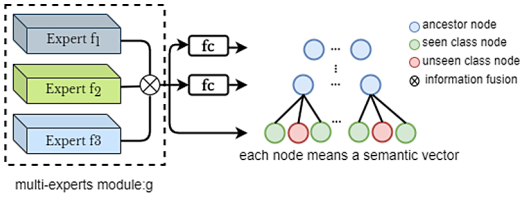


Fig. 3. The architecture of the regressor used in our method.

3.3 Details of Our Method

The basic framework used in our method is CWGAN, where the input of the generator is the random noise (i.e., z, $z \sim N(0,1)$) and the semantic vector (i.e., a) corresponding to each specific class, and its output is the generated visual features. The loss of the generator can be obtained as follows:

$$\mathcal{L}_G = -E[D(G(z,a),a)] - \eta E[logP(y|G(z,a)) - \beta E[t|R(G(z,a)]] \tag{1}$$

where the first term is the Wasserstein loss [3], the second item is the classification loss for evaluating the quality of the generated features, and the last item is the reconstruction loss. η and β are the trade-off parameters. t is the identifier corresponding to the hierarchical tree.

The loss of the discriminator can be calculated as follows:

$$\mathcal{L}_D = E[D(G(z,a),a)] - E[D(x,a)] + \lambda E[(\|\nabla_{\hat{x}} D(\hat{x},a)\|_2 - 1)]^2 \tag{2}$$

where the last term refers to the penalty loss, which is also known as the Lipschitz constraint [11], in which $\hat{x} = \mu x + (1-\mu)\tilde{x}, \mu \sim U(0,1)$. Here λ is a penalty parameter, which was set to 10 in our study.

The global optimization objective of our model is:

$$\begin{aligned} \min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}_{GAN} = E[D(x,a)] - E[D(G(z,a),a)] \\ -\lambda E[(\|\nabla_{\hat{x}} D(\hat{x},a)\|_2 - 1)]^2 - \eta E[logP(y|G(z,a)) - \beta E[t|R(G(z,a))]] \end{aligned} \tag{3}$$

After training, one can use the generator to augment the original training data set of the ZSL task, thereby improving the generalization ability of the model. It is worth mentioning that to improve the prediction accuracy of the ZSL model, we adopted the ensemble learning framework designed in [17] to make predictions. Specifically, we trained classifiers f_v and f_s based on the training data provided by visual features and labels, and the training data provided by semantic vectors and labels, respectively. The final prediction results of the model are determined by these two classifiers:

$$y = argmax\,[f_v(x) + \hat{\lambda} f_s((g(x))] \tag{4}$$

where $\hat{\lambda}$ is the trade-off parameter.

The learning framework of our method is shown in Fig. 1, which contains a generator, a discriminator (D), a classifier (P), and a regressor (R). The training process of the regressor can be summarized as Algorithm 1. The training methods of other modules are the same as traditional methods.

4 Experiments

4.1 Datasets

In the experiment, we selected five benchmark data sets to evaluate the performance of our method, including Animals with Attributes1 (**AWA1**) [16], Animals with Attributes2 (**AWA2**) [30], Caltech-UCSD Bird-200-2011 (**CUB**) [29],

Oxford Flowers (**FLO**) [21], and SUN Scene Recognition (**SUN**) [23]. The details of the data sets are shown in Table 1.

Algorithm 1. The training process of the regressor

Input: Training data $\mathcal{D}$=(X_S^{Tr}, A_S, A_U); the number of nodes at each level in the class hierarchy tree, which is represented by array arr, and $arr[0]$ represents the number of leaf nodes; the learning rate α; and the trade-off parameters β_0, β_1, ..., β_n.
Output: The parameters (i.e., θ_r) of the regressor.
Step1: K-means clustering
if $len(arr) == 1$:
    return A;
$temp\_att = A$;
while $i < len(arr) - 1$ do:
    $node = arr[i]$;
    kmean = Kmeans(n_clusters=$node$).fit($temp\_att$);
    $cluster\_center$ = kmean.cluster_center_;
    save $cluster\_center$;
    $temp\_att = cluster\_center$;
Step2: Build the framework of the regressor R according to the architecture of each expert module.
Step3: Let T_{ij} represent the a_j of the i-th layer.
while not done do:
    $pred_0, pred_1, ..., pred_n = R(x|x \in jth\ class)$;
    $\mathcal{L} = \beta_i \sum_{i=0}^{n} \|pred_i - T_{ij}\|^2$;
    $\theta_r = \theta_r - \alpha \nabla_{\theta_r}(L)$;
return θ_r;

Table 1. The details of the experimental data sets

Dataset	AWA1	AWA2	CUB	FLO	SUN
Images	30475	37322	11788	8189	14340
Attributes	85	85	312	1024	102
Seen classes	40	40	150	82	645
Unseen classes	10	10	50	20	72

To make a fair comparison with other ZSL algorithms, we used the same division method as [30] for the AWA1, AWA2, CUB, and SUN data sets; for FLO, we used the division method mentioned in [26]. A pre-trained ResNet-101 [12] model was chosen as the feature extractor for all the ZSL algorithms.

4.2 Implementation Details

As shown in Fig. 1, our learning framework mainly includes three functional modules: generator, discriminator, and regressor. In our experiments, for the generator and the discriminator, we used the multilayer perceptron (MLP) with a single hidden layer, in which the number of nodes in the hidden layer is 4096 and the activation function is LeakyReLU [18]. The output layer of the generator contains 2048 nodes and uses ReLU as its activation function; while the output layer of the discriminator has one output node but does not use any activation function. For the regressor, we used the neural network with a single hidden layer as the basic expert module (three expert modules in total), the activation function of the hidden layer is LeakyReLU, and the activation function of the output layer is ReLU. For AWA1, AWA2, and CUB, the number of hidden nodes corresponding to these three expert modules are 2048, 3072, and 4096, respectively; for SUN, they are 7168, 8189, and 9216, respectively; for FLO, they are 3072, 4096, and 5120, respectively. In this study, Adam [15] was chosen as the optimization algorithm and the learning rate was set to $1e-4$. In addition, the level of the class hierarchy tree was set to 3.

In Eq. (1), the last term can be expanded as follows:

$$\beta_i E[t|R(G(z,a)) = \beta_{seen} \left\| pred_0 - a^s \right\|^2 + \beta_{unseen} \sum_{i=0}^{2} \left\| pred_i - T_{ij} \right\|^2, \quad (5)$$

where β_{seen} is a constant, β_{unseen} is a vector, and $\mathcal{T}_{ij}$ refers to the ancestor of a_j when $\mathrm{i} > 0$ and refers to the a_j itself when $\mathrm{i} = 0$. The first term is used for constraining the generated seen classes features to reconstruct to their corresponding semantic vectors, while the last term is used for constraining the generated unseen classes features to reconstruct to their ancestors and their corresponding semantic vectors. The values of these parameters in our experiments are shown in Table 2.

Moreover, the value of the $\hat{\lambda}$ in Eq. (4) was set to 2 in SUN, and was set to 1.5 in other data sets.

Table 2. The parameter settings of the regressor.

Dataset	**AWA1**	**AWA2**	**CUB**	**FLO**	**SUN**
Node number in HT	[50, 20, 5]	[50, 20, 5]	[200, 20, 5]	[102, 50, 10]	[717, 300, 50]
β_{seen}	1,	1,	1,	1,	10,
β_{unseen}	[0.1, 1, 1]	[0.1, 1, 1]	[0.1, 1, 1]	[1,1,1]	[0.1, 5, 5]

Note: HT means the class hierarchy tree. $\beta_{unseen} = [\beta_0, \beta_1, \beta_2]$ correspond to the weight of loss evaluation from bottom to top of the class hierarchy tree.

4.3 Experimental Results

In this study, we used the harmonic mean $\mathcal{H}$ to evaluate the performance of the proposed algorithm, which can be obtained as follows:

$$\mathcal{H} = \frac{2 \times A_s \times A_u}{A_s + A_u} \tag{6}$$

where A_s and A_u are the top-1 accuracy of the model on the seen classes and the unseen classes, respectively. Note that the larger the value of $\mathcal{H}$, the better the generalization ability of the model.

We compared our method with the other seven popular ZSL algorithms (i.e., SJE [2], ESZSL [27], ALE [1], DEVISE [9], GAZSL [33], f-CLSWGAN [31], and AFC-GAN [17]), and the corresponding results are shown in Table 3.

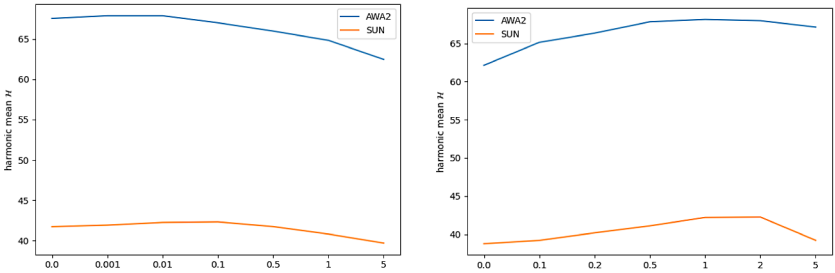
Table 3. Experimental results of our method and other ZSL algorithms on benchmarks

Method	AWA1			AWA2			CUB			FLO			SUN		
	As	*Au*	$\mathcal{H}$	*As*	*Au*	$\mathcal{H}$	*As*	*Au*	$\mathcal{H}$	*As*	*Au*	$\mathcal{H}$	*As*	*Au*	$\mathcal{H}$
SJE [2]	74.6	11.3	19.6	73.9	8.0	14.4	59.2	23.5	33.6	47.6	13.9	21.5	47.6	13.9	21.5
ESZSL [27]	75.6	6.6	12.1	77.8	5.9	11.0	68.3	12.6	21.0	56.8	11.4	19.0	11.0	27.9	15.8
ALE [1]	16.8	76.1	27.5	81.8	14.0	23.9	62.8	23.7	34.4	61.6	13.3	21.6	33.1	21.8	26.3
DEVISE [9]	68.7	13.4	22.4	74.7	17.1	27.8	53.0	23.8	32.8	44.2	9.9	16.2	27.4	16.9	20.9
GAZSL [33]	84.2	29.6	43.8	86.9	35.4	50.3	61.3	31.7	41.8	77.4	28.1	41.2	39.3	22.1	28.3
f-CLSWGAN [31]	61.4	57.9	59.6	68.9	52.1	59.4	57.7	43.7	49.7	73.8	59.0	65.6	42.6	36.6	39.4
AFC-GAN [17]	66.8	58.2	62.2	-	-	-	59.7	53.5	56.4	80.0	60.2	68.7	36.1	49.1	41.6
ours	72.4	61.5	66.5	77.3	60.9	68.1	59.0	54.7	56.8	78.6	65.0	71.2	38.0	47.9	42.4

From Table 3, one can observe that our model can achieve the highest $\mathcal{H}$ on all the benchmark data sets, which implies that our method has better generalization ability than others. Moreover, our model has higher prediction accuracy for the unseen classes (i.e., A_u) than other algorithms on most of the data sets (4/5, i.e., AWA1, AWA2, CUB, and FLO), which indirectly reflects that our strategy does improve the quality of the generated unseen classes features.

4.4 Parameter Sensitivity Analysis

To explore the impact of the three hyper-parameters (η, $\hat{\lambda}$, and β_{unseen}) in our method on model performance, here we use the one-variable-at-a-time method to analyze the sensitivity of the model to them. As mentioned in Section III, η is used to weigh the loss generated by the classifier, and its impact on the performance of the model on AWA2 and SUN is shown in Fig. 4(a). It can be observed from Fig. 4(a) that the value of η should not be set too large. Generally, as long as its value is fixed at 0.01, the model can achieve good performance. $\hat{\lambda}$ is used to weigh the prediction results of the two classifiers in Eq. (4), and

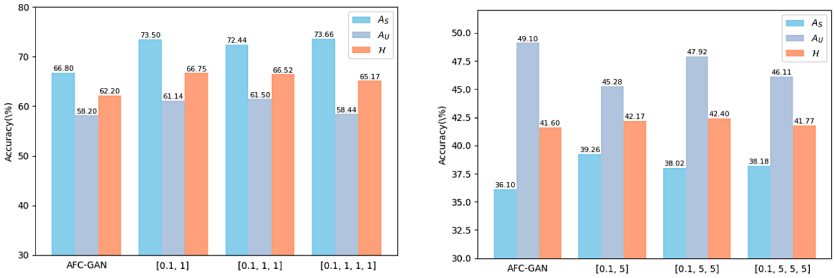


(a) weight of classification loss (η) (b) weight of classifier ($\hat{\lambda}$)

Fig. 4. Parameter sensitivity

its impact on model performance is shown in Fig. 4(b). It can be observed from Fig. 4(b) that setting the value of $\hat{\lambda}$ to 1.5 is a good choice.

β_{unseen} is an important parameter in building the class hierarchy tree, and its influence on the generalization ability of the final model is shown in Fig. 5(a)–Fig. 5(b). The elements in β_{unseen} represent the weights of the reconstruction loss in building the tree from the leaf nodes to the root node, and the length of β_{unseen} is the height of the tree. From Fig. 5(a)–Fig. 5(b), one can observe that when the height of the tree is 2 or 3, the performance of the model can be significantly improved, which again verifies the effectiveness of our method.



(a) β_{unseen} on AWA1 (b) β_{unseen} on SUN

Fig. 5. The recognition results of the model with different β_{unseen} on AWA1 and SUN

5 Conclusions

To improve the generalization ability of the generative ZSL algorithms, we designed a hierarchical-tree-based method to enhance the quality of the generated features in this paper. Specifically, our method can evaluate the quality of the generated visual features from multiple levels and tune the generator through the corresponding loss feedback. Moreover, to enhance the mapping

ability of the model from the visual space to the semantic space, we added the multi-expert module to the traditional single mapping channel to realize the multi-level extraction and transformation of visual features, which is conducive to the model for mining the relationship between the visual features and the semantic features. We evaluated the effectiveness of the proposed algorithm on five benchmark data sets and the experimental results show that our model can achieve higher prediction accuracy than the other seven popular ZSL algorithms. In the future, we will consider incorporating non-iterative algorithms [4,5] to our method to improve the training efficiency of its classifier.

References

1. Akata, Z., Perronnin, F., Harchaoui, Z., Schmid, C.: Label-embedding for attribute-based classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 819–826 (2013)
2. Akata, Z., Reed, S., Walter, D., Lee, H., Schiele, B.: Evaluation of output embeddings for fine-grained image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2927–2936 (2015)
3. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 70, pp. 214–223 (2017)
4. Cao, W., Gao, J., Ming, Z., Cai, S., Shan, Z.: Fuzziness-based online sequential extreme learning machine for classification problems. Soft Comput. **22**(11), 3487–3494 (2018). https://doi.org/10.1007/s00500-018-3021-4
5. Cao, W., Wang, X., Ming, Z., Gao, J.: A review on neural networks with random weights. Neurocomputing **275**, 278–287 (2018)
6. Ebenezer, J.P., Das, B., Mukhopadhyay, S.: Single image haze removal using conditional wasserstein generative adversarial networks. In: 2019 27th European Signal Processing Conference, pp. 1–5. IEEE (2019)
7. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1778–1785. IEEE (2009)
8. Felix, R., Kumar, V.B., Reid, I., Carneiro, G.: Multi-modal cycle-consistent generalized zero-shot learning. In: Proceedings of the European Conference on Computer Vision, pp. 21–37 (2018)
9. Frome, A., et al.: DeViSe: a deep visual-semantic embedding model. In: Advances in Neural Information Processing Systems, pp. 2121–2129 (2013)
10. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
11. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein GANs. In: Advances in Neural Information Processing Systems, pp. 5767–5777 (2017)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

13. Jiang, H., Wang, R., Shan, S., Chen, X.: Transferable contrastive network for generalized zero-shot learning. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 9765–9774 (2019)
14. Karessli, N., Akata, Z., Schiele, B., Bulling, A.: Gaze embeddings for zero-shot image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4525–4534 (2017)
15. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 951–958. IEEE (2009)
17. Li, J., Jing, M., Lu, K., Zhu, L., Yang, Y., Huang, Z.: Alleviating feature confusion for generative zero-shot learning. In: Proceedings of the 27th ACM International Conference on Multimedia, pp. 1587–1595 (2019)
18. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of ICML, vol. 30, p. 3 (2013)
19. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
20. Mishra, A., Krishna Reddy, S., Mittal, A., Murthy, H.A.: A generative model for zero shot learning using conditional variational autoencoders. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 2188–2196 (2018)
21. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics and Image Processing, pp. 722–729. IEEE (2008)
22. Palatucci, M., Pomerleau, D., Hinton, G.E., Mitchell, T.M.: Zero-shot learning with semantic output codes. In: Advances in Neural Information Processing Systems, pp. 1410–1418 (2009)
23. Patterson, G., Xu, C., Su, H., Hays, J.: The sun attribute database: beyond categories for deeper scene understanding. Int. J. Comput. Vis. **108**(1–2), 59–81 (2014)
24. Radovanović, M., Nanopoulos, A., Ivanović, M.: Hubs in space: popular nearest neighbors in high-dimensional data. J. Mach. Learn. Res. **11**(Sep), 2487–2531 (2010)
25. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. arXiv preprint arXiv:1605.05396 (2016)
26. Reed, S.E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., Lee, H.: Learning what and where to draw. In: Advances in Neural Information Processing Systems, pp. 217–225 (2016)
27. Romera-Paredes, B., Torr, P.: An embarrassingly simple approach to zero-shot learning. In: International Conference on Machine Learning, pp. 2152–2161 (2015)
28. Socher, R., Ganjoo, M., Manning, C.D., Ng, A.: Zero-shot learning through cross-modal transfer. In: Advances in Neural Information Processing Systems, pp. 935–943 (2013)
29. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
30. Xian, Y., Lampert, C.H., Schiele, B., Akata, Z.: Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. IEEE Trans. Pattern Anal. Mach. Intell. **41**(9), 2251–2265 (2018)
31. Xian, Y., Lorenz, T., Schiele, B., Akata, Z.: Feature generating networks for zero-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5542–5551 (2018)

32. Zheng, M., et al.: Conditional wasserstein generative adversarial network-gradient penalty-based approach to alleviating imbalanced data classification. Inf. Sci. **512**, 1009–1023 (2020)
33. Zhu, Y., Elhoseiny, M., Liu, B., Peng, X., Elgammal, A.: A generative adversarial approach for zero-shot learning from noisy texts. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1004–1013 (2018)