GT-GAN: A General Transductive Zero-Shot Learning Method Based on GAN

JUNHAO DONG, BO XIAO, BO DING, AND HAOYU WANG
School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
Corresponding author: Bo Xiao (xiaobo@bupt.edu.cn)
This work was supported by the Research Innovation Fund for College Students of the Beijing University of Posts and Telecommunications.

ABSTRACT Most zero-shot learning (ZSL) methods based on generative adversarial networks (GANs) utilize random noise and semantic descriptions to synthesize image features of unseen classes, which alleviates the problem of training data imbalance between seen and unseen classes. However, these methods usually only learn the distributions of seen classes in the training stage, ignoring the unseen ones. Due to the different distributions of seen and unseen samples, i.e., image features, these methods cannot generate unseen features of sufficient quality, so the performances are also limited, especially for the generalized zero-shot learning (GZSL) setting. In this article, we propose a general transductive method based on GANs, called GT-GAN, which can improve the quality of generated unseen image features and therefore benefit the classification. A new loss function is introduced to make the relative positions between each unseen image and its $k$ nearest neighbors in the feature space as consistent as possible with their relative positions in the semantic space; this loss function may be easily applied in most existing GAN-based models. Experimental results on five benchmark datasets show a significant improvement in accuracy compared with that of original models, especially in the GZSL setting.

INDEX TERMS Image classification, generative adversarial network (GAN), transductive zero-shot learning, generalized zero-shot learning.

I. INTRODUCTION
Supervised learning methods have achieved great success in the domain of machine learning. They are widely applied in various fields, e.g., image classification, and the performance is no worse than that of humans. However, supervised learning requires large amounts of labeled data for training, and the learned classifier can only recognize samples in classes that appear in the training stage, which is not suitable for certain specific real-world applications. Among the reasons for this deficiency are an insufficient number of training samples for some classes, high labeling costs for some samples, and changes in the target classes over time. To solve these problems, zero-shot learning [1]–[4] (ZSL, trained with seen samples and tested by unseen ones) has been proposed.

Given that the key point of ZSL is to address the lack of labeled unseen samples and that the supervised classification models based on labeled data are relatively advanced, a natural idea is to generate corresponding data so that a conventional ZSL can be converted to a supervised classification problem. Fortunately, with the development of generative adversarial networks (GANs) [5], researchers have proposed novel approaches that can use noise vectors and semantic information to generate fake samples.

Fig. 1 shows the main idea of conventional ZSL methods [6]–[8] with GANs. First, the GANs are trained. A GAN [5] is a framework used for capturing the distributions of real seen samples (image features) to generate fake samples from the same distributions. It consists of two distinct subnetworks, a generator network ($G$) and a discriminator network ($D$). The job of $G$ is to leverage random noises and semantic representations of seen classes to generate fake seen samples that resemble the real ones, while the job of $D$ is to determine whether the inputs are real or fake. These two models are trained in an adversarial way to synthesize better features. In addition, these methods also use other losses or regularizations to achieve better training effects, which are not illustrated in the figure. Second, the trained $G$ is used to generate fake unseen samples with noise and unseen-class semantic descriptions and train a classifier. Third, in the testing stage, the trained classifier is leveraged to classify the real unseen samples and output the classification results.
However, these methods have a common problem, i.e., their GANs can only learn the distributions of seen samples in the training stage, ignoring the unseen ones. Due to the different distributions of seen and unseen samples, the generator cannot synthesize unseen features of sufficient quality to train a discriminative classifier. Thus, the classification performance is also limited. Specifically, for the generalized zero-shot learning (GZSL, trained with seen samples and tested by both seen and unseen ones) setting, because the test samples come from both seen and unseen categories, the classifier may bias the predictions toward the seen classes, which may lead to a more significant drop in accuracy. To alleviate these problems, we would ideally implement a GAN that can also learn some information on the unseen classes at the training stage to generate better unseen samples that are more similar to the real ones without considering their real categories.

In this work, we propose a novel transductive GAN-based method called GT-GAN, which can learn from unlabeled unseen features, to alleviate the inequality between seen and unseen classes in the training phase. First, in addition to the conventional training dataset consisting of seen categories, we also introduce unlabeled unseen image features and use a softmax classifier, which is trained with seen features, to classify them. We regard the outputs, which are rendered as probability distributions, as the similarities between these features and the seen classes. Then, we compute weighted semantic descriptions for each unseen feature with similarities to and the semantic information of the seen classes and select the $k$-closest unseen semantic descriptions to generate fake features. We introduce the average samples of these fake features and regularize them such that each one is close to the original feature differently, according to the distances of the semantic descriptions, e.g., if one unseen semantic vector is closer to a weighted one, the features generated with it should be closer to the original feature. A new loss function is introduced to achieve this goal and make the relative positions between each unseen image and its $k$-nearest neighbors in the feature space as consistent as possible with their relative positions in the semantic space. In addition, to the best of our knowledge, most GAN-based methods do not use unlabeled unseen samples for training or only implement their transductive versions for their own models. Our proposed method can be used in various existing GAN-based models to achieve better classification results than the original methods. We summarize our contributions as follows.

(1) We propose a general transductive ZSL method based on GANs, called GT-GAN, which introduces unlabeled unseen samples for training, to improve the performance of the generator.

(2) A new contrastive loss function is proposed to make the relative positions between each unseen image and its $k$-nearest neighbors in the feature space as consistent as possible with their relative positions in the semantic space. This loss function can also help synthesize unseen image features closer to the real features. It can also alleviate the effect of the misclassification of each unseen feature because we select $k$ unseen-class semantic descriptions instead of one.
for each unseen feature for the first classification to generate fake features. Furthermore, this function can be easily applied to various existing GAN-based ZSL methods.

(3) Experimental results on five benchmark datasets prove that the existing models improved by our method can achieve significantly higher accuracy than the original ones, especially in the GZSL setting.

II. RELATED WORKS

A. GENERATIVE ADVERSARIAL NETWORK

A generative adversarial network (GAN) [5] is a framework designed for capturing the distributions of inputs, such as images, to generate new data from the same distribution. However, it suffers from some problems, such as unstable training and mode collapse. Hence, some improved models have been proposed to mitigate these problems. WGAN [9] introduces the Wasserstein distance instead of the Jensen-Shannon divergence to measure the deviation between synthetic and real data distributions. LSGAN [10] replaces the cross-entropy loss with a least-squares loss to tackle the vanishing gradient problem at the training stage. In most recent studies, the proposed models can generate higher-quality samples than the previous methods as well as solve these problems. Saliencygan [11] demonstrates a better ability to capture the data distributions without mode collapse. TPSDiyc [12] achieves better alignment between the source and target data while maintaining superior image quality. DAGAN [13] can stabilize the generator and provide superior generation with preserved perceptual image details. MS-GAN [14] achieves more stabilized and efficient training and improves the perceptual quality of the superresolved results. These advanced methods may become potential targets of our proposed method.

B. OVERVIEW OF ZERO-SHOT LEARNING METHODS

Zero-shot learning (ZSL) was proposed to recognize unseen samples, which are disjointed from training samples. ZSL methods are usually categorized into two categories: classifier-based methods and instance-based methods [15]. The former aims to directly learn a classifier for unseen classes in distinct ways, e.g., in [16]–[18], the methods learn a correspondence function between feature space and semantic space, while the methods in [19], [20] leverage relationships among classes. The latter focuses on obtaining labeled samples of unseen classes. Recently, assisted by data generation models, such as GANs [5] and variational autoencoders (VAEs) [6], [21], many new synthesizing methods have been proposed to directly generate unseen samples from their semantic attributes, which can transform zero-shot learning to a conventional supervised learning problem.

C. GAN-BASED ZSL METHODS

Leveraging GAN to generate features can convert the ZSL problem to a conventional classification problem. Because GAN suffers from unstable training problems and mode collapse, most recent models adopt improved GANs, e.g., WGAN [9], to undergo a stable training process and avoid mode collapse. f-CLSWGAN [7] initially uses the conditional WGAN to directly generate image features to train a classifier and introduces a classification loss to make the generated features more discriminative. GAZSL [8] utilizes ACGAN added with a visual pivot regularizer (VPG) to ensure that generated samples of each class are close to the average of the corresponding real features. LisGAN [22] introduces soul samples (average samples) of each class, which can represent the most semantically meaningful aspects of each image feature in the same class, to improve the generated features of the conditional WGAN. In our study, we mainly describe our method as it is applied to models based on the conditional WGAN because it is more widely used in existing works. Certainly, the idea of our method can also be used in other GAN-based models.

However, because GAN-based ZSL methods have to train the generator on seen features and apply it to unseen ones, an inevitable issue, named feature confusion, has been highlighted recently. Feature confusion refers to the fact that the synthesized unseen features are prone to seen references and are incapable of reflecting the real aspects of the unseen instances. AFC-GAN [23] leverages a boundary loss that estimates and then maximizes the decision boundary between the seen and unseen features to handle the issue. It also introduces a multimodal cycle-consistent loss to promote diversity and preserve the semantic consistency of the synthesized features, which can also alleviate feature confusion. In our work, we directly introduce unlabeled unseen features to make the generator learn from them at the training stage and then synthesize better unseen samples that are closer to the real samples.

In addition, the different distributions of seen and unseen features lead to a typical domain adaptation problem, which can also influence the effects of the models. [24] and [25] recently addressed this issue. In [24], a novel ATM method is proposed to minimize the interdomain divergence, maximize the intraclass density and address the equilibrium challenge issue in adversarial domain adaptation, while in [25], a novel approach is proposed to jointly exploit feature adaptation with distribution matching and sample adaptation with landmark selection. These studies may also be leveraged in our method to improve the classification results.

III. PROPOSED METHOD

A. DEFINITIONS AND NOTATIONS

We first define $S = \{(x, y, t)|x \in X_s, y \in Y_s, t \in T_s\}$ as the training samples of the seen classes, where $x \in R^d$ represents the image features, $t \in R^m$ stands for the semantic attributes, and $y$ is the class label of each sample. In addition, we also have $X_u$ and $U = \{(y, t)|y \in Y_u, t \in T_u\}$ as auxiliary training data of the unseen classes for a transductive setting. $X_u$ denotes a set of unlabeled visual features, while $U$ consists of semantic vectors and the corresponding labels.
Specifically, we have $Y_s \cap Y_u = \emptyset$. With these training examples, the task of ZSL is to learn the function $f_{ZSL}: X_u \rightarrow Y_u$, while GZSL is tasked to learn the function $f_{GZSL}: \{X_s, X_u\} \rightarrow Y_s \cup Y_u$.

**B. MAIN IDEA OF GT-GAN**

In this section, a novel transductive GAN-based method is presented to alleviate the inequality between seen and unseen categories in the training stage. It can be decomposed into four steps, as illustrated in Fig. 2.

First, suppose that a seen-class image classifier, i.e., a softmax classifier, has been trained by the real seen-class image features from $X_s$ with their corresponding labels from $Y_s$; we then classify the unlabeled unseen image features from $X_u$ with this classifier. The output can be regarded as a probability distribution matrix $P \in \mathbb{R}^{n \times k_s}$, which represents the probabilities that each sample is assigned to each of the seen classes, where $n$ is the number of input features, and $k_s$ stands for the number of seen classes.

Second, we regard $P$ as the similarity matrix between each unlabeled unseen sample and seen class. In this case, a new matrix $T_s \in \mathbb{R}^{k_s \times m}$, consisting of all the seen-class semantic attributes, is introduced to compute a weighted semantic matrix $T_w \in \mathbb{R}^{n \times m}$ with $P$ as the weights:

$$T_w = PT_s$$

Hence, every unlabeled unseen sample has its own semantic vector $t_w \in \mathbb{R}^{m}$ from each row of $T_w$.

Third, since each weighted semantic attribute $t_w$ and unseen-class attribute $t_u$ are both $m$-dimensional vectors, we calculate the $L_2$ distances between each $t_w$ and all $t_u$, where the latter constitutes the unseen-class semantic space $T_u$. Then, we select $k t_u$ with the smallest distances. This is the equivalent of mapping each weighted semantic vector to the unseen-class semantic space $T_u$ and selecting the $k$-nearest unseen semantic vectors. The distance of the $i$-th sample and its $j$-th selected unseen semantic vector can be formulated as:

$$d_{ij}^2 = ||t_w(i) - t_u(j)||^2,$$
where \( i \in [1, n] \) and \( j \in [1, k] \). It should be noted that for traditional ZSL models, selecting the nearest one as the label of the unseen-class image means completing the classification task.

Finally, we utilize \( k \) unseen semantic vectors and random noises to generate \( C \) sample features and calculate the soul sample \([22]\), i.e., the average sample, for each \( x_i \), which is illustrated as follows:

\[
\tilde{z}_{ij} = \frac{1}{C} \sum_{l=1}^{C} z_{ij}^{(l)},
\]

where \( \tilde{z}_{ij} \) denotes the soul sample corresponding to the \( j \)-th nearest neighbor for \( x_i \), and \( z_{ij}^{(l)} \) is the \( l \)-th generated fake image feature. Then, these \( k \) soul samples can be compared with the original image feature, which corresponds to the weighted semantic vector \( t_w \), and a contrastive loss with semantic distances can be computed:

\[
L_{Ra} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \lambda_{ij} ||\tilde{z}_{ij} - x_i||^2_2,
\]

where \( x_i \) is the \( i \)-th original unlabeled unseen image feature, and \( \lambda_{ij} \) is the weight parameter of the loss between the \( j \)-th soul sample of the \( i \)-th weighted vector, i.e., \( \tilde{z}_{ij} \), and \( x_i \). Specifically, for each \( i \), let \( \{\frac{1}{2d_{i1}^2}, \frac{1}{2d_{i2}^2}, \ldots, \frac{1}{2d_{ik}^2}\} \) be normalized as \( \{\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{ik}\} \) (\( \sum_{j=1}^{k} \lambda_{ij} = 1 \)) to guarantee that the closer \( t_u \) is to \( t_w \), the generated fake sample should be closer to the original one. Obviously, \( L_{Ra} \) makes the relative positions between each image and its \( k \)-nearest neighbors in the feature space as consistent as possible with their relative positions in the semantic space.

\section{Overall Model}

In this section, we will introduce the GT-GAN model, which is based on LisGAN \([22]\). The full procedure is illustrated in Fig. 3, where the two blocks at the top of the image represent the training process and the two blocks below represent the classification.

1) \textbf{Loss Functions}

The conditional WGAN \([7]\) consists of a conditional generator \( G \) and a conditional discriminator \( D \), where \( G \) takes the attribute vector \( t \) and noise \( z \) as inputs to generate a fake image feature \( \tilde{x} = G(z, t) \), and \( D \) takes real image feature \( x \), fake feature \( \tilde{x} \) and \( t \) as its inputs to discriminate whether a feature of one category is real or fake. The loss function of the conditional WGAN can be defined as

\[
L_G = -E[D(\tilde{x}, t)]
\]

\[
L_D = E[D(x, t)] - E[D(\tilde{x}, t)] + \beta E[||\nabla_x D(\tilde{x}, t)||_2 - 1]^2,
\]

where \( \hat{x} = x + (1 - \varepsilon)\tilde{x} \) with \( \varepsilon \sim U(0, 1) \), and \( \beta \) is a hyperparameter. The first two terms approximate the
Wasserstein distance, and the last term is the gradient penalty, which enforces the Lipschitz constraint [26]. In this article, we set $\beta = 10$, which has been found to work well across a variety of architectures and datasets in [26].

To generate better unseen features that are closer to the real features, we introduce $L_{Ra}$ in Section III-B as a regularization of $L_G$, so the full loss function of $G$ can be formulated as follows:

$$L_G^f = L_G + \gamma_u L_{Ra},$$

where $\gamma_u$ is the hyperparameter weighting of this regularization, whose settings are described in Section IV-B. Specifically, the existing models may use different regularizations to modify $L_G$, but the ratio of $L_{Ra}$ and $-E[D(\tilde{x}, t)]$ can be set as a constant, i.e., $\gamma_u$.

2) TRAINING THE MODEL

As shown in Fig. 3, the top two blocks represent training process for the generator, where the first represents the training process of the general GAN-based model, e.g., LisGAN, and the second represents the use of $L_{Ra}$ for training. The specific training process for each epoch can be described as follows:

For the first block, given dataset $S$ and random noises $z \sim N(0, 1)$, we can leverage the generator $G$ that utilizes the seen-class attribute vectors $t_s$ and $z$ to generate fake seen image features $\tilde{x}_s = G(z, t_s)$. To ensure that the generated features are as close to the real features as possible, the discriminator $D$ usually takes real seen features $x_s$, $\tilde{x}_s$ and $t_s$ as its inputs to determine whether a feature of the specific class is real or fake. In addition, to make the synthesized features more discriminative, a softmax classifier (the seen-class image classifier in Fig. 3), which has been trained with real seen features from $X_s$, is also introduced to classify $\tilde{x}_s$ and output a classification loss. In addition, GT-GAN also leverages soul sample regularization to improve feature generation. Both the classification loss and soul sample regularization are the same as those in LisGAN [22].

In this case, we already have the loss function of $G(L_G)$, including a Wasserstein loss $-E[D(\tilde{x}, t)]$, a classification loss and soul sample regularization. The details of the loss function and the specific settings of the weighting hyperparameters can be found in [22]; we use the same settings here. We also obtain the loss of $D(L_D)$, which is the same as in Eq. 6. We will first perform the backpropagation and update the parameters of $D$ with $L_D$ but will not train $G$ because we need to obtain loss $L_{Ra}$ in the second block.

As described in Section III-B, we introduce unlabeled unseen features $x_u^{it}$ to obtain a contrastive loss $L_{Ra}$ by leveraging the seen-class image classifier and the semantic attributes of both seen and unseen categories. Hence, the full loss function of $G(L_G)$ can be composed of $L_{Ra}$ and $L_G$ as defined in Eq. 7. Then, the parameters of $G$ can be updated, completing the training of one epoch. Obviously, the main idea of GT-GAN is easily applied to other GAN-based models; in the following experiments, we also add $L_{Ra}$ to f-CLSWGAN [7] to observe the improvement in performance.

3) CLASSIFICATION

We utilize the same classification process as in the general GAN-based model, i.e., the two lower blocks in Fig. 3. First, the fake unseen image features $\tilde{x}_u$ are synthesized by the trained generator with noise $z$ and unseen semantic descriptions $t_u$. Then, we can train a softmax classifier, i.e., the unseen-class image classifier, on these features to classify the real unseen features $x'_u$. In addition, we leverage the confident samples that are considered to be correctly classified to fine-tune the classification results as in [22].

D. FULL ALGORITHM PROCEDURE

In this section, we summarize the training and the classification process of GT-GAN in Algorithm 1 and Algorithm 2, respectively. It should be noted that we omit the generator’s parameters $\theta_G$ and the discriminator’s parameters $\theta_D$ in the loss functions formulated earlier. Actually, if the generator and the discriminator are used in the calculation of a loss function, this loss function should be related to their parameters and the gradient can also be calculated.

**Algorithm 1** The Training Process of GT-GAN

**Input:** $z$, random noises. $x_s$, real seen features. $x_u$, real unseen features. $t_s$, seen-class attribute vectors. $t_u$, unseen-class attribute vectors.

**Output:** $\theta_G$, the generator’s parameters.

**Initialization:** $\theta_{G_0}$, initial generator’s parameters. $\theta_{D_0}$, initial discriminator’s parameters. $\tilde{n}_{epoch}$, the number of the training epochs. $n_{train}$, the number of the training samples. $M$, the batch size. $n_{iter}$, the number of iterations of the discriminator per generator iteration. All experiments in the article used the values $M = 64$, $n_{iter} = 5$ as suggested in [9]. The values of $\tilde{n}_{epoch}$ and $n_{train}$ changed in different datasets.

1: \textbf{for} $e = 1, 2, \ldots, \tilde{n}_{epoch}$ \textbf{do}
2: \hspace{1em} \textbf{for} $t = 1, 2, \ldots, \lfloor n_{train}/M \rfloor$ \textbf{do}
3: \hspace{2em} \textbf{for} $iter = 1, 2, \ldots, n_{iter}$ \textbf{do}
4: \hspace{3em} Get a batch of real unseen features \{s_u^{it}\}_{i=1}^M from the training data.
5: \hspace{3em} Generate a batch of fake features \{\tilde{s}_u^{it}\}_{i=1}^M by the generator with $[c_u^{it}, t_u^{it}]_i = 1$.
6: \hspace{3em} Calculate the loss $L_D$ by Eq. 6 and the gradient of $\frac{\partial L_D}{\partial \theta_D}$.
7: \hspace{3em} Update the parameters $\theta_D$ by backpropagation.
8: \hspace{2em} \textbf{end for}
9: \hspace{1em} Generate a batch of fake features \{\tilde{s}_u^{it}\}_{i=1}^M by the generator with $[c_u^{it}, t_u^{it}]_i = 1$.
10: \hspace{1em} Calculate the loss $L_G$ according to [22].
11: \hspace{1em} Get a batch of real unseen features \{s_u^{it}\}_{i=1}^M.
12: \hspace{1em} Calculate the loss $L_{Ra}$ according to Section III-B.
13: \hspace{1em} Calculate the full loss function $L_G^f$ by Eq. 7 and the gradient of $\frac{\partial L_G^f}{\partial \theta_G}$.
14: \hspace{1em} Update the parameters $\theta_G$ by backpropagation.
15: \hspace{1em} \textbf{end for}
16: \textbf{end for}
Algorithm 2: The Classification Process of GT-GAN

Input: $z$, random noises. $t_u$, unseen-class attribute vectors. $x_u$, real unseen features.

Output: The final classification results of $x_u$.

1: Generate fake unseen features $\tilde{x}_u$ by the trained generator with $t_u$ and $z$.
2: Train a softmax classifier on these synthesized features $\tilde{x}_u$.
3: Classify the real unseen features $x_u$ with the trained classifier.
4: Fine-tune the classification results with the confident samples according to [22].

IV. EXPERIMENTS

A. DATASETS AND EVALUATION PROTOCOL

We evaluate our method on five popular benchmark datasets, i.e., Caltech-UCSD-Birds 200-2011 (CUB) [27], Oxford Flowers (FLO) [28], SUN Attribute (SUN) [29], Animals with Attributes (AWA) [30], and aPascal/aYahoo (aPY) [31] for both ZSL and GZSL settings. CUB is an image dataset with 11,788 photos of 200 bird species, which is annotated with 312 attributes. FLO consists of 8,189 images from 102 different categories of flowers. SUN contains 14,340 images, which come from 717 types of scenes annotated with 102 attribute labels. AWA is a small-scale dataset with 30,475 images, 50 classes and 85 attributes. Finally, aPY is a small-scale dataset with 15,339 pictures originating from 32 categories and a 64-dimensional attribute vector for each category. Among these datasets, CUB, FLO and SUN are all fine-grained datasets, while AWA and aPY are coarse-grained datasets. The statistics and splitting settings of the datasets are presented in Table 1, which are the same as those described in [2] and [32].

| Dataset | attributes | images | seen classes (train/val) | unseen classes |
|---------|------------|--------|-------------------------|---------------|
| CUB     | 312        | 11788  | 106/50                  | 50            |
| FLO     | 1024       | 8189   | 62/20                   | 20            |
| SUN     | 102        | 14340  | 58/65                   | 72            |
| AWA     | 85         | 30475  | 23/13                   | 10            |
| aPY     | 64         | 15339  | 15/5                    | 12            |

To obtain the image features for further training, we use ResNet-101, which is pretrained on ImageNet, to extract 2048-dimensional features. For the semantic attributes, we utilize the default attributes in CUB, SUN, AWA and aPY. Since FLO has no attribute annotations, we use the 1024-dimensional CNN-RNN features from a previous work [32].

At the testing stage, we adopt the average per-class top-1 accuracy as in [7], [22] to evaluate the performance of each method with ZSL. In the GZSL setting, we report the harmonic mean (H) to measure the overall performance on the seen and unseen classes, which can be defined as

$$H = \frac{2 \times S \times U}{S + U},$$

where $S$ and $U$ represent the average per-class top-1 accuracy for the seen and unseen classes, respectively.

B. IMPLEMENTATION DETAILS

Because the proposed GT-GAN works on the existing GAN-based ZSL models, the generator it leverages is the one from the corresponding model. The classifier it uses is a standard $k_s$-way softmax classifier, which has been trained with seen samples in advance. Specifically, in this article, we set $\gamma_u = 0.04$, $k = 4$ and $C = 20$ for the ZSL setting. In the GZSL setting, we set $\gamma_u = 0.1$, $k = 5$ and $C = 20$. Actually, we have run experiments on the different values of the three parameters for both ZSL and GZSL settings, as shown in Fig. 4. According to the results, we select these two different parameter settings for ZSL and GZSL to achieve the best overall effects, respectively.

C. GENERAL RESULTS

In this section, we evaluate GT-GAN and some other inductive and transductive models on five popular benchmark datasets, i.e., CUB, FLO, SUN, AWA and aPY with ZSL and GZSL settings, and compare the best effects with those from the classic models and the state-of-the-art models. We also apply $L_{\text{d}}$ of GT-GAN to f-CLSWGAN [7] to evaluate the performance, which is represented as GT-GAN$_f$-CLSWGAN in the result tables.

1) ZERO-SHOT LEARNING

We first provide the results of each model for these five datasets in Table 2. For the ZSL setting, the classification result for each sample is restricted to $y \sim Y_u$. It can be seen that GT-GAN and GT-GAN$_f$-CLSWGAN perform better than LisGAN and f-CLSWGAN, respectively. For f-CLSWGAN, the improved model achieves 1.0%, 0.3%, 1.1% and 2.4% improvements on CUB, FLO, SUN and aPY and

| Model     | CUB | FLO | SUN | AWA | aPY |
|-----------|-----|-----|-----|-----|-----|
| DAP [33]  | 40.0| -   | 39.9| 44.1| 33.8|
| CMT [34]  | 34.6| -   | 39.9| 39.5| 28.0|
| CONSE [35]| 34.3| -   | 38.5| 45.6| 26.9|
| SSE [19]  | 43.9| -   | 51.5| 60.1| 34.0|
| DEVISE [16]| 52.0| 45.9| 56.5| 54.2| 39.8|
| SIE [18]  | 53.9| 53.4| 53.7| 65.6| 32.9|
| LATEM [36]| 49.3| 40.4| 55.5| 55.1| 35.2|
| EZZSL [37]| 53.9| 51.0| 54.5| 58.2| 33.3|
| ALE [17]  | 54.9| 48.5| 58.1| 59.9| 39.7|
| SYNC [20]  | 55.6| -   | 56.3| 54.0| 23.9|
| SAE [38]  | 33.3| -   | 40.3| 53.0| 8.3 |
| DEM [39]  | 51.7| -   | 61.9| 68.4| 35.0|
| GAIZSL [8]| 55.8| 60.5| 61.3| 68.2| 41.1|
| TCN [40]  | 59.5| -   | 61.5| 70.3| 38.9|
| f-CLSWGAN [7]| 57.3| 67.2| 60.8| 68.2| 40.5|
| LisGAN [22]| 58.8| 69.6| 61.7| 70.6| 43.1|
| ALE-train [41]| 54.5| 48.3| 55.7| 66.5| 46.7|
| DSRL [3]  | 48.7| 37.7| 56.8| 74.7| -   |
| SE-ZSL [21]| 59.6| -   | 63.4| 69.5| -   |
| GT-GAN$_f$-CLSWGAN| 58.3| 67.5| 60.8| 69.3| 42.9|
| GT-GAN  | 59.5| 71.3| 60.7| 71.6| 45.0|

VOLUME 8, 2020

TABLE 2. Results from evaluating ZSL on the five datasets. I: inductive methods, T: transductive methods. The best and second best results are in bold.
FIGURE 4. Hyperparameter sensitivities of $\gamma_u$, $k$ and $C$ for five datasets in both the ZSL and GZSL settings.

almost the same accuracy on SUN. In addition, the GT-GAN also achieves 0.7%, 1.7%, 1.0% and 1.9% improvements on CUB, FLO, AWA and aPY and a very close accuracy on SUN with respect to LisGAN. Specifically, it achieves almost state-of-the-art performance. According to these results, our method can actually help recognize unseen samples by making the generator synthesize better unseen features that are more discriminative and closer to real features.

2) GENERALIZED ZERO-SHOT LEARNING
The GZSL experiment results are demonstrated in Table 3. In this setting, both seen and unseen samples need to be recognized, i.e., the classification result of each sample is $y \sim Y_s \cup Y_u$. Hence, we adopt the harmonic mean (H) to measure the overall performance of each method on the seen and unseen categories. From the results, we can see that the GT-GAN$_{f$-CLSWGAN$}$ method improves the accuracy over the original f-CLSWGAN by 4.9%, 4.5%, 3.1%, 7.5% and 6.9% on CUB, FLO, SUN, AWA and aPY, respectively, and GT-GAN achieves 4.0%, 4.9%, 2.6%, 5.4%, and 5.8% improvements over LisGAN. We can also see that GT-GAN generally outperforms the other models on all five datasets. Although some models reach a higher accuracy on either seen or unseen classes, they are unable to show maximal accuracy for both classes combined. Hence, our method can make the models more generalized and stronger than their original forms.
Comparing the effects of ZSL and GZSL, it can be seen that our proposed GT-GAN can actually be applied to various existing GAN-based ZSL models and improve their performances. We can also observe that our method achieves more substantial gains in GZSL. This is because in the ZSL setting, the final classifier is trained on fake unseen samples, and it only classifies real unseen samples at test time. Therefore, the generated unseen features in each category during classifier training do not need to be very similar to the real features but are nevertheless relatively discriminative and can guarantee correct classification. However, our method helps synthesize highly related unseen samples with real samples, which cannot obviously improve the accuracy of recognition. In the GZSL setting, we utilize the final classifier trained on real seen samples and generated unseen samples to recognize both seen and unseen real samples. In this case, if the generated unseen features cannot reach a sufficient quality, the classifier may bias the predictions to the seen classes, which may lead to a drop in accuracy, i.e., the feature confusion problem. Since our method helps generate unseen samples that are close to the real samples, it can alleviate this issue and obtain significantly higher accuracy.

D. MODEL ANALYSIS

In this section, we analyze our model by performing an ablation study, investigating hyperparameter sensitivity, testing stability and convergence, and assessing the effect of the number of synthesized unseen samples based on GT-GAN. In addition, a further analysis is also performed to directly prove that our method actually helps synthesize unseen samples that are close to real samples instead of inferring as such from the improvement in accuracy.

1) ABLATION STUDY

In this paper, we apply our GT-GAN to LisGAN [22]. Since the previous ablation study in [22] was sufficient and the experiments in the current paper illustrated the effect of our proposed method, we will not perform more ablation studies.

2) HYPERPARAMETER SENSITIVITY

We investigate the sensitivities of three hyperparameters, $\gamma_u$, $k$, and $C$, for both ZSL and GZSL settings. $\gamma_u$ denotes the weighting coefficient of contrastive loss, whose effect is reported in Fig. 4(a) and Fig. 4(b). $k$ is the number of selected attributes as well as the number of soul samples, and its influence is shown in Fig. 4(c) and Fig. 4(d). Similarly, we also introduce $C$ as the number of synthesized features of each attribute used to measure how many generated samples can represent the overall effect of the generator, whose sensitivity is shown in Fig. 4(e) and Fig. 4(f). From these results, we can see that the weighting coefficient $\gamma_u$ should be relatively small for ZSL and GZSL; otherwise, the performances on most datasets would degrade dramatically as $\gamma_u$ increases. In addition, our model achieves the best result on each dataset with different $k$ but is not very sensitive to this parameter. Thus, we set $k = 4$ and $k = 5$ for ZSL and GZSL, respectively, to achieve the best overall performances. Similarly, it is easy to observe that our model is not sensitive to $C$, so we set this parameter 20 to reduce the computational costs.

3) STABILITY AND CONVERGENCE

Since our proposed model is based on GAN, which needs several training epochs to achieve the balance between the generator and the discriminator, it is essential to test the stability and convergence of our model during training. We report the results for both ZSL and GZSL settings with increasing training epochs in terms of testing errors in Fig. 5. It can be.

### TABLE 3. Results from evaluating GZSL on five datasets. I: inductive methods, T: transductive methods. We report the average per-class top-1 accuracy of seen classes (S) and unseen classes (U) and their harmonic mean (H) to evaluate each method. The best and second best results are in bold.

| Model       | CUB | FLO | SUN | AWA | aPY |
|-------------|-----|-----|-----|-----|-----|
| U | S | H | S | H | S | H | S | H | S | H |
| DAP [33]    | 4.7 | 67.9 | 3.3 | - | - | - | 4.2 | 25.2 | 7.2 | 0.0 | 88.7 | 0.0 | 4.8 | 78.3 | 9.0 |
| CMT [34]    | 7.2 | 49.8 | 12.6 | - | - | - | 3.1 | 21.8 | 11.8 | 0.9 | 87.6 | 1.8 | 1.4 | 85.2 | 2.8 |
| CONSE [39]  | 1.6 | 72.2 | 3.1 | - | - | - | 6.8 | 39.9 | 11.6 | 0.4 | 88.6 | 0.8 | 0.0 | 91.2 | 0.0 |
| SSE [19]    | 8.5 | 46.9 | 14.4 | - | - | - | 2.1 | 36.4 | 4.0 | 7.0 | 80.5 | 12.9 | 0.2 | 78.9 | 0.4 |
| DEVISE [16] | 23.8 | 53.0 | 32.8 | - | - | - | 16.9 | 24.5 | 16.2 | 13.4 | 68.7 | 22.4 | 4.9 | 76.9 | 9.2 |
| SJE [18]    | 23.5 | 59.2 | 33.6 | - | - | - | 14.7 | 30.5 | 19.8 | 11.3 | 74.6 | 19.6 | 3.7 | 55.7 | 6.9 |
| LATEM [36]  | 15.2 | 57.9 | 24.0 | - | - | - | 14.7 | 28.8 | 19.5 | 7.3 | 71.7 | 13.3 | 0.1 | 73.0 | 0.2 |
| BZSL [37]   | 3.4 | 70.1 | 4.6 | - | - | - | 11.0 | 25.9 | 15.8 | 5.9 | 77.8 | 11.0 | 2.4 | 70.1 | 4.6 |
| ALA [17]    | 4.6 | 73.7 | 8.7 | - | - | - | 23.1 | 33.6 | 26.3 | 14.0 | 81.8 | 23.9 | 4.6 | 73.7 | 8.7 |
| SYNC [20]   | 7.4 | 66.3 | 13.3 | - | - | - | 7.9 | 43.3 | 13.4 | 10.0 | 90.5 | 18.0 | 7.4 | 66.3 | 13.3 |
| SAE [38]    | 7.8 | 54.0 | 13.6 | - | - | - | 8.8 | 18.0 | 11.8 | 1.1 | 82.2 | 2.2 | 0.4 | 80.9 | 0.9 |
| DEM [39]    | 11.1 | 75.1 | 19.4 | - | - | - | 20.5 | 34.3 | 25.6 | 30.5 | 86.4 | 45.1 | 11.1 | 75.1 | 19.4 |
| GAZSL [8]   | 23.9 | 60.6 | 34.3 | - | - | - | 21.7 | 34.5 | 26.7 | 19.2 | 86.2 | 31.4 | 14.2 | 78.6 | 24.0 |
| TCN [40]    | 52.6 | 52.0 | 52.5 | - | - | - | 31.2 | 37.3 | 34.0 | 49.4 | 76.3 | 60.0 | 24.1 | 64.0 | 35.1 |
| 4-CLS-GAN [7] | 43.7 | 57.7 | 49.7 | 59.0 | 73.8 | 65.6 | 42.6 | 36.6 | 39.4 | 57.9 | 61.4 | 59.6 | 32.9 | 61.7 | 42.9 |
| LisGAN [22] | 46.5 | 57.9 | 51.6 | 57.7 | 83.8 | 68.3 | 42.9 | 37.8 | 40.2 | 52.6 | 76.3 | 62.3 | 34.3 | 68.2 | 45.7 |
| ALE-train [41] | 23.5 | 45.1 | 30.9 | 13.6 | 61.4 | 22.2 | 19.9 | 22.6 | 21.2 | - | - | - | - | - | - |
| GPZSL [42]  | 24.9 | 45.8 | 32.2 | 21.8 | 75.0 | 33.3 | 0.0 | 41.6 | 0.0 | - | - | - | - | - | - |
| DSRL [3]    | 17.3 | 39.0 | 24.0 | 26.9 | 64.3 | 37.9 | 17.7 | 25.0 | 20.7 | - | - | - | - | - | - |
| GT-GAN-L4-CLS-GAN | 49.7 | 60.7 | 54.6 | 63.8 | 77.8 | 70.1 | 51.0 | 36.4 | 42.5 | 58.2 | 79.4 | 67.1 | 37.8 | 73.0 | 49.8 |
| GT-GAN     | 48.9 | 64.3 | 55.6 | 64.7 | 84.4 | 73.2 | 50.8 | 36.9 | 42.8 | 58.3 | 80.7 | 67.7 | 38.9 | 76.2 | 51.5 |
seen that the training process of our model is generally stable although there are some small fluctuations. In addition, our model can converge to a stable state with 20 epochs.

4) EFFECT OF THE NUMBER OF SYNTHESIZED SAMPLES

We also analyze how the number of generated unseen samples $N$ for classifier training influences the test accuracy, as shown in Fig. 6. We can see that the accuracy on all datasets increases rapidly as $N$ increases from 1 to 10, and the changes decrease as $N$ increases to 50. When $N>50$, the model is not very sensitive to it. In addition, we can also observe that the accuracy has larger improvements on CUB and SUN than on the other datasets, which may be because they have more unseen classes and need more synthesized samples to help recognize real features among these classes.

5) FURTHER ANALYSIS

We introduce an evaluation to directly prove that compared with LisGAN, GT-GAN actually synthesizes unseen samples that are close to the real samples, as shown in Fig. 7. As a soul sample can be regarded as the centre of a cluster, we calculate the square of the distance between the real soul sample $s^c$ and the fake soul sample $\tilde{s}^c$ of each unseen class to evaluate the difference between the overall generation and real sample:

$$L_E = \frac{1}{k_u} \sum_{c=1}^{k_u} ||\tilde{s}^c - s^c||_2^2,$$

where $k_u$ is the number of unseen categories. In Fig. 7, we can see that GT-GAN can generate better samples, which are close to the real soul samples, for each epoch to train the classifier. Thus, we can achieve higher classification accuracy than the original model.

V. CONCLUSION

In this paper, we propose a general transductive GAN-based method called GT-GAN, which leverages unlabeled unseen samples for training to improve the performance of the original generator. A new loss function is also introduced to help generate unseen features that are close to the real features and make the relative positions between each unseen image and its $k$-nearest neighbors in the feature space as consistent as possible with their relative positions in the semantic space; this loss function may be used in most existing GAN-based models. Extensive experiments on five benchmark datasets show that our method can be applied to various existing GAN-based (conditional WGAN) models and achieves substantially higher accuracy than the original models, especially in the GZSL setting. The model analysis also demonstrates the sensitivity of each parameter and verifies that our model is able to better generate features for the unseen categories.

However, there are several limitations or directions for improvements. First, the synthesized unseen samples cannot represent most aspects of one class because we cannot obtain...
soul samples of unlabeled features and learn from them at the training stage. Second, we cannot guarantee that the categories represented by the selected $k$ semantic vectors are all highly related to the real category of the original unlabeled unseen feature. In other words, the generator still suffers a risk of being incorrectly trained. Further work will be required to solve these problems.

REFERENCES

[1] Z. Ding, M. Shao, and Y. Fu, “Low-rank embedded ensemble semantic dictionary for zero-shot learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2050–2058.

[2] Y. Xian, B. Schiele, and Z. Akata, “Zero-shot learning — The good, the bad and the ugly,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 4582–4591.

[3] M. Ye and Y. Guo, “Zero-shot classification with discriminative semantic representation learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 7140–7148.

[4] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and W. Wang, V. W. Zheng, H. Yu, and C. Miao, “A survey of zero-shot learning,” in Proc. IEEE/ACM Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 7402–7411.

[5] J. Li, M. Jing, K. Lu, Z. Ding, L. Zhu, and Z. Huang, “Leveraging the invariant side of generative zero-shot learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 7402–7411.

[6] S. Reed, Z. Akata, H. Lee, and B. Schiele, “Learning deep representations for zero-shot learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 7402–7411.

[7] Y. Xian, T. Lorezzi, B. Schiele, and Z. Akata, “Feature generating networks for zero-shot learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2018, pp. 5542–5551.

[8] Y. Zhu, M. Elhoseiny, B. Liu, X. Peng, and A. Elgammal, “A generative adversarial approach for zero-shot learning from noisy texts,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1004–1013.

[9] S. Martin Arjovsky and L. Bottou, “Wasserstein generative adversarial networks,” in Proc. 34th Int. Conf. Mach. Learn., Sydney, NSW, Australia, 2017., pp. 214–223.

[10] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, “Least squares generative adversarial networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2794–2802.

[11] C. Wang, S. Dong, X. Zhao, G. Papantastisou, H. Zhang, and G. Yang, “SaliscyGAN: Deep learning semisupervised salient object detection in the fog of IoT,” IEEE Trans. Intell. Syst., vol. 16, no. 3, pp. 567–577, Dec. 2020.

[12] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5767–5777.

[13] C. Wang, S. Bronson, P. Perona, and S. Belongie, “The caltech-ucsd birds-200-2011 dataset,” Caltech, Pasadena, CA, USA, Tech. Rep. CNS-TR2011-001, 2011.

[14] G. Patterson and J. Hays, “SUN attribute database: Discovering, annotating, and recognizing scene attributes,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2012, pp. 2751–2758.

[15] C. H. Lampert, H. Nickisch, and S. Harmeling, “Learning to detect unseen object classes by between-class attribute transfer,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2009, pp. 951–958.

[16] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, “Describing objects by their attributes,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2009, pp. 1778–1785.

[17] S. Reed, Z. Akata, H. Lee, and B. Schiele, “Learning deep representations of fine-grained visual descriptions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 49–58.

[18] C. H. Lampert, H. Nickisch, and S. Harmeling, “Attribute-based classification for zero-shot visual object categorization,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 3, pp. 453–465, Mar. 2014.

[19] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, “Zero-shot learning through cross-modal transfer,” in Proc. Adv. Neural Inf. Process. Syst., pp. 935–943, 2013.

[20] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean, “Zero-shot learning by convex combination of semantic embeddings,” 2013, arXiv:1312.5650. [Online]. Available: http://arxiv.org/abs/1312.5650.

[21] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, “Latent embeddings for zero-shot classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 69–77.

[22] B. Romero-Paredes and P. Torr, “An embarrassingly simple approach to zero-shot learning,” in Proc. Int. Conf. Mach. Learn., 2015, pp. 2152–2161.

[23] E. Kodirov, T. Xiang, and S. Gong, “Semantic autoencoder for zero-shot learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 3174–3183.

[24] L. Zhang, T. Xiang, and S. Gong, “Learning a deep embedding model for zero-shot learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2021–2030.

[25] H. Jiang, R. Wang, S. Shan, and X. Chen, “Transferable contrastive network for generalized zero-shot learning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 9765–9774.

[26] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, “Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 9, pp. 2251–2265, Jun. 2019.

[27] V. K. Verma and P. Rai, “A simple exponential family framework for zero-shot learning,” in Proc. Joint Euro. Conf. Mach. Learn. Knowl. Discovery Databases, Cham, Switzerland: Springer, 2017, pp. 792–808.
JUNHAO DONG was born in Bengbu, Anhui, China, in 1999. He is currently pursuing the bachelor’s degree majoring in telecommunication engineering with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China. His research interests include machine learning and computer vision.

BO XIAO was born in Changyi, Shandong, China, in 1975. He received the B.S. degree in image transmission and processing, the M.S. degree in computer science, and the Ph.D. degree in signal and information processing from the Beijing University of Posts and Telecommunications, Beijing, China, in 1998, 2005, and 2009, respectively. He has been with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, since 1998, where he has been an Associate Professor, since 2010. From September 2018 to August 2019, he was a Visiting Scholar with the University of Windsor, Windsor, ON, Canada. He has coauthored five books and more than 40 journals and conference papers. His research interests include data mining, pattern recognition, deep learning, computer vision, and intelligent wireless communication.

BO DING was born in Dancheng, Henan, China, in 2000. He is currently pursuing the bachelor’s degree majoring in information engineering with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China. His research interests include natural language processing, machine learning, and computer vision.

HAOYU WANG was born in Chaoyang, Liaoning, China, in 1997. He received the B.S. degree in information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2019, where he is currently pursuing the degree majoring in information and communication engineering. His research interests include natural language processing and computer vision.

***