Zero-Shot Learning Based on Quality-Verifying Adversarial Network

Siyang Deng, Quanxue Gao, Wei Xia, Gang Xiang, and Xinbo Gao *Senior Member, IEEE*

***Abstract*—Recently, generative adversarial network (GAN)-based zero-shot learning methods have attracted widespread attention. However, due to the randomness of GAN generation, most existing methods cannot well guarantee to generate sufficiently reliable features and have good generalization ability. Targeting at these problems, we propose an effective Quality-Verifying Adversarial Network (QVAN) that consists of one generator and double discriminators. Adversarial learning between the former discriminator and generator is to generate visual features, which can be partitioned into pseudo-generated features and reliable-generated features. The latter discriminator is used for quality-verifying that will guide the generator to generate more reliable features that are near the real visual features. To avoid over-fitting and ensure intra-class diversity, we set the threshold for each class to distinguish pseudo-generated features and reliable-generated features. To further preserve both compactness and discriminability of the samples, we introduce the class metric constraint, which are more conducive to classification. Moreover, we introduce $\ell_{1,2}$-norm constraint to fully consider the specific distribution among different classes, thus making the generated features more discriminant. Extensive experiments on several real-world datasets show the effectiveness of the proposed approach, which demonstrate the advantage over the state-of-the-art methods.**

***Index Terms*—Zero-shot learning, Generative adversarial network, Quality-verifying, Class metric constraint, ℓ_{12}-norm constraint**

I. Introduction

In real world applications, with the rapid development of Internet and smart devices, zero-shot learning has attracted much attention in dealing with multimedia data [48], [32], [9]. Due to its extraordinary performance, numerous zero-shot learning methods have been widely proposed [41], [47], [17], [14], [53] in the past decade. Zero-shot learning is to identify categories that have never been seen before. Given semantic descriptions of object classes, the purpose of zero-shot learning is to accurately recognize objects of the unseen classes, from which no examples are available at the training stage, by associating them to the seen classes, from which labeled examples are provided [1].

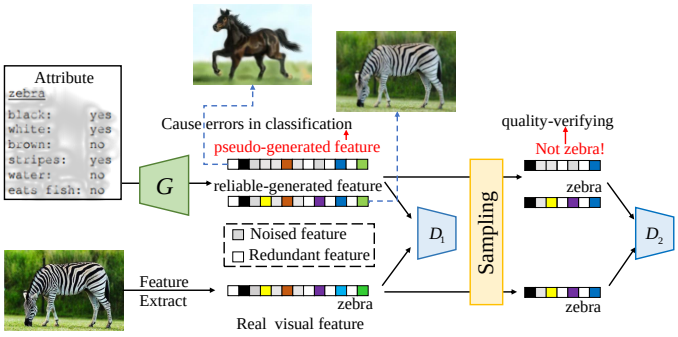


Fig. 1: Our general framework and the motivation of quality-verifying.

The underlying secret ensuring the success of zero-shot learning is to find a mapping that links the semantics to the image, and then apply the mapping to unseen classes to realize the migration from the seen class to the unseen class. Much effort has been made to learn the mapping. Qi et al. [29] learned a transfer function as a bridge to propagate the labels between text and image spaces, as a result, text and visual data can be jointly explored. Changpinyo et al. [1] aligned the semantic space that was derived from external information to the model space that concerned itself with recognizing visual features, they introduced a set of "phantom" object classes whose coordinates lived in both the semantic space and the model space. Qiao et al. [31] proposed an $\ell_{2,1}$-norm based objective function which can simultaneously suppress the noisy signal in the text and learn a function to match the text document and visual features. Long et al. [43] retained one-to-one mapping strategy and synthesized visual data via mapping attributes of classes or instances to the visual space.

Recent years, Generative Adversarial Network (GAN) [11] has been widely used in zero-shot learning due to its promising performance on learning a mapping from one modal to other modal. Therefore, many zero-shot learning methods based on GAN are widely proposed and achieve good experimental results. They consider the features generated by the generator to be reliable after adversarial learning or some regularization constraints. However, due to the randomness of GAN generation, the generated features may not reliable enough. As shown in Fig. 1, the generated feature is relatively high dimensional, it contains important category information, noise and irrelevant items. When the real image sample is a zebra, the two generated features are similar to the real visual features

Manuscript received ********; revised ********; accepted ********. ("current version" is the date the typeset version is posted on Xplore). This work is supported by National Natural Science Foundation of China under Grants 61773302 and 61372069.

Corresponding author: Quanxue Gao (e-mail:qxgao@xidian.edu.cn).

S. Deng, Q. Gao, and W. Xia are with the state key laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China.

G. Xiang is with Beijing Aerospace Automatic Control Institute, Beijing 100854, China and with School of Automation and Electrical Engineering, Beihang University, Beijing 100191, China.

X. Gao is with the Chongqing Key Laboratory of Image Cognition, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.

Digital Object Identifier (inserted by IEEE).

in most dimensions, thus confusing discriminator D_1 into thinking that both generated features are zebras. However, one of them is the reliable-generated feature which corresponds to a zebra, the other is the pseudo-generated feature which corresponds to a black horse, the existence of pseudo features makes the results of the model suboptimal, and will cause errors in the classification process. Therefore, we need to distinguish between reliable features and pseudo features in generated features, which motivates us to conduct quality-verifying on the generated features. If we consider all the features generated by GAN as unreliable, it will promote the generator to continuously produce more realistic features, which will lead to overfitting (small training errors and poor generalization ability). This is because that for the generated features with good quality, they are realistic enough and we do not need to deal with them, while for the generated features with poor quality, we hope to make them more realistic through model training. In the model, if all the generated features are considered to be reliable, the quality of the generated features will be not good enough, leading to errors in the subsequent classification process. Therefore, how to conduct quality-verifying and how to distinguish the reliable-generated features and the unreliable-generated features are the highlights of our paper.

In this paper, targeting at the limitation of GAN in generating features, we propose an effective Quality-Verifying Adversarial Network (QVAN) to improve it. Our main contributions are as follows. (1) We use the double discriminators for quality-verifying as shown in Fig. 1, low dimensional features are obtained by sampling, and pseudo-generated features can be detected in the process of quality-verifying. Such supervised information will guide the generator to generate more realistic and reliable features. Moreover, to avoid over-fitting and ensure intra-class diversity, we set the threshold which is the maximum reconstruction error we can accept for each class to distinguish pseudo-generated features and reliable-generated features. (2) To further preserve both compactness and discriminability of the generated features, we introduce the class metric constraint, which is more conducive to classification. (3) We use the ℓ_{12}-norm constraint to fully consider the specific distribution among different classes, so that features of each class are only distributed in the important dimensions, thus making the generated features more discriminant. Extensive experiments on several real-world datasets show the effectiveness of the proposed approach, which demonstrate the advantage over the state-of-the-art methods.

II. Related works

A. Generative Adversarial Network

Generative Adversarial Network (GAN) [11] is a promising deep learning model which was proposed in 2014. It consists of two submodules which are generator and discriminator. In the training process, the goal of generator is to generate realistic samples to confuse the discriminator, while the goal of discriminator is to try to distinguish the generated (fake) samples from the real ones. Thus, generator and discriminator constitute a dynamic " adversarial process ". In the end, when the discriminator is unable to distinguish the real sample from the fake one, the goal is achieved. That is, through adversarial learning, the generator generates really realistic samples.

Due to the characteristics of GAN, it is widely used in many fields. For example, in video coding, Zhu et al. [50] an intra prediction method to improve the video coding performance, in which GAN is adopted to intelligently remove the spatial redundancy with the inference process. In the field of high resolution image synthesis, Guo et al. [13] devise an auto-embedding generative adversarial net that generates high resolution images by learning a latent embedding extracted from an autoencoder. where GAN exploits the high-level photo-structure and acts as a bridge to connect the distributions of the input noise and real data. In the field of multi-source image fusion, Li et al. [20] integrate multi-scale attention mechanism into both generator and discriminator of GAN to fuse infrared and visible images. In the field of zero-shot learning, GAN can learn a mapping from one modal to other modal, so it can deal with semantics and images well.

B. Zero-Shot Learning Strategy

The common scenario of zero-shot learning is to train with the data of seen classes, such as the text descriptions(or semantic attributes) and images of the seen class [16], [49], [42], [35], [15], [51], and then use the trained model to identify the unseen class data. Therefore, the key problem of zero-shot learning is to learn a mapping that compares two different modals of data in the same space, then the model is applied to the unseen class to realize the migration from the seen class to the unseen class [1].

Due to the excellent performance of GAN in image generation, people generate images through GAN, so as to make a comparison at the level of image. However, this way of generating images not only has high dimension, but also the quality of the generated images is not good enough, resulting in unsatisfactory experimental performance. As a result, Xian et al.[41] proposed a GAN-based method that generated CNN features from semantic instead of generating images, they extract the CNN features from the real image as the real visual features. Then the generated features and real visual features are sent into the discriminator. Through the process of adversarial learning to make the generator to generate enough realistic features. For unseen classes during testing process, the trained model also has the ability to generate corresponding visual features according to the semantic of the unseen class, so as to realize the recognition of unseen class.

Inspired by generated features in GAN, many zero-shot learning methods based on GAN are widely proposed [16], [49], [42], [35], [15], [4], [10], and many improvements have been made to make the generated features more realistic and discriminative. Qi [28] proposed a method LS-GAN, which used a Lipschitz regularity condition on the density of real data to regularize the loss function, so that the regularized model can better generalize to produce new data from a reasonable number of training examples than the classic GAN. Later on, Qi et al. [30] proposed a novel localized generative adversarial net to learn on the manifold of real data, this way enables local

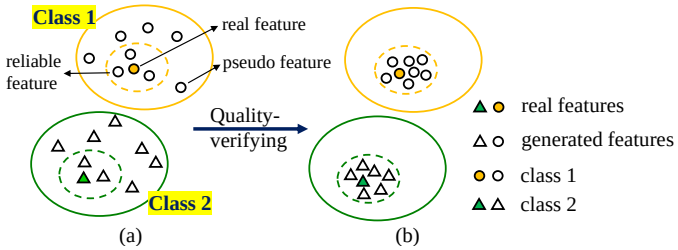


Fig. 2: Illustration our motivation of quality-verifying. Different shapes represent different classes, solid points represent real features, and hollow points represent generated features.

generators to adapt to and directly access the local geometry without need to invert the generator in a global GAN. Zhu et al. [52] proposed the conditional generative model which took the noisy texts as input and hallucinated the visual features for corresponding classes, they introduce visual pivot regularization to make the generated features more discriminative. Li et al. [21] trained a conditional Wasserstein GAN, to improve the quality of generated features, they introduced soul samples which were defined as the representations of each category. Yu et al. [45] aligned the visual-semantic interactions by formulating both the visual prototype generation and the class semantic inference into an adversarial framework and captured the discriminative information with an efficient multi-modal cross-entropy loss. Huang et al.[15] used a regressor to map each visual feature back to its corresponding class embedding, so that the generated visual features were more discriminative between classes. Narayan et al. [25] added semantic embedding decoder to transform visual features into semantic embeddings and construct circular consistency loss, and used feedback mechanism to guide the generation of features. Felix et al.[8] reconstructed the generated features back into the semantic space and minimized the reconstruction errors, so as to better establish the mapping of semantic to visual features. And based on this idea, Ye et al.[44] not only minimized the loss of reconstruction, but also calculated the structural loss between the generated features and the real visual features.

III. Motivation

Since the shortcoming of GAN-based zero-shot methods is that the generated features are not reliable enough. As shown in Fig. 2 (a), it represents the spatial distribution of generated features. The solid line circle represents the range where the discriminator thinks the generated features belong to this class. In other words, if the generated feature is within the solid line circle of this class, it can confuse the discriminator. The features in the dotted circle means that the generated features are of good quality, they are reliable-generated features. While the annular area between the dotted circle and the solid line circle indicates that the generated features are of poor quality, and we call them pseudo-generated features. The existence of pseudo-generated features makes the GAN-based ZSL methods obtain unreliable generated features. The spatial distributions of pseudo-features are far away from the real features, which may cause errors in the classification process. This situation motivates us to carry out quality-verifying.

How do we determine the range of the dotted box and the pseudo-features? We can extract the principal components from all generated features, calculate the reconstruction errors, and set the real features' maximum reconstruction error as the threshold of this class. In the low dimensional space, as for reliable-generated features, they are near the real visual features (The reconstruction error is less than the threshold). On the contrary, as for pseudo-generated features, the reconstruction errors are greater than the threshold. This information will be fed back to the generator to produce more stable and reliable features as shown in Fig. 2 (b).

In the process of generating features, we hope the features of the same classes are closer and of different classes are further apart, so we introduce the class metric constraint. Moreover, motivated by ℓ_{12}-norm, which has been widely used for feature selection [23], [18], [24], we find that it also has an important role of characterizing class-specificity distribution in dimension space. Therefore, we introduce ℓ_{12}-norm constraint to fully consider the specific distribution among different classes, thus making the generated features more discriminant.

IV. Method Framework

Fig. 3 shows the overall architecture of QVAN. It mainly contains 5 parts which are Generator G, Discriminant enhancement module, Discriminator D_1, Sampler S and Discriminator D_2, respectively. The details of each part will be described in the following.

Notations: $T \in R^{c\times l}$ is the text features, where c represents the number of classes in a batch of samples in the training process, and l is the dimension of text features. $\tilde{X} \in R^{n\times d_1}$ represents the set of features generated by the generator, of which $\tilde{x}$ is one of the generated features (The lowercase letters that correspond to the capital letters below have the similar meaning). $X \in R^{n\times d_1}$ is the real visual features, where n is the number of samples of a batch in the training process, and d_1 is the dimension of real visual features. $\tilde{Y} \in R^{n\times d_2}$ is the low-dimensional features of $\tilde{X}$ after sampling. Noise $z \in R^{n\times d_3}$, where d_3 is the dimension of the noise.

A. Generator G

The input of generator G is noisy text description T. The text encoder encodes T into features, it first marks the input text description as words, removes stop words, and then extracts Term Frequency Inverse Document Frequency (TF-IDF) features vector, which is also a pre-processing process for the input data. Then the encoded text passes through a fully connected layer to reduce the dimension. and the fully connected layer also contributes a lot to noise suppression. Next, we concatenate the text feature that was output in the previous step with the random noise z, where z obeys Gaussian distribution. Adding noise increases the variety of generator generation to the extent that the generator does not produce the same result every time. In this more diverse case, it is easier to find more appropriate generator parameters in iterative updates. Finally, we feed concatenated noisy features into the

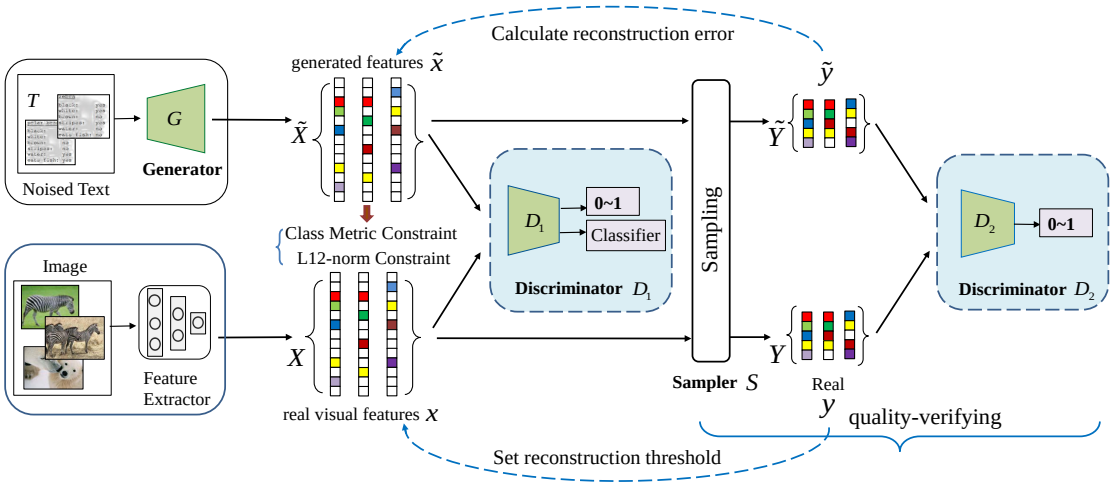


Fig. 3: Illustration of QVAN's overall architecture: The Generator G is used to generate realistic visual feature, and the former Discriminator D_1 is used to distinguish the generated features from the real visual features and to classify the generated features into the correct class. Before classification in Discriminator D_1, class metric constraint is used to make the features of the same class closer and of different classes more discriminative. The Sampler S and the later Discriminator D_2 is used for quality-verifying.

two fully connected layers, and the activation functions of the two fully connected layers are Leaky ReLU and Tanh respectively. The loss of the generator is defined as:

$$L(G) = -\mathbb{E}_{z\sim p_z}[D_1(\tilde{x})] + L_{cls}(\tilde{x}) - \mathbb{E}_{z\sim p_z}[D_2(\tilde{y}_{pse})] \tag{1}$$

where $\tilde{x} = G_\theta(T, z)$ is the generated feature based on the input text T and noise z, θ is the parameter of the generator. $\tilde{Y}$ is the low-dimensional features of $\tilde{X}$ after sampling, $\tilde{y}_{pse}$ represents the samples of pseudo features in $\tilde{Y}$. $\tilde{y}_{pse}$ is obtained by traversing the samples in $\tilde{Y}$, if the reconstruction error of a sample in $\tilde{Y}$ is greater than the threshold value of its class, then the sample is a pseudo feature. The first term is Wasserstein loss from discriminator D_1. The second term L_{cls} is the additional classification loss, it is the cross entropy loss between the classification result of the classifier in D_1 and the real class labels. The classifier is contained in the structure of discriminator D_1. The last term is also the Wasserstein loss which comes from discriminator D_2.

B. Discriminant Enhancement Module

The discriminant enhancement module mainly contains class metric constraint and ℓ_{12}-norm constraint.

1) Class Metric Constraint: The class metric constraint contains class central constraint and intra-class constraint, it is introduced to make the features of the same class closer and of different classes farther. The specific process is shown in Fig. 4.

Class Central Constraint: In order to encourage the generator to generate features that match real visual features, we propose class central constraint to facilitate this process. $X_{all} \in R^{n_{all}\times d_1}$ is the all real visual features in the dataset, where n_{all} is the number of samples and d_1 is the dimension. The specific implementation process is in Fig. 4: Since the real visual features are known from the dataset, we divide all of them into classes and calculate the class center for each class, the class center is calculated by summing and averaging samples of the same class. This process is equivalent to pre-processing in advance, taking the calculated class center(the black point) as the input of the model. Therefore, the class centers of the real features are fixed and reliable. Then during the training process, according to the samples sent by each batch, we calculate the generated features' class center of each class (the red point). For the class centers of the generated features and the real features, we hope that the class centers of the same class are closer and of different classes are further in Fig. 4 (b). Finally, we define class central constraint as:

$$\begin{aligned} L_e = \frac{1}{C}\sum_{c=1}^{C}(||\mathbb{E}_{\tilde{x}_c\sim p_g^c}[\tilde{x}_c] - \mathbb{E}_{x_c\sim p_{data}^c}[x_c]||^2 \\ - \sum_{j=1,j\neq c}^{C}||\mathbb{E}_{\tilde{x}_c\sim p_g^c}[\tilde{x}_c] - \mathbb{E}_{x_j\sim p_{data}^j}[x_j]||^2) \end{aligned} \tag{2}$$

where C is the number of seen classes, $\tilde{x}_c$ is the generated feature of class c which obeys conditional distribution p_c^g. x_c is the real feature of class c which obeys conditional distribution p_{data}^c . When calculating the class centers, we approximate it as $\mathbb{E}_{x_c\sim p_{data}^c}[x_c] = \frac{1}{N_c}\sum_{i=1}^{N_c} x_c^i$, where N_c is the number of real features of class c. Similarly, $\mathbb{E}_{x_j\sim p_{data}^j}[x_j] = \frac{1}{N_j}\sum_{i=1}^{N_j} x_j^i$. As for generated feature, it is approximated by averaging the generated high-dimensional vectors for class c, $\mathbb{E}_{\tilde{x}_c\sim p_g^c}[\tilde{x}_c] = \frac{1}{N_c^s}\sum_{i=1}^{N_c^s}\tilde{x}_c^i$, where $\tilde{x}_c^i = G(T_c, z_i)$ is the i-th generated feature of class c, and N_c^s is the number of generated features of class c.

Intra-Class Constraint: In Fig. 4 (b), the class center constraint only makes the class centers of the same classes closer and of the different classes farther, it does not consider the compactness of the distribution within the same class. Therefore, we introduce the intra-class constraint which is defined as:

$$L_{in} = \frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{N_c}(||\tilde{x}_c^i - E_{x_c\sim p_{data}^c}[x_c]||^2) \tag{3}$$

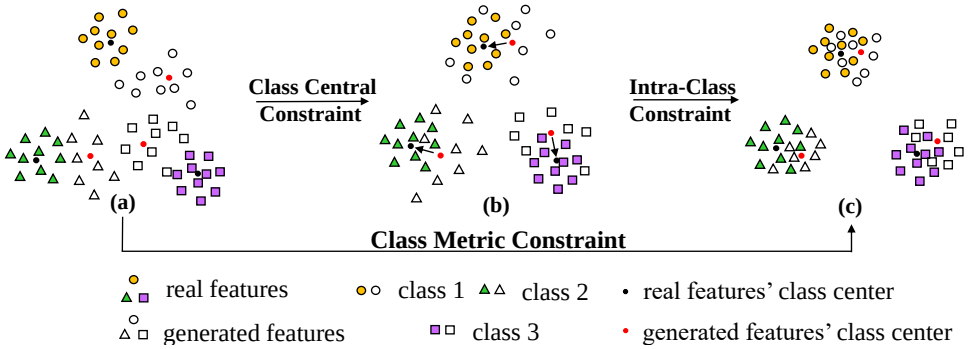


Fig. 4: The description of class metric constraint. It consists of class central constraint and intra-class constraint.

where the definitions of $\tilde{x}_c^i$ and $E_{x_c \sim p_{data}^c}[x_c]$ have been mentioned in Eq. (2). Through intra-class constraint in Fig. 4 (c), generated features of the same class are clustered near the center of its class and are closely distributed.

Therefore, class metric constraint is the sum of class central constraint and intra-class constraint.

$$L_{class} = L_e + L_{in} \tag{4}$$

2) ℓ_{12}-norm Constraint: The generated feature $\tilde{x}$ by generator may have a large amount of redundant information, which may result some features to be incorrectly classified. In order to make the features of different classes more discriminant in the distribution of feature space, we introduce the ℓ_{12}-norm constraint on the generated feature $\tilde{x}$. The generated feature $\tilde{x}$ by generator may have a large amount of redundant information, which results some features to be incorrectly classified. To make the learned representation more discriminative, we introduce the ℓ_{12}-norm constraint term on the generated representation feature matrix. The loss function can be shown as:

$$L_{12} = ||\tilde{X}||_{1,2}^2 + ||\tilde{X}^T||_{1,2}^2 \tag{5}$$

where $\tilde{X}$ is the generated feature, and $\tilde{X}^T$ is the transpose of $\tilde{X}$. The dimension of $\tilde{X}$ is the same as that of the real feature X, $\tilde{X} \in R^{n \times d_1}$, where n is the number of samples of a batch in the training process, and d_1 is the dimension of real visual features. $||\tilde{X}||_{1,2}$ is to want the rows to be sparse, that is, the samples are distributed only in their important feature dimensions, and are zero or small value in the other dimensions. However, $||\tilde{X}||_{1,2}$ can only be sparse in rows. In order to make classes more discriminant, we want the columns to be sparse as well, so on this basis, we add $||\tilde{X}^T||_{1,2}$.

The ℓ_{12}-norm is defined as follows:

$$||A||_{1,2} = \sqrt{\sum_{j=1}^{k} (\sum_{i=1}^{d} |a_{ij}|)}^2 \tag{6}$$

where A denotes a matrix, a_{ij} is the j-th element in the i-th row, d represents the number of rows and k represents the number of columns.

By minimizing Eq.(5), for a fixed class such as the j-th class, different elements in each projected data compete with each other, only a few elements corresponding to the j-th class will survive (nonzero) [23]. Thus, the learned generated features well preserve class-specificity distribution.

C. Discriminator D_1

The input of D_1 is the real visual feature X and the generated feature $\tilde{X}$. Both X and $\tilde{X}$ first pass through a full connection layer with ReLU activator function, and then through two branches of the discriminator: (i) one is the full connection layer for binary classifiers to distinguish between real (real visual feature x) and fake (generated feature $\tilde{x}$) input features. (ii) The other is full connection layer for n-way classifiers which is used to classify generated features into correct classes, this is also the classifier that we use to predict the features' labels. The loss of discriminator D_1 is:

$$\begin{aligned} L(D_1) = \mathbb{E}_{z \sim p_z}[D_1(\tilde{x})] - \mathbb{E}_{x \sim p_{data}}[D_1(x)] \\ + \lambda L_{GP} + \tfrac{1}{2}(L_{cls}(\tilde{x}) + L_{cls}(x)) \end{aligned} \tag{7}$$

where the first two terms approximate Wasserstein distance of the distribution of real features and generated features. The third term is the gradient penalty, which is used to avoid the pathological behavior in WGAN. In previous WGAN, the use of weight clipping in WGAN to enforce a Lipschitz constraint on the critic can lead to the problems of generating poor samples or fail to converge. Therefore, we use the L_{GP} to clipping eights: penalize the norm of gradient of the critic with respect to its input. This method of Lipschitz constraint can avoid the bad situations mentioned above, and it has been proved in the reference [12]. $L_{GP} = (||\nabla_{\hat{x}} D_1(\hat{x})||_2 - 1)^2$, where $\hat{x}$ is the linear interpolation of the real feature x and the generated feature $\tilde{x}$. For more information about L_{GP} can refer to reference [12]. The last two terms are classification losses of real feature x and generated features $\tilde{x}$ corresponding to class labels.

D. Sampler S

In the Sampler S, We use one of the most classical method which is Principal Component Analysis (PCA), and people also often use the idea of PCA to sample, such as Zhang et al. [46] use PCA to project samples into subspaces for learning. There are three main steps: we first calculate projection matrix W for each class, then calculate low-dimensional features $\tilde{Y}$ and Y by W, finally we calculate the reconstruction threshold

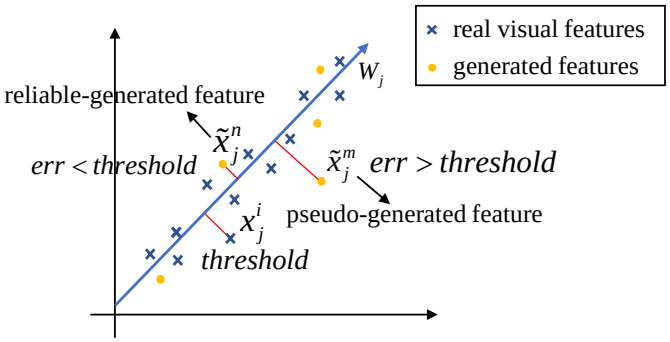


Fig. 5: Determine the projection direction and reconstruction threshold for the j-th class.

for each class from low-dimensional features Y. The specific steps are as follows:

Calculate projection matrix: The training sample consists of two parts, one is the real visual feature, the other is the generated feature. We make full use of prior knowledge, divide the real visual features $X \in R^{n \times d_1}$ into c classes, where n is the total number of visual feature X, and d_1 is the dimension of the features. We use PCA to calculate the projection matrix $W = \{W_1, W_2, \cdots, W_c\}$ of each class. For the sample $X_i \in R^{n_i \times d_1}$ of class i, each row is a sample, n_i is the number of samples X_i, and d_1 is the dimension. The specific process is as follows: (1) Calculate the mean of each of these rows in X_i, and then subtract the mean from each of these rows, the samples at this time are represented by X_i'. (2) Calculate the covariance matrix of X_i'. (3) Find the eigenvalue and eigenvector of the covariance matrix. (4) Sort the eigenvalues in order from large to small, select the largest d_2 of them, and then use the corresponding d_2 eigenvectors as column vectors to form the eigenvector matrix $W_i \in R^{d_1 \times d_2}$.

For the i-th class, the projection matrix W_i is a unique property of this class. When using PCA to reduce dimension, we choose an appropriate dimension d_2 with a contribution rate between 0.8 and 0.85 for each class, so as to reduce the complexity of subsequent processing. Then $W_i \in R^{d_1 \times d_2}$, for sample $X_i \in R^{n_i \times d_1}$ of the i-th class, $Y_i = X_i W_i \in R^{n_i \times d_2}$ is the feature after dimension reduction, where n_i is the number of the features of the i-th class.

Calculate low dimensional features: Through the above steps, we have calculated the projection matrix W of each class through the real visual features. During the training, the generated features $\tilde{X}$ and the real visual features X are paired and share the same label. We reduce the dimension of $\tilde{X}$ and X by the corresponding class of W, the features after dimension reduction are $\tilde{Y}$ and Y. Specifically, for sample x_j^i in X, it is the i-th sample of class j. Its corresponding low dimensional feature is $y_j^i = x_j^i W_j$.

Calculate reconstruction threshold: From PCA, we get $\tilde{Y}$ and Y. We calculate the reconstruction error through reconstructing the low-dimensional visual features Y back to the high-dimensional features X. Assume that there are m classes of samples sent in a batch ($n \leq c$), where c is the total number of classes in the training sample. For the features of the j-th class, there are N_j^s features $\{x_j^1, x_j^2, \cdots, x_j^{N_j^s}\}$. We calculate the reconstruction error for each feature separately. For feature x_j^i, the reconstruction error is $err_j^i = ||x_j^i - (x_j^i W_j W_j^T + \bar{x}_j^i)||^2$, where $(x_j^i W_j W_j^T + \bar{x}_j^i)$ is the reconstructed high-dimensional feature and $\bar{x}_j^i$ is the mean of the x_j^i, and the reason why adding $\bar{x}_j^i$ is because we're subtracting the mean when PCA reduced dimensions. For the j-th class, the reconstruction threshold is $err_j = \max_{1 \leq i \leq N_j^s} \{err_j^i\}$. This threshold represents the maximum error we can accept when reconstructing the real feature of the j-th class. As shown in Fig. 5, the blue points are the real visual features of the j-th class. We determined the projection direction W_j of this class (the blue line with arrows) based on these blue points. When the blue point is reconstructed back, x_j^i is the farthest point from the projection direction, and the distance represented by the red line of x_j^i is the reconstruction threshold of this class. The blue points are all real visual features, therefore, the threshold is the maximum range of the reconstruction error that we can accept. We use the threshold to distinguish reliable-generated features and pseudo-generated feature. In Fig. 5, The reconstruction error of $\tilde{x}_j^n$ is less than the threshold, it is considered to be reliable-generated feature, and the reconstruction error of $\tilde{x}_j^m$ is greater than the threshold, it is considered to be pseudo-generated feature.

It is important to calculate the respective reconstruction errors for the different classes of samples, as this is also a property specific to each class. As shown in TABLE I, we visualize the reconstruction thresholds of the first 40 classes of samples on the CUB dataset. As can be seen from TABLE I, reconstruction thresholds for different classes of samples have great differences, which also proves the significance of setting reconstruction thresholds according to different classes.

In addition to the PCA, we have also tried other sampling methods like auto-encoder. We have tried to make no distinction between classes, using an autoencoder to encode generated and real features into the low-dimensional space.

Algorithm 1: Training procedure of QVAN

Input: The images' real visual features X, matching texts T, corresponding labels, the random noise z, the batch size n, the maximal loops N, the iteration number of discriminators is $n_d = 5$, hyper-parameter $\lambda = 10$, the visual pivots $\{\bar{x}_c\}_{c=1}^C$, the projection matrix W and reconstruction threshold err calculated by PCA for each class.

for $iter = 1, 2, \cdots, N$ do
  for $t = 1, 2, \cdots, n_d$ do
    1: $\tilde{x} \leftarrow G_\theta(T, z)$
    2: Compute the loss $L(D_1)$ of discriminator using Eq. (7)
    3: Update the parameter w_1 of D_1
    4: Compute $\tilde{y}$ and y in low-dimensional space according to corresponding W of each class.
    5: For each $\tilde{y}$, calculate its reconstruction error and compared with the threshold of its class, then divided into reliable feature y_{rel} or pseudo feature y_{pse}.
    6: Compute the loss $L(D_2)$ of discriminator using Eq. (8)
    7: Update the parameter w_2 of D_2
  end for
  Compute the loss L_{class} of class central constraint using Eq. (4)
  Compute the loss L_{12} of $\ell 12$-norm Constraint using Eq. (5)
  $\tilde{x} \leftarrow G_\theta(T, z)$
  Compute the loss $L(G)$ of generator G using Eq. (1)
  Update the parameter θ of the generator G using $L(G) + L_{class} + L_{12}$
end for

However, in this way, we can't calculate the reconstruction threshold for each class, so we can't distinguish the reliable features and pseudo features in the generated features. Therefore, the effect is not as good as PCA. In order to calculate the reconstruction threshold for each class of samples, we have tried to use an autoencoder for each class of samples, but there are usually many classes of samples. This method will greatly increase the complexity of the network and is not feasible in practice. Therefore, we finally choose the PCA method, which can pre-process the real features in advance, so as to calculate the reconstruction threshold of each class of samples, without participating in the training process of the network, and also reduce the complexity of the network.

E. Discriminator D_2

The structure of the discriminator D_2 is roughly the same as that of D_1, except that the branch of the classifier is removed from D_1, and D_2 only contains the function of distinguishing true and fake features. The inputs of discriminator D_2 are generated features $\tilde{Y}$ and real visual features Y, which are low dimensional features corresponding to $\tilde{X}$ and X. The function of discriminator D_2 is to distinguish between $\tilde{Y}$ and Y. It consists of a full connection layer with ReLU activation function and a full connection layer for binary classifiers. As shown in Fig. 5, for the reliable-generated features, to avoid over-fitting and ensure intra-class diversity, we consider it to be real features, and for the pseudo-generated features, we consider them to be fake features, thus guiding the generator to produce more realistic and reliable features. The loss function of discriminator D_2 is the Wasserstein distance loss, which can be defined as:

$$\begin{aligned} L(D_2) = &-\mathbb{E}_{x\sim p_{data}}[D_2(y)] - \mathbb{E}_{x\sim p_{data}}[D_2(\tilde{y}_{rel})] \\ &+ \mathbb{E}_{z\sim p_z}[D_2(\tilde{y}_{pse})] + \lambda L_{GP} \end{aligned} \tag{8}$$

where $\tilde{y}_{rel}$ is the reliable features in $\tilde{Y}$, and $\tilde{y}_{pse}$ is the pseudo features in $\tilde{Y}$, and $\tilde{Y}_{rel} \cup \tilde{Y}_{pse} = \tilde{Y}$, $\tilde{Y}_{rel} \cap \tilde{Y}_{pse} = \phi$. By feeding back such supervision to the front, the generator is then enforced to maximize the discriminator loss, leading to better representations and classification results.

V. Experiments

A. Datasets

We adopt two types of zero-shot learning datasets, both of which contain samples of two modals of semantics and visual features. However, these two types of databases have different types of semantic representations, one is text description, the other is semantic attributes.

I) Datasets with text descriptions and images

CUB(Caltech UCSD Birds-2011) [38] contains 200 categories of bird species with a total of 11,788 images. Each species is associated with an article from Wikipedia [7] and organized according to scientific classification (order, family, genus, species). The samples in the CUB dataset consists of two parts, one is text description of the bird and the other is real visual features extracted from the images. As for text description, Term Frequency Inverse Document Frequency (TF-IDF) of word frequency [34] is used to represent, and the dimension of TF-IDF features for CUB is 7551.The dimension of visual features for CUB is 3584. We split the dataset using two different split settings [7], named Super Category-Shared (SCS) and Super-Category-Exclusive (SCE), in term of how close the seen classes are related to the unseen classes. In traditional zero-shot learning, seen classes are used for training and unseen classes are used for testing. For each unseen class in SCS, there exists one or more seen classes that belong to the same parent class. For example, both Coopers Hawk in the training set and Harriss Hawk in the testing set are under the parent category Hawks. On the contrary, in SCE, the parent categories of unseen classes are exclusive to those of the seen classes. Therefore, SCE is much harder than SCS as the relevance between seen and unseen classes is minimized. We follow both split settings to evaluate the capability of our approach. We divide CUB dataset into **CUB-SCS, CUB-SCE**. In CUB-SCS, there are 200 classes, 150 classes for training and 50 classes for testing. In CUB-SCE, there are 160 classes for training and 40 classes for testing.

NAB(North America Birds)[37] is a larger dataset of birds with 1011 classes and 48,562 images. The taxonomy for this dataset contains 1011 nodes, and the categories cover the most common North American birds. As for textual representation in NAB, The dimension of TF-IDF features is 13217, and the dimension of visual features is 3072. We use the split settings mentioned above to divide the NAB dataset into **NAB-SCS, NAB-SCE**. In NAB-SCS and NAB-SCE, there are 404 classes, 323 classes for training and 81 classes for testing.

II) Datasets with semantic attributes and images

CUB(Caltech UCSD Birds) [38] is a medium-size, fine-grained dataset with 312 attributes, containing 11,788 bird pictures and 200 categories, 150 classes for training and 50 classes for testing.

AWA1(Animals with Attributes 1)[19] is a medium-size, coarse-grained dataset with 85 attributes, containing 30,475 animal images in 50 categories, with 40 classes for training and 10 classes for testing.

AWA2(Animals with Attributes 2)[40]has the same 50 animal categories as AWA1, but all of its 37,332 images are obtained from public websites. The attributes and of AwA2 are defined in the same way as AwA1.

SUN(SUN Attribute)[27] is a medium-size, coarse-grained scene data set with 102 attributes, containing 14,340 images and 717 scene categories, of which 645 classes are used for training and 72 classes are used for testing.

FLO(Oxford Flowers)[26]consists of 8,189 images which comes from 102 flower categories. Each class consists of between 40 and 258 images. The images have large scale, pose and light variations.

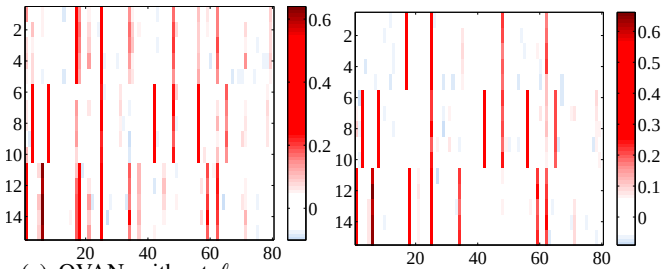
B. Superiority of Our Method

We conduct experiments on both types of datasets to demonstrate the superiority of our method. TABLE II and TABLE III show the performances of all of the comparison methods and our method.

The datasets in II contain text descriptions and images, TABLE II reports that our method has a better performance

TABLE I: In the CUB dataset, these are the reconstruction thresholds for the first 40 classes of samples.

Class	1	2	3	4	5	6	7	8	9	10
Threshold	473.16	529.30	616.47	492.14	536.58	571.68	540.71	514.00	437.69	856.08
Class	11	12	13	14	15	16	17	18	19	20
Threshold	450.25	627.89	525.04	603.51	627.95	727.74	687.07	796.45	529.08	648.81
Class	21	22	23	24	25	26	27	28	29	30
Threshold	629.30	568.79	568.79	568.79	419.02	594.62	356.61	604.04	353.19	329.01
Class	31	32	33	34	35	36	37	38	39	40
Threshold	483.92	514.25	465.53	515.26	648.15	932.79	886.62	462.01	403.38	401.06

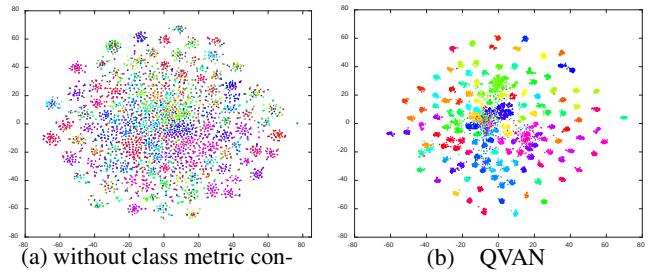


(a) QVAN without ℓ_{12}-norm constraint (b) QVAN

Fig. 6: Visualization of the generated features on CUB database. Each row is a sample, there are 3 classes, 5 samples per class.

TABLE II: Top-1 accuracy (%) on CUB and NAB datasets with two split settings, the datasets are the types that contain text descriptions and images.

Datasets	CUB		NAB	
Methods	SCS	SCE	SCS	SCE
WACZSL [6]	27.0	5.0	-	-
ESZSL [33]	28.5	7.4	24.3	6.3
SynC [1]	28.2	8.7	18.3	3.9
ZSLNS [31]	29.1	7.3	24.5	6.8
WACPVC [5]	33.4	7.6	11.5	6.2
ZSLPP [7]	37.3	9.7	30.3	8.1
f-CLSWGAN [41]	42.6	9.4	-	-
GAZSL [52]	43.7	10.3	35.6	8.6
QVAN	**45.82**	**14.84**	**37.52**	**9.96**



(a) without class metric constraint and quality-verifying network (b) QVAN

Fig. 7: Visualization of the generated features in the training process on CUB database, there are 150 classes, different colors represent different classes.

TABLE III: Top-1 accuracy (%) on different datasets, the datasets are the types that contain semantic attributes and images.

Methods	CUB	AWA1	AWA2	SUN	FLO
ESZSL [33]	53.9	58.2	58.6	54.5	51.0
SynC [1]	55.6	54.0	46.6	56.3	-
f-CLSWGAN [41]	57.3	68.2	-	60.8	67.2
DEM [22]	51.7	68.4	67.2	61.9	-
SP-AEN [3]	55.4	-	58.5	59.2	-
SR-GAN [44]	55.4	71.8	-	62.3	-
LisGAN [21]	58.8	70.6	-	61.7	69.6
GAZSL [52]	60.8	73.2	70.3	66.5	65.6
GMGN [36]	64.6	73.9	-	64.1	-
EBP [45]	72.4	74.4	73.4	-	83.7
SDGN [39]	74.9	-	-	68.4	81.8
QVAN	**75.5**	**74.8**	**74.2**	**68.8**	**84.0**

on both datasets. Compared with the GAZSL method, which has the current best classification performance, our method achieves 4.8%, 44.1% improvement on CUB with SCS and SCE splitting, and outperforms 5.3% and 15.8% on NAB with SCS and SCE splitting. For the SCE database lacking super category sharing, our method has a much higher accuracy than other methods, indicating that our method has greater advantages on the databases which are difficult to categorize.

The datasets in III contain semantic attributes and images, TABLE III reports that our method has a better performance on all datasets, which demonstrates the effectiveness of the proposed method.

TABLE IV: Effects of different components on zero-shot classification accuracy (%) on CUB and NAB datasets with SCS and SCE split setting

Datasets	CUB		NAB	
Ablation Study	SCS	SCE	SCS	SCE
QVAN Without L_{12}	43.20	13.97	35.82	9.01
QVAN Without L_{class}	24.85	6.76	26.07	5.241
QVAN Without S and D_2	41.97	9.82	34.35	8.21
QVAN	**45.82**	**14.84**	**37.52**	**9.96**

C. Ablation Study

In this section, we study the effectiveness of our three contributions which are ℓ_{12}-norm constraint, class metric constraint and quality-verifying network. Therefore, the ablation

experiments are carried out in this section to illustrate our contributions from the final classification accuracy and the feature visualization in the training process.

1) Classification Accuracy: TABLE IV shows the classification accuracy of different situations. In TABLE IV, Without L_{12} is that our method only removes ℓ_{12}-norm constraint, Without L_{class} means that our method only removes class metric constraint, and Without S and D_2 is that our method only removes the quality-verifying network. From TABLE IV, we can see that our contributions play important roles in the model, which can improve the classification accuracy of the model, especially on the databases which are difficult to categorize.

2) Feature Visualization: TABLE IV shows the role of our three contributions from the perspective of the final classification accuracy. Next, we visualize the features to further study the effectiveness of our three contributions. First, we study the role of the ℓ_{12}-norm constraint by visualizing features from the perspective of the feature space. Then, we use t-SNE method to visualize the overall distribution of features to study the role of class metric constraint and quality-verifying network.

Fig. 6 is the visualization of the generated features in the feature space on CUB-SCS database. Each row is a sample, there are 15 samples of 3 classes, 5 samples per class. Compared with Fig. 6 (a) and (b), when there is no ℓ_{12}-norm constraint, the distribution of features is a little chaotic. Through the ℓ_{12}-norm constraint, features are only distributed in some important dimensions, and the value in other dimensions is 0 or very small. This makes the features more discriminative among different classes, and thus more conducive to classification. In the meanwhile, we can see from Fig. 6 (a) and (b), ℓ_{12}-norm is a relatively loose constraint, which does not force the distribution of samples. Therefore, it does not limit the generator, which proves the reasonability of the ℓ_{12}-norm constraint.

Fig. 7 is the overall distribution of features on CUB-SCS database. There are 150 classes, each point is a sample, different colors represent different classes. From Fig. 7 (a), we can see that when there is no class metric constraint and quality-verifying network, the generated features are very messy. In Fig. 7 (b), when these two contributions are included in our model, the features generated in the training process can well simulate the distribution of real visual features. The distributions of the same class are compact and of the different classes are far apart.

Fig. 7 is to visualize the feature distribution from a macro perspective. In order to observe the effect of class metric constraint and quality-verifying network more clearly, we randomly selected 4 classes for specific research in Fig. 8.

In Fig. 8, the cross points are real visual features, their distribution can always be classified. As for generated features, Fig.8 (a) is the model without class metric constraint and quality-verifying network, the generated features are very messy. Compare Fig. 8 (b) and Fig. 8 (d), we can see that the class metric constraint can make the distributions of the same class closer and of different classes more discriminative. Compare Fig. 8 (c) and Fig. 8 (d), we can conclude that the quality-verifying network also plays a key role in the model. From quality-verifying network, the generated features are of better quality and closer to the real visual features.

TABLE V: AUSUC scores on CUB and NAB datasets with two split settings

Datasets	CUB		NAB	
Methods	SCS	SCE	SCS	SCE
ESZSL	0.185	0.045	0.092	0.029
ZSLNS	0.147	0.044	0.093	0.023
WACPVC	0.225	0.054	0.007	0.023
SynC	0.131	0.040	0.027	0.008
ZSLPP	0.304	0.061	0.126	0.035
GAZSL	0.354	0.087	0.204	0.058
QVAN	**0.396**	**0.122**	**0.242**	**0.065**

D. Extend to Generalized zero-shot learning

In conventional zero-shot learning, seen classes are used for training and unseen classes are used for testing. However, as the seen classes are often the most common objects, it is unrealistic to assume that we will never encounter them during the test phase[2]. Therefore, we can extend our method to the Generalized Zero-Shot Learning(GZSL). In this section, we first introduce the evaluation metric and then compare our method with the others.

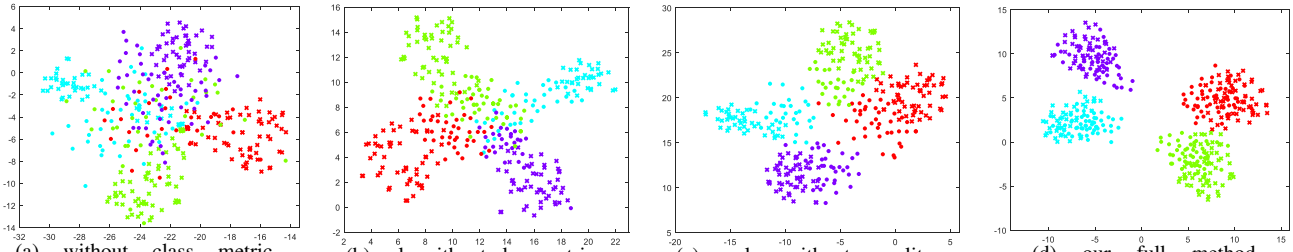
1) Evaluation Metric: Chao et al. [2] proposed a general metric for GZSL that involves classifying images of both seen classes S and unseen classes U into $T{=}S \cup U$. The accuracies are denoted as $A_{S\to T}$ and $A_{U\to T}$ respectively. They introduced a balancing parameter γ to draw Seen-Unseen accuracy Curve(SUC) and use Area Under SUC to measure the general capability of methods for ZSL. The higher the area is, the better an algorithm is able to balance $A_{S\to T}$ and $A_{U\to T}$.

2) Superiority of Our Method in GZSL: As can be seen from TABLE V, when extended to the GZSL, our model is still superior to the existing algorithm. Compared with the GAZSL method, which has the current best classification performance, our method achieves 11.9%, 40.2% improvement on CUB with SCS and SCE splitting, and outperforms 18.6% and 12.1% on NAB with SCS and SCE splitting.

VI. Discussion

A. Different Reconstruction Threshold Settings

In our model, the threshold of reconstruction error is determined by the farthest point of the real visual features. Here, we investigate the effect of different reconstruction threshold setting methods on the final performance. Therefore, the mean value of the reconstruction errors of real visual features is studied as the reconstruction threshold. We conducted experiments on CUB and NAB datasets, and the experimental results are shown in TABLE VI. QVAN-Threshold-max is the reconstruction threshold used in the method proposed in our paper, QVAN-Threshold-mean represents that we use the mean value of the reconstruction errors of real visual features. As can be seen from the experimental results in the TABLE VI,



(a) without class metric constraint and quality-verifying network (b) only without class metric constraint (c) only without quality-verifying network (d) our full method

Fig. 8: Visualization of the generated features and real visual features in the training process on CUB dataset, there are randomly selected 4 classes, Different colors represent different classes. The solid points of the circle represent generated features, and cross points represent real visual features.

QVAN-Threshold-max has a better effect. This may be because the furthest point of the real samples, as the reconstruction threshold, is also the maximum error range that we can accept, so it is more reasonable.

TABLE VI: Study on different reconstruction threshold setting methods on CUB and NAB datasets.

Datasets	CUB		NAB	
Different Thresholds	SCS	SCE	SCS	SCE
QVAN-Threshold-mean	45.65	14.52	37.20	9.15
QVAN-Threshold-max	**45.82**	**14.84**	**37.52**	**9.96**

B. Different Quality-Verifying Methods

In our model, in the process of quality-verifying, we project the high-dimensional features into the low-dimensional space, and take the maximum reconstruction error of the real features as the threshold value to judge the quality of the generated features. Here, we study different quality-verifying method, that is we only do quality-verifying in the high-dimensional space, we directly compare the generated features with the real features, then measure the difference between the two by the ℓ_1-norm. In each class, we take the maximum ℓ_1-norm difference between the real features as a threshold, indicating the maximum difference we can accept within the real range. If the difference between the generated feature and the corresponding real feature is greater than the threshold value, it indicates that the generated feature is of good quality, otherwise it is poor. This method is called QVAN-only high dim, and the experimental results are shown in TABLE VII.

TABLE VII: Study the different quality-verifying methods. The values in the table represent the final accuracy of the experiments.

Datasets	CUB		NAB	
Different Quality-Verifying	SCS	SCE	SCS	SCE
QVAN-only high dim	44.10	10.91	36.20	9.05
QVAN	**45.82**	**14.84**	**37.52**	**9.96**

As can be seen from the results in the TABLE VII, the results of quality-verifying only in high dimensional space are not as good as our method. This may be because the higher dimensional features contain a lot of noise and irrelevant terms, which will affect the results.

C. Different Sampling Dimensions

In our model, when using PCA to sample, we choose an appropriate dimension with a contribution rate between 0.8 and 0.85 for each class, so as to reduce the complexity of subsequent processing. Here, we study the different dimensions in the quality-verifying stage. The experimental results are shown in TABLE VIII, QVAN-rate 0.60-0.65 indicates that we choose the dimension with the contribution rate between 0.6 and 0.65, the same as below.

As can be seen from the TABLE VIII, with the increase of the contribution rate in the process of PCA dimension reduction, the overall accuracy rate presents a general upward trend. However, when the contribution rate is too high (greater than 0.9), it cannot play a good role in sampling. The contribution rates of 0.8 to 0.85 and 0.85 to 0.9 achieve good experimental results, which indicate that when the contribution rates are within this range, the low-dimensional features can not only remove irrelevant information and noise direction, but also preserve the structural information of features well.

TABLE VIII: Study the different quality-verifying methods. The values in the table represent the final accuracy of the experiments.

Datasets	CUB		NAB	
Different Contribution Rates	SCS	SCE	SCS	SCE
QVAN-rate 0.60-0.65	43.72	12.92	36.25	9.16
QVAN-rate 0.65-0.70	44.60	12.82	36.30	9.15
QVAN-rate 0.70-0.75	44.53	12.72	36.54	9.08
QVAN-rate 0.75-0.80	45.73	13.51	36.30	9.15
QVAN-rate 0.80-0.85	**45.82**	**14.84**	37.52	**9.96**
QVAN-rate 0.85-0.90	45.81	14.82	**37.62**	9.85
QVAN-rate 0.90-0.95	45.52	14.52	37.10	9.25

D. Discriminant performance of the model

In order to study the performance of the discriminator with the number of iterations, we conduct experiments on the CUB-SCS dataset. Fig. 9 shows the change of model accuracy as

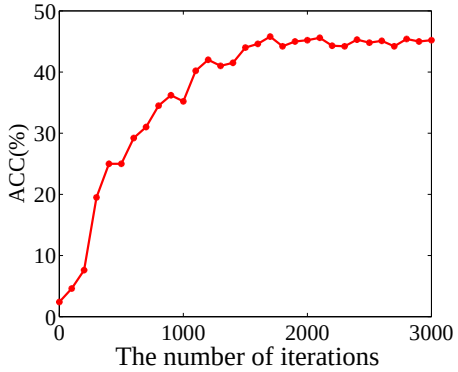


Fig. 9: On CUB-SCS dataset, the change curve of accuracy with the number of iterations.

the number of iterations increases. It can be seen that the classification accuracy increased rapidly at the beginning, and then gradually stabilized. This shows that with the increase of the number of iterations, the discriminant ability gradually increases, and then gradually becomes stable.

VII. Conclusions

In this paper, we present an effective quality-verifying adversarial network (QVAN) consisting of a generator and two discriminators, through which we can obtain reliable and stable generated features. In the meanwhile, we avoid overfitting and ensure intra-class diversity through setting the reconstruction threshold for each class. In order to consider both compactness and discriminability of generated features, we introduce the class metric constraint. And we use the ℓ_{12}-norm to fully consider the specific distribution among different classes, thus making the generated features more discriminant. Extensive experiments on several real-world datasets show the effectiveness of proposed approach, which demonstrate the advantage over the state-of-the-art methods, especially on the databases which are difficult to categorize. In the future, the idea of quality-verifying can also be applied to other GAN-based classification and clustering work.

References

[1] S. Changpinyo, W. Chao, B. Gong, and F. Sha. Synthesized classifiers for zero-shot learning. *IEEE CVPR*, pages 5327–5336, 2016.
[2] W. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. *IEEE CVPR*, 2016.
[3] L. Chen, H. Zhang, J. Xiao, W. Liu, and S.-F. Chang. Zero-shot visual recognition using semantics-preserving adversarial embedding networks. pages 1043–1052, 2017.
[4] J. Dong, B. Xiao, B. Ding, and H. Wang. Gt-gan: A general transductive zero-shot learning method based on gan. *IEEE Access*, PP(99):1–1, 2020.
[5] M. Elhoseiny, A. M. Elgammal, and B. Saleh. Write a classifier: Predicting visual classifiers from unstructured text. *IEEE TPAMI*, 39(12):2539–2553, 2017.
[6] M. Elhoseiny, B. Saleh, and A. M. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. *ICCV*, pages 2584–2591, 2013.
[7] M. Elhoseiny, Y. Zhu, H. Zhang, and A. M. Elgammal. Link the head to the "beak": Zero shot learning from noisy text description at part precision. *IEEE CVPR*, pages 6288–6297, 2017.
[8] R. Felix, B. G. V. Kumar, I. Reid, and G. Carneiro. Multi-modal cycle-consistent generalized zero-shot learning. *ECCV*, pages 21–37, 2018.
[9] J. Gao and C. Xu. CI-GNN: building a category-instance graph for zero-shot video classification. *IEEE Trans. Multim.*, 22(12):3088–3100, 2020.
[10] R. Gao, X. Hou, J. Qin, J. Chen, and L. Shao. Zero-vae-gan: Generating unseen features for generalized and transductive zero-shot learning. *IEEE TIP*, PP(99):1–1, 2020.
[11] I. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *NIPS*, pages 2672–2680, 2014.
[12] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *NIPS*, pages 5767–5777, 2017.
[13] Y. Guo, Q. Chen, J. Chen, Q. Wu, Q. Shi, and M. Tan. Auto-embedding generative adversarial networks for high resolution image synthesis. *IEEE TMM*, 21(11):2726–2737, 2019.
[14] Y. Guo, G. Ding, J. Han, and Y. Gao. Zero-shot learning with transferred samples. *IEEE TIP*, 26(7):3277–3290, 2017.
[15] H. Huang, C. Wang, P. S. Yu, and C. Wang. Generative dual adversarial network for generalized zero-shot learning. *IEEE CVPR*, 2018.
[16] H. Huang, C. Wang, P. S. Yu, and C. Wang. Generative dual adversarial network for generalized zero-shot learning. *IEEE CVPR*, pages 801–810, 2019.
[17] N. Karessli, Z. Akata, B. Schiele, and A. Bulling. Gaze embeddings for zero-shot image classification. *IEEE CVPR*, pages 6412–6421, 2017.
[18] D. Kong, R. Fujimaki, J. Liu, F. Nie, and C. Ding. Exclusive feature learning on arbitrary structures via $\ell_{1,2}$-norm. *NIPS*, 2:1655–1663, 2014.
[19] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE CVPR*, pages 951–958, 2009.
[20] J. Li, H. T. Huo, C. Li, R. Wang, and Q. Feng. Attentionfgan: Infrared and visible image fusion using attention-based generative adversarial networks. *IEEE TMM*, 2020.
[21] J. Li, M. Jing, K. Lu, Z. Ding, L. Zhu, and Z. . Huang. Leveraging the invariant side of generative zero-shot learning. In *IEEE CVPR*, pages 7402–7411, 2019.
[22] Z. Li, X. Tao, and S. Gong. Learning a deep embedding model for zero-shot learning. In *IEEE CVPR*, pages 3010–3019, 2017.
[23] D. Ming and C. Ding. Robust flexible feature selection via exclusive l21 regularization. *IJCAI*, pages 3158–3164, 2019.
[24] D. Ming, C. Ding, and F. Nie. A probabilistic derivation of lasso and l12-norm feature selections. *AAAI*, 33:4586–4593, 2019.
[25] S. Narayan, A. Gupta, F. S. Khan, C. G. M. Snoek, and L. Shao. Latent embedding feedback and discriminative features for zero-shot classification. In *ECCV*, volume 12367, pages 479–495, 2020.
[26] M. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008.
[27] G. Patterson and J. Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *IEEE CVPR*, pages 2751–2758, 2012.
[28] G. J. Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *International Journal of Computer Vision*, (5), 2017.
[29] G. J. Qi, W. Liu, C. Aggarwal, and T. S. Huang. Joint intermodal and intramodal label transfers for extremely rare or unseen classes. *IEEE TPAMI*, pages 1–1, 2016.
[30] G. J. Qi, L. Zhang, H. Hu, M. Edraki, J. Wang, and X. S. Hua. Global versus localized generative adversarial nets. In *IEEE CVPR*, pages 1517–1525, 2018.
[31] R. Qiao, L. Liu, C. Shen, and A. V. Den Hengel. Less is more: Zero-shot learning from online textual documents with noise suppression. *IEEE CVPR*, pages 2249–2257, 2016.
[32] S. Rahman, S. H. Khan, and N. Barnes. Deep0tag: Deep multiple instance learning for zero-shot image tagging. *IEEE Trans. Multim.*, 22(1):242–255, 2020.
[33] B. Romeraparedes and P. H. S. Torr. An embarrassingly simple approach to zero-shot learning. *ICML*, pages 2152–2161, 2015.
[34] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
[35] M. B. Sariyildiz and R. G. Cinbis. Gradient matching generative networks for zero-shot learning. *IEEE CVPR*, pages 2168–2178, 2019.
[36] M. B. Sariyildiz and R. G. Cinbis. Gradient matching generative networks for zero-shot learning. In *IEEE CVPR*, pages 2168–2178, 2019.
[37] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. H. Barry, P. Ipeirotis, P. Perona, and S. J. Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. *IEEE CVPR*, pages 595–604, 2015.
[38] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds200-2011 dataset. *California Institute of Technology*, 2011.

[39] J. Wu, T. Zhang, Z.-J. Zha, J. Luo, and F. Wu. Self-supervised domain-aware generative network for generalized zero-shot learning. In *IEEE CVPR*, pages 12764–12773, 2020.
[40] Y. Xian, C. H. Lampert, S. Bernt, and A. Zeynep. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. *IEEE TPAMI*, PP(99):1–1, 2017.
[41] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata. Feature generating networks for zero-shot learning. *IEEE CVPR*, pages 5542–5551, 2018.
[42] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. *IEEE CVPR*, pages 1316–1324, 2018.
[43] L. Yang, L. Li, S. Ling, F. Shen, and J. Han. From zero-shot learning to conventional supervised classification: Unseen visual data synthesis. In *IEEE CVPR*, 2017.
[44] Z. Ye, F. Lyu, L. Li, Q. Fu, J. Ren, and F. Hu. Sr-gan: Semantic rectifying generative adversarial network for zero-shot learning. *IEEE ICME*, pages 85–90, 2019.
[45] Y. Yu, Z. Ji, J. Han, and Z. Zhang. Episode-based prototype generating network for zero-shot learning. In *IEEE CVPR*, pages 14032–14041, 2020.
[46] L. Zhang, M. Edraki, and G. J. Qi. Cappronet: Deep feature learning via orthogonal projections onto capsule subspaces. *NIPS*, pages 5819–5828, 2018.
[47] L. Zhang, T. Xiang, and S. Gong. Learning a deep embedding model for zero-shot learning. *IEEE CVPR*, pages 3010–3019, 2017.
[48] X. Zhang, S. Gui, Z. Zhu, Y. Zhao, and J. Liu. Hierarchical prototype learning for zero-shot recognition. *IEEE Trans. Multim.*, 22(7):1692–1703, 2020.
[49] Z. Zhang, Y. Xie, and L. Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. *IEEE CVPR*, pages 6199–6208, 2018.
[50] L. Zhu, S. Kwong, Y. Zhang, S. Wang, and X. Wang. Generative adversarial network-based intra prediction for video coding. *IEEE TMM*, 22(1):45–58, 2020.
[51] P. Zhu, H. Wang, and V. Saligrama. Generalized zero-shot recognition based on visually semantic embedding. *IEEE CVPR*, 2018.
[52] Y. Zhu, M. Elhoseiny, B. Liu, X. Peng, and A. Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. *IEEE CVPR*, pages 1004–1013, 2018.
[53] Y. Zhu, J. Xie, B. Liu, and A. Elgammal. Learning feature-to-feature translator by alternating back-propagation for generative zero-shot learning. In *ICCV*, 2019.