Abstract—Zero Shot Learning (ZSL), a type of structured multi-output learning, has attracted much attention due to its requirement of no training data for target classes. Conventional ZSL methods usually project visual features into semantic space and assign labels by finding their nearest prototypes. However, this type of Nearest Neighbor Search (NNS) based methods often suffer from great performance degradation because of the non-uniform variances between different categories. In this paper, we propose a probabilistic framework by taking covariance into account to deal with the problem mentioned above. In this framework, we define a new latent space, which has two characteristics. The first is the features in this space should gather within classes and scatter between classes, which is implemented by triplet learning, the second is the prototypes of unseen classes are synthesized with nonnegative coefficients which are generated by Nonnegative Matrix Factorization (NMF) of relations between the seen classes and unseen classes in attribute space. During training, the learned parameters are the projection model for triplet network and the nonnegative coefficients between unseen classes and seen classes. In the testing phase, visual features are projected into latent space and assigned with the labels that have the maximum probability among unseen classes for classic ZSL or within all classes for Generalized ZSL. Extensive experiments are conducted on four popular datasets, and the results show that the proposed method can outperform the state-of-the-art methods in most circumstances.

Index Terms—Nonnegative Matrix Factorization (NMF), Triplet Network, Zero Shot Learning (ZSL), Prototype Synthesis.

I. INTRODUCTION

RECENT efforts on image classification research have been focusing on large-scale image recognition issues, such as the challenge on ImageNet [1]. Since the latest Convolutional Neural Network (CNN) based deep learning methods have achieved significant improvement and reached the accuracy of over 95% [2][3], a question is raised that are we already able to solve large-scale classification problems. This problem can be answered based on two conditions: 1) Can the training set include all classes over the world and contain enough training samples? 2) Can the training model be extended without retraining to the classes which are not included in the training set and still retain high performance? The first condition is almost impossible to be reached because there are 8.7 million classes only in animal species [4]. Therefore, many researchers try to fulfill the second requirement with Transfer Learning [5] and Zero Shot Learning (ZSL) [6][7].

ZSL aims to recognize the categories which have no labeled data available during training. This is usually realized by introducing auxiliary semantic information, such as attribute vectors [8] and word embeddings [9], which are often used as the prototypes of ZSL for final classification. To this end, ZSL usually learns to generate structured vectors in attribute space, which are then used to find the best prototypes as the test samples’ labels. From this perspective, ZSL can be regarded as a type of Structured Multi-output Learning (SML). During training, the relationship between visual features and semantic attributes is learned with only the data from seen classes. The prediction is conducted by directly applying the learned model on the data of unseen classes. The most popular ZSL methods are based on Nearest Neighbor Search (NNS), which projects visual feature into attribute space and find the nearest attribute...
vector as its label, or projects attribute prototypes into feature space as the feature prototypes which are then exploited for comparing with input visual features to find their class labels.

However, these NNS based methods often suffer from great performance degradation due to the neglect of variances between different classes. For example, in Fig. 1 there are two classes (denoted as blue and yellow) which have different variances, the new input feature (illustrated as a green square) is closer to the yellow class than the blue one. In the conventional NNS based methods, this new data should be classified as the yellow class, but in fact, it should belong to the blue class because it has a higher probability to be assigned with the blue label than the yellow one by considering the class variances.

Verma et al. first proposed a simple exponential framework for ZSL [10], [11], which treats each class distribution as an exponential family distribution, and the mean and the variance of each seen/unseen category are defined as a linear projection of the corresponding predefined class attribute. The projection is learned with only the seen classes and can be exploited to predict the parameters of the class-conditional distribution of each unseen class by utilizing the attributes of unseen classes. Then, the classification of the input unseen data can be conducted by finding the maximum probability in visual space. Wang et al. proposed an end-to-end Variational AutoEncoder (VAE) network based ZSL method (VZSL) [12], which represents each seen/unseen class using a class-specific latent-space distribution conditioned on class attributes. These latent-space distributions are utilized as a prior for a supervised VAE, which can facilitate to learn discriminative feature representations for the inputs. Although these two methods with the probabilistic framework can achieve significant improvement comparing to traditional NNS based methods, there still exist some problems that may hinder them to obtain better results:

1) This framework defines a linear projection from attribute space to feature space to fit the prototypes, which can be denoted as $\mu = W_{\mu} a_s$ and $\sigma^2 = W_{\sigma^2} a_s$, where $\mu$ and $\sigma^2$ are the mean and the variance of seen classes in visual feature space respectively, and they can also be considered as the prototypes of the seen classes. $a_s$ is the corresponding class level attribute, and $W_{\mu}$ and $W_{\sigma^2}$ are its projection parameters, and directly calculated by optimizing the Least Square Error (LSE) loss, that is to say, $\mu$ and $\sigma^2$ are weighted linear combination of attributes. However, there is no constraint for $W_{\mu}$ and $W_{\sigma^2}$, since direct optimization with LSE might lead to negative values for $W_{\mu}$ and $W_{\sigma^2}$, i.e., the synthesized unseen prototypes sometimes may come from negative attribute, which is unreasonable for realistic scenarios. Besides, no constraint for the projection parameters also leads to over-fitting, which obtains good results on training data, but leads to bad performance on test data.

2) In the equation $\sigma^2 = W_{\sigma^2} a_s$, $W_{\sigma^2}$ is a matrix, and $a_s$ is a vector, so the result $\sigma^2$ should be a vector. Thus, the covariance of the synthesized prototype can only be considered as $\text{diag}(\sigma^2_1, \cdots, \sigma^2_n)$, which means that the entries of the whole matrix except the diagonal are all zeros. It is known that the diagonal matrix assumes that the elements of the features are independent of each other, which is a strong assumption and ignores the correlation between attributes. VZSL [12] uses a nonlinear deep network to generate a diagonal covariance matrix, but the problem remains unresolved.

3) As it is known that the distribution of seen data is usually different from that of unseen data, thus using the model generated with only seen data to approximate the unseen data often leads to domain shift problem [13]. The methods proposed by Verma et al. [10] and Wang et al. [12] utilize only the seen data to compute the projection matrix or network parameters and apply them on unseen prototype synthesis, which will inevitably cause that problem.

4) Verma et al. [10] directly used the original visual features to compute the prototypes of seen classes. However, the distribution of these features often overlapped between different classes, which will be verified in the final experiments, i.e., many instances may have high probabilities to two or more classes, which will easily lead to the wrong classification. The method VZSL [12] also has such a problem. Therefore, to reduce the overlapped area of the data, it is necessary to find another space, where the processed samples are more discriminative.

To deal with these problems, in this paper, we propose a novel probabilistic method with Latent Nonnegative Prototypes Synthesis (LNPS) for unseen classes. In this method, we make three great efforts, the first one is the definition of a latent space, where the original features are projected into it by triplet learning, which can make the projected data in this space has less overlapped areas, and to be more discriminative, this effort can well solve the fourth problem mentioned above. Second, to mitigate the domain shift problem, we design a nonnegative combination model to synthesize the unseen prototypes from the seen data, and the nonnegative coefficients are generated with the relationship between the attributes of seen classes and unseen classes, which is illustrated in Fig. 2. For example, the prototype of ‘alpaca’, including the mean and the variance, is synthesized with the prototypes of ‘giraffe’, ‘deer’, ‘tiger’ and so on. This effort has built connections between the seen classes and the unseen classes, which can well alleviate the domain shift problem, and improve the performance on the more realistic Generalized ZSL (GZSL) setting significantly. Third, we make the prototype of each
category in the latent space as a Gaussian distribution, which consists of a mean vector and a variation matrix, and find the maximum probability of a test sample with respect to all the synthesized prototypes. Such effort constrains the prediction within a probabilistic framework in the latent space, thus can circumvent the problem caused by the NNS when class distributions are overlapping.

Our contributions are briefly listed as follows:

- We design a latent space for zero shot classification with triplet network. In this latent space, features belonging to the same category are gathered together, otherwise they are scattered among the classes, which can greatly reduce the overlapped areas, and make the projected data more discriminative.
- We build relationships between the seen classes and the unseen classes to solve the domain shift problem. In this manipulation, Nonnegative Matrix Factorization (NMF) is utilized to learn the nonnegative coefficients to generate the unseen attributes from the seen attributes, and then the coefficients are used to synthesize unseen prototypes from the seen prototypes in latent space.
- We build relationships between the seen classes and the unseen classes to solve the domain shift problem. In this manipulation, Nonnegative Matrix Factorization (NMF) is utilized to learn the nonnegative coefficients to generate the unseen attributes from the seen attributes, and then the coefficients are used to synthesize unseen prototypes from the seen prototypes in latent space.
- Extensive experiments are conducted on four popular datasets for both ZSL and GZSL, the results show that the proposed method can outperform the state-of-the-art methods in most circumstances, especially on the more realistic GZSL setting.

The main content of this paper is organized as follows: In section II we briefly introduce the existing methods for ZSL. Section III describes the proposed method in detail. Section IV gives the experimental results of comparison with existing methods for both conventional ZSL and GZSL. Finally in section V, we conclude this paper.

II. RELATED WORK

According to the usage of unseen data during training, ZSL methods can be coarsely divided into two categories: Inductive ZSL, which assumes that the unseen data should be strictly inaccessible during training, and Transductive ZSL, which utilizes the unlabeled data of unseen classes as part of the training set.

Inductive ZSL Since visual attribute learning [14] has been proposed, many researchers conduct their work to discover the intermediate attribute classifiers for zero-shot learning. One of the most popular frameworks is compatibility learning, which learns linear or non-linear mapping functions with only using seen data and attributes, and then applying on unseen data. DAP [15] is one of the earliest compatibility frameworks, which learns probabilistic attribute classifiers and estimates the label by integrating the ranks of the learned classifiers. ALE [16], SJE [17], and DEVISE [18] employ bilinear compatibility function to project features into semantic embedding space, where the features and attributes belong to the same class with depending on the correlation is maximal or minimal. Embarrassingly Simple Zero Shot Learning (ESZSL) [19] adds an additional regularization term to the unregularized risk minimization equation.

To improve the performance and reduce the usage of manual attributes, some hybrid methods are proposed, e.g. Combination of Semantic Embeddings (CONSE) [20] and Semantic Similarity Embedding (SSE) [21] exploits seen classes to construct the attributes of unseen classes.

Synthetic learning is a novel type of method, which typically synthesizes pseudo features from semantic attributes. The classifiers are trained by using conventional algorithms such as Decision Tree (DT) [22] or Support Vector Machine (SVM) [23]. Some well-known methods have a similar structure as the standard one. For example, Synthesised Classifiers (SYNC) [24], Unseen Visual Data Synthesis (UVDS) [8] and Generating Pseudo Feature Representation (GPFR) [24]. The most relevant method to ours is the exponential framework GFZSL [10, 11], which treats each class distribution as an exponential family distribution, and defines the mean and variance of visual prototype for each category as the projection of the class attribute. Besides, Wang et al. proposed an end-to-end VAE based VZSL method [12], which represents each seen/unseen class using a class-specific latent-space distribution conditioned on class attributes. These latent-space distributions are utilized as a prior for a supervised VAE, which can facilitate to learn discriminative feature representations for the inputs. However, both GFZSL and VZSL assume that all dimensions of the prototype variance are independent of each other. In addition, they may generate prototypes with negative coefficients for unseen classes, which will not happen in real feature representations.

Since the distribution of seen data often differs from that of the unseen data, thus just using the model generated with only seen data inevitably leads to the domain shift problem, i.e., if the projection model from visual feature to semantic embedding is learned only from the seen classes, the projection of unseen class image is likely to be shifted due to the bias distribution of the training seen classes. Sometimes this bias might be far away from the correct unseen class prototype, leading to an error of the subsequent nearest neighbor search. Therefore, the best way to solve this problem is to include unlabeled unseen data into training, which is called transductive ZSL.

The earliest concept of transductive ZSL was proposed by Y. Fu et al. [26], who learned a multi-label regression to generalize the model to unseen classes by utilizing both seen and unseen data. Semi-supervised framework [27] takes both labeled and unlabeled data as input, and jointly learns a multi-class classification model on all classes. The framework can consistently learn both the label representations and the model parameters across the seen classes and unseen classes. Unsupervised Domain Adaptation (UDA) [28] casts the visual-embedding projection learning problem as a sparse coding problem, which sets each dimension of the semantic embedding space corresponds to a dictionary basis vector. The coefficients/sparse code of each visual feature vector is its projection in the semantic embedding space. Y. Guo et al. [29] proposed a method to solve transductive ZSL with a shared model space (SMS) with replacing the shared attribute space in existing works. Recently, Y. Li et al. [30] exploits the intrinsic relationship between the semantic space manifold and the
III. METHODOLOGY

A. Problem Definition

Let \( \mathcal{S} = \{s_1, \cdots, s_p\} \) denotes a set of seen classes and \( \mathcal{U} = \{u_1, \cdots, u_q\} \) denotes a set of unseen classes, where \( p \) and \( q \) are the number of seen and unseen classes respectively.

The two sets are disjoint, that is to say, \( \mathcal{S} \cap \mathcal{U} = \emptyset \). Besides, \( \mathcal{A}^s = \{\alpha_1^s, \cdots, \alpha_p^s\} \in \mathbb{R}^{d_x \times p} \) and \( \mathcal{A}^u = \{\alpha_1^u, \cdots, \alpha_q^u\} \in \mathbb{R}^{d_u \times q} \) represent for the corresponding seen and unseen class level semantic representations, such as attributes or word embeddings, where \( d_x \) is the dimension of attribute vector.

Given a set of labeled training data of seen classes \( \mathbf{X}^s = \{x_1^s, \cdots, x_p^s\} \in \mathbb{R}^{d_x \times p_s} \), where \( p_s \) is the dimensionality of a single feature vector \( x_i^s \), and \( N_s \) is the number of training data. Each feature \( x_i^s \) is simultaneously associated with a label \( y_i \in \mathcal{S} \) and its corresponding attribute \( \alpha_{y_i} \in \mathcal{A}^s \). Let \( \mathbf{X}^u = \{x_1^u, \cdots, x_u^u, \cdots, x_{N_u}^u\} \in \mathbb{R}^{d_x \times N_u} \) represents for a set of test data, which is not assigned with its corresponding labels and semantic representations, where \( N_u \) is the number of test data. The objective of ZSL is to predict the labels of the test data \( \mathbf{X}^u \) by learning a classifier \( \mathcal{F} : \mathbf{X}^u \rightarrow \mathcal{U} \) with the training data \( \mathbf{X}^s \) and the whole attribute set \( \mathcal{A}^s \cup \mathcal{A}^u \).

B. The Probabilistic Framework

As the problems described in the section of the introduction, the NNS based methods might cause great performance degradation due to their neglect of data distribution, thus the classification should be determined in the probabilistic framework. Furthermore, for the sake of circumventing a large number of overlapping areas in original visual space, the features should be mapped into a latent space, where the features gather together within a class and spread out between classes. Therefore, for a feature vector \( x_i^u \) in original visual space, it should be first mapped into latent space as \( z_i^u \), and then the classification can be conducted with the multivariate Gaussian distribution,

\[
\begin{align*}
\mu_i^u &= \arg \max_{j \in \mathcal{U}} \mathcal{N}(z_i^u | \mu_j^u, \Sigma_j^u) \\
&= \arg \max_{j \in \mathcal{U}} \frac{1}{(2\pi)^{\frac{d_u}{2}} |\Sigma_j^u|^\frac{1}{2}} \exp\left(-\frac{1}{2}(z_i^u - \mu_j^u)^T(\Sigma_j^u)^{-1}(z_i^u - \mu_j^u)\right),
\end{align*}
\]

(1)
where, $\mu^u_i$ and $\Sigma^u_i$ are the mean vector and covariance matrix respectively of each unseen prototype, and $d_z$ is the dimensionality of the vector in latent space.

According to Eq. 1 there are two things left to complete for this framework, one is to find a projection function $G: X^u \rightarrow Z^u$ to map the original features into latent space, where the projected features are more discriminative; another is to synthesize the Gaussian distribution parameters $\mu^u_i$ and $\Sigma^u_i$ of each unseen prototype in latent space.

For the former one, the best way is adopting the triplet based metric learning method, thus, we exploit the labeled training data $X_s$ and their corresponding label $S$ to generate triplets, and learn a mapping function $G': X^s \rightarrow Z^s$, which is then used as $G$ when computing $Z^u$. For the latter one, since we have already obtained the seen attributes $A^s$ and the unseen attributes $A^u$, it is feasible to compute nonnegative coefficients $H \in \mathbb{R}^{s \times u}$ ($H_{ij} > 0$) to synthesize $A^u$ from $A^s$, which can be addressed by applying Non-negative Matrix Factorization (NMF). Furthermore, if $Z^s$ has been generated with the previous manipulation, it is easy to obtain the mean vector $\mu^u_i$ and covariance matrix $\Sigma^u_i$ of each seen prototype in latent space. With the precomputed $H$, calculating the distribution of unseen prototypes will be an easy operation.

The whole framework is illustrated in Fig. 3 and the details are clarified in the following several subsections.

### C. Triplet Network

Since the data distribution of each class in visual feature space is often overlapped with each other, which makes the features difficult to be classified, it is necessary to define a latent space to project the original features into it, where the data points are more discriminative. The latent space should have two characteristics, one is the data should be gathered within a class and scattered between classes, another is to avoid the curse of dimensionality, the data points in latent space should have less dimensionality than that in original feature space. The best way to fulfill the above two requirements is metric learning [34] [35], such as Siamese network and triplet network.

Siamese network exploits pairwise samples, including similar and dissimilar pairs, it encourages the network to pull together similar pairs and push away dissimilar pairs, while triplet network uses triplets, it conducts the same work as the Siamese network within one sample. Lots of experiments have proved that the triplet network is better than the Siamese network in most situations. Therefore, in our method, we adopt the triplet network. There are three important elements in the triplet network, including objective function, triplet selection, and network architecture, which are described details as follows.

#### Objective

The latent embedding is denoted as $z = G'(x)$, which embeds a visual feature $x$ into a low dimensional latent space. Here we intend to ensure that a visual feature $x_a^i$ (anchor) of a special class is closer to the feature $x_p^i$ (positive) of the same class than it is to any feature $x_n^i$ (positive) of any other class.

Since the triplet model in FaceNet [36] has achieved great success in face recognition, we adopt the concept and define,

$$\|G'(x_a^i) - G'(x_p^i)\|^2 + \alpha < \|G'(x_a^i) - G'(x_n^i)\|^2,$$

where, $\alpha$ is a margin, and $(x_a^i, x_p^i, x_n^i)$ is a triplet sample. Therefore, the objective loss function can be defined as,

$$L = \sum_{i=1}^{N_s} \max(0,\|G'(x_a^i) - G'(x_p^i)\|^2 - \|G'(x_a^i) - G'(x_n^i)\|^2 + \alpha).$$

#### Triplet Selection

Choosing suitable triplets to use is very important for achieving good performance. Inspired by FaceNet [36], we adopt the semi-hard mining strategy to select triplets, which is described as,

$$\|G'(x_a^i) - G'(x_p^i)\|^2 < \|G'(x_a^i) - G'(x_n^i)\|^2 < \|G'(x_a^i) - G'(x_p^i)\|^2 + \alpha.$$
zero, which is unreasonable for feature synthesis and often leads to over-fitting.

To circumvent these problems, we build a connection between the attributes of seen classes and unseen classes, and make the nonnegative constraint on the parameters, which can be represented as,

\[ A^u = A^s H \]

\[ \text{s.t. } H_{ij} \geq 0, \]  

where, \( H \) is the synthesis coefficient matrix, and \( H_{ij} \) is the entry of \( H \) in row \( i \) and column \( j \).

**Optimization**

Due to the nonnegative constraint, Eq. 5 is not convex, thus, it cannot be solved with a closed-form solution. Here, we adopt the concept of NMF to minimize the LSE loss of each entry, and define the following loss function,

\[ J(H) = \frac{1}{2} \sum_{i,j} (A_{ij}^s - (A^s H)_{ij}). \]  

(6)

Since \((A^s H)_{ij}\) can be represented as,

\[ (A^s H)_{ij} = \sum_k A_{ik}^s H_{kj}, \]  

(7)

thence,

\[ \frac{\partial (A^s H)_{ij}}{\partial H_{kj}} = A_{ik}^s. \]  

(8)

We make derivative of Eq. 6 with respect to \( H_{kj} \), and obtain,

\[ \frac{\partial (J(H))}{\partial H_{kj}} = \sum_i A_{ik}^s ((A^s H)_{ij} - A_{ij}^u) \]

\[ = \sum_i A_{ik}^s (A^s H)_{ij} - \sum_i A_{ik}^s A_{ij}^u \]

\[ = ((A^s)^T A^s H)_{kj} - ((A^s)^T A^u)_{kj}. \]  

(9)

Then we can use the Gradient Descent (GD) to compute \( H_{kj} \), which can be denoted as,

\[ H_{kj} = H_{kj} - \lambda \cdot ((A^s)^T A^s H)_{kj} - ((A^s)^T A^u)_{kj}, \]  

(10)

where, \( \lambda \) is the learning rate.

For Eq. 10, we set,

\[ \lambda = \frac{H_{kj}}{((A^s)^T A^s H)_{kj}}, \]  

(11)

then, it can be simplified as,

\[ H_{kj} = H_{kj} \cdot \frac{((A^s)^T A^u)_{kj}}{((A^s)^T A^s H)_{kj}}. \]  

(12)

Eq. 12 is a typical iterative process, which can guarantee \( H_{kj} \) is nonnegative during optimization, and the convergence proof can be found in [37].

**Synthesis**

Due to the fact that the class attributes annotated by human experts are mostly according to their visual appearance, thus we assume that the distribution of interclass similarity in visual and attribute spaces are consistent. Furthermore, the triplet network is applied to reduce the intraclass variance so as to further alleviate interclass similarity shift. To verify our assumption, the normalized similarity matrices of the seen classes of AWA in both latent space and attribute space are shown in Fig. 5. Yellow indicates high similarity and blue is the opposite. It is clear that the distributions of interclass similarity in latent and attribute spaces are consistent. Since Coefficient Matrix \( H \) aims to capture the interclass similarities between unseen and seen classes, but the latent feature of unseen classes is not available before the ZSL test, we can estimate it from attribute space and apply it to the latent feature space as an approximate alternative.

In the last subsection, we have obtained the projection function \( G \) from visual space to latent space, thus, it is easy to get the projected features \( Z^v \) and \( Z^a \), which are then exploited to compute the seen prototypes,

\[ \left\{ \begin{array}{ll}
\mu_c^s = \sum_i z_i^s \cdot 1(\ell(z_i^s) == c), \quad & \text{where } c \in S \\
\Sigma_c^s = \frac{1}{N_c} \sum_i ((z_i^s - \mu_c^s)(z_i^s - \mu_c^s)^T) \cdot 1(\ell(z_i^s) == c), \quad & \end{array} \right. \]  

(13)

where, \( \ell(z_i^s) \) is the label function of \( z_i^s \), and \( 1(\cdot) \) is the indicator function when the condition is satisfied the output is 1, otherwise 0. \( N_s \) is the number of all seen data, and \( N_c = \sum_i 1(\ell(z_i^s) == c) \) denotes the number of features falling into the \( c^{th} \) category.

Similar to the attribute synthesis, we exploit the same coefficient matrix \( H \) to synthesize the unseen prototypes in latent space,

\[ \left\{ \begin{array}{ll}
\mu_c^u = \sum_i z_i^u \cdot H_{ic} \quad & \\
\Sigma_c^u = \left( \sum_i H_{ic} \right) \cdot \left( \sum_i \frac{H_{ic}}{\Sigma_i} \right)^{-1}, \quad & \end{array} \right. \]  

(14)

where, \( c \in U \). Given an input test data \( x_i^u \), we can map it into latent space with the function \( G(\cdot) \), which is learned with triplet network described in last subsection, and get \( z_i^u = G(x_i^u) \). According to Eq. 1 it is easy to obtain the label of the new unseen data \( x_i^u \).

**E. Transductive Setting**

In standard ZSL setting, the parameters to estimate the unseen classes are learned only from the data of seen classes, and this setting is often called **Inductive Setting**. But in realistic scenarios, the distribution of unseen data often differs from
that of seen classes, which will lead to great performance
degradation due to the domain shift problem. In our proposed
method, we exploit the unseen attributes in the training phase
to creatively construct the relationship between seen classes
and unseen classes, which can alleviate the domain shift
problem to a certain degree. However, only one attribute
vector of each unseen class cannot thoroughly capture the
distribution, thus it is unable to solve the domain shift problem
perfectly.

Sometimes, we may have the opportunity to access the
unlabeled data from unseen classes. Therefore, we can include
them in the training phase to obtain its real distribution, which
will definitely improve the performance. This setting is named
Transductive Setting. In our work, the objective is to leverage
the unlabeled unseen data to further improve the estimation
\{\mu^u_c, \Sigma^u_c\}, c \in \mathcal{U} upon the inductive results.

Suppose the given unlabeled data of each category follows
the Gaussian distribution independently, the entire data set can
be modeled with a Gaussian Mixture Model (GMM) [38],
which can be represented as,
\[ p(z^u_i) = \sum_{k=1}^{q} \pi_k N(z^u_i | \mu^u_k, \Sigma^u_k), \]
where, \( \pi_k \) is the mixing coefficient, and follows two con-
straints, which are 0 \( \leq \pi_k \leq 1 \) and \( \sum_{k=1}^{q} \pi_k = 1 \). The log of
the likelihood function of whole dataset of the unseen classes
is given by,
\[ \ln p(z^u_i | \pi, \mu^u, \Sigma^u) = \sum_{n=1}^{N_u} \ln \sum_{k=1}^{q} \pi_k N(z^u_i | \mu^u_k, \Sigma^u_k). \]

Maximizing the formulation can be addressed by an eleg-
ant and powerful method, namely Expectation-Maximization
(EM) algorithm [38], which can be expressed as the following
procedure,

- Initialize the mean vectors \( \mu^u_k, k \in \mathcal{U} \) and covariance
  matrices \( \Sigma^u_k, k \in \mathcal{U} \) with the synthesized results from
  inductive setting in last subsection. Initialize the mixing
  coefficients \( \pi_k = \frac{1}{q} \), and evaluate the initial value of the
  log likelihood with Eq. [16]
- **E-step.** Estimate the expected values using the current
  parameters,
  \[ \gamma(\ell_{nk}) = \frac{\pi_k N(z^u_n | \mu^u_k, \Sigma^u_k)}{\sum_{j=1}^{q} \pi_k N(z^u_n | \mu^u_j, \Sigma^u_j)}, \]
  where, \( \ell_{nk} \) means assigning the \( n^{th} \) data point with the
  \( k^{th} \) label.
- **M-step.** Re-evaluate the parameters using the current
  expected values,
  \[ \mu^u_k = \frac{1}{N_k} \sum_{n=1}^{N_u} \gamma(\ell_{nk}) z^u_n, \]
  \[ \Sigma^u_k = \frac{1}{N_k} \sum_{n=1}^{N_u} \gamma(\ell_{nk})(z^u_n - \mu^u_k)(z^u_n - \mu^u_k)^T, \]
  \[ \pi_k = \frac{N_k}{N_u}, \]

where,
\[ N_k = \sum_{n=1}^{N_u} \gamma(\ell_{nk}). \]

- Evaluate the log-likelihood with Eq. [16] and check for
  convergence of either the log-likelihood or the parameters.
  If the convergence criterion is not satisfied, return to
  E-step.

After the EM step, we can leverage the converged para-
eters and Eq. [1] to predict the label of new input unseen features.

\section*{F. Computational Complexity}

In this subsection, we discuss the computational complexity
of our method. Since our method mainly consists of a triplet
network, prototype synthesis, and GMM, we analyze them
separately. For the triplet network, we assume the neurons
in each layer is \( n \) for simplicity. Since the forward process
mainly contains the matrix multiplication, the computational
complexity for the forward network is \( O(\ell n^3) \), where \( \ell \) is
the number of layers. The backpropagation part mainly contains
error propagation and gradient computation, complexity of the
error propagation in all layers is \( O(\ell n^3) \) too, and if we assume that
there are \( n \) gradient iterations, the total complexity of
backpropagation is \( n \times O(\ell n^3) = O(\ell n^4) \). Combing both
forward and backward processes, we can conclude the com-
putational complexity of the triplet network is \( O(\ell n^4) \) in a single
iteration. For prototype synthesis, the computation of NMF
mainly contains matrix multiplication and matrix element-
wise division. Since the dimension of the attribute is \( a_d \), and
if we assume the iteration number is \( m \), the computational
complexity of prototype synthesis is \( O(m ad^3) \). For the process
of EM, the main computation of E-step is the probability
computation, which has the complexity of \( O(iq\beta N_u) \), where,
\( i \) is the iteration number for EM. The main computation
of M-step is the calculation of the covariance, which has
the complexity of \( O(i\beta^2 N_u) \). Therefore, the computational
complexity of GMM is \( O(i\beta(i + q)N_u) \).

\section*{IV. EXPERIMENTS}

To verify the effectiveness of our method, we use four
popular datasets to evaluate our model for ZSL, and compare
it with a number of state-of-the-art baselines. We conduct our
experiments on two settings, including inductive setting and
transductive setting, and report the results on classical ZSL
and GZSL respectively. In this section, we will first briefly
introduce the four datasets, and then show the performance
of our method comparing to some baselines, at last, the detailed
analysis will be demonstrated to show the importance of some
hyper-parameters.

\subsection*{A. Datasets and Settings}

\textbf{Datasets}

Similar to many other ZSL methods [33, 41, 42], we also
use the same four popular datasets. For the sake of fair
comparison, we leverage the same split like that in [33], and
the statistics of the datasets are summarized in Tab. I. The
description of the datasets is as follows,
### Table I
The summarization of the four popular datasets used in our experiments.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Attribute Dim</th>
<th>Samples</th>
<th>Seen/Unseen Class</th>
<th>Test samples (unseen)</th>
<th>Test samples (seen)</th>
<th>Train (seen)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN [39]</td>
<td>102</td>
<td>14,340</td>
<td>645/102</td>
<td>1,440</td>
<td>2,580</td>
<td>10,320</td>
</tr>
<tr>
<td>CUB [10]</td>
<td>512</td>
<td>11,788</td>
<td>150/50</td>
<td>2,967</td>
<td>1,764</td>
<td>7,057</td>
</tr>
<tr>
<td>AWA [15]</td>
<td>35</td>
<td>30,475</td>
<td>40/10</td>
<td>5,685</td>
<td>4,958</td>
<td>11,788</td>
</tr>
<tr>
<td>aPY [40]</td>
<td>64</td>
<td>15,339</td>
<td>20/12</td>
<td>7,924</td>
<td>1,483</td>
<td>5,932</td>
</tr>
</tbody>
</table>

- **SUN attribute (SUN):** The SUN dataset [39] contains 14,340 images with 645 seen classes (training set) and 72 unseen classes (test set). There are 20 images in each class, which is also associated with a 102-dimensional real value class-attribute vector. Among the seen classes, we select 4 images from each class to build the seen test set, and the left is composed of the training set of seen classes.

- **Caltech-UCSD Birds-200-2011 (CUB-200):** The CUB-200 [43] is a fine-grained dataset, which contains 11,788 images with 150 seen classes (training set) and 50 unseen classes (test set). Each image has a real value 312-dimensional class-attribute vector, specifying the presence or absence of various attributes of that image. The attribute vectors for all images in a class are averaged to construct the continuous class-attribute vector. Besides, the data of seen classes are divided into two parts, one containing 7,057 images is used for seen train, and the other containing 1,764 images is employed for the seen test.

- **Animal with Attribute (AwA):** The AwA dataset is a coarse-grained dataset [15], which contains 30,475 images with 40 seen classes (training set) and 10 unseen classes (test set). Each class has a predefined real-value 85-dimensional class-attribute vector. The set of unseen classes has 5,685 images, and the set of seen classes has 24,790 images, among which 4,958 images are used for the seen test, and the remaining is used for training.

- **a Pascal & Yahoo attribute (aPY):** aPY [40] is also a coarse-grained dataset, which contains 15,339 images with 32 classes, and 20 of them (Pascal set) are used as seen classes and the left 12 (Yahoo set) are treated as unseen classes. Besides, each class is associated with a 64-dimensional attribute. Among the data of 20 seen classes, 5,932 images are used for seen train, and the left 1,438 are for the seen test.

### Table II
The optimal dimensions of latent space for triplet network.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SUN</th>
<th>CUB</th>
<th>AWA</th>
<th>aPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZSL</td>
<td>102</td>
<td>512</td>
<td>256</td>
<td>32</td>
</tr>
<tr>
<td>GZSL</td>
<td>1024</td>
<td>512</td>
<td>256</td>
<td>32</td>
</tr>
</tbody>
</table>

### B. Comparison with Baselines

Conventional ZSL metric often focuses on the average accuracy of all test data, but it has a drawback that if there is a big class occupies more than 50% of the data, then the result will be determined only by this class, such as the class ‘person’ in aPY contains over 60% of the dataset, thus the class ‘person’ will dominate the entire result. However, the purpose of ZSL is to achieve good performance on all unseen categories, so it is better to compute the accuracies in each class, and then average them as the final result.

Besides, the ZSL metric assumes that the test data in advance are known belonging to unseen classes, and will be tested only on unseen classes, which is unreasonable in realistic scenarios. We usually do not know the ascription of the test data in advance, thus it is necessary to find the best assignment on both seen and unseen classes. Furthermore, the model should be not only suitable for unseen classes but also should maintain the performance on seen classes. This metric is called Generalized ZSL (GZSL), which is described as follows,

- **Seen test accuracy tr:** Average per-class classification accuracy for seen test samples;
- **Unseen test accuracy ts:** Average per-class classification accuracy for unseen test samples;
- **Harmonic accuracy H:** traditional arithmetic mean $H = (tr + ts)/2$, which computes the average value of $tr$ and $ts$, can still generate good results when one of $tr$ and $ts$ is high and the other is very low. However, very low accuracy on single metric often means the trained model fails, thus here we use harmonic accuracy $H = (2 × tr × ts)/(tr + ts)$ [33] to replace the arithmetic mean.

We compare our algorithm with 20 recently proposed inductive and transductive methods. The inductive methods include DAP [15], CONSE [20], CMT [44], SSE [21], LATEM [45], ALE [16], DEVISE [18], SJE [17], ESZSL [19], SYNC [24], SAE [46] CVAEZSL [47], PRESERVE model [48].
TABLE III
Comparison with state-of-the-art baselines on GZSL setting. '-' means not reported or not available.

<table>
<thead>
<tr>
<th>Method</th>
<th>SUN tr</th>
<th>SUN H</th>
<th>CUB tr</th>
<th>CUB H</th>
<th>AWA tr</th>
<th>AWA H</th>
<th>aPY tr</th>
<th>aPY H</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAP [15]</td>
<td>4.2</td>
<td>25.1</td>
<td>7.5</td>
<td>1.6</td>
<td>67.9</td>
<td>3.3</td>
<td>0.0</td>
<td>88.7</td>
</tr>
<tr>
<td>CONSE [20]</td>
<td>6.8</td>
<td>39.9</td>
<td>11.6</td>
<td>1.6</td>
<td>72.2</td>
<td>3.1</td>
<td>0.4</td>
<td>88.6</td>
</tr>
<tr>
<td>CMT [44]</td>
<td>8.1</td>
<td>21.8</td>
<td>11.8</td>
<td>7.2</td>
<td>49.8</td>
<td>12.6</td>
<td>0.9</td>
<td>87.6</td>
</tr>
<tr>
<td>SSE [21]</td>
<td>2.1</td>
<td>36.4</td>
<td>4.0</td>
<td>8.5</td>
<td>46.9</td>
<td>14.4</td>
<td>7.0</td>
<td>80.5</td>
</tr>
<tr>
<td>LATEM [45]</td>
<td>14.7</td>
<td>28.8</td>
<td>19.5</td>
<td>15.2</td>
<td>57.3</td>
<td>24.0</td>
<td>7.3</td>
<td>71.7</td>
</tr>
<tr>
<td>ALE [16]</td>
<td>21.8</td>
<td>33.1</td>
<td>26.3</td>
<td>23.7</td>
<td>62.8</td>
<td>34.4</td>
<td>16.8</td>
<td>76.1</td>
</tr>
<tr>
<td>DEVISE [18]</td>
<td>16.9</td>
<td>27.4</td>
<td>20.9</td>
<td>23.8</td>
<td>53.0</td>
<td>32.8</td>
<td>13.4</td>
<td>68.7</td>
</tr>
<tr>
<td>SJE [17]</td>
<td>14.7</td>
<td>30.5</td>
<td>19.8</td>
<td>23.5</td>
<td>59.2</td>
<td>33.6</td>
<td>11.3</td>
<td>74.6</td>
</tr>
<tr>
<td>ESZSL [19]</td>
<td>11.0</td>
<td>27.9</td>
<td>15.8</td>
<td>12.6</td>
<td>63.8</td>
<td>21.0</td>
<td>6.6</td>
<td>75.6</td>
</tr>
<tr>
<td>SAE [46]</td>
<td>8.8</td>
<td>18.0</td>
<td>11.8</td>
<td>7.8</td>
<td>54.0</td>
<td>13.6</td>
<td>1.8</td>
<td>77.1</td>
</tr>
<tr>
<td>SYNC [24]</td>
<td>7.0</td>
<td>43.3</td>
<td>13.4</td>
<td>11.5</td>
<td>70.9</td>
<td>19.8</td>
<td>8.9</td>
<td>87.3</td>
</tr>
<tr>
<td>CVAE-ZSL [47]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>47.2</td>
<td>-</td>
</tr>
<tr>
<td>PRESERVE [48]</td>
<td>23.7</td>
<td>26.7</td>
<td>54.3</td>
<td>33.9</td>
<td>-</td>
<td>-</td>
<td>13.5</td>
<td>51.4</td>
</tr>
<tr>
<td>CDA [49]</td>
<td>21.5</td>
<td>34.7</td>
<td>26.5</td>
<td>23.5</td>
<td>55.2</td>
<td>32.9</td>
<td>28.1</td>
<td>73.5</td>
</tr>
<tr>
<td>GFZSL [10]</td>
<td>0.0</td>
<td>39.6</td>
<td>0.0</td>
<td>0.0</td>
<td>45.7</td>
<td>0.0</td>
<td>1.8</td>
<td>80.3</td>
</tr>
<tr>
<td>LAGO [50]</td>
<td>18.8</td>
<td>33.1</td>
<td>23.9</td>
<td>21.8</td>
<td>73.6</td>
<td>33.7</td>
<td>23.8</td>
<td>67.0</td>
</tr>
<tr>
<td>PSEUDO [51]</td>
<td>19.0</td>
<td>32.7</td>
<td>24.0</td>
<td>23.0</td>
<td>51.6</td>
<td>31.8</td>
<td>22.4</td>
<td>80.6</td>
</tr>
<tr>
<td>KERNEL [52]</td>
<td>21.0</td>
<td>31.0</td>
<td>25.1</td>
<td>24.2</td>
<td>63.9</td>
<td>35.1</td>
<td>18.3</td>
<td>79.3</td>
</tr>
<tr>
<td>TVN [6]</td>
<td>18.2</td>
<td>28.9</td>
<td>22.3</td>
<td>21.6</td>
<td>47.5</td>
<td>29.7</td>
<td>18.2</td>
<td>87.5</td>
</tr>
<tr>
<td>VZSL [13]</td>
<td>15.2</td>
<td>23.8</td>
<td>18.6</td>
<td>17.1</td>
<td>37.1</td>
<td>23.8</td>
<td>22.3</td>
<td>77.5</td>
</tr>
<tr>
<td>Our Method</td>
<td>39.7</td>
<td>38.9</td>
<td>39.3</td>
<td>37.8</td>
<td>58.2</td>
<td>45.9</td>
<td>37.0</td>
<td>51.4</td>
</tr>
</tbody>
</table>

Table IV
Comparison with state-of-the-art ZSL baselines on both inductive setting and transductive setting.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SUN</th>
<th>CUB</th>
<th>AWA</th>
<th>aPY</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAP [15]</td>
<td>39.9</td>
<td>40.0</td>
<td>44.1</td>
<td>33.8</td>
<td>39.5</td>
</tr>
<tr>
<td>CONSE [20]</td>
<td>38.8</td>
<td>34.3</td>
<td>45.6</td>
<td>26.9</td>
<td>36.4</td>
</tr>
<tr>
<td>CMT [44]</td>
<td>39.9</td>
<td>34.6</td>
<td>39.6</td>
<td>28.0</td>
<td>35.5</td>
</tr>
<tr>
<td>SSE [21]</td>
<td>51.5</td>
<td>43.9</td>
<td>60.1</td>
<td>34.0</td>
<td>47.4</td>
</tr>
<tr>
<td>LATEM [45]</td>
<td>13.0</td>
<td>49.3</td>
<td>55.1</td>
<td>35.2</td>
<td>48.7</td>
</tr>
<tr>
<td>ALE [16]</td>
<td>56.5</td>
<td>52.0</td>
<td>54.2</td>
<td>39.8</td>
<td>50.6</td>
</tr>
<tr>
<td>DEVISE [18]</td>
<td>53.7</td>
<td>53.9</td>
<td>65.6</td>
<td>32.9</td>
<td>51.5</td>
</tr>
<tr>
<td>SJE [17]</td>
<td>54.5</td>
<td>53.9</td>
<td>58.2</td>
<td>38.3</td>
<td>51.2</td>
</tr>
<tr>
<td>ESZSL [19]</td>
<td>56.3</td>
<td>55.6</td>
<td>54.0</td>
<td>23.9</td>
<td>47.5</td>
</tr>
<tr>
<td>SYNC [24]</td>
<td>40.3</td>
<td>33.3</td>
<td>53.0</td>
<td>8.3</td>
<td>36.2</td>
</tr>
<tr>
<td>GFZSL [10]</td>
<td>60.6</td>
<td>49.3</td>
<td>68.3</td>
<td>38.4</td>
<td>54.2</td>
</tr>
<tr>
<td>TVN [6]</td>
<td>59.3</td>
<td>54.9</td>
<td>64.7</td>
<td>40.9</td>
<td>55.0</td>
</tr>
<tr>
<td>VZSL [13]</td>
<td>52.0</td>
<td>43.8</td>
<td>63.7</td>
<td>30.3</td>
<td>47.5</td>
</tr>
<tr>
<td>Our Method</td>
<td>60.4</td>
<td>53.2</td>
<td>67.4</td>
<td>24.8</td>
<td>56.0</td>
</tr>
<tr>
<td>QFSL [31]</td>
<td>62.7</td>
<td>56.2</td>
<td>60.4</td>
<td>38.6</td>
<td>54.7</td>
</tr>
<tr>
<td>GFZSL-Trans [10]</td>
<td>59.4</td>
<td>45.2</td>
<td>74.7</td>
<td>35.9</td>
<td>53.8</td>
</tr>
<tr>
<td>VZSL-Trans [12]</td>
<td>57.6</td>
<td>49.3</td>
<td>69.1</td>
<td>35.7</td>
<td>52.9</td>
</tr>
<tr>
<td>Our Transductive</td>
<td>61.5</td>
<td>59.3</td>
<td>82.2</td>
<td>44.4</td>
<td>61.9</td>
</tr>
</tbody>
</table>

Coupled Dictionary Learning (CDL) [49], Probabilistic AND-OR Attribute Grouping Model (LAGO) [50], Pseudo Transfer (PSEUDO) [51], KERNEL model [52], Triple Verification Network (TVN) [6], and Conditional Variational ZSL (VZSL) [12]. The transductive methods include GFZSL-Trans [10], VZSL-Trans [12] and QFSL [31], and all the results are recorded in Tab. IV and Tab. III, among which VZSL and QFSL are implemented by us according to the algorithms described in their original papers, and the others are directly cited from [33] or the results reported by themselves. Besides, since we have already known the ascription of the test data in transductive setting, we do not report the performances of transductive methods such as QFSL [31] and GFZSL-Trans [10] on GZSL.

Comparison on ZSL

We conduct experiments on all the four datasets for the metric ZSL and report the results in Tab. V. From the table, we can discover that our method of inductive setting can achieve the best performance on aPY, and exceed DEVISE, the best method on the corresponding dataset, by 3.0%. On the datasets SUN, AWA and CUB, our method can achieve the second, second and fifth place and lower than the best methods by 2.1%, 0.9% and 2.4% respectively. In the transductive setting, our method can reach the first place on three datasets, including CUB, AWA and aPY, and obtain 3.1%, 7.5% and 4.6% higher compared with the best methods QFSL, GFZSL-Trans and DEVISE respectively.

Although our method gets a bit worse performance than the best methods on SUN, AWA and CUB (on the inductive setting), the average performance on all dataset is much higher than the best baselines, no matter on inductive setting or transductive setting. Concretely, our method can achieve 56.0% on inductive setting and 61.9% on the transductive setting and surpass the best method TVN and QFSL by 1.0% and 7.2% respectively.

In our opinion, the standard to evaluate a method cannot only rely on the performance on a single dataset, while it should depend on as many datasets as possible and take the best performance of the average score as the winner. Therefore, we argue that our method can outperform the state-of-the-art methods of ZSL.

Comparison on GZSL

In conventional ZSL, we assume that the search range is fixed on the unseen classes, which is unrealistic because it is unable to know whether the new sample belongs to the seen classes or the unseen classes in advance. Therefore, the more realistic way is to test on all classes, i.e., GZSL. In this subsection, we conduct the experiments and give the analysis on GZSL.

Since the transductive setting assumes that the ascription of the test to seen classes or unseen classes, it is pointless to report the result on GZSL. The experiments on GZSL are also carried out on the four datasets, and the results are shown in Tab. III.
From Tab. III it can be observed that our method can achieve the best performance on $ts$ and $H$ on all four datasets. Specifically, as for the metric $ts$, we can achieve 17.9%, 13.6%, 8.9%, and 6.1% higher than the best baselines; for the harmonic accuracy $H$, 12.6%, 10.8%, 4.2%, and 11.0% improvement can be obtained compared to the best methods on each dataset.

As for $ts$, the seen test accuracy, although our method cannot reach the first place on any dataset, it can still keep the performance on the upper-middle level. Some methods such as DAP and CONSE can get the highest score on $ts$ on the datasets CUB, AWA and aPY, but they almost get no correct recognition for the unseen test set, which finally leads to bad performance on $H$. This phenomenon reveals that those methods are over-fitting on seen classes, thus they perform badly on unseen classes.

Besides, we further compare the results in Tab. III and Tab. IV. It is observed that although our method cannot get the best performances on ZSL on datasets SUN, CUB and AWA, it can surpass the best methods by a large margin on GZSL. Since we all have known that GZSL is more reasonable than ZSL in realistic scenarios due to the ascription of the input data to seen or unseen classes is usually unknown, we can conclude that our method is superior to the state-of-the-art methods.

In Fig. 6, we take AWA as an example and illustrate the data points before and after the process of triplet network with t-SNE [53]. The upper row shows the data points of seen classes, and the bottom row shows the data points of unseen classes. From this figure, we can clearly observe that the points belong to the same category gather together and the overlap between classes is greatly reduced for both seen classes and unseen classes.

### C. Ablation Study

In this section, we will give some detailed analysis of several hyperparameters, such as the effect of triplet network, the dimension of the latent space $\beta$ and the influence of the LNPS.

#### Effect of Triplet Network

Since we have claimed that directly using visual features extracted from ResNet will cause the overlap between the distribution of each class and lead to performance degradation, we exploit a triplet network to process the visual features. Here, we demonstrate three issues, the first one is can the triplet network reduce the overlapped area between each class, the second one is how much does the triplet network affect the performance, and the last one is whether the proposed two-layer full connection network is the optimal choice.

In Fig. 7 we take AWA as an example and illustrate the data points before and after the process of triplet network with t-SNE [53]. The upper row shows the data points of seen classes, and the bottom row shows the data points of unseen classes. From this figure, we can clearly observe that the points belong to the same category gather together and the overlap between classes is greatly reduced for both seen classes and unseen classes.

Since we have claimed that the process of triplet network can improve the performance of ZSL, we conduct experiments on all four datasets to verify whether the triplet network...
the Siamese network, which is caused by the fact that the model with the triplet network slightly outperforms that with the final performance. Furthermore, we can also find that our network in our method plays an important role in improving discriminative. APY is a coarse-grained dataset and also has i.e., the data points gather together within a class and there are many overlapped areas. After the process of triplet network, the data points are very similar in visual space, thus for this phenomenon is that CUB is a fine-grained dataset, that with neither of triplet and Siamese networks. The reason performance significantly, especially on CUB, comparing to discover that our method with triplet network can improve the results are recorded in Fig. 7. From the figure, we can have a positive effect. Besides, we also experiment with the Siamese network to show our method is better than it too, and the results are recorded in Fig. 7. From the figure, we can discover that our method with triplet network can improve the performance significantly, especially on CUB, comparing to that with neither of triplet and Siamese networks. The reason for this phenomenon is that CUB is a fine-grained dataset, where the data points are very similar in visual space, thus there are many overlapped areas. After the process of triplet network, the data points gather together within a class and spread out between classes, i.e., the data points are more discriminative. APY is a coarse-grained dataset and also has a little improvement, which further proved that the triplet network in our method plays an important role in improving the final performance. Furthermore, we can also find that our model with the triplet network slightly outperforms that with the Siamese network, which is caused by the fact that the Siamese network cannot deal with both negative and positive samples simultaneously.

To answer the third question, we design an experiment by modifying the layers of triplet network and report the results on four datasets in Fig. 8. In this experiment, besides the proposed two-layer model, we compose another three network architectures, including one-layer (2048 → β), three-layer (2048 → 1024(ReLU) → 1024(ReLU) → β) and four-layer (2048 → 1024(ReLU) → 1024(ReLU) → 1024(ReLU) → β). From the figure, we can find the best results always appear in the two-layer model on all four datasets, which reveals the two-layer model is optimal, and the one-layer model suffers from under-fitting due to the few parameters, while the three/four-layer model falls into over-fitting because of its excessive parameters.

**Effect of probabilistic framework**

Since we have claimed that our method with the probabilistic framework is better than that with NNS in Fig. 1, we experiment with NNS to show the superiority of our model. In this experiment, we use NNS instead of the probability model can exceed that with NNS by a large margin on both ZSL and GZSL. In this experiment, we use NNS to show the superiority of our model.

**Different Dimensions of Latent Space**

The dimension of latent space is determined by cross-validation, and optimal values are recorded in Tab. From this table, we can clearly observe that our method with the probabilistic model can exceed that with NNS by a large margin on both ZSL and GZSL.

**Different Dimensions of Latent Space**

The dimension of latent space is determined by cross-validation, and optimal values are recorded in Tab. However, we also argue that it is necessary to find out how much does the dimension of latent space affect the final results. Thus, in this subsection, we experiment with different dimensions of latent space to analyze the influence on the performance on four datasets. In this experiment, we set β = {16, 32, 64, 128, 256, 512, 1024, 2048} respectively and illustrate the classification accuracies in Fig. 10. From this figure, we can discover that the final results on the real unseen test are consistent with the optimal parameters of cross-validation in most circumstances. In addition, the dimension of latent space plays a more important role on SUN
In this subsection, we try to prove its superiority from multi perspectives in this subsection. 

The main creative part of our method is LNPS, thus we try to prove its superiority from multi perspectives in this subsection. 

The Eq. 5 is solved by NMF, which can guarantee that the coefficients are non-negative. However, there is another method, namely Matching Pursuit (MP) [54], can also be used to generate the nonnegative coefficients. Concretely, MP first finds the most similar attributes $a^u_i$ of $a^v_j$, and their coefficients $\kappa$ by addressing the following formulation,

$$c = \arg \min_{j \in S} \|a^u_i - \kappa a^v_j\|_2^2$$

subject to $\kappa \geq 0,$

and then sets the residual $a^u_i - \kappa a^v_j$ as $a^u_i$ to find the next $a^v_j$ and $\kappa$ with the formulation (22) until convergence (such as when residual reaches a very small value). Therefore, we conduct experiments on both MP and NMF to find which one is better. We first exploit MP and NMF to generate the coefficients on AWA, and choose the class ‘Giraffe’ as an example, whose coefficients are drawn in Fig. 9. From the figure, we can observe that the coefficients generated by MP haves only three non-zero values, and among these values there is a big one 0.8 corresponding to the class ‘deer’, while the coefficients produced by NMF have seven non-zero values, and all the values are not so big and nearly equal to each other. The phenomenon produced by MP will lead to a very serious problem that the synthesized unseen prototype of ‘Giraffe’ is very similar to the seen class ‘deer’, which will subsequently influence the classification, especially on GZSL.

We illustrate the result with NMF and MP via a histogram in Fig. 11, from which we can find that the results on ZSL with MP are a little worse than that with NMF, while the performance with MP on GZSL is lower than that with NMF by a large margin. This phenomenon is caused by the reason that MP tries to find the most similar seen classes, among which the first seen class contributes the most, while NMF attempts to synthesize the unseen class with multi reasonable seen classes. When testing on ZSL, the search range is only focused on the unseen classes, so the performances with MP and NMF are both significant. However, when testing on GZSL, the search range is extended to all the classes, thus the new test sample of unseen classes will be misclassified to be the similar seen class by the method with MP, while the method with NMF will not make such error because the synthesized unseen class is not very similar to any of the seen classes. For example, many instances of the class ‘Giraffe’ will be misclassified to the category ‘deer’ when testing on GZSL according to Fig. 9.

In addition, to verify the importance of our proposed LNPS, we also design another experiment comparing our method with GFZSL [10]. In our experiments, we first employ the triplet network to preprocess the input data before applying GFZSL, and the results are demonstrated in Fig. 12. In this figure, we can observe that our method can outperform the GFZSL significantly on CUB, AWA and aPY, except that the performance on SUN has a little degradation. Since the two experiments use the same preprocessing, we can argue that the probabilistic model of our method is better than that of GFZSL.

Results on Few Shot Learning

To be more generalizable, we extend our method on Few Shot Learning (FSL) and conduct an experiment to show its effect. In this experiment, we set the number of labeled samples as $\{2, 5, 10, 15, 20\}$, and report the results on the datasets of AWA and CUB. The experiment includes both our method and VZSL [12], and the results are shown in Fig. 13. From this figure, it can be clearly discovered that our method can significantly outperform VZSL under each number.
of labeled data. The superiority of our method mainly comes from the nonnegative synthesis of the unseen prototypes.

**Fig. 13.** Results of our method on FSL compared with VZSL [12].

**Task/Modality Shift Problem**

Task-shift is an inevitable issue in the ZSL problem because the disjoint seen/unseen distribution in the visual modality needs to be reconciled by an extra auxiliary domain. Although semantic attributes or Word2Vec [55] representation is widely adopted, they both suffer from the severe visual-semantic gap. From the experimental results in Fig. 14 it can be seen that Word2Vec suffers from more task-shift than visual attributes and the performance is remarkably degraded. This is because the semantic attribute is more associated with the visual appearance, such as color, stripe, and leg, whereas Word2Vec captures more relationships between categories with text description rather than the visual features. Therefore, we can treat semantic attribute as another type of visual representation and the task shift problem has a slight impact on final performance. However, such task-shift results in that the prototypes in the original attribute space are still not totally consistent with that in the visual feature space. Therefore, our latent visual feature is directly extracted from the visual image by deep network using the triple network as a constraint to maximally mitigate the visual-semantic discrepancy. Evidence can be clearly observed in Tab. III Tab. IV and Fig. 7 where the performance of using the synthesized prototype in the latent feature modality significantly outperforms that of using original visual prototypes. Therefore, the task-shift problem is alleviated by the proposed prototypes in the latent space.

**V. Conclusion**

In this paper, to alleviate the misclassification problem in traditional NNS based methods, we have proposed a probabilistic framework for ZSL. In this method, the visual features of seen classes are first used to train a triplet network to gather within a class and scatter between classes in latent space, and then further be leveraged to generate seen prototypes. The most creative part of the proposed work is the latent nonnegative prototypes synthesis of unseen classes, which exploits the relations between seen attributes and unseen attributes to compute the synthesis coefficients, which is subsequently utilized to synthesize prototypes of unseen classes. In addition, we also extend our method on the transductive setting. Extensive results on both ZSL and GZSL have proved that our method can outperform most state-of-the-art methods on four popular datasets, and the detailed analysis of some hyper-parameters also shows the superiority of the proposed method.

**References**


Haofeng Zhang received the B.Eng. degree and the Ph.D. degree in 2003 and 2007 respectively from School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. He is currently an associate professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. His research interests include computer vision and robotics.

Huaqi Mao received the B.S. degree in Software Engineering from Chongqing University of Technology, Chongqing, China, in 2017, and is currently working towards the master degree in the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His current research interests include image classification and deep learning.

Yang Long is currently an Assistant Professor with the Department of Computer Science, Durham University, UK. He received his Ph. D. degree in Computer Vision and Machine Learning from the Department of Electronic and Electrical Engineering, the University of Sheffield, UK, in 2017. He received the M.Sc. degree from the same institution, in 2014. His research interests include Artificial Intelligence, Machine Learning, Computer Vision, Deep Learning, Zero-shot Learning, with focus on Transparent AI for Healthcare Data Science.

Wankou Yang received the B.S., M.S., and Ph.D. degrees from the School of Computer Science and Technology, Nanjing University of Science and Technology, China, in 2002, 2004, and 2009, respectively. From 2009 to 2011, he was a Post-Doctoral Fellow with the School of Automation, Southeast University, China. From 2010 to 2011, he was a Post-Doctoral Fellow with the Face Aging Group, University of North Carolina at Wilmington, Wilmington, NC, USA. From 2011 to 2015, he was an Assistant Professor with the School of Automation, Southeast University. He is currently an Associate Professor with the School of Automation, Southeast University. His research interests include pattern recognition, computer vision, and machine learning.

Ling Shao (M’09–SM’10) is the CEO and Chief Scientist of the Inception Institute of Artificial Intelligence (IAI), Abu Dhabi, United Arab Emirates. His research interests include computer vision, machine learning and medical imaging. He is an associate editor of *IEEE Transactions on Image Processing*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Cybernetics*, and several other journals. He is a fellow of the International Association of Pattern Recognition, the Institution of Engineering and Technology and the British Computer Society.