Compressing Unknown Images with Product Quantizer for Efficient Zero-Shot Classification

Jin Li[1,2], Xuguang Lan[1,2*], Yang Liu[3], Le Wang[1,2], Nanning Zheng[1,2]
[1] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, China
[2] National Engineering Laboratory for Visual Information
Processing and Applications, Xi'an Jiaotong University, China,
[3] State Key Laboratory of Integrated Services Networks, Xidian University, China
{j.lixjtu, liuyangxidian}@gmail.com, {xglan, lewang, nnzheng}@mail.xjtu.edu.cn

Abstract

For Zero-Shot Learning (ZSL), the Nearest Neighbor (NN) search is generally conducted for classification, which may cause unacceptable computational complexity for large-scale datasets. To compress zero-shot classes by the trained quantizer for efficient search, it tends to induce large quantization error because distributions between seen and unseen classes are different. However, as semantic attributes of classes are available in ZSL, both seen and unseen classes have the same distribution for one specific property, e.g., animals have or do not have spots. Based on this intuition, a Product Quantization Zero-Shot Learning (PQZSL) method is proposed to learn embeddings as well as quantizers to compress visual features into compact codes for Approximate NN (ANN) search. Particularly, visual features are projected into an orthogonal semantic space, and then the Product Quantization (PQ) is utilized to quantize individual properties. Experimental results on five benchmark datasets demonstrate that unseen classes are represented by the Cartesian product of quantized properties with little quantization error. As classes in orthogonal common space are more discriminative, the classification based on PQZSL achieves state-of-the-art performance in Generalized Zero-Shot Learning (GZSL) task, meanwhile, the speed of ANN search is 10-100 times higher than traditional NN search.

1. Introduction

As deep learning-based architectures achieve many successes [23] [40], large-scale labeled training images are more and more important for computer vision applications recently. However, large-scale well-annotated datasets are difficult to be established. One of the main reasons is that the frequencies of natural images obtained by people follow long-tailed distributions [47]. The number of newly defined visual concepts and products is growing rapidly. Meanwhile, some objects are simply rare by nature and there are little labeled samples for training. Besides, sometimes image annotation requires options from experts, which is expensive. Alternatively, Zero-Short Learning (ZSL) has been proposed as a feasible solution [24]. Images in limited classes are used in the training phase while the learned model aims to recognize images in totally new classes without additional data collections. Aiming at this goal, ZSL utilizes semantic auxiliary information such as word descriptions and attributes to preserve inter-class associations between seen and unseen classes [24] [10] [11] [48] [14]. In the training phase, embeddings and compatibility functions are learned given images and attributes of seen classes. In the testing phase, the task of unseen class recognition becomes a typical classification problem, which can be solved by employing the Nearest Neighbor (NN) search with learned compatibility functions.

Nowadays, as the number of obtained images and new concepts grows rapidly, very large space and long time are required to store and annotate images. For ZSL, to achieve high efficiency of classification in the large-scale dataset such as ImageNet [9] is a big challenge. Compressing data into compact code is a powerful technique for enabling efficient Approximate Nearest Neighbor (ANN) search. Particularly, the methods of generating compact code can be divided into two categories, i.e., hashing-based and quantization-based methods. Hashing learns embeddings mapping high-dimension vectors into binary codes, and thus can significantly save storage space [42] [36] [26]. More importantly, in the binary space, the Euclidean distance is replaced by the Hamming distance, which can be efficiently calculated by the bit-wise XOR operation. Howev-

*Corresponding author: X. Lan. (xglan@mail.xjtu.edu.cn)

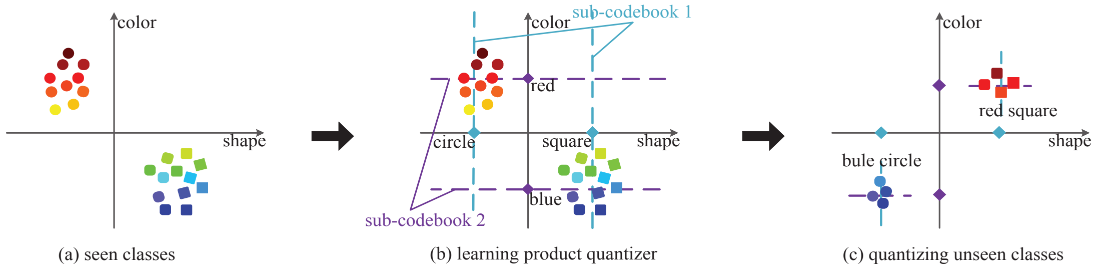

Figure 1. Illustration of PQZSL. (a) Samples in two seen clusters with two properties, i.e., shape and color. (b) Product quantizer trained to quantize individual property. The sub-codebook of shape contains two codewords, circle and square; the sub-codebook of color contains red and blue. (c) Using Cartesian product of properties to represent new classes, i.e., bule circles and red squares.

er, there are two main disadvantages in hashing-based ANN search. One is that the value range of Hamming distance is limited, which may decrease the accuracy when the nearest neighbors are sorted by Hamming distance. For example, as vectors are compressed into 8-bits codes, distances only vary in 8 integer values. The other one is that hashing can only use the Symmetric Distance Computation (SDC)[19], which has been proved to be worse than Asymmetric Distance Computation (ADC) in ANN search [19].

Quantization-based methods aim to learn a set of codebooks that contain codewords to reconstruct vectors or subvectors [19] [16] [5]. In particular, original vectors are mapped into integer codes that index codewords for reconstruction. For ANN search, distances between the query and codewords are pre-computed and stored in look-up tables, then they are read from these tables by their codes. When vectors are compressed into 8-bits codes, quantization methods can learn a codebook containing 2^8 codewords, thus the range of distance has 256 values. In addition, quantization methods support the ADC, where only the database is compressed. In order to quantize zero-shot classes, the problem is that only distributions of seen classes can be obtained, which may be very different from test unseen classes. Therefore, if the codebook trained by features in seen classes are used to quantize unseen classes, it tends to cause large quantization error that may significantly decrease the search accuracy.

Since attributes of classes are introduced, we assume that there is a semantic space, where the value range of each semantic property is the same in both seen and unseen classes. As illustrated in Figure 1 (a), a semantic space contains two seen classes, varying in the color and shape. Although seen classes cluster in small regions of the semantic space, all kinds of colors and shapes are included. Based on this intuition, we propose a Product Quantization Zero-Shot Learning (PQZSL), which compresses large-scale database into compact codes for efficient ANN search. To reconcile the visual and semantic space, a common space is defined, where dimensions of projected features are required to be orthogonal to satisfy the independence condition of Product Quantization (PQ). In the common space, we train a sub-codebook to quantize specific properties. For example, two sub-codebooks are learned from the color and shape individually in Figure 1 (b). After sub-codebooks are generated, new classes can be quantized by Cartesian products of sub-codewords with little errors, which is shown in Figure 1 (c). Our intuition is demonstrated by experimental results on both synthesized dataset and ImageNet [9]. In the final classification, ANN search is utilized to compute distances between projected features and class-level attributes, which comes from two different modalities. Consequently, it is difficult to train a unified codebook to simultaneously quantizer features and attributes with little quantization errors. To solve this problem, ADC search strategy is introduced, where only the large-scale database is compressed while attributes are uncompressed in the testing phase. In practical applications, when the number of databases is much larger than that of classes, ADC has similar efficiency but achieves higher accuracy than SDC.

2. Related Works

In this section, we briefly review related work in zero-shot learning, vector quantization and approximate nearest neighbor search.

2.1. Zero-Shot Learning

In order to train a model from seen classes that can be generalized to classify unseen classes, ZSL generally requires a set of high-dimension vectors in visual space and semantic space to describe images and their characteristics, which are named features and attributes in this paper. In the training phase, features and class-level attributes in seen classes are obtained to learn embeddings or compatibility functions. In the testing phase, test images are required to be classified into the correct unseen classes identified by their attributes via the NN search. For Generalized Zero-Shot Learning (GZSL) [7], test images may come from either seen or unseen classes, which is more reasonable for the real-world task.

To unify visual features and semantic attributes into the same space, the first idea is mapping features to semantic space[24] [2] [43] [38] [13] [34] [4], which is similar to the multi-label classification. However, this direction of the projection may induce the hubness problem [37], where much features may be classified into one class in the testing phase. In contrast, mapping attributes into visual space can reduce the hubness problem, where the projected attributes are regarded as centroids of classes [33] [29]. Semantic Auto-Encoder (SAE) requires features and attributes to mutually reconstruct others using the same parameters, which can reduce the domain shift problem [22]. Moreover, a low-rank constraint is added in [27] to preserve the intrinsic structure in attributes.

Another group of methods map both visual features and semantic attributes into intermediate spaces [6] [45] [1] [46]. In Semantic Similarity Embedding (SSE) [45], the mixture of seen class proportions is regarded as the common space, where images belong to the same class should have a similar mixture pattern. Preserving Semantic Relations (PSR) [3] defines a common space which preserves semantic relations between classes. In Joint Latent Similarity Embedding (JLSE) [46], features and attributes are mapped into two separate latent spaces, and compatibility function is learned to measure their similarities. More exactly, projected class-level attributes in common space are also named prototypes, which represent centroids of classes in the NN search. For classification, the class label of test sample $\mathbf{u}_i$ is defined as

$$\hat{y}(\mathbf{u}_i) = \arg\min_j d(\mathbf{u}_i, \mathbf{v}_j), \tag{1}$$

where $d(\cdot,\cdot)$ is the measure of similarity, which is generally defined as the Euclidean or Cosine distance in ZSL. Assume $\mathbf{U} = \{\mathbf{u}_i\}_{i=1}^N$ and $\mathbf{V} = \{\mathbf{v}_j\}_{j=1}^T$ denote features and attributes in the testing phase respectively, where N is the number of images and T is the number of candidate classes. As similarities between features and attributes are exhaustively computed, the computational complexity is $O(D^2NT)$, where D is the dimension of common space. For large-scale classification task, the complexity is very large so that the quantization-based ANN search [19] is proposed to improve the efficiency.

2.2. Vector Quantization

Vector Quantization (VQ) [25] has been widely used in data compression and fast retrieval applications. For a D-dimension vector $\mathbf{u}_i \in \mathbb{R}^D$, a quantizer is a function mapping $\mathbf{u}_i$ to a vector $q(\mathbf{u}_i) \in \mathbf{C}$, where $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K]$ is the codebook that contains K D-dimension codewords $\mathbf{c}_i$. Then the mapping is defined as

$$q(\mathbf{u}_i) = \arg\min_{\mathbf{c}_k} ||\mathbf{u}_i - \mathbf{c}_k||_F^2, \tag{2}$$

where the D-dimension vector $\mathbf{u}_i$ is compressed into a scalar. Given a training set $\mathbf{U}$, the objective function is defined as

$$\begin{aligned} &\min \sum_{i=1}^N ||\mathbf{u}_i - \mathbf{C}\mathbf{b}_i||_F^2, \\ &s.t. \quad \mathbf{b}_i = \{0,1\}^K, ||\mathbf{b}_i||_1 = 1, \end{aligned} \tag{3}$$

where $\mathbf{b}_i$ is the code (a one-hot vector) to represent sample $\mathbf{u}_i$. Lloyd's algorithm [28] iteratively updates the codebook and codes to minimize the loss. However, the size of the codebook in VQ rises exponentially with the length of the code. PQ [19] is proposed to overcome this limitation.

In PQ, a vector $\mathbf{u}_i$ is split into M sub-vectors $\mathbf{u}_{i1}, \mathbf{u}_{i2}, \cdots, \mathbf{u}_{iM}$, and then the m-th sub-codebooks is trained to quantize corresponding sub-vectors,

$$q_m(\mathbf{u}_{im}) = \mathbf{c}_{mk}, \mathbf{c}_{mk} \in \mathbf{C}_m, \tag{4}$$

where $\mathbf{C}_m = [\mathbf{c}_m 1, \mathbf{c}_m 2, \cdots, \mathbf{c}_m K]$ is the m-th sub-codebooks and k is the k-th codeword in $\mathbf{C}_m$. And the codebook is defined as the Cartesian product of M sub-codebooks

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{C}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{C}_M \end{bmatrix}. \tag{5}$$

Let $\mathbf{b}_{im}$ denote a sub-vector in $\mathbf{b}_i$, which indexes a codeword in sub-codebook $\mathbf{C}_m$ to represent $\mathbf{u}_{im}$, thus the objective function of PQ can be defined as

$$\begin{aligned} &\min \sum_{i=1}^N ||\mathbf{u}_i - \mathbf{C}\mathbf{b}_i||_F^2, \\ s.t. \quad &\mathbf{b}_i = [\mathbf{b}_{i1}^T, \cdots, \mathbf{b}_{iM}^T]^T, \\ &\mathbf{b}_{im} = \{0,1\}^K, \quad ||\mathbf{b}_{im}||_1 = 1. \end{aligned} \tag{6}$$

To reduce the quantization error in PQ, we can increase the number of sub-codebooks M. However, PQ assumes that the splits of $\mathbf{u}_i$ are independent of others. To satisfy this condition, a transformation that reduces correlation among dimensions in $\mathbf{u}_i$ is learned before training the product quantizer[16]. In this paper, we add the orthogonal restriction among dimensions in $\mathbf{u}_i$ to satisfy the independent condition, which is discussed in Section 3.1 in detail.

2.3. Approximate Nearest Neighbor Search

The quantization-based ANN search method is proved to be an efficient way to reduce the cost of distance computation [19]. After the vectors in the database are compressed via the learned quantizer, there are two strategies to approximate the NN search. The first way is SDC, where the similarity distance measure in NN search can be approximated as

$$\hat{d}(\mathbf{u}, \mathbf{v}) = d(q(\mathbf{u}), q(\mathbf{v})) = d(\mathbf{c}_k, \mathbf{c}_l), \tag{7}$$

where $d(\mathbf{c}_k, \mathbf{c}_l)$ can be pre-computed and stored in a look-up table. The second way is ADC that can be represented as

$$\tilde{d}(\mathbf{u}, \mathbf{v}) = d(q(\mathbf{u}), \mathbf{v}) = d(\mathbf{c}_k, \mathbf{v}), \tag{8}$$

where the query $\mathbf{v}$ is not compressed. Similar to SDC, we first compute store $d(\mathbf{c}_k, \mathbf{v})$ $(k = 1, \cdots, K)$ in a look-up table. After pre-computation, ANN search only requires to look up the table for N times to compute distances between $\mathbf{v}$ and $\mathbf{U} = \{\mathbf{u}_i\}_{i=1}^{N}$. When the time consumption of quantizing $\mathbf{v}$ is also considered, SDC and ADC have the same computational complexity in ANN search, which has been proved in [19].

3. Approach

The training dataset of ZSL is defined by a series of triplets $(\mathbf{x}_i^s, \mathbf{y}_i^s, \mathbf{a}_i^s)_{i=1}^{N_s} \in \mathbf{X}_s \times \mathbf{Y}_s \times \mathbf{A}_s$, where N_s is the number of training samples. $\mathbf{X}_s \in \mathbb{R}^{d_x \times N_s}$ represents the set of d_x dimensional visual features of images. $\mathbf{Y}_s \in \mathbb{R}^{L \times N_s}$ denotes class labels, where each column is a one-hot vector. Moreover, in ZSL, only L classes are available in training, which are regarded as seen classes. $\mathbf{A}_s \in \mathbb{R}^{d_a \times N_s}$ is the d_a-dim semantic representation such as attributes, which is the augment of class-level attributes $\mathbf{A}$, i.e., $\mathbf{A}_s = \mathbf{A}\mathbf{Y}_s$. In this paper, features and attributes are projected into a common space. $f(\mathbf{x}_i^s)$ and $g(\mathbf{a}_i^s)$ indicate the visual embedding and semantic embedding, respectively. In the testing phase, visual and semantic samples of unseen classes are given, i.e., $(\mathbf{x}_i^t, \mathbf{a}_i^t)_{i=1}^{N_t} \in \mathbf{X}_t \times \mathbf{A}_t$. Similar to seen classes, $\mathbf{X}_t$ and $\mathbf{A}_t$ contain N_t features and attributes, respectively. Moreover, ZSL requires that $\mathbf{A}_s \cap \mathbf{A}_t = \varnothing$. The NN search is employed to predict the label $\hat{y}$ of a sample $\mathbf{x}_i^t$, which can be defined in Eq. (1). Then, embeddings are learned for efficient zero-shot classification.

3.1. Product Quantization Zero-Shot Learning

Our method structure an orthogonal common space, where each dimension is independent to others. In the common space, a codebook (like $\mathbf{C}$ in Eq. (5)) is trained, where each sub-codebook is trained to represent corresponding dimensions. Then visual and semantic embeddings are learned to project features and attributes into the common space. By this way, the final objective function contains a loss containing three parts: quantization loss, visual embedding loss, and semantic embedding loss, meanwhile all parameters are optimized simultaneously. In the test phase, projected features are compressed into compact codes via the codebook while a projected attribute is regarded as a

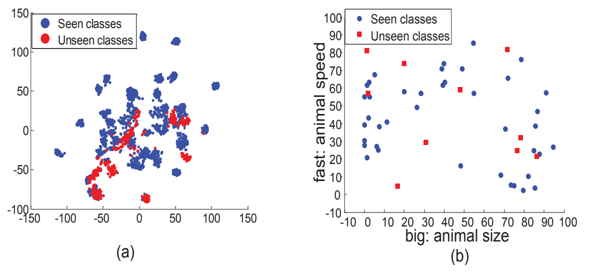

Figure 2. Visualization of distributions of seen and unseen classes in dataset Animal with Attributes (AwA). (a) 2-D t-SNE result of features. (b) Two properties class-level attributes: size and speed of animals

query $\mathbf{v}$ in Eq. (8). Asymmetric distances between features and attributes are computed via Eq. (8) for classification.

3.1.1 Product Quantization

Distributions of visual features in seen and unseen classes are different in zero-shot classification problem, which can be demonstrated by the t-SNE result [30] of features in Animal with Attributes (AwA) dataset [24] [44] shown in Figure 2 (a). Therefore, if the vector quantizer is trained using seen classes, it tends to cause large quantization error when codewords are utilized to represent unseen classes. Since attributes of classes can be obtained, the value range of each property of seen and unseen classes is similar. For example, properties “big” and “fast” of 50 kind animals in AwA dataset are shown in Figure 2 (b), which describe the size and speed with continuous value. For every single dimension, the value of both seen and unseen classes vary in the same range. This means each property can be completely seen in the training phase.

In fact, if the learned model is required to recognize all unseen objects in ZSL, seen classes must contains as much range as possible of properties. For example, if all training samples are animals, the model definitely cannot know airplane (just like ancient people). Therefore, we think the assumption of diversity of training classes is existed in all ZSL problem. Based on this observation, we assume that there is an orthogonal common space where dimensions are required to be independent of others. Compared to the semantic space of class-level attributes, the orthogonal common space achieves two advantages. First, projected features can be split into independent sub-vectors and quantized individually. Second, assume the visual embedding f consists of a family functions $f_1, f_2, \cdots$, each of them projects features into one dimension. The independence among dimensions in common space makes f simpler and more discriminative because functions will not confuse others, which is pointed in [18] and [21].

As discussed above, the quantization loss is defined as

$$\mathcal{L}_q = \sum_{i=1}^{N_s} \|f(\mathbf{x}_i^s) - \mathbf{C}\mathbf{b}_i\|_F^2 = \|f(\mathbf{X}_s) - \mathbf{C}\mathbf{B}\|_F^2, \quad s.t. \quad f(\mathbf{X}_s)f(\mathbf{X}_s)^T = \mathbf{I}. \tag{9}$$

Here $f(\mathbf{X}_s)$ maps each column in $\mathbf{X}_s$ one by one. $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{N_s}]$ represents the set of codes of features. In quantization method, Eq. (9) is generally optimized by iteratively update $\mathbf{C}$ and $\mathbf{B}$. After sub-codebooks are trained, the Cartesian product of sub-codewords can represent unseen classes in the common space with little quantization error for ANN search.

3.1.2 Embedding Learning

In ZSL, the visual embedding f is learned to project features into different class centers, which is similar to the classifier. In this paper, the center loss [17] is introduced,

$$\mathcal{L}_{e1} = \sum_{i=1}^{N_s} \|f(\mathbf{x}_i^s) - \mathbf{z}_{y_i^s}\|_F^2 = \|f(\mathbf{X}_s) - \mathbf{Z}_s\|_F^2, \quad s.t. \quad f(\mathbf{X}_s)f(\mathbf{X}_s)^T = \mathbf{I}. \tag{10}$$

Here $\mathbf{z}_{y_i^s}$ represents the center of projected features of class indexed by $\mathbf{y}_i^s$ in common space. In the training phase, visual embedding maps features into L centers $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_L]$, which represent L seen classes. Then $\mathbf{Z}_s$ in Eq. (10) is the augment of $\mathbf{Z}$ by class labels $\mathbf{Y}_s$, i.e., $\mathbf{Z}_s = \mathbf{Z}\mathbf{Y}_s$.

In Eq. (9), the orthogonal constraint makes f difficult to be optimized. Therefore, we relaxed this constraint by letting dimensions in the class center be orthogonal. And then the objective function Eq. (10) can be modified to

$$\mathcal{L}_{e1} = \|f(\mathbf{X}_s) - \mathbf{Z}_s\|_F^2, \quad s.t. \quad \mathbf{Z}_s = \mathbf{Z}\mathbf{Y}_s, \quad \mathbf{Z}\mathbf{Z}^T = \mathbf{I}. \tag{11}$$

The semantic embedding maps class-level attributes into class centers in the common space. In order to preserve semantic inter-class associations among class centers while avoiding the domain shift problem, the auto-encoder framework is introduced like [22], i.e.

$$\mathcal{L}_{e2} = \sum_{i=1}^{N_s} \left\|\mathbf{a}_i^s - g^{-1}(g(\mathbf{a}_i^s))\right\|_F^2, \quad s.t. \quad \mathbf{z}_i^s = g(\mathbf{a}_i^s), \tag{12}$$

where $g^{-1}(\cdot)$ is a decoder to reconstruct attributes. According to SAE [22]. The constraint in $\mathcal{L}_{e2}$ can be relaxed and the compact form is represented as

$$\mathcal{L}_{e2} = \left\|\mathbf{A}_s - g^{-1}(\mathbf{Z}_s)\right\|_F^2 + \lambda \|\mathbf{Z}_s - g(\mathbf{A}_s)\|_F^2. \tag{13}$$

To train a codebook for ZSL, $\mathcal{L}_{e1}$ and $\mathcal{L}_{e2}$ respectively guarantee the independence and semantics in projected points. Hence, the objective function of PQZSL is

$$\mathcal{L} = \mathcal{L}_q + \alpha\mathcal{L}_{e1} + \beta\mathcal{L}_{e2}. \tag{14}$$

3.2. Optimization

In this paper, both visual and semantic embeddings are defined as linear matrixes. Similar to SAE, the same projection matrix is simultaneously used for encoder and decoder. The objective function is deduced as

$$\min_{\mathbf{P},\mathbf{Q},\mathbf{C},\mathbf{B},\mathbf{Z}_s,\mathbf{Z}} \|\mathbf{P}\mathbf{X}_s - \mathbf{C}\mathbf{B}\|_F^2 + \alpha\|\mathbf{P}\mathbf{X}_s - \mathbf{Z}_s\|_F^2 + \beta\left\|\mathbf{A}_s - \mathbf{Q}^T\mathbf{Z}_s\right\|_F^2 + \alpha\lambda\|\mathbf{Q}\mathbf{A}_s - \mathbf{Z}_s\|_F^2, \quad s.t. \quad \mathbf{Z}_s = \mathbf{Z}\mathbf{Y}_s, \quad \mathbf{Z}\mathbf{Z}^T = \mathbf{I}. \tag{15}$$

3.2.1 Parameter Updating

To optimize this function, embeddings, codebook, codes and class centers are updated iteratively.

Update P: Fixing all parameters except $\mathbf{P}$, and let the derivative with respect to $\mathbf{P}$ be $\mathbf{0}$, the visual embedding is updated by

$$\mathbf{P} = (\mathbf{C}\mathbf{B} + \alpha\mathbf{Z}_s)\mathbf{X}_s^T[(1+\alpha)(\mathbf{X}_s\mathbf{X}_s^T)]^{-1}. \tag{16}$$

Update Q: To learn the semantic embedding $\mathbf{Q}$, the other parameters are fixed. Let the derivative with respect to $\mathbf{Q}$ be 0, a Sylvester Equation [39] is obtained,

$$\beta\mathbf{Z}_s\mathbf{Z}_s^T\mathbf{Q} + \lambda\mathbf{Q}\mathbf{A}_s\mathbf{A}_s^T = (\beta+\lambda)\mathbf{Z}_s\mathbf{A}_s^T, \tag{17}$$

which can be solved in a single line in Matlab or Python [1]. In fact, divide both sides of Eq. (17) by β, there is only one hyper-parameter that influence $\mathbf{Q}$.

Update C: In PQ, sub-codebooks can be optimized independently. Let $\mathbf{P} = [\mathbf{P}_1^T, \cdots, \mathbf{P}_M^T]^T$, the sub-codebook $\mathbf{C}_m$ refers to

$$\|\mathbf{P}_m\mathbf{X}_s - \mathbf{C}_m\mathbf{B}_m\|_F^2, \tag{18}$$

where $\mathbf{B}_m = [\mathbf{b}_{1m}, \mathbf{b}_{2m}, \cdots, \mathbf{b}_{N_s m}]$. Similar to Eq. (16), $\mathbf{C}_m$ can be updated by

$$\mathbf{C}_m = (\mathbf{P}_m\mathbf{X}\mathbf{B}_m^T)(\mathbf{B}_m\mathbf{B}_m^T)^{-1}. \tag{19}$$

Update B: Given a sample $\mathbf{x}_i^s$, its code $\mathbf{b}_{im}$ of each sub-codebook $\mathbf{C}_m$ is also independent. We update $\mathbf{b}_{im}$ by exhaustively search sub-codewords in $\mathbf{C}_m$, which achieves the minimal quantization error in $\|\mathbf{P}_m\mathbf{x}_i^s - \mathbf{C}_m\|_F^2$.

Update $\mathbf{Z}_s$: When $\mathbf{Z}_s$ is updated, we relax the constraint

[1] For example, Matlab can use sylvester() or lyap() function.

$\mathbf{Z}_s = \mathbf{Z}\mathbf{Y}_s$ by adding an additional term $\|\mathbf{Z}_s - \mathbf{Z}\mathbf{Y}_s\|_F^2$, and the objective function can be re-written as

$$\begin{aligned}&\alpha\|\mathbf{P}\mathbf{X}_s - \mathbf{Z}_s\|_F^2 + \beta\left\|\mathbf{A}_s - \mathbf{Q}^T\mathbf{Z}_s\right\|_F^2\\&+\lambda\|\mathbf{Q}\mathbf{A}_s - \mathbf{Z}_s\|_F^2 + \gamma\|\mathbf{Z}_s - \mathbf{Z}\mathbf{Y}_s\|_F^2.\end{aligned} \tag{20}$$

Let the derivative with respect to $\mathbf{Z}_s$ be 0, it can be updated by

$$\begin{aligned}\mathbf{Z}_s =& \left[(\alpha+\lambda+\gamma)\mathbf{I} + \beta\mathbf{Q}\mathbf{Q}^T\right]^{-1}\\&[(\beta+\lambda)\mathbf{Q}\mathbf{A}_s + \alpha\mathbf{P}\mathbf{X}_s + \gamma\mathbf{Z}\mathbf{Y}_s].\end{aligned} \tag{21}$$

here $\mathbf{I}$ is an identity matrix with the same size as $\mathbf{Q}\mathbf{Q}^T$. To guarantee the independence among dimensions in $\mathbf{Z}_s$, we set $\gamma \gg \alpha, \beta, \lambda$.

Update Z: To optimize $\mathbf{Z}$, we solve the objective function

$$\|\mathbf{Z}_s - \mathbf{Z}\mathbf{Y}_s\|_F^2, \quad s.t. \quad \mathbf{Z}\mathbf{Z}^T = \mathbf{I}. \tag{22}$$

Notice that there is no constraint in the number of dimensions in common space. To solve Eq. (22), we restrict $\mathbf{Z}$ to be a square matrix with size $L \times L$. It can be easily proved that if rows of $\mathbf{Z}$ are orthogonal, and then columns of $\mathbf{Z}$ are also orthogonal [15], i.e., $\mathbf{Z}^T\mathbf{Z} = \mathbf{Z}\mathbf{Z}^T = \mathbf{I}$. According to [16], Eq. (22) has a closed-form solution. Firstly, we apply Singular Value Decomposition to $\mathbf{Y}_s\mathbf{Z}_s^T = \mathbf{R}\mathbf{S}\mathbf{W}^T$, and $\mathbf{Z} = \mathbf{W}\mathbf{R}^T$.

3.2.2 Hyper-parameters and Initialization

There are many hyper-parameters in Eq. (15). However, $\mathbf{C}$, $\mathbf{B}$ and $\mathbf{Z}$ only present in one term thus are not influenced by hyper-parameters. As we set $\gamma \gg \alpha, \beta, \lambda$ to satisfy the independence constraint, $\mathbf{Z}_s$ is also slightly influenced by all the hyper-parameters. More importantly, when updating $\mathbf{P}$ and $\mathbf{Q}$, they are only depended on α and λ respectively. By this means, we individually consider these parameters based on the performance in cross-validation in training data. For ImageNet [9], $\gamma = 10^4$, $\alpha = 300$, $\lambda = 10$ and $\beta = 1$. The hyper-parameters in other datasets can be obtained in same way.

To initialize parameters, we generate a real orthogonal matrix $\mathbf{Z}$ by orthogonalizing a random matrix and let $\mathbf{Z}_s = \mathbf{Z}\mathbf{Y}$. $\mathbf{Q}$ can be calculated by solving Eq. (17). The visual embedding is initialized without considering quantization loss, i.e., $\mathbf{P} = \mathbf{Z}_s\mathbf{X}_s^T(\mathbf{X}_s\mathbf{X}_s^T)^{-1}$, and then Lloyd's method is utilized to compute $\mathbf{C}$ and $\mathbf{B}$ given $\mathbf{P}\mathbf{X}_s$.

3.3. ANN Search-based Classification

In the testing phase, the ADC strategy is employed. The main reason is that asymmetric distance estimation is more accurate than symmetric distance. Moreover, it is difficult to train the unified codebook to quantize projected features and attributes, because there is a bias between the projections from two different modalities. Following ADC strategy, visual features are firstly projected into common space by $\mathbf{U}_t = \mathbf{P}\mathbf{X}_t$, which are further quantized as $q(\mathbf{U}_t) = \mathbf{C}\mathbf{B}$ and stored. Semantic attributes are also projected into common space, where $\mathbf{V}_t = \mathbf{Q}\mathbf{A}_t$ is regarded as the set of uncompressed queries. Then the ANN-based classification is conducted by using Eqs. (1) and (8). Specifically, when $d(\cdot,\cdot)$ is the Euclidean distance, Eq. (8) can be re-written as

$$\begin{aligned}\tilde{d}(\mathbf{u},\mathbf{v}) &= \sqrt{\|q(\mathbf{u})\|_2 + \|\mathbf{v}\|_2 - 2q(\mathbf{u})^T\mathbf{v}}\\&= \sqrt{\sum_{m=1}^{M}\|q_m(\mathbf{u}_m)\|_2 + \|\mathbf{v}_m\|_2 - 2q_m(\mathbf{u}_m)^T\mathbf{v}_m}.\end{aligned} \tag{23}$$

When $d(\cdot,\cdot)$ is defined as cosine distance, Eq. (8) becomes

$$\begin{aligned}\tilde{d}(\mathbf{u},\mathbf{v}) &= 1 - \frac{2q(\mathbf{u})^T\mathbf{v}}{\|q(\mathbf{u})\|_2\|\mathbf{v}\|_2}\\&= 1 - \frac{\sum_{m=1}^{M} q_m(\mathbf{u}_m)^T\mathbf{v}_m}{\sum_{m=1}^{M}\|q_m(\mathbf{u}_m)\|_2\sum_{m=1}^{M}\|\mathbf{v}_m\|_2}.\end{aligned} \tag{24}$$

Notice that each-codebook contains K=256 codewords, thus $q_m(\mathbf{u}_m)$ only has 256 possible values. In Eqs. (23) and (24), terms $q_m(\mathbf{u}_m)^T\mathbf{v}_m$ and $\|q_m(\mathbf{u}_m)\|_F^2$ also has 256 values that can be pre-computed and stored in look-up tables, and then the computational complexity is independent from the size of database.

3.4. Computational Complexity Analysis

Assume N features are required classified into T classes, then the computational complexity is $O(D^2NT)$ when the final classification is conducted in the D-dim common space. As ADC strategy is introduced, the computational complexity to establish tables is $O(D^2KT)$. The time of table lookup is $O(MNT)$, where M is the number of sub-codebooks. Let $D = \delta M, 1 \leqslant \delta \leqslant D$, the ratio of the complexity of NN and ANN search can be represented as

$$\begin{aligned}&\frac{O(D^2NT)}{O(D^2KT)+O(MNT)}\\=&\frac{O(\delta DMNT)}{O(\delta DMKT)+O(MNT)}\\=&\frac{O(\delta DN)}{O(\delta DK)+O(N)}.\end{aligned} \tag{25}$$

where K is generally fixed to 256, which is smaller than the number of features N. D is the dimension of uncompressed vectors. δ is depended on M, which is a hyper-parameter influencing the quantization error and search speed. In the large-scale datasets, $N \gg DK$, thus the time consumption can be significantly reduced according to Eq. (25). For medium-scale datasets, the main cost of ANN is establishing look-up tables. And the ratio of the speed of ANN and NN search is depended on the size of the database.

4. Experiments

The proposed method is mainly evaluated in large-scale dataset ImageNet 21K dataset and four medium-scale datasets, where classification in the generalized zero-shot task is tested. Moreover, details about product quantization are shown, including quantization error and search efficiency. For fair comparison, are features and settings are same as that in [44], which provides a standard of ZSL and GZSL task.

4.1. Datasets and Evaluation

The large-scale dataset ImageNet [9] contains 21,841 classes with more than 10 million images collected from the real-world, where 1K seen classes containing 1.2 million images are employed to learn embeddings. There are different splits for the test. 2-hops/3-hops refers to test classes belonging to 2/3 tree hops away from 1K train classes in the WordNet hierarchy[6], which contains 1,509/7,678 unseen classes. Classes that contain the top 500/1K/5K maximum images as well as top 500/1K/5K minimum images are also used for test splits respectively. Finally, all 20K classes are tested, which is very challenging. Four medium-scale datasets are Attribute Pascal and Yahoo (aPY) [12], Animals with Attributes (AWA) [44], Caltech-UCSD-Birds (CUB) [41] and SUN attributes (SUN) [32]. To split the dataset for training and testing, we follow the same settings with [44], where unseen classes are not included in the sets of deep neural network training. Hence, these unseen classes are really "unknown" for the trained model[2].

As quantization method is used, the quantization loss and its influence on accuracy are shown. And the time consumption of ANN search is evaluated to show the improvement of classification efficiency. To evaluate the accuracy of classification, the average of per-class precision (AP) is measured. In GZSL task where features come from either seen classes or unseen classes, '**tr**' represents AP of test features which belong to seen classes. In contrast, '**ts**' is the AP that unseen features are classified into all classes. '**H**' is the harmonic mean of **tr** and **ts**.

4.2. Main Results

We first train product quantizer in a synthesized dataset to demonstrate the effectiveness in representing zero-shot

[2]All image feautures and standard splites are published here: http://www.mpi-inf.mpg.de/zsl-benchmark

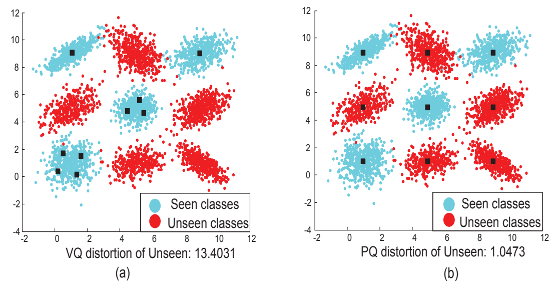

Figure 3. Training codebooks with different quantization method in ZSL, where codewords are denoted by black squares. (a) Vector Quantization. (b) Product Quantization

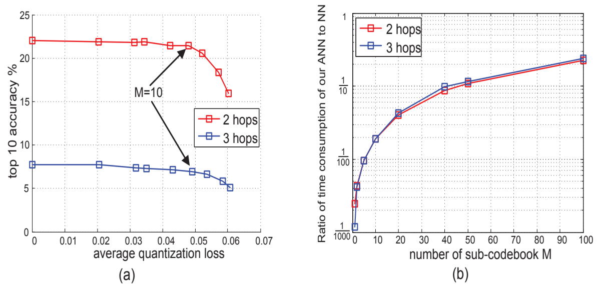

Figure 4. Accuracy and efficiency of quantization (a) Accuracy VS. quantization loss under different M. (b) Ratio of time consumption of 2/3 hops classes under different M.

classes. In detailed, we sample 4000 2-D vectors from a Gaussian Mixture Model, which has 9 clusters. Vectors drawn from 4 clusters are treated as seen classes, while others are unseen. The comparison between VQ and PQ is shown in Figure 3, where 9 codewords are trained. If VQ is directly used, it tends to minimize the quantization loss of known classes, which cause over-fitting when unseen classes are introduced. Since the information in each dimension is completely obtained from seen classes, the sub-codebook can accurately represent both seen and unseen classes.

To present the relationship between classification accuracy and quantization loss, we vary the number of sub-codebooks to adjust the loss. GZSL performance (**ts**) in 2/3 hops under different quantization loss are evaluated in Figure 4 (a). When quantization loss is larger than 0.05 ($M = 10$), the **ts** accuracy decreases rapidly. In Figure 4 (b), the ratio of time consumption of ANN and NN search via the different number of sub-codebooks is compared. The efficiency of ANN is 10-100 times higher than that of NN search in the same common space. According to the requirement of accuracy, we can set M from 10 to 50.

Finally, the overall results of PQZSL are compared with state-of-the-art baselines. The GZSL accuracy on ImageNet is evaluated in Table 1. To quantize projected features, we train 50 sub-codebooks where each of them contains 256 codewords. In this way, one visual feature is compressed

Table 1. Generalized Zero-Shot Learning comparisons on ImageNet dataset. We measure Top-10 accuracy in %.

	Hierarchy		Most populated			Least populated			All
Method	2-hops	3-hops	500	1K	5K	500	1K	5K	20K
CONSE [31]	0.86	7.14	23.47	18.38	9.92	0.00	0.00	0.66	3.43
CMT [38]	7.80	2.77	9.65	7.73	3.83	3.37	2.71	1.45	1.25
LATEM [43]	16.99	6.28	23.61	18.65	8.73	8.73	7.60	3.50	2.71
ALE [1]	17.79	6.34	24.93	19.37	9.12	**10.38**	**8.46**	3.63	2.77
DEVISE [13]	17.59	6.28	24.66	19.11	8.99	10.11	8.26	3.63	2.71
SJE [2]	17.46	6.21	23.61	18.45	8.79	9.85	8.00	3.50	2.71
ESZSL [35]	19.24	6.81	26.52	20.56	9.72	9.12	7.73	**3.76**	3.10
SYNC [6]	14.55	5.62	16.33	13.82	7.87	2.77	2.44	1.78	2.64
SAE [22]	13.55	4.82	20.76	16.60	7.60	3.43	2.57	1.58	2.24
PQZSL	**21.80**	**7.41**	**29.30**	**23.75**	**11.3**	9.42	7.87	3.72	**3.45**

Table 2. Generalized Zero-Shot Learning results on SUN, CUB, AWA and aPY. We measure the AP of Top-1 accuracy in %.

	SUN			CUB			AWA			aPY		
Method	**ts**	**tr**	**H**	**ts**	**tr**	**H**	**ts**	**tr**	**H**	**ts**	**tr**	**H**
CMT [38]	8.7	28.0	13.3	4.7	60.1	8.7	8.7	89.0	15.9	10.9	74.2	19.0
SSE [45]	2.1	36.4	4.0	8.5	46.9	14.4	8.1	82.5	14.8	0.2	**78.9**	0.4
LATEM [43]	14.7	28.8	19.5	15.2	57.3	24.0	11.5	77.3	20.0	0.1	73.0	0.2
ALE [1]	21.8	33.1	26.3	23.7	62.8	34.4	14.0	81.8	23.9	4.6	73.7	8.7
DEVISE [13]	16.9	27.4	20.9	23.8	53.0	32.8	17.1	74.7	27.8	4.9	76.9	9.2
SJE [2]	14.7	30.5	19.8	23.5	59.2	33.6	8.0	73.9	14.4	3.7	55.7	6.9
ESZSL [35]	11.0	27.9	15.8	12.6	63.8	21.0	5.9	77.8	11.0	2.4	70.1	4.6
SYNC [6]	7.9	**43.3**	13.4	11.5	**70.9**	19.8	10.0	90.5	18.0	7.4	66.3	13.3
SAE [22]	17.8	32.0	22.8	18.8	58.5	29.0	16.7	82.5	27.8	12.3	72.5	20.9
LESAE [27]	21.9	34.7	26.9	24.3	53.0	33.3	21.8	70.6	33.3	12.7	56.1	20.1
PSR [3]	20.8	37.2	26.7	24.6	54.3	33.9	20.7	73.8	32.3	13.5	51.4	21.4
SP-ANE[8]	24.9	38.6	30.3	34.7	70.6	46.6	23.3	**90.9**	37.1	13.7	63.4	22.6
CDL [20]	21.5	34.7	26.5	23.5	55.2	32.9	28.1	73.5	40.6	19.8	48.6	28.1
PQZSL	**35.1**	35.3	**35.2**	**43.2**	51.4	**46.9**	**31.7**	70.9	**43.8**	**27.9**	64.1	**38.8**

into a 50-Bytes code and the compression ratio is about 160. Compared with plenty of baselines, PQZSL achieves the best performance in most splits. The comparison demonstrates that compact codes can be used for replacing original features for zero-shot classification. Results in Table 2 show our advantages in the other four datasets. According to the dimension of common space in different datasets, the M is set to 30, 129, 10 and 5 for CUB, SUN, AWA and aPY respectively. Compared to other baselines, our approach obtains highest **ts** and **H** values in all the datasets. As we define the orthogonal semantic space, class centers are more discriminative. This is the main reason to achieve improvement. More importantly, the product quantization can quantize unseen classes with little errors, which slightly decreases the search accuracy.

5. Conclusions

In this paper, we propose a novel Product Quantization Zero-Shot Learning method, which learns product quantizer from seen classes to quantize unseen classes. In this way, the database can be compressed and stored as compact codes for efficient nearest neighbor search, which is helpful to large-scale classification. Given visual features and semantic attributes, the quantizer as well as embeddings, are learned iteratively. Experimental results in synthesized datasets demonstrate that codebook can well represent unseen classes. More importantly, the search speed is improved when ANN search is employed. The proposed method also achieves the state-of-the-art performance in GZSL in ImageNet and four medium-scale datasets.

Acknowledgements

This work was supported in part by the key project of Trico-Robot plan of NSFC under Grant No.91748208, National Science and Technology Major Project under Grant No. 2018ZX01028-101, key Project of Shaanxi province under Grant No.2018ZDCXLGY0607, and NSFC under Grant No.61573268.

References

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1425–1438, 2016. 3, 8

[2] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, pages 2927–2936, 2015. 3, 8

[3] Y. Annadani and S. Biswas. Preserving semantic relations for zero-shot learning. In *CVPR*, pages 7603–7612, 2018. 3, 8

[4] L. J. Ba, K. Swersky, S. Fidler, and R. Salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *ICCV*, pages 4247–4255, 2015. 3

[5] A. Babenko and V. Lempitsky. Additive quantization for extreme vector compression. In *CVPR*, pages 931–938, 2014. 2

[6] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Synthesized classifiers for zero-shot learning. In *CVPR*, pages 5327–5336, 2016. 3, 7, 8

[7] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *ECCV*, pages 52–68, 2016. 2

[8] L. Chen, H. Zhang, J. Xiao, W. Liu, and S.-F. Chang. Zero-shot visual recognition using semantics-preserving adversarial embedding network. In *CVPR*, pages 1043–1052, 2018. 8

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 1, 2, 6, 7

[10] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *CVPR*, pages 2584–2591, 2013. 1

[11] M. Elhoseiny, Y. Zhu, H. Zhang, and A. Elgammal. Link the head to the" beak": Zero shot learning from noisy text description at part precision. In *CVPR*, pages 6288–6297, 2017. 1

[12] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, pages 1778–1785, 2009. 7

[13] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, pages 2121–2129, 2013. 3, 8

[14] Y. Fu and L. Sigal. Semi-supervised vocabulary-informed learning. In *CVPR*, pages 5337–5346, 2016. 1

[15] F. R. Gantmacher. Matrix theory. *Chelsea, New York*, 21, 1959. 6

[16] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):744–755, 2014. 2, 3, 6

[17] Y. Guo, G. Ding, J. Han, and Y. Gao. Sitnet: Discrete similarity transfer network for zero-shot hashing. In *IJCAI*, pages 1767–1773, 2017. 5

[18] D. Jayaraman, F. Sha, and K. Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *CVPR*, pages 1629–1636, 2014. 4

[19] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2011. 2, 3, 4

[20] H. Jiang, R. Wang, S. Shan, and X. Chen. Learning class prototypes via structure alignment for zero-shot recognition. In *ECCV*, pages 118–134, 2018. 8

[21] P. Kankuekul, A. Kawewong, S. Tangruamsub, and O. Hasegawa. Online incremental attribute-based zero-shot learning. In *CVPR*, pages 3657–3664, 2012. 4

[22] E. Kodirov, T. Xiang, and S. Gong. Semantic autoencoder for zero-shot learning. In *CVPR*, pages 3174–3183, 2017. 3, 5, 8

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1

[24] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014. 1, 3, 4

[25] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transaction on Communication*, 28(1):84–95, 1980. 3

[26] L. Liu, M. Yu, and L. Shao. Latent structure preserving hashing. *International Journal of Computer Vision*, 122(3):439–457, 2017. 1

[27] Y. Liu, Q. Gao, J. Li, J. Han, and L. Shao. Zero shot learning via low-rank embedded semantic autoencoder. In *IJCAI*, pages 2490–2496, 2018. 3, 8

[28] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 3

[29] Y. Long, L. Liu, L. Shao, F. Shen, G. Ding, and J. Han. From zero-shot learning to conventional supervised classification: Unseen visual data synthesis. In *CVPR*, pages 1627–1636, 2017. 3

[30] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 4

[31] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. 2013. 8

[32] G. Patterson, C. Xu, H. Su, and J. Hays. The sun attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision*, 108(1-2):59–81, 2014. 7

[33] M. Radovanović, A. Nanopoulos, and M. Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(Sep):2487–2531, 2010. 3

[34] S. Reed, Z. Akata, H. Lee, and B. Schiele. Learning deep representations of fine-grained visual descriptions. In *CVPR*, pages 49–58, 2016. 3

[35] B. Romera-Paredes and P. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, pages 2152–2161, 2015. 8

[36] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015. 1

[37] Y. Shigeto, I. Suzuki, K. Hara, M. Shimbo, and Y. Matsumoto. Ridge regression, hubness, and zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–151, 2015. 3

[38] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, pages 935–943, 2013. 3, 8

[39] J. J. Sylvester. Sur l'équation en matrices px=xq. *CR Acad. Sci. Paris*, 99(2):67–71, 1884. 5

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 1

[41] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 7

[42] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2009. 1

[43] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele. Latent embeddings for zero-shot classification. In *CVPR*, pages 69–77, 2016. 3, 8

[44] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *arXiv preprint arXiv:1707.00600*, 2017. 4, 7

[45] Z. Zhang and V. Saligrama. Zero-shot learning via semantic similarity embedding. In *ICCV*, pages 4166–4174, 2015. 3, 8

[46] Z. Zhang and V. Saligrama. Zero-shot learning via joint latent similarity embedding. In *CVPR*, pages 6034–6042, 2016. 3

[47] X. Zhu, D. Anguelov, and D. Ramanan. Capturing long-tail distributions of object subcategories. In *CVPR*, pages 915–922, 2014. 1

[48] Y. Zhu, M. Elhoseiny, B. Liu, X. Peng, and A. Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In *CVPR*, pages 1004–1013, 2018. 1