Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2023) 31: 216 – 237
© TÜBİTAK
doi:10.55730/1300-0632.3979

LVQ Treatment for Zero-Shot Learning

Fırat İSMAİLOĞLU*
Department of Computer Engineering, Faculty of Engineering, Sivas Cumhuriyet University, Sivas, Turkey

Received: 14.02.2022 • **Accepted/Published Online:** 22.12.2022 • **Final Version:** 19.01.2023

Abstract: In image classification, there are no labeled training instances for some classes, which are therefore called unseen classes or test classes. To classify these classes, zero-shot learning (ZSL) was developed, which typically attempts to learn a mapping from the (visual) feature space to the semantic space in which the classes are represented by a list of semantically meaningful attributes. However, the fact that this mapping is learned without using instances of the test classes affects the performance of ZSL, which is known as the domain shift problem. In this study, we propose to apply the learning vector quantization (LVQ) algorithm in the semantic space once the mapping is determined. First and foremost, this allows us to refine the prototypes of the test classes with respect to the learned mapping, which reduces the effects of the domain shift problem. Secondly, the LVQ algorithm increases the margin of the 1-NN classifier used in ZSL, resulting in better classification. Moreover, for this work, we consider a range of LVQ algorithms, from initial to advanced variants, and applied them to a number of state-of-the-art ZSL methods, then obtained their LVQ extensions. The experiments based on five ZSL benchmark datasets showed that the LVQ-empowered extensions of the ZSL methods are superior to their original counterparts in almost all settings.

Key words: Zero-shot learning, learning vector quantization, image classification, prototype learning, large margin classifiers

1. Introduction

When dealing with the problem of image classification/visual recognition, we usually assume that a number of labeled training instances is available for each class of interest. However, in practice, this may not be feasible, since collecting and annotating instances for each class results in a huge cost. Moreover, after training a classifier, new unseen classes may emerge dynamically [1]. In fact, new plant and animal species are constantly being discovered, making the classification of such target classes a challenge for image classification [1–3].

To address the above problem, zero-shot learning (ZSL) was developed, which was inspired by the ability of humans to identify novel cases/classes given a high-level description about them [3, 6]. In the context of ZSL, such descriptions are generally given as a list of semantically meaningful properties called attributes. These can be continuous word vectors [4] or binary vectors of visual properties, such as ”has tail”, ”is red” [3]. Using these attributes, ZSL aims to classify test/target classes for which no labeled training instances are available. ZSL achieves this task in the following way.

ZSL assumes that the training classes, i.e. the classes whose labeled instances are available during training, are also represented by the same set of the attributes. This results in a space known as semantic (embedding) space that contains representations of both the test classes and training classes, i.e. their prototypes. A few

*Correspondence: fismailoglu@cumhuriyet.edu.tr studies in ZSL aim to transfer images in the visual feature space and their class representations in the semantic space to a different intermediate space though the main approach is to embed the images in the feature space directly into the semantic space, learning only one mapping [2, 5]. Regardless of which approach is preferred, the transformation is learned using only images of the training classes, while ZSL is concerned with classifying images of test classes. This leads to a problem referred to as the (projection) domain shift problem [1, 5, 6, 9].

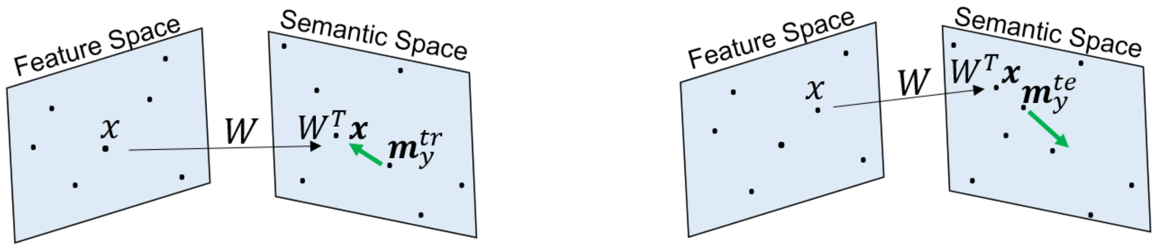
The domain shift problem also arises in domain adaptation/transfer learning, where a classifier is trained mainly using a source (auxiliary) domain and then used to classify instances of a target domain [7]. This has led to adopt several unsupervised domain adaptation methods to address the domain shift problem in ZSL [5, 8]. In this regard, [5] exploits unlabeled target data to create a multiview embedding space consisting of feature views and semantic views estimated using canonical correlation analysis (CCA). Similarly, [8] uses both labeled source and unlabeled target data to learn a projection matrix casting the learning problem as a sparse coding problem. However, in the case of ZSL, the source domain, i.e. the training data, and the target domain, i.e. the test data, have completely different tasks/classes. Therefore, the domain shift problem is more severe in ZSL compared to in transfer learning. For this reason, there are also a few studies that consider the domain shift problem solely from the perspective of ZSL [9, 10]. However, these studies require that all possible test data be available, which is usually impossible since new classes appear dynamically.

Another issue with ZSL is that the classification task performed in the semantic space is usually based on 1-nearest neighbor search. That is, a test instance is first mapped to the semantic space, and then a test class with the nearest prototype is assigned to the test instance. However, unlike the feature space, where the images are represented with discriminative features, the semantic space is constructed with attributes that are not necessarily discriminative [3, 5]. For example, although the noses of elephants and pigs have a different shape and appearance, the attribute ”has nose” is true for both animals in the semantic space but is indistinguishable. This also holds true when they are word vectors or they encode a linguistic context [3]. As a result, prototypes of both training and test classes become close to each other in the semantic space, which in turn leads to the nearest neighbor classifier having a low margin, i.e. being less confident in its predictions [11].

In this study, we propose to address the inherent problems of ZSL described above using learning vector quantization (LVQ) algorithm as a treatment [12]. In particular, we employ the LVQ algorithm in the semantic space which helps to refine the positions of the class representations of training and test classes so that they are more separable. This improves the classification performance of the nearest neighbor classifier of ZSL by increasing its margin. In fact, new positions of the test classes in the semantic space are determined in relation to the mapping learned at the training phase. Thus, we conclude that the proposed LVQ-empowered method helps to bridge the training and testing phases of ZSL and mitigates the domain shift problem to some extent.

Figure 1 illustrates the proposed idea based on applying the LVQ algorithm in the semantic space. Here, W is a matrix that allows a linear mapping from the feature space into the semantic space. On the left hand side 1a, the nearest prototype $\mathbf{m}_y^{tr}$ corresponds to the class of $\mathbf{x}$, an instance from the training class y, is therefore attracted to $\mathbf{x}$. On the right hand side (ii), however, the nearest prototype $\mathbf{m}_y^{te}$ corresponds to a test class y, and it is therefore repelled from $\mathbf{x}$.

To determine the projection matrix, i.e. W, the literature presents a range of methods with different motivations, including DeViSE [13], ALE [14], SJE [15], ESZSL [16], and SAE [8]. In this study, we employ all these methods to evaluate the general usability of the proposed method for ZSL. Particularly, we aim to understand which state-of-the-art ZSL method can benefit from the proposed method and to what extent. In



LVQ updates $\mathbf{m}_y^{tr}$ in that it is attracted to its instance. LVQ updates $\mathbf{m}_y^{te}$ in that it is repelled from an instance of a training class.

Figure 1. An illustration of the proposed LVQ-empowered ZSL method.

this way, we can identify which method is required to maximize the margin in the semantic space. We note here that the proposed method is limited to linear projections from the feature space to the semantic space. However, some nonlinear projections can also be found in the literature, including LATEM [17] and CMT [18].

The idea we propose in this paper can also be considered in the field of *prototype learning*, since it is based on the LVQ algorithm [19]. LVQ starts with a small set of exemplars (prototypes) for each class and then iteratively improves them so that the final prototypes can better represent the classes. Thus, combining ZSL with LVQ yields better representation of classes (both training and test classes), which, in turn, leads to a better classification performance, since in ZSL, classification is performed via class prototypes.

Below, we summarize the main contributions of this study.

- Thanks to the LVQ algorithm, we are able to refine the positions of the prototypes of the test classes with respect to the mapping learned with the instances of the training classes. This helps to mitigate the adverse effects of the domain shift problem that arises in ZSL.

- We make use of the LVQ algorithm to separate classes in semantic space, which is not necessarily discriminative. This, in turn, leads to the ZSL classifier that is based on 1-nearest neighbor principle having a high margin.

- Since the LVQ algorithm is studied under the field of prototype learning, we provide an insight into ZSL from the perspective of prototype learning, which has not yet been extensively studied.

- For the proposed method, we study not only the initial versions of the LVQ algorithm but also some more advanced versions, i.e. those that include an optimization process. Furthermore, we consider a number of state-of-the-art ZSL methods in evaluating the use of LVQ for ZSL.

In the remainder of the paper, we first provide an overview of the work related to ZSL, highlighting the milestones in its history, its relation with prototype learning, and its future. We then proceed to the methodology section, where we formally introduce the problem of interest and explain how we approach it using a family of LVQ algorithms. We reserve the following section for the presentation of the results of the experiments and the related discussion section. We conclude the paper with the final section summarizing the results and pointing out future research directions.

2. Related work

We divide the related work into four topics. First, we provide a brief history of Zero-Shot Learning highlighting the milestones in its history. Then, we give an overview of the projection methods used in ZSL, since the proposed method can be viewed as a post-processing step to such projection methods. We then discuss prototype based learning methods in ZSL, since LVQ, on which the proposed method is based, is an example of this kind. Finally, we conclude this section with the current research on ZSL.

2.1. Milestones of ZSL

Considering ZSL a particular form of lifelong learning, where the learning is a continuous process involving a sequence of new tasks, may date the history of ZSL to earlier times. However, the first notable study with the name of zero-data learning was conducted in 2008 in the context of drug discovery [20]. This study was followed by [1], where a formal framework for ZSL was provided for the first time in the literature. In this sense, semantic (feature) space, semantic knowledge bases (i.e. class prototype vectors) and semantic output code classifier (i.e. a classifier that infers a class label to instances via the semantic space) were introduced. Lampert et al. proposed to train a (probabilistic) classifier for each attribute using all instances (images) from all training classes to directly predict the attributes of the instances [3]. Later, this method was criticized with the argument that the classifiers are trained in isolation, i.e. without considering an interaction between the attributes [5, 14]. Furthermore, it suffers from the domain shift problem, since all classifiers are trained using the instances of the target classes only.

In the following years, ZSL has started to be considered a subfield of transfer learning [7], realizing the fact that the main idea in ZSL is to transfer the knowledge encapsulated in the instances of the target classes to the task of classifying instances of the test classes [6]. More specifically, it was agreed that ZSL belongs to heterogeneous transfer learning, as the label space of the source data (training classes), and that of the target data (test classes) are different, which makes the learning tasks different. As a result, many studies on ZSL have been motivated from the perspective of heterogeneous transfer learning [5, 21]. Another source of motivation for ZSL is metric learning, which can be defined as learning a task-specific distance metric. Since classification for test classes is determined using k-NN, a distance-based classifier, the success of ZSL is highly dependent on the metric used. Therefore, an important part of ZSL studies is devoted to the development of metric learning methods for ZSL [22, 23]. A common point of these studies is to define the metric learning task in such a way that instances are forced to be closer to their class prototypes than to any other prototype in the semantic space.

2.2. Projection methods in ZSL

Both training and test images are in the visual feature space, but their class labels are in the semantic space formed by a set of attributes, each corresponding to a property of these classes. In the realm of ZSL, images and their labels usually meet in the semantic space, although a few studies try to bring them together in an intermediate space in which both are projected [24]. To embed the images from the feature space directly into the semantic space, many projection methods have been developed. These methods can use either linear projections [8, 13–16] or nonlinear projections [17, 18] though; linear methods predominate in the literature, hence in our study. What these methods have in common is that they all aim to maximize the compatibility between (training) instances and their corresponding class prototypes in the semantic space, while differing in the form of the defined objective functions. For example, SAE [8] and ESZSL [16] include an explicit regularization term in their objective functions, but the rest prefers early stopping to regularize the mapping implicitly. We leave the details of how they work to the following section.

2.3. Prototype learning for ZSL

Prototype learning refers to learning a small set of exemplars, i.e. prototypes, that can represent the entire set of observations by capturing their typical features [25]. These prototypes can be utilized in unsupervised learning to reduce memory requirements and/or to uncover underlying structures in the data [12]. However, they are mainly used in supervised learning so that at least one prototype is generated for each class. As a result, instance-based classifiers, such as the 1-NN classifier, only consider these prototypes in the testing phase leading to a fast classification.

As mentioned earlier, in ZSL, both training and test classes are represented by a prototype in semantic space created by a set of attributes, which makes prototype learning methods applicable to ZSL in principle. Nevertheless, the use of these methods for ZSL remains elusive in the literature with a few exceptions. The common approach of these few studies is that they aim at learning prototypes for the feature (visual) space, which is more discriminative compared to the semantic space. A noticeable study in [26] proposes to learn visual prototypes and superprototypes in a transductive setting, i.e. using objects from test classes in the training phase. Here, the visual prototypes are learned in the visual space using instances of both training and test classes and hence transductive, and then these prototypes are coupled with semantic prototypes, resulting in superprototypes. This helps to align the feature space and the semantic space, as in learning a projection matrix. Similarly, Ji et al. introduced unseen prototype learning (UPL) model, which learns visual prototypes for test classes based on their semantic prototypes [27]. In this process, the visual prototypes are supervised with the class average visual representations in order for them to be more discriminative. UPL then treats each learned visual prototype as a latent classifier for the test classes.

2.4. Current research on zero-shot learning

We conclude this section with a brief note on research on ZSL published in the last two years. A recent trend in ZSL is the development of generative models that can generate training instances for test classes based on their semantic attributes [28]. The final classifier is then built over these synthesized instances as well as the original instances of the training classes. For this purpose, an algorithm based on either a generative adversarial network (GAN) or variational autoencoder is considered [29]. Another ongoing research in ZSL, including this research, is devoted to enforcing class separability realizing that ZSL is ultimately a classification problem [30, 31]. The study in [30] proposes to generate visual feature classifiers from the semantic attributes using a deep neural network, which retains the visual feature discriminability intrinsically. Similarly, the study in [31] focuses on correctly classifying hard samples of test classes that are closer to other classes to increase the generalizability of the deep neural network with respect to the test classes.

Finally, we review recent research on ZSL from the perspective of prototype learning, since our proposed ZSL method, has its roots in the LVQ algorithm, a type of prototype learning method. The first method worth mentioning in this regard is [32], where a method called ZSL-CPLSR is proposed. Basically, ZSL-CPLSR aims to learn class prototypes in the feature space for both training and test classes. In doing so, ZSL-CPLSR tries to ensure that the relationship between the training and test classes present in the semantic space is preserved in the feature space. Although ZSL-CPLSR and our proposed method both attempt to learn class prototype vectors, ZSL-CPLSR differs in two ways. Firstly, ZSL-CPLSR tries to solve a nonconvex optimization problem, while we rely on the LVQ algorithm, which is entirely heuristic, and secondly, ZSL-CPLSR creates the prototype vectors in the feature space, while it is the semantic space in our case. Another notable work is [33], which proposes convolutional prototype learning that performs in the feature space. The distinguishing element of this work is the assumption that the training and test classes follow similar distributions at task-level, meaning that they share similar zero-shot tasks, while the generic models in ZSL, including ours, assume that the training and test classes follow the same sampling distribution. We conclude this section with [34], where a model called discriminative domain-invariant prototypes (DDIP) is proposed. Similar to ZSL-CPLSR and our method, DDIP also attempts to learn prototype vectors for both the training and test classes, with the difference that these prototypes are learned in a latent hyperspherical space. Moreover, DDIP imposes an orthogonal constraint to ensure that the resulting prototypes are orthogonal to each other, which, in turn, makes them more discriminative in the latent space. As in ZSL-CPLSR, however, DDIP deals with a nonconvex optimization problem and thus obtains suboptimal solutions.

3. Methodology

This section begins with a formal definition of the Zero-Shot Learning (ZSL) problem, then moves to an explanation of the LVQ algorithm and its variants in which we are interested, and is concluded with an explanation of how we use the LVQ to treat the domain shift problem of ZSL.

3.1. Problem setup

In ZSL, there are two sets of classes: the set of training classes, i.e. seen classes; and the set of test classes, i.e. unseen classes, where we denote these sets as $\mathcal{S}$ and $\mathcal{U}$, respectively. We assume that these sets are disjoint: $\mathcal{S} \cap \mathcal{U} = \emptyset$ and that we have semantic representations of the training and test classes, which we denote as k-dimensional vectors: $\mathbf{m}_y \in \mathbb{R}^k$, where y belongs to $\mathcal{S}$ or $\mathcal{U}$. These representations are also known as class prototypes. $X \in \mathbb{R}^d$ denotes d-dimensional feature space in which both the training and test instances are originally represented. Given the semantic representations and a training set $\mathcal{D}^{tr} = \{(\mathbf{x}_i, y_i) \in X \times \mathcal{S}\}_{i=1}^{N_{tr}}$, we aim in ZSL to assign the instances of a test set $\mathcal{D}^{te} = \{\mathbf{x}_i \in X\}_{i=1}^{N_{te}}$ with a test class. To this end, we learn a mapping $f : X \rightarrow \mathcal{U}$ defined as follows:

$$\underset{y \in \mathcal{U}}{\operatorname{argmax}}\, F(\mathbf{x}, y; W), \tag{1}$$

where F measures the compatibility between $\mathbf{x}$ and y and is parameterized by a matrix W of size $d \times k$. Note that this compatibility is measured in the semantic space after $\mathbf{x}$ is projected through W. In this study, we restrict ourselves to linear compatibility between instances and classes, i.e. $\mathbf{x}$ and y, thus define F as dot-product similarity:

$$F(\mathbf{x}, y; W) = \mathbf{x}^T W \mathbf{m}_y \tag{2}$$

When it comes to determining the matrix W, the literature offers a number of methods motivated by different needs. Theoretically, the idea we propose in this paper is compatible with all of them, but we only consider the state-of-the-art and explain it below.

3.2. Projection methods to learn W

The methods presented in this context have in common that they aim at producing a higher compatibility between training instances and their class prototypes than between training instances and other class prototypes in the semantic space. In doing so, they define different objective functions, but in the following we give a unified formulation for these methods following the notation in [2].

Deep visual semantic embedding (DeViSE) [13] uses hinge rank loss as in unregularized ranking SVM:

$$\sum_{y \in \mathcal{S}} \max\left[0, \Delta\left(y_i, y\right) + F(\mathbf{x}_i, y; W) - F(\mathbf{x}_i, y_i; W)\right]. \tag{3}$$

Here the term $\Delta(y_i, y)$ provides a confidence in classifying the i-th instance, i.e. the margin of the classifier, and defined as $\Delta(y_i, y) = 0$ if $y_i = y$, 1 otherwise.

Attribute label embedding (ALE) [14] uses an objective function which ensures that correct labels are ranked higher than incorrect ones, and its objective function is:

$$\sum_{y \in \mathcal{S}} \frac{\beta_{r\Delta(\mathbf{x}_i, y_i)}}{r\Delta(\mathbf{x}_i, y_i)} \max\left[0, \Delta\left(y_i, y\right) + F(\mathbf{x}_i, y; W) - F(\mathbf{x}_i, y_i; W)\right]. \tag{4}$$

Defining $r(\mathbf{x}_i, y_i)$ as the rank of y_i for instance $\mathbf{x}_i$; $r\Delta(\mathbf{x}_i, y_i)$ indicates how many classes achieve a comparable compatibility score considering the score of the correct class, i.e. y_i, is thus defined as

$$\sum_{y \in \mathcal{S}} \Delta\left(y_i, y\right) + F(\mathbf{x}_i, y; W) > F(\mathbf{x}_i, y_i; W), \tag{5}$$

where is a boolean function, equals 1 if its argument is true and 0 otherwise. Also note that $r(\mathbf{x}_i, y_i)$ is an upper-bound on the rank of y_i for $\mathbf{x}_i$. This being said, $\beta_{r\Delta(\mathbf{x}_i, y_i)}$ is equal to $\sum_{j=1}^{r(\mathbf{x}_i, y_i)} \alpha_j$, where α_j corresponds to the penalty incurred by moving from rank j to $j+1$ and defines a decreasing sequence $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_{|\mathcal{S}|} \geq 0$. Akata et al. propose to choose $\alpha_j = 1/j$ [14].

Structured joint embedding (SJE) [15] considers only the maximum violating class, i.e. an incorrect class, but still achieves the highest compatibility score. Thus, the objective function of SJE, inspired by the structured SVM, is defined as:

$$\max[0, \max_{y \in \mathcal{S}}[\Delta(y_i, y) + F(\mathbf{x}_i, y; W) - F(\mathbf{x}_i, y_i; W)]]. \tag{6}$$

We note that since SJE does not take into account the other incorrect classes, it tends to have low performance compared to DeViSE and ALE.

Embarrassingly simple zero-shot learning (ESZSL) [16] considers the following regularization term:

$$\gamma \|W \mathbf{m}_y\|_2 + \lambda \|W^T \mathbf{x}\|_2 + \beta \|W\|_{Fro}, \tag{7}$$

where γ, λ, and β are regularization parameters. Here, the first term controls the (Euclidean) norm of the projected class prototypes in the feature space, which allows for a fair comparison between prototypes even when the training set is highly imbalanced. The second term, on the other hand, bounds the projection of the training instances into the semantic space so that the projection does not depend on the distribution of the training features. Finally, the third term penalizes the Frobenius norm of the projection matrix W.

Semantic AutoEncoder (SAE) [8] first projects the feature space into the semantic space through W and then reconstructs the original feature space using the transpose of W. In doing so, SAE tries to minimize the error that may occur when reconstructing the original feature space. With this in mind, SAE attempts to minimize the following objective:

$$\underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \|\mathbf{x} - W\mathbf{m}_y\|_2 + \lambda\|W^T\mathbf{x} - \mathbf{m}_y\|_2, \tag{8}$$

where λ is a regularization parameter that can control the importance of the projection from the semantic space to the feature space.

We conclude this section with a note on how to optimize the objective functions defined above. Concretely, DeViSE, ALE, and SJE use stochastic gradient descent-based methods that perform early stopping to allow implicit regularization for W. ESZSL and SAE, on the other hand, deal with a convex objective function and can therefore provide a closed-form solution.

3.3. LVQ treatment for ZSL

In the previous section, we discussed the ZSL methods that can make instances and class prototypes compatible through a mapping W. From now on, we turn our attention to refining the positions of the class prototypes with respect to W learned using the instances of training classes. This enables us to connect ZSL's training and testing phases. For this purpose, we make use of the learning vector quantization (LVQ) algorithm [12]. In fact, LVQ helps to pull the prototypes towards their instances while pulling them away from the instances of the other classes. In this study, we explore a number of the most commonly used LVQ variants from the point of view of ZSL.

LVQ refers to a family of algorithms that aim to learn class prototypes representing class regions, i.e. Voronoi partitions [12, 35]. Starting from initial positions for the prototypes, LVQ updates the positions iterating over the training data. After the prototypes are learned, classification is performed according to the winner-takes-all rule, i.e. test instance is assigned to the class whose prototype is the nearest. Since ZSL also deals with class prototypes, a relationship between ZSL and LVQ is highly likely, which leads us to conduct this study.

Kohonen introduced the first versions of LVQ, including LVQ1, LVQ2.1, LVQ3, and OLVQ, about 30 years ago [12]. Since then, several variants have been developed [35–38]. Although it was initially developed as a heuristic method, its theoretical and algorithmical aspects, such as its generalization bounds and the loss functions associated with it, were later described [11, 39, 40]. In the following, we first review the initial versions of LVQ and then continue with its optimized versions.

3.3.1. The initial versions of LVQ for ZSL

The LVQ algorithms in this group employ a heuristic method to determine the positions of the class prototypes, and therefore do not require a cost function to be optimized. Nevertheless, their empirical results are still respectable compared to versions using more advanced methods [35].

LVQ1, by far the most popular LVQ algorithm, corrects only the winner prototype, i.e. the one closest to the instance of interest. If the winner prototype corresponds to the true class, then it is updated with

$$\mathbf{m}_y(t+1) = \mathbf{m}_y(t) + \alpha(t)\left[W^T\mathbf{x} - \mathbf{m}_y(t)\right], \tag{9}$$

else, it is updated with:

$$\mathbf{m}_y(t+1) = \mathbf{m}_y(t) - \alpha(t)\left[W^T\mathbf{x} - \mathbf{m}_y(t)\right]. \tag{10}$$

In (9), $\mathbf{m}_y$ is pulled towards $\mathbf{x}$; thus, it moves away from the class boundaries, while in (10), $\mathbf{m}_y$ is pulled away from $\mathbf{x}$. For $\alpha(t)$, it is a constant between 0 and 1. Alternatively, one may wish to decrease $\alpha(t)$ over time to reduce its effect in later iterations. A popular approach that we also consider in this study is:

$$\alpha(t) = \alpha(1 - \frac{t}{maxIter})$$

where α is an initial step size, $maxIter$ is the total number of iterations and t denotes time.

LVQ2.1 differs from LVQ1 in that it updates two prototypes simultaneously. The first is the one corresponding to the true class and is updated with (9), and the second is the one corresponding to the nearest class considering the other classes and is updated with (10). However, these updates only occur when an instance falls into a window of the decision border, which depends on whether the following inequality holds:

$$\min\left(\frac{dist_i}{dist_j}, \frac{dist_j}{dist_i}\right) > \frac{1-w}{1+w} \tag{11}$$

with w denoting width of the window. Larger w values can lead to unnecessary updates of class prototypes. Kohonen therefore recommends choosing a value between 0.2 and 0.3. Specifically for this study, we tried a range of values between 0.2 and 0.3 but did not find any noticeable difference in the performance of the ZSL methods equipped with LVQ2.1. However, since a value of 0.3 generally provides better results, we set w to 0.3 during the experiments. Moreover, in equation (11), $dist_i$ (resp. $dist_j$) denotes the Euclidean distance between instance of interest and the prototype of its true class (resp. that of the other closest class). With these update rules, LVQ2.1 attempts to approximate the Bayes decision boundaries. Nevertheless, LVQ2.1 has been criticized for being unstable because LVQ2.1 makes adjustments only in the presence of errors, i.e. when an instance is in a local window [35, 39].

OLVQ, another common LVQ variant, refers to optimized-learning LVQ and assigns an optimal learning rate to each prototype. Instead of considering a one-size-fits-all learning rate, OLVQ uses the following learning rate specific to $\mathbf{m}_y$:

$$\alpha_y(t+1) = \frac{\alpha_y(t)}{1 + s(t)\alpha_y(t)}, \tag{12}$$

where $s(t)$ is set to 1, if y is the true class of an instance, and -1 otherwise. As in LVQ1, only the closest prototype is updated, using either (9) or (10) depending on whether y is the true class. Moreover, it is common practice to set all initial learning rates to 0.3 [12]. The main advantage of OLVQ over the other two LVQ algorithms is that it converges quickly.

We note that there is another popular variant of LVQ introduced by Kohonen, LVQ3 [12], which we do not consider in this study. The reason is that LVQ3 requires multiple prototypes for each class, which is impossible under the conventional setting of ZSL. Therefore, we cannot make use of LVQ3 as a treatment for ZSL.

3.3.2. The generalized versions of LVQ for ZSL

The Kohonen versions of LVQ rely on heuristic arguments that raise concerns such as stability, generalizability, and convergence [40]. This has led to the development of versions that have solid mathematical foundations. In this context, Sato et al. introduced generalized LVQ (GLVQ), which considers an explicit cost function for the

LVQ for the first time in the literature [36]. Specifically, GLVQ tries to minimize the following cost function:

$$\sum_{i=1}^{N_{tr}} f(\mu_i) \text{ where } \mu_i = \frac{dist_i - dist_j}{dist_i + dist_j} \tag{13}$$

where $dist_i$ and $dist_j$ are the distances defined as in (11). Here, f is chosen to be a monotonically increasing function, such as sigmoid function [36, 37]. (13) is minimized with respect to the prototype vectors using the steepest descent method [36]. We note that minimizing (13) amounts to maximizing the hypothesis margin of an LVQ classifier due to term $dist_i - dist_j$ because the lower this term is, the more confidence the classifier gains [11].

GRLVQ [37] refers to generalized relevance learning vector quantization that combines GLVQ with metric learning [41]. Specifically, GRLVQ substitutes the conventional Euclidean distance used to determine the distances between instances and prototypes with the following metric:

$$\|W^T\mathbf{x} - \mathbf{m}_y\|_\Lambda^2 = (W^T\mathbf{x} - \mathbf{m}_y)^T\Lambda(W^T\mathbf{x} - \mathbf{m}_y), \tag{14}$$

where Λ is a diagonal matrix with entries $\Lambda_{i,i} = \lambda_i \in \mathbb{R}$ for $i \in \{1, ..., k\}$. Here λ_i scales the dimensions of the semantic space according to their relevance; hence, they are called relevance factors [37]. In practice, this means that one can use GRLVQ for ZSL to take into account the relevance of the attributes when computing the distances to the class prototypes in the semantic space. In fact, GRLVQ learns both the positions of the prototypes and the relevance factors. As in GLVQ, GRLVQ also employs a stochastic gradient descent for the optimization problem.

GMLVQ [38], which is referred to as generalized matrix learning vector quantization, estimates a full $k \times k$ matrix Λ for (14). This enables GMLVQ to consider correlations between the attributes.

Finally, in Figure 2, we summarize the proposed LVQ-empowered ZSL method in the form of a block diagram. This diagram starts with the training set and the class representations of both the training classes and test classes as input, proceeds to generate a projection matrix based on a ZSL method, then updates the class prototypes using an LVQ method of choice, and ends with the output of the refined class prototypes.

Algorithm 1 The LVQ-empowered ZSL method.

```
Input: Training set: $\mathcal{D}^{tr} = \{(\mathbf{x}_i, y_i) \in X \times \mathcal{S}\}_{i=1}^{N_{tr}}$ ($X \subset \mathbb{R}^d$), sets of seen and unseen classes: $\mathcal{S}$, and $\mathcal{U}$, class prototypes of the training and test classes $\{\mathbf{m}_y\} \subset \mathbb{R}^k$ ($y \in \mathcal{S}$ or $\mathcal{U}$), a ZSL method that can generate a projection matrix $W \in \mathbb{R}^{d \times k}$, a LVQ method, e.g., LVQ1, GLVQ,.., etc., max iteration number $maxIter$.
Output: Updated class prototypes $\{\mathbf{m}_y\} \subset \mathbb{R}^k$ ($y \in \mathcal{S}$ or $\mathcal{U}$) wrt the projection matrix $W$.
 1: Project the train instances $\mathbf{x} \in X$ into the semantic space using $W$, i.e. $W^T\mathbf{x}$.
 2: for $t = 1 : maxIter$ do
 3:     for $(\mathbf{x}, y) \in \mathcal{D}^{tr}$ do
 4:         Compare $W^T\mathbf{x}$ with all of the class prototypes $\{\mathbf{m}_y\}$ and find the nearest one: $\mathbf{m}_y^*$
 5:         if $\mathbf{m}_y^*$ corresponds to $y$ then
 6:             Update $\mathbf{m}_y^*$ so that it is pulled towards $\mathbf{x}$ based on the LVQ method of choice.
 7:         else
 8:             Update $\mathbf{m}_y^*$ so that it is pulled away from $\mathbf{x}$ based on the LVQ method of choice.
 9:         end if
10:     end for
11: end for
```

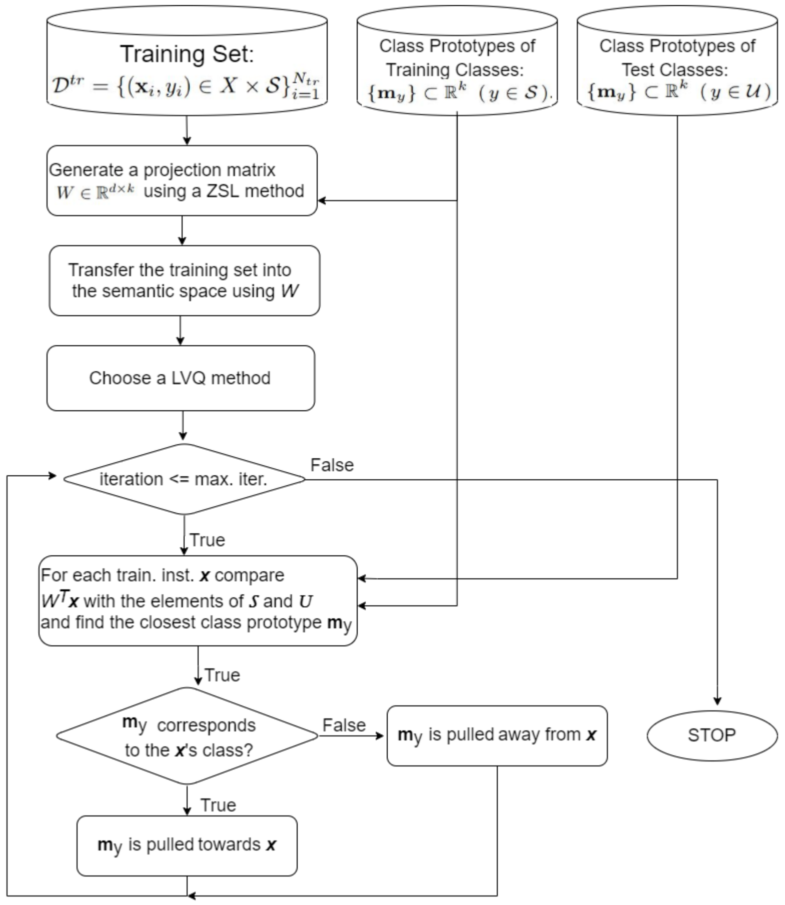

Training Set:
$\mathcal{D}^{tr} = \{(\mathbf{x}_i, y_i) \in X \times \mathcal{S}\}_{i=1}^{N_{tr}}$
Class Prototypes of Training Classes: $\{\mathbf{m}_y\} \subset \mathbb{R}^k \ (y \in \mathcal{S})$
Class Prototypes of Test Classes: $\{\mathbf{m}_y\} \subset \mathbb{R}^k \ (y \in \mathcal{U})$
Generate a projection matrix $W \in \mathbb{R}^{d \times k}$ using a ZSL method
Transfer the training set into the semantic space using W
Choose a LVQ method
iteration <= max. iter.
False
True
For each train. inst. $\boldsymbol{x}$ compare $W^T\boldsymbol{x}$ with the elements of $\mathcal{S}$ and $\mathcal{U}$ and find the closest class prototype $\mathbf{m}_y$
True
$\mathbf{m}_y$ corresponds to the $\boldsymbol{x}$'s class?
False
$\mathbf{m}_y$ is pulled away from $\boldsymbol{x}$
STOP
True
$\mathbf{m}_y$ is pulled towards $\boldsymbol{x}$

Figure 2. A block diagram for the LVQ-empowered ZSL method.

4. Experiments

In this section, we begin by introducing the datasets in which we are interested in this study. We then explain the settings of the ZSL methods and the LVQ methods, and finally present the results along with a detailed discussion.

4.1. Datasets

For this study, we use five benchmark datasets for evaluation: AWA1 [3], AWA2 [2], CUB [42], SUN [43], and aPY [44]. We extract features from these datasets following the procedure explained in a survey on ZSL [2]. Concretely, for datasets AWA1, AWA2, CUB, and SUN, we consider the entire image, while for aPY we consider bounding boxes for feature extraction. To extract the features, we use deep convolutional neural networks (CNNs) trained on ImageNet-1K resulting in 101-layered ResNet. We then use the 2048-dimensional top-layer pooling units of the network.

AWA1 is the original form of the famous animal with attributes dataset introduced by Lampert et al. [3]. It consists of images of 50 animals represented by 85 visual attributes. AWA2 contains images of the same 50 animals represented by the same 85 attributes but uses images from public sources such as Wikipedia and Filickr [2]. CUB and SUN are both fine-grained and thus challenging datasets, with CUB containing 200 bird species and SUN containing 717 scene categories [42, 43]. Finally, aPY contains aPascal/aYahoo objects from 32 categories [44]. More details about these datasets can be found in Table 1.

Table 1. Statistics of the benchmark datasets used for evaluation.

Dataset	# Attributes	# Seen cls.	# Unseen cls.	# Train. insts.	# Test insts.	# Total insts.
AWA1 [3]	85	40	10	24,295	6180	30,475
AWA2 [2]	85	40	10	30,337	6985	37,322
CUB [42]	312	150	50	8855	2933	11,788
SUN [43]	102	645	72	12,900	1440	14340
aPY [44]	64	20	12	12,695	2644	15,339

4.2. Evaluation criteria

In ZSL, the Top-1 accuracy is the most common measure for evaluating a classifier's performance in classifying unseen classes, which is basically an accuracy averaged over the classes [2, 8, 14]. This prevents the accuracy measure from being dominated by densely populated classes. In fact, the Top-1 accuracy is nothing more than a class-averaged recall measure, since it is concerned with the ratio of correctly predicted instances per class. Accordingly, the Top-1 accuracy is calculated as follows:

$$Top-1 = Recall = \frac{1}{|\mathcal{U}|}\sum_{c\in\mathcal{U}} \frac{\#\text{ correct predictions in } c}{\#\text{ instances in } c}. \tag{15}$$

In addition to the Top-1 accuracy, there are other performance measures that are commonly used in machine learning/computer vision, namely precision, and F1-score. As in the recall measure, to compute the precision measure for ZSL, we can estimate the precision per (unseen) class and then average them which can be defined as follows:

$$Precision = \frac{1}{|\mathcal{U}|}\sum_{c\in\mathcal{U}} \frac{\#\text{ correct predictions in } c}{\#\text{ instances predicted as } c}. \tag{16}$$

F1-score combines the precision and recall and is defined as the harmonic mean of the two:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{17}$$

4.3. Results and discussion

Recall that we use five different methods to obtain a mapping from feature space to semantic space: DeViSE [13], ALE [14], SJE [15], ESZSL [16], and SAE [8]. Of these, DeViSE, ALE, and SJE use gradient descent optimization, where the learning rate plays a crucial role in the success of the optimization problem. We therefore report results for a range of different learning rates. In doing so, we aim to show the effectiveness of the LVQ algorithms for the ZSL methods in different situations. Specifically, we let the learning rate range over the set $\{0.0001, 0.001, 0.01, 0.1, 1\}$. For each dataset and for each learning rate, we obtain the Top-1 accuracy for the gradient-based LVQ methods, i.e. DeViSE, ALE, and SJE, and for their LVQ extensions. Tables 2–4 show these results.

Unlike the above mentioned ZSL methods, ESZSL [16] has a closed form solution; thus, no iterative algorithm, such as gradient descent, is required, which, in turn, makes ESZSL independent of the learning rate. However, ESZSL has three regularization parameters γ, λ, and β that must be set, see (7). In [16], it is recommended to multiply γ and λ to get β, which reduces the number of parameters to 2, i.e. γ and λ. Furthermore, the survey [2] and original paper [16] state that the ideal value of the parameter λ is 3, independent of the parameter gamma. This leaves us with only one parameter to set, namely gamma γ. As a result, we give the Top-1 accuracy scores of ESZSL for different values of γ. For this purpose, we consider the set $\{-2, -1, 0, 1, 2\}$ from which γ takes a value. This is consistent with the survey [2] and the original paper [16]. SAE, on the other hand, has a regularization parameter λ that need to be carefully tuned. Considering the related works, we use the following parameters for λ: $\{0.1, 1, 5, 10, 50]\}$. Tables 5 and 6 show the Top-1 accuracy scores of ESZSL and SAE, respectively.

Table 2. Top-1 accuracy scores of DeViSE [13] and its LVQ extensions.

Dataset	Learn. rate	DeViSE	DeViSE +LVQ1	DeViSE +LVQ2.1	DeViSE +OLVQ	DeViSE +GLVQ	DeViSE +GRLVQ	DeViSE +GMLVQ
AWA1	0.0001	0.428	0.444	0.393	**0.455**	0.428	0.411	0.415
	0.001	0.563	0.575	0.426	**0.583**	0.569	0.505	0.564
	0.01	0.552	0.552	0.378	**0.573**	0.562	0.531	0.554
	0.1	0.529	**0.548**	0.444	0.544	0.535	0.485	0.529
	1	0.495	0.435	0.442	0.495	**0.499**	0.433	0.495
AWA2	0.0001	0.571	0.571	0.434	**0.585**	0.573	0.509	0.509
	0.001	0.576	0.576	0.434	**0.589**	0.576	0.511	0.508
	0.01	0.584	**0.601**	0.444	0.597	0.581	0.511	0.52
	0.1	0.528	0.528	0.443	**0.546**	0.525	0.497	0.528
	1	0.504	**0.524**	0.441	0.504	0.506	0.392	0.504
CUB	0.0001	0.02	0.065	0.0673	**0.071**	0.027	0.02	0.02
	0.001	0.072	0.276	**0.294**	0.245	0.073	0.101	0.107
	0.01	0.076	0.246	**0.295**	0.248	0.173	0.103	0.106
	0.1	0.423	0.451	**0.533**	0.462	0.422	0.456	0.465
	1	0.441	0.461	**0.547**	0.477	0.444	0.436	0.438
SUN	0.0001	0.334	**0.506**	0.469	0.468	0.351	0.382	0.387
	0.001	0.541	**0.588**	0.545	0.582	0.542	0.553	0.553
	0.01	0.556	0.568	0.432	**0.572**	0.556	0.544	0.564
	0.1	0.519	0.338	0.308	0.444	0.521	**0.541**	0.537
	1	0.537	0.336	0.299	0.444	**0.541**	0.497	0.538
aPY	0.0001	0.083	0.171	**0.221**	0.181	0.099	0.099	0.099
	0.001	0.083	0.205	**0.224**	0.169	0.091	0.099	0.083
	0.01	0.268	0.282	**0.319**	0.309	0.274	0.273	0.274
	0.1	0.318	0.369	0.352	**0.384**	0.324	0.331	0.331
	1	0.333	0.361	0.369	**0.394**	0.336	0.338	0.338

The immediate result that can be derived from Tables 2–6 is that the accuracy of the ZSL methods improves when they are equipped with an LVQ algorithm, regardless of the dataset used. More precisely, it is useful to update the positions of the class prototypes in the semantic space with respect to the learned mapping

Table 3. Top-1 accuracy scores of ALE [14] and its LVQ extensions.

Dataset	Learn. rate	ALE	ALE +LVQ1	ALE +LVQ2.1	ALE +OLVQ	ALE +GLVQ	ALE +GRLVQ	ALE +GMLVQ
AWA1	0.0001	0.457	0.467	**0.499**	0.472	0.455	0.497	0.497
	0.001	0.565	0.582	0.539	0.577	0.566	0.606	**0.607**
	0.01	0.561	0.587	0.512	0.568	0.567	**0.606**	**0.606**
	0.1	0.565	0.577	0.498	0.586	0.569	0.604	**0.605**
	1	0.547	0.547	0.499	**0.579**	0.547	0.529	0.531
AWA2	0.0001	0.481	0.481	0.465	0.482	0.476	0.487	**0.488**
	0.001	0.518	**0.581**	0.506	0.525	0.551	0.556	0.557
	0.01	0.522	**0.583**	0.485	0.529	0.557	0.563	0.565
	0.1	0.532	**0.564**	0.485	0.545	0.561	0.552	0.552
	1	0.515	**0.561**	0.481	0.516	0.521	0.548	0.548
CUB	0.0001	0.101	**0.325**	0.301	0.248	0.181	0.138	0.153
	0.001	0.334	0.409	**0.441**	0.408	0.359	0.377	0.382
	0.01	0.471	0.488	**0.541**	0.492	0.473	0.436	0.437
	0.1	0.484	0.497	**0.547**	0.501	0.485	0.447	0.451
	1	0.474	0.488	**0.543**	0.495	0.477	0.457	0.457
SUN	0.0001	0.329	0.444	**0.464**	0.426	0.369	0.361	0.363
	0.001	0.522	0.549	0.551	**0.553**	0.527	0.543	0.543
	0.01	0.609	**0.623**	0.622	0.621	0.611	0.614	0.615
	0.1	0.617	0.626	**0.633**	0.627	0.621	0.626	0.627
	1	0.603	**0.613**	0.609	0.611	0.602	0.612	**0.613**
aPY	0.0001	0.225	0.292	**0.359**	0.296	0.316	0.271	0.213
	0.001	0.303	0.316	**0.356**	0.339	0.301	0.301	0.299
	0.01	0.291	0.309	**0.361**	0.331	0.302	0.298	0.299
	0.1	0.333	0.354	**0.392**	0.361	0.359	0.284	0.284
	1	0.367	0.371	0.384	0.381	0.364	**0.407**	**0.407**

using an LVQ method. The improvement is up to 20%, especially when the ZSL method performs poorly, which occurs due to inappropriate parameter settings. When the ZSL methods show better performance, applying the LVQ algorithms results in a smaller increase in the accuracy. In this case, the improvement ranges from 2% to 10% for DeViSE, 2% to 6% for ALE, 0.05% to 2% for SJE, 0.05% to 2% for ESZSL, and 1% to 10% for SAE. This implies that DeViSE and SAE gain the most from the LVQ algorithms, but SJE and ESZSL gain the least. We also note that the highest improvement in the accuracy is obtained with CUB and aPY datasets in general, where the improvements are 4.5% on average.

Considering the parameters that lead to the highest accuracy for each ZSL method and for each dataset, we also calculated the class-averaged precision and F1 scores as in (16) and (17). Tables 7 and 8 show the precision and F1 scores of the original ZSL methods and their LVQ extensions, respectively.

Based on the precision scores shown in Table 7, as in the case of the Top-1 accuracy scores, we claim that using an LVQ algorithm together with a ZSL classifier increases the precision of the classifier. The increase here is up to 11%, though it is mostly between 1% and 4%. Comparing the ZSL methods based on their precision gain resulting from the use of an LVQ algorithm, SAE is in the first place with an average gain of 5%, followed

Table 4. Top-1 accuracy scores of SJE [15] and its LVQ extensions.

Dataset	Learn. Rate	SJE	SJE +LVQ1	SJE +LVQ2.1	SJE +OLVQ	SJE +GLVQ	SJE +GRLVQ	SJE +GMLVQ
AWA1	0.0001	0.101	0.283	**0.371**	0.163	0.175	0.101	0.101
	0.001	0.206	0.372	**0.399**	0.388	0.381	0.271	0.331
	0.01	0.567	**0.591**	0.521	0.578	0.562	0.515	0.515
	0.1	0.586	**0.602**	0.505	0.591	0.582	0.577	0.579
	1	0.588	0.588	0.483	0.618	0.584	0.593	**0.594**
AWA2	0.0001	0.131	0.293	**0.355**	0.201	0.205	0.131	0.131
	0.001	0.307	0.392	**0.401**	0.399	0.391	0.394	0.394
	0.01	0.567	0.581	0.572	**0.588**	0.574	0.574	0.576
	0.1	0.583	0.583	0.577	**0.588**	0.577	0.574	0.574
	1	0.508	0.503	0.528	**0.545**	0.533	0.533	0.533
CUB	0.0001	0.108	0.259	0.249	**0.262**	0.132	0.109	0.184
	0.001	0.362	0.409	0.331	**0.414**	0.381	0.373	0.383
	0.01	0.489	0.481	0.389	0.479	**0.491**	0.465	**0.49**
	0.1	0.495	0.393	0.377	0.388	0.495	0.503	**0.512**
	1	**0.493**	0.383	0.383	0.386	**0.493**	0.487	0.487
SUN	0.0001	0.101	0.284	0.281	**0.285**	0.125	0.106	0.126
	0.001	0.416	**0.499**	0.475	0.497	0.437	0.429	0.444
	0.01	0.536	0.526	0.483	**0.541**	**0.541**	0.511	0.533
	0.1	0.531	0.499	0.483	0.494	0.538	0.507	**0.541**
	1	0.534	0.489	0.461	0.496	0.538	0.534	**0.551**
aPY	0.0001	0.083	0.226	0.214	**0.244**	0.107	0.135	0.135
	0.001	0.251	0.277	0.194	0.286	**0.302**	0.287	0.297
	0.01	0.312	0.321	0.255	0.322	**0.345**	**0.345**	**0.345**
	0.1	0.331	0.336	0.288	**0.341**	0.339	0.335	0.336
	1	0.341	0.341	0.298	**0.346**	0.342	0.342	0.341

by DeViSE with an average gain of 3%. The precision gain is the lowest for SJE and ESZSL. This result is consistent with that of the Top-1 accuracy results. As with the Top-1 scores, the highest gain in precision is observed using an LVQ algorithm on the CUB dataset.

4.4. Comparison of the LVQ versions

It may be also necessary to figure out which LVQ is the most appropriate for each ZSL method. For this purpose, we consider the F1-scores listed in Table 8, as they reflect both the Top-1 accuracy, i.e. recall and the precision scores. Based on these scores, we compare the average ranks of the LVQ methods for each ZSL method. In particular, for each ZSL method, and for each dataset, we rank the results, i.e. the F1 scores; obtained with the original form of the ZSL method of interest and its LVQ extensions. We then average these ranks across all datasets. The rows in Table 9 correspond to these average ranks for each ZSL method.

In general, we conclude that LVQ1 and OLVQ achieve the highest improvement for the ZSL methods; thus, they are recommended to be used with the ZSL methods considered in this paper. These LVQ versions are followed by two generalized LVQ methods GLVQ and GMLVQ in terms of their contribution to the success

Table 5. Top-1 accuracy scores of ESZSL [16] and its LVQ extensions.

Dataset	Gamma γ	ESZSL	ESZSL +LVQ1	ESZSL +LVQ2.1	ESZSL +OLVQ	ESZSL +GLVQ	ESZSL +GRLVQ	ESZSL +GMLVQ
AWA1	-2	0.488	0.494	0.487	0.492	**0.501**	0.485	0.485
	-1	0.553	**0.556**	0.538	0.553	0.553	0.549	0.548
	0	0.556	0.556	0.551	0.556	0.566	0.557	**0.559**
	1	0.506	0.506	0.506	0.506	**0.524**	0.506	0.506
	2	0.499	0.501	0.501	0.499	**0.511**	0.499	0.499
AWA2	-2	0.501	0.501	0.475	**0.502**	0.498	0.485	0.488
	-1	**0.547**	**0.547**	0.522	**0.547**	0.541	0.499	0.503
	0	0.537	0.537	0.528	0.537	**0.548**	0.523	0.523
	1	0.489	0.492	0.489	0.489	**0.516**	0.489	0.489
	2	0.473	0.474	0.473	0.474	**0.505**	0.473	0.473
CUB	-2	0.439	0.443	**0.445**	0.444	0.442	0.423	0.426
	-1	0.485	0.485	0.492	0.486	0.486	**0.497**	**0.497**
	0	0.472	**0.475**	0.473	0.472	0.473	0.472	0.474
	1	0.428	0.429	0.429	0.428	**0.431**	0.428	0.428
	2	0.433	**0.439**	**0.439**	0.433	0.433	0.433	0.433
SUN	-2	0.468	0.471	**0.479**	0.471	0.469	0.471	0.477
	-1	0.501	0.504	**0.508**	0.504	0.501	0.501	0.501
	0	0.552	0.556	**0.558**	0.555	0.552	0.552	**0.558**
	1	0.547	**0.549**	0.548	0.548	0.555	0.548	0.548
	2	0.554	0.554	0.554	0.554	**0.555**	**0.555**	**0.555**
aPY	-2	0.304	**0.345**	0.322	0.321	0.304	0.334	0.334
	-1	0.344	**0.352**	0.341	0.346	0.344	0.343	0.346
	0	0.367	0.367	0.366	**0.369**	0.367	0.365	0.367
	1	0.335	**0.338**	0.335	0.335	0.335	0.336	0.336
	2	0.314	0.314	0.314	0.314	0.314	0.314	0.314

of the ZSL methods. The case of LVQ2.1 is worth mentioning here. Although LVQ2.1 is the second best LVQ extension for SAE and increases the Top-1 accuracy of SAE by up to 15% for aPY, LVQ2.1 tends to degrade the classification ability of SJE and ESZSL, see Table 9. This can be attributed to the unstable nature of LVQ2.1, as mentioned in 3.3.1. Therefore, the use of LVQ2.1 in conjunction with the ZSL methods raises concerns.

We also want to determine if the differences between the observed average ranks, i.e. those shown in Table 9, are statistically significant. For this purpose, we performed the Freidman test [45]. For each ZSL method, with seven classifiers (the original one plus six LVQ extensions) and five datasets, the test statistic of the Freidman test, F_F, follows the F distribution with $7-1=6$ and $(7-1)\times(5-1)=24$ degrees of freedom. The critical value of $F(6,24)$ is 2.51 at 95% confidence level. Since the test statistics F_F for the different ZSL methods are always below the critical value of 2.51 and vary between 0.93 and 1.13, we cannot reject the null hypothesis that the average ranks are significantly different from the mean rank of 4. We thus conclude that the conducted Freidman test is not powerful enough to reject the null hypothesis that all LVQ extensions contribute equally to the accuracy of a ZSL method.

Table 6. Top-1 accuracy scores of SAE [8] and its LVQ extensions.

Dataset	Reg. Param. λ	SAE	SAE +LVQ1	SAE +LVQ2.1	SAE +OLVQ	SAE +GLVQ	SAE +GRLVQ	SAE +GMLVQ
AWA1	0.1	0.515	0.521	0.526	0.521	0.532	**0.561**	**0.561**
	1	0.526	0.535	0.543	0.535	**0.571**	0.569	0.569
	5	0.511	0.519	0.542	0.519	**0.561**	0.553	0.553
	10	0.505	0.515	0.545	0.514	**0.551**	0.549	0.548
	50	0.494	0.511	**0.537**	0.511	0.531	0.531	0.529
AWA2	0.1	0.498	0.503	0.515	0.504	**0.559**	0.551	0.551
	1	0.514	0.528	0.543	0.528	0.568	**0.571**	**0.571**
	5	0.493	0.521	0.548	0.507	0.542	**0.571**	0.568
	10	0.482	0.494	0.549	0.496	0.545	**0.571**	0.568
	50	0.469	0.485	**0.536**	0.482	**0.536**	0.526	0.522
CUB	0.1	0.322	0.334	0.339	0.337	0.312	**0.349**	**0.349**
	1	0.289	0.324	0.326	0.331	0.311	**0.353**	**0.353**
	5	0.331	**0.391**	0.383	0.369	0.349	0.386	0.387
	10	0.354	0.384	**0.402**	0.383	0.363	0.396	0.396
	50	0.382	0.408	**0.433**	0.406	0.401	0.424	0.429
SUN	0.1	0.523	0.551	0.546	0.551	0.531	0.555	**0.556**
	1	**0.534**	0.511	0.502	0.512	0.484	0.512	0.512
	5	0.509	**0.511**	0.493	0.497	0.482	0.489	0.491
	10	0.496	**0.522**	0.504	0.504	0.474	0.494	0.495
	50	0.488	**0.515**	0.507	0.491	0.483	0.493	0.499
aPY	0.1	0.086	0.233	0.242	0.234	0.234	**0.266**	**0.266**
	1	0.166	0.246	**0.273**	0.248	0.243	0.259	0.261
	5	0.144	0.264	**0.305**	0.266	0.261	0.274	0.281
	10	0.131	**0.322**	0.319	0.295	0.273	0.293	0.301
	50	0.102	0.304	**0.334**	0.303	0.301	0.305	0.303

5. Conclusions and future work

Zero-shot learning (ZSL) refers to learning to classify images (instances) of classes for which there are no labeled instances in the training set, i.e. unseen (test) classes. In doing so, ZSL leverages high-level descriptions of the classes given in the form of a list of attributes corresponding to semantically meaningful properties. These attributes form class prototypes the in semantic space. ZSL methods generally aim to map the instances of test classes in the feature space (visual space) to the semantic space. However, this mapping is learned through training classes, i.e. those that are present in the training phase, which leads to the domain shift problem.

In this study, we suggest employing the LVQ algorithm for ZSL. This offers a number of advantages. Firstly, it updates the prototypes of the test classes with respect to learned mapping, which helps to reduce the discrepancy between the training and testing phases of ZSL and thus alleviates the effects of the domain shift problem. Another advantage is that LVQ helps to separate class prototypes in the semantic space, which is much less discriminative compared to the feature space. As a result, we obtain better classification results. Furthermore, since LVQ is a prototype learning method, this study sheds light on the link between ZSL and prototype learning, which has not been well studied so far, even though ZSL inherently uses class prototypes.

Table 7. Precision scores of the ZSL methods and their LVQ extensions.

Dataset	ZSL method						
	DeViSE	DeViSE +LVQ1	DeViSE +LVQ2.1	DeViSE +OLVQ	DeViSE +GLVQ	DeViSE +GRLVQ	DeViSE +GMLVQ
AWA1	0.498	0.498	0.401	**0.529**	0.508	0.473	0.503
AWA2	0.502	0.502	0.384	**0.511**	0.501	0.488	0.493
CUB	0.548	**0.579**	0.564	0.571	0.549	0.546	0.547
SUN	0.591	**0.596**	0.495	0.611	0.589	0.605	0.594
aPY	0.201	**0.266**	0.242	0.215	0.203	0.208	0.208
Dataset	ALE	ALE +LVQ1	ALE +LVQ2.1	ALE +OLVQ	ALE +GLVQ	ALE +GRLVQ	ALE +GMLVQ
AWA1	0.524	0.543	0.522	0.543	**0.558**	0.554	0.554
AWA2	0.535	**0.541**	0.514	0.531	0.528	0.453	0.462
CUB	0.546	0.551	**0.566**	0.546	0.552	0.511	0.512
SUN	**0.652**	**0.652**	0.643	**0.652**	**0.652**	0.648	0.648
aPY	**0.314**	0.301	0.289	0.288	0.292	0.313	0.311
Dataset	SJE	SJE +LVQ1	SJE +LVQ2.1	SJE +OLVQ	SJE +GLVQ	SJE +GRLVQ	SJE +GMLVQ
AWA1	0.577	**0.617**	0.411	0.604	0.571	0.543	0.541
AWA2	0.583	0.583	0.524	**0.585**	0.582	0.509	0.502
CUB	0.509	0.463	0.451	0.504	0.517	0.509	**0.511**
SUN	**0.558**	0.542	0.495	**0.558**	**0.558**	0.527	0.557
aPY	0.181	0.161	0.158	0.156	0.177	0.162	**0.188**
Dataset	ESZSL	ESZSL +LVQ1	ESZSL +LVQ2.1	ESZSL +OLVQ	ESZSL +GLVQ	ESZSL +GRLVQ	ESZSL +GMLVQ
AWA1	0.523	0.523	0.518	0.523	**0.531**	0.525	0.526
AWA2	0.515	0.515	0.508	0.515	**0.519**	0.507	0.507
CUB	0.498	0.499	0.503	0.499	0.501	**0.511**	**0.511**
SUN	0.588	0.588	0.588	0.588	0.588	0.588	0.588
aPY	0.226	0.238	0.229	0.226	0.226	0.241	**0.243**
Dataset	SAE	SAE +LVQ1	SAE +LVQ2.1	SAE +OLVQ	SAE +GLVQ	SAE +GRLVQ	SAE +GMLVQ
AWA1	0.495	0.544	0.545	0.495	0.586	0.678	**0.682**
AWA2	0.563	0.563	0.562	0.563	0.589	**0.614**	**0.614**
CUB	0.481	0.485	**0.571**	0.481	0.495	0.517	0.521
SUN	0.612	**0.645**	0.632	0.628	0.614	0.605	0.605
aPY	0.109	0.107	**0.132**	0.113	0.109	0.115	0.116

We consider a range of LVQ algorithms from the initial versions to more advanced ones. For the ZSL methods that enable mapping from the feature space to the semantic space, we utilize DeViSE [13], ALE [14], SJE [15], ESZSL [16], and SAE [8]. Combining the ZSL methods with the LVQ algorithms, we obtain their LVQ extensions, which have been shown to outperform their original counterparts in the experiments on five benchmark datasets. We measure the performance of the ZSL methods and their LVQ extensions in terms of

Table 8. F1 scores of the ZSL methods and their LVQ extensions.

Dataset	ZSL method						
	DeViSE	DeViSE +LVQ1	DeViSE +LVQ2.1	DeViSE +OLVQ	DeViSE +GLVQ	DeViSE +GRLVQ	DeViSE +GMLVQ
AWA1	0.524	0.524	0.408	**0.549**	0.532	0.475	0.528
AWA2	0.537	0.537	0.401	**0.547**	0.535	0.446	0.449
CUB	0.488	0.511	**0.556**	0.519	0.489	0.468	0.471
SUN	0.573	**0.582**	0.433	0.591	0.572	0.571	0.578
aPY	0.251	**0.297**	0.293	0.278	0.253	0.248	0.249
Dataset	ALE	ALE +LVQ1	ALE +LVQ2.1	ALE +OLVQ	ALE +GLVQ	ALE +GRLVQ	ALE +GMLVQ
AWA1	0.543	0.549	0.522	0.554	0.565	**0.578**	**0.578**
AWA2	0.531	**0.539**	0.455	0.532	0.546	0.501	0.529
CUB	0.502	0.513	**0.557**	0.515	0.508	0.476	0.478
SUN	0.635	**0.643**	0.636	0.642	0.636	0.634	0.636
aPY	0.307	0.294	0.311	**0.314**	0.288	0.294	0.292
Dataset	SJE	SJE +LVQ1	SJE +LVQ2.1	SJE +OLVQ	SJE +GLVQ	SJE +GRLVQ	SJE +GMLVQ
AWA1	0.583	**0.601**	0.459	0.611	0.577	0.548	0.549
AWA2	0.586	0.583	0.549	0.587	**0.588**	0.525	0.528
CUB	0.501	0.419	0.414	0.491	**0.505**	0.501	0.502
SUN	0.547	0.534	0.488	0.548	0.549	0.519	**0.551**
aPY	**0.236**	0.213	0.208	0.211	0.233	0.221	0.222
Dataset	ESZSL	ESZSL +LVQ1	ESZSL +LVQ2.1	ESZSL +OLVQ	ESZSL +GLVQ	ESZSL +GRLVQ	ESZSL +GMLVQ
AWA1	0.539	0.539	0.534	0.539	**0.545**	0.537	0.538
AWA2	0.526	0.526	0.518	0.526	**0.533**	0.515	0.515
CUB	0.492	0.492	0.496	0.492	0.493	**0.503**	**0.503**
SUN	0.571	0.577	0.561	**0.581**	0.571	0.564	0.574
aPY	0.272	**0.285**	0.274	0.273	0.272	0.279	0.281
Dataset	SAE	SAE +LVQ1	SAE +LVQ2.1	SAE +OLVQ	SAE +GLVQ	SAE +GRLVQ	SAE +GMLVQ
AWA1	0.514	0.542	0.544	0.514	0.579	**0.623**	**0.624**
AWA2	0.545	0.545	0.552	0.545	0.578	**0.592**	**0.592**
CUB	0.422	0.441	**0.492**	0.438	0.442	0.466	0.471
SUN	0.539	**0.572**	0.561	0.564	0.541	0.554	0.554
aPY	0.151	0.151	**0.178**	0.155	0.151	0.159	0.161

(class-averaged) Top-1 accuracy (recall) and precision scores. Based on the Top-1 accuracy scores, we manage to improve the performance of DeViSE and SAE by up to 10%, while the improvement for the other ZSL methods is around 2%. Based on the precision scores, we reach similar conclusions. We also try to compare the LVQ algorithms in terms of their contribution to improving the success of the ZSL methods. For this purpose, for each ZSL method and for each dataset, we rank the F1 scores of the original ZSL method and its LVQ

Table 9. The average ranks of the original ZSL methods and their LVQ extensions based on the F1 scores.

ZSL method	Original	+LVQ1	+LVQ2.1	+OLVQ	+GLVQ	+GRLVQ	+GMLVQ
DeViSE	4.2	2.6	4.8	**1.6**	3.8	6.4	4.6
ALE	4.8	3.1	4.2	**2.4**	3.8	5.2	4.5
SJE	2.9	4.4	6.6	3.4	**2**	5.3	3.4
ESZSL	4.6	**3**	5.2	3.6	3.4	4.6	3.6
SAE	6.5	4.6	2.6	4.9	4.4	2.8	**2.2**

extensions, and then average these ranks accross the datasets. The averaged ranks indicate that LVQ1 and OLVQ are the largest contributors for most of the ZSL methods considered, followed by GLVQ and GMLVQ. Therefore, we recommend using LVQ1 and OLVQ to improve a ZSL method with the LVQ algorithm.

The work presented in this paper can be extended in several directions. One may consider multiple prototypes for training and test classes, which is consistent with the nature of the LVQ algorithm. In this way, a famous LVQ variant LVQ3 becomes applicable to ZSL. Indeed, other LVQ variants can also be considered for ZSL, as we do in this work. In particular, the advanced versions GRLVQ [37] and GMLVQ [38] produce a matrix with entries corresponding to the relevance of the attributes in the semantic space. Using this matrix in a Mahalanobis metric, one can perform metric learning in the semantic space. Finally, without requiring any adaptation, the proposed method can be applied to few-shot learning, where the test classes have few labeled instances.

References

[1] Palatucci M, Pomerleau D, Hinton GE, Mitchell TM. Zero-shot learning with semantic output codes. Advances in neural information processing systems 2009; 22.

[2] Xian Y, Lampert C, Schiele B, Akata Z. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. IEEE transactions on pattern analysis and machine intelligence 2018; 41 (9): 2251-2265. doi: 10.1109/TPAMI.2018.2857768

[3] Lampert C, Nickisch H, Harmeling S. Attribute-based classification for zero-shot visual object categorization. IEEE transactions on pattern analysis and machine intelligence 2013; 36 (3): doi:453-465. 10.1109/TPAMI.2013.140

[4] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems 2013; 26.

[5] Fu Y, Hospedales TM, Xiang T, Gong S. Transductive multi-view zero-shot learning. IEEE transactions on pattern analysis and machine intelligence 2015; 37 (11): 2332-2345. doi:10.1109/TPAMI.2015.2408354

[6] Wang W, Zheng VW, Yu H, Miao C. A survey of zero-shot learning: Settings, methods, and applications. ACM Transactions on Intelligent Systems and Technology (TIST)2019; 10 (2): 1-37. doi:10.1145/3293318

[7] Pan SJ, Yang Q. A survey on transfer learning. IEEE Transactions on knowledge and data engineering 2009; 22 (10): 1345-1359. doi:10.1109/TKDE.2009.191.

[8] Kodirov E, Xiang T, Gong S. Semantic autoencoder for zero-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition 2017; 3174-3183. doi:10.1109/CVPR.2017.473

[9] Kodirov E, Xiang T, Fu Z, Gong S. Unsupervised domain adaptation for zero-shot learning. In Proceedings of the IEEE international conference on computer vision 2015; 2452-2460. doi:10.1109/ICCV.2015.282

[10] Rohrbach M, Ebert S, Schiele B . Transfer learning in a transductive setting. Advances in neural information processing systems 2013; 26.

[11] Crammer K, Gilad-Bachrach R, Navot A, Tishby N. Margin analysis of the LVQ algorithm. Advances in neural information processing systems 2002; 15.

[12] Kohonen T.Self-Organizing Maps. 3rd ed. Berlin, Heidelberg: Springer, 2001. doi:10.1007/978-3-642-56927-2

[13] Frome A, Corrado GS, Shlens J, Bengio S, Dean J et al. Devise: A deep visual-semantic embedding model. Advances in neural information processing systems 2013; 26.

[14] Akata Z, Perronnin F, Harchaoui Z, Schmid C. Label-embedding for image classification. IEEE transactions on pattern analysis and machine intelligence 2015; 38 (7); 1425-1438. doi:10.1109/TPAMI.2015.2487986

[15] Akata Z, Reed S, Walter D, Lee H, Schiele B. (2015). Evaluation of output embeddings for fine-grained image classification. In Proc. of the IEEE conference on computer vision and pattern recognition 2015; 2927-2936. doi:10.1109/CVPR.2015.7298911

[16] Romera-Paredes B, Torr P. An embarrassingly simple approach to zero-shot learning. In International conference on machine learning 2015; 2152-2161. doi:10.1007/978-3-319-50077

[17] Xian Y, Akata Z, Sharma G, Nguyen Q, Hein M et al. Latent embeddings for zero-shot classification. In Proc. of the IEEE conference on computer vision and pattern recognition 2016;69-77. doi:10.1109/CVPR.2016.15

[18] Socher R, Ganjoo M, Manning C, Ng A. Zero-shot learning through cross-modal transfer. Advances in neural information processing systems 2013; 26.

[19] Xu W, Xian Y, Wang J, Schiele B, Akata Z. Attribute prototype network for zero-shot learning. Advances in Neural Information Processing Systems 2020; 33: 21969-21980.

[20] Larochelle H, Erhan D, Bengio Y. Zero-data learning of new tasks. In AAAI 2008; 1 (2): 3-13.

[21] Gavves E, Mensink T, Tommasi T, Snoek CG, Tuytelaars T. Active transfer learning with zero-shot priors: Reusing past datasets for future tasks. In Proc. of the IEEE International Conference on Computer Vision 2015; 2731-2739. doi:10.1109/ICCV.2015.313

[22] Mensink T, Verbeek J, Perronnin F, Csurka G. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In European Conference on Computer Vision 2012; 488-501. doi:10.1007/978-3-642-33709

[23] Jiang H, Wang R, Shan S, Chen X. Adaptive metric learning for zero-shot recognition. IEEE Signal Processing Letters 26 (9):1270-1274. doi:10.1109/LSP.2019.2917148

[24] Zhang Z, Saligrama V. Zero-shot learning via semantic similarity embedding. In Proc. of the IEEE international conference on computer vision 2015; 4166-4174. doi:10.1109/ICCV.2015.474

[25] Biehl M, Hammer B, Villman T. Prototype-based models in machine learning. Wiley Interdisciplinary Reviews: Cognitive Science 2016; 7 (2): 92-111. doi:10.1002/wcs.1378

[26] Zhang X, Gui S, Zhu Z, Zhao Y, Liu J. Hierarchical prototype learning for zero-shot recognition. IEEE Transactions on Multimedia 2019; 22 (7): 1692-1703. doi: 10.1109/TMM.2019.2959433

[27] Ji Z, Cui B, Yu Y, Pang Y, Zhang Z. Zero-shot classification with unseen prototype learning. Neural computing and applications 2021; 1-11. doi:10.1007/s00521-021-05746-9

[28] Sariyildiz MB, Cinbis RG. Gradient matching generative networks for zero-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019; 2168-2178. doi:10.1109/CVPR.2019.00227

[29] Reed S, Akata Z, Yan X, Logeswaran L, Schiele B et al. Generative adversarial text to image synthesis. In International conference on machine learning 2016; 1060-1069.

[30] Li K, Min MR, Fu Y. Rethinking zero-shot learning: A conditional visual classification perspective. In Proc. of the IEEE/CVF International Conference on Computer Vision 2019; 3583-3592. doi: 10.1109/ICCV.2019.00368

[31] Keshari R, Singh R, Vatsa M. Generalized zero-shot learning via over-complete distribution. In Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020; 13300-13308. doi:10.1109/CVPR42600.2020.01331

[32] Zhao P, Zhang S, Liu J, Liu H. Zero-shot Learning via the fusion of generation and embedding for image recognition. Information Sciences 2021; 578: 831-847. doi:10.1016/j.ins.2021.08.061

[33] Liu Z, Zhang X, Zhu Z, Zheng S, Zhao Y et al. Convolutional prototype learning for zero-shot recognition. Image and Vision Computing, 98:103924. doi:10.1016/j.imavis.2020.103924

[34] Wang Y, Zhang H, Zhang Z, Long Y, Shao L. Learning discriminative domain-invariant prototypes for generalized zero shot learning. Knowledge-Based Systems 2020; 196: 105796. doi:10.1016/j.knosys.2020.105796

[35] Nova D, Estévez PA. A review of learning vector quantization classifiers. Neural Computing and Applications 2014; 25 (3):511-524. doi:10.1007/s00521-013-1535-3

[36] Sato A, Yamada K. Generalized learning vector quantization. Advances in neural information processing systems 1995; 8.

[37] Hammer B, Villmann T. Generalized relevance learning vector quantization. Neural Networks 2002;15 (8); 1059-1068. doi:10.1016/S0893-6080(02)00079-5

[38] Schneider P, Biehl M, Hammer B. Adaptive relevance matrices in learning vector quantization. Neural computation 2009; 21 (12) :3532-3561. doi: 10.1162/neco.2009.11-08-908

[39] Villmann T, Bohnsack A, Kaden M. Can learning vector quantization be an alternative to SVM and deep learning?- Recent trends and advanced variants of learning vector quantization for classification learning. Journal of Artificial Intelligence and Soft Computing Research 2017; 7 (1): 65-81. doi:10.1515/jaiscr-2017-0005

[40] Biehl M, Ghosh A, Hammer B. Dynamics and Generalization Ability of LVQ Algorithms. Journal of Machine Learning Research 2007; 8 (2).

[41] Kulis B. Metric learning: A survey. Foundations and trends in machine learning 2012,; 5 (4): 287-364. doi:10.1561/2200000019

[42] Welinder P, Branson S, Mita T, Wah C, Schroff F et al. Caltech-UCSD birds 200 2010.

[43] Patterson G, Hays J. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In IEEE Conference on Computer Vision and Pattern Recognition 2012; 2751-2758. doi:10.1109/CVPR.2012.6247998

[44] Farhadi A, Endres I, Hoiem D, Forsyth D. Describing objects by their attributes. In IEEE conference on computer vision and pattern recognition 2009; 1778-1785. doi: 10.1109/CVPRW.2009.5206772

[45] Janez D. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research 2006; 7:1-30.