Neural Computing and Applications (2023) 35:9931–9945
https://doi.org/10.1007/s00521-023-08299-1

ORIGINAL ARTICLE

Zero-shot learning via self-organizing maps

Firat Ismailoglu[1]

Received: 1 August 2022 / Accepted: 6 January 2023 / Published online: 25 January 2023

Abstract
Collecting-labeled images from all possible classes related to the task at hand is highly impractical and may even be impossible. At this point, Zero-Shot Learning (ZSL) can enable the classification of new test classes for which there are no labeled images for training. The vast majority of existing ZSL methods aim to learn a projection from the feature space into the semantic space, where all classes are represented by a list of semantic attributes. To this end, they usually try to solve a complex optimization problem. Nevertheless, the semantic features (attributes) may not be suitable to represent the images because they are derived based on human knowledge and are, therefore, abstract. Alternatively, in this study, we introduce a novel ZSL method called SOMZSL, which has its roots in Self-Organizing Maps (SOM), a famous data visualization method. In particular, SOMZSL builds two SOMs of the same size and shape, one for the feature space and one for the attribute space, and then establishes a correspondence between them. Instead of considering a direct projection between the feature space and the attribute space, which is inherently different, SOMZSL connects them through comparable intermediate layers, i.e., SOMs. In terms of performance, SOMZSL can classify novel test classes as well or even better than existing ZSL methods without dealing with a complex optimization problem, thanks to the heuristic nature of SOM on which it is based. Finally, SOMZSL uses unlabeled test images in the construction of SOMs and can thus mitigate the domain shift problem inherent in ZSL.

Keywords Image classification · Zero-shot learning · Self-organizing maps · Domain shift problem

1 Introduction

Just like humans, machines need to see objects before they can recognize them. This is a key to any autonomous object recognition system. In particular, in the context of image classification/recognition, this phenomenon requires that images from each class of interest be present in the training set. However, the environment in which we live is dynamic; new classes emerge and evolve over time. For example, new types of viruses are constantly being discovered and futuristic objects such as the next generation of concept cars are being designed. Additionally, labeling images involves human effort and is therefore costly, making the collection of labeled images from all possible classes of interest impractical. Different from traditional classification methods that are unable to handle new classes, at this point we need a new technology to classify these classes. Zero-Shot Learning (ZSL) was developed for this purpose [1–3].

ZSL can classify images even if they have no labeled instances in a given training set. Typically, ZSL accomplishes this by leveraging labeled instances of other classes, i.e., those that exist during training and are therefore called training classes. ZSL assumes that both the training classes and the test classes, i.e., the classes to be classified, are represented with the same semantically meaningful attributes, such as visual properties "has four legs", "is green" [2]. Using the (labeled) images of the training classes together with their semantic attributes, ZSL learns how to describe the images of the test classes with the semantic attributes. Using these descriptions, ZSL eventually recognizes the test classes much like a human identifies new cases when given a high-level description about them.

The initial studies on ZSL aimed at learning a set of independent classifiers, one for each semantic attribute [2, 4]. This was later criticized on the grounds that no

✉ Firat Ismailoglu
fismailoglu@cumhuriyet.edu.tr

1 Department of Computer Engineering, Sivas Cumhuriyet University, 58140 Sivas, Turkey

interaction between the classifiers is allowed since they are learned separately [3, 5]. Since then, the studies capable of projecting the *feature (visual) space* of images into the *attribute (semantic) space* of classes have come to dominate the research on ZSL [5–9]. These studies have in common that they aim at projecting the images near their classes, which are represented by prototype vectors in the attribute space, using a projection matrix. Here, the columns of the projection matrices correspond to the semantic attributes and are learned jointly. However, learning such a matrix is nontrivial task and usually requires handling a complex optimization function. In addition, representing the images that are physically exist by the semantic features may not be appropriate because such features are derived on the basis of human knowledge and are thus abstract [10].

The above mentioned methods also suffer from the *projection domain shift problem*, which arises because the projection is learned using only training class images, while the ultimate goal of ZSL is to classify test classes [11]. This is a severe problem, considering that the data distributions of the classes are likely to be different, causing the method to be biased towards the training class images, which in turn degrades the performance of the ZSL method. To overcome this problem, the obvious solution is to incorporate the unlabelled test data into the learning process, which is achieved by a tiny fraction of ZSL methods [11].

In this paper, we introduce a fairly new method for ZSL that has its roots in Self-Organizing Maps (SOM) [12]. Therefore, we first outline SOM and then explain our proposed method. Briefly, SOM is a simple but robust data visualization method that can represent data in high dimensions in a regular grid (lattice) of neurons (units). SOM achieves this visualization by assigning each data instance to a neuron that best matches the instance and is therefore called Best Matching Unit (BMU). In practise, BMU is the neuron whose weight vector is closest to the instance at hand and can be considered as a proxy for the instance in SOM.

Learning the weight vectors specific to each neuron is iterative, where at each iteration an instance is selected and then the corresponding BMU is determined. Based on the distance to the coordinates of the BMU, the weight vectors corresponding to the neurons are updated, i.e., they converge to the instance.

For ZSL, we propose to create two SOMs of the same size and shape, one for the feature (input) space and one for the semantic space, calling the former the input SOM and the latter the attribute SOM. Our motivation is to transform input and semantic spaces, which are inherently different, into similar forms to make them comparable. To establish a correspondence between two SOMs, we first utilize labeled images of the training classes in the following way. We take a pair consisting of an image and its class, find the BMU in the input SOM for the image and the BMU in the attribute SOM for its class based on its attributes. We then update the weight vectors of the input SOM based on the BMU in the attribute SOM. As a result, the input SOM is created under the supervision of the semantic space. For the attribute SOM, its weight vectors are updated based on the BMU in the input SOM, similar to the previous case. This creates a correspondence between the two SOMs, which eventually leads to the input space and the attribute space being connected via the SOMs.

In the second phase of SOMZSL, both unlabeled test images and the semantic representations, i.e., class prototypes, of the test classes come into play. In particular, the test images are used to further train the input SOM in an unsupervised manner, i.e., using only the features of the images, not their classes. Likewise, the semantic representations are used to further train the attribute SOM again in an unsupervised manner, i.e., using only the attributes of the classes. Here, the neurons of the two SOMs are updated in a conventional way, i.e., based on the BMUs in their own SOM.

As mentioned earlier, BMUs in a SOM are proxies for instances in the original space, and training a SOM based on these BMUs provides a representation of the original space. In this sense, in the first phase of SOMZSL, we update the input SOM based on the BMUs corresponding to classes represented by a set of attributes. Similarly, the attribute SOM is trained with the BMUs in the input SOM. As a result, the input SOM (or attribute SOM) becomes a representative of the attribute space (or input space) at the end of the first phase. In the second phase of SOMZSL, the SOMs start to represent their own spaces by considering their own BMUs. Overall, both SOMs are trained with instances of the input space and class representations in the attribute space which establishes a correspondence between them. The associated code for SOMZSL can be found at https://github.com/FiratIsmailoglu/SOMZSL.

When it comes to classifying a test image, it is first mapped to its BMU in the input SOM and then directly transferred to the corresponding BMU in the attribute SOM, which shares the same coordinates with the one in the input SOM. As a result, the image is represented by the weight vector of the BMU in the attribute SOM, whose dimension is the same as that of the semantic space. This, in turn, enables the representation of the test image in the semantic space. To illustrate this classification rule, we give Fig. 1.

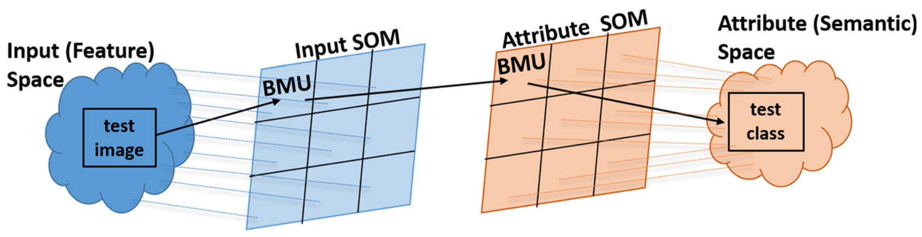


Fig. 1 The classification rule of SOMZSL

1.1 Research questions

In this study, by proposing a new ZSL method that is based on SOM, we seek to answer the following research questions:

- Is it possible to bridge the gap between the feature and semantic space in ZSL that are inherently different by abstracting and simplifying these spaces?
- Can we develop a simple but effective method to enable the classification in ZSL without having to deal with an optimization problem?
- Can SOM, a well-known and appreciated method for data visualization, also be used for transfer learning?

1.2 Novelties and advantages of SOMZSL

To address the research questions listed above, we develop a SOM-based ZSL method, SOMZSL, in this paper. SOMZSL differs from existing ZSL methods in several ways. In the following, we summarize them.

- Considering the training class images as auxiliary (source) data and the test class images as target data, ZSL is a form of transfer learning / domain adaptation. As far as we know, SOMZSL is the first SOM-based method not only for ZSL but also for transfer learning/domain adaptation problem.
- SOMZSL establishes a correspondence between two SOMs from two different spaces.
- Unlike most ZSL methods, which transform either only the feature space or only the semantic space, SOMZSL transforms both spaces into two different yet comparable grids, i.e., SOMs.
- SOMZSL relies on a widely appreciated data visualization method, SOM, which is easy to use.

SOMZSL offers several advantages over existing ZSL methods, including:

- not having to deal with a costly optimization function,
- preserving the topology of the feature space and the semantic space thanks to the SOMs included in it,
- mitigating the adverse effects of the projection domain shift problem inherent in ZSL by allowing the use of unlabeled test images as well as the semantic representations of the test classes, which endows SOMZSL with a trunsductive property.

In the remainder of the paper, we first review the work related to ZSL and SOM. In the next section, we begin by formalizing the problem of interest in this study, and then proceed with a detailed explanation of the proposed SOMZSL. We then report the results of the experiments conducted to validate the performance of SOMZSL. In the last section, we summarize what has been done in this study and then provide an outlook on what can be done next.

2 Related work

To structure the related work, we divide this section into several parts. First, we give an overview of the history of the ZSL problem, since ZSL is our main concern in this study. Then we continue with the main approaches to ZSL and then with the brief history of SOM, which is our approach to ZSL. We also discuss uses of SOM more related to the problem of interest. Finally, we conclude this section in a way that leads us to the proposed SOM based ZSL method.

Brief History of the ZSL Problem The lack of labeled training instances for each class of interest is not a new problem, nor is it specific to the image classification. In fact, this problem has been studied for decades in the field of lifelong learning, where learned knowledge is used for new emerging tasks [13]. Nevertheless, most of the related research is concerned with the image classification problem and is considered in the context of Zero-Shot Learning (ZSL) [3]. One of the earliest studies in this regard is [4], in which Lampert et. al. introduced the concept of (semantic) attributes, which can be defined as high-level descriptions of classes. Using these attributes, they proposed to transfer information from seen to unseen classes. Meanwhile Palatucci et. al. formalized the ZSL problem by defining the necessary notions such as attribute space, class prototype vectors and semantic output code classifiers [1]. Later, this problem attracted the attention of the transfer learning/domain adaptation community, since the underlying idea of ZSL is nothing but transferring information from seen classes to unseen classes [14]. This led to an explosion of studies that aim to project the feature (input) space into the attribute space [5, 7]. While these studies remain popular, a new trend is to develop generative models for ZSL [15]. Specifically, these models generate synthetic training instances for unseen classes based on their semantic attributes, resulting in a transformation of the ZSL problem into a traditional image classification problem.

Main Approaches to ZSL Early approaches to ZSL propose to learn a separate (binary) classifier for each semantic attribute using the instances of training classes. They then combine the resulting classifiers to obtain the final classifier for test classes. DAP and IAP are the famous examples of this kind [2]. However, these approaches are highly criticized for not considering correlations between the attributes. Later approaches focus on learning a projection matrix that helps transfer instances (i.e., images) in the input space to the attribute space, resulting in both instances and class prototypes being represented in the same space. ALE [5], ESZSL [9] and SJE [8] are some successful examples in this category. These methods try to project the instances to points that are close to the corresponding classes, while dealing with different optimization problems. Moreover, they take into account correlations between the attributes since the columns of the projection matrix they generate correspond to the attributes and are learned jointly.

The projection methods mentioned above are also criticized for allowing the classification to take place in the semantic space, which leads to discarding the discriminative power of visual features. With this in mind, some studies prefer to use the feature (input) space as the embedding space into which the semantically represented classes are projected [10]. Nevertheless, these studies are also subject to criticism, because of their inability to preserve the semantic properties of the classes [16]. Alternatively, [17] introduced *model space*, where the coordinates correspond to the parameters of the classifiers (models), each of which was trained to classify a particular (seen) class. They then created two weighted graphs, one in the semantic space and one in the model space, where the class representations and the classifier parameters correspond to the vertices of these graphs, respectively, and finally aligned these two graphs using manifold learning. Speaking of the ZSL methods that are embedding-based, a small fraction of them aim at learning a common intermediate embedding space where both the feature space and the semantic space are mapped [18]. However, these methods tend to assign test instances to seen training classes due to the lack of labeled instances from test classes [10].

The problem of ZSL has also been studied from the point of view of deep learning. In general, deep learning serves ZSL in two ways. First, a Convolutional Neural Network (CNN) can be built where the number of output neurons is equal to the number of attributes, i.e., the dimension of the attribute space [19]. In this case, the output space of the CNN is served as the common embedding space. Alternatively, [20] proposed learning two multilayer perceptrons (MLPs) simultaneously, one for the feature space and one for the semantic space, to embed both spaces in an intermediate embedding space. However, both approaches are prone to the domain shift problem as the CNNs and MLPs are trained using only training (seen) classes. Second, deep learning, particularly Generative Adversarial Networks (GANs), is utilized to create synthetic instances for test classes that are then used in training a ZSL model [15]. Parallel to this idea, [21] utilizes a conditional variational autoencoder to estimate the underlying probability distribution of image features conditioned on the class embeddings, leading to the generation of the required synthetic instances.

Brief History of SOM Teuvo Kohonen invented SOM in the 1980s as a dimensionality reduction technique that can provide an intuitive understanding of patterns and relationships in high-dimensional data [12]. Since then, SOM has received widespread use and attendance, mainly due to two reasons: (i) its ability to preserve the topology of input data and (ii) its ease of implementation [22]. In such applications, SOM has been used primarily for clustering, abstracting, explaining, and visualizing high-dimensional data. In addition, SOM has also been used for supervised learning, such as classification [23] or even time series prediction [24]. In the meantime, several extensions of SOM have been proposed. One notable extension in the history of SOM is Growing SOM (GSOM) [25]. Unlike conventional SOM, in GSOM the number of neurons is not predetermined, but can grow based on a heuristic: if the error value of a neuron is above a certain growth threshold, then let it grow in all possible directions, otherwise increase the error value of the neuron. This provides some flexibility in determining the SOM size. Another important version is Batch SOM, which proposes to update the weight vectors only at the end of each epoch, after all instances have been seen [12]. This makes SOM completely deterministic, while the original online version is stochastic. Speaking of determinism, Akinduko et. al. proposed a PCA-based initialization scheme for the weight vectors of SOM [26]. Instead of starting with completely random vectors, this approach determines the weight vectors based on linear combinations of the first two principal components of the input space, guaranteeing that the same weights are started with each run. Finally, a few words about the theoretical aspects of SOM. Although originally a heuristic method, there are quite a few studies in the history of SOM that aim to explain the theory behind it [27]. These studies focus on three aspects: (i) stability of the results (ii) organization of the weight vectors, (iii) convergence. Of them, convergence is by far the most studied theoretical aspect of SOM, but there is still no well-approved study that can clearly explain whether and/or when a SOM converges [27].

More Relevant Uses of SOM To our knowledge, SOM has not been used for the zero-shot learning problem, nor for the more general transfer learning problem. This may be because SOM is inherently an unsupervised method, whereas ZSL is a type of classification problem and therefore requires supervised methods. The same is true for transfer learning, where labeled data is expected from the source data. Nevertheless, in terms of creating multiple SOMs, as we do, it is possible to encounter relevant studies. In these studies, first multiple (base) SOMs with the same size and shape are generated based on the same dataset with different initializations or based on different datasets created using the bagging method. Then, these SOMs are combined (aggregated) to obtain the final SOM while preserving their topological properties. In determining the final SOM, it is common to seek correspondence between the base SOMs. The underlying motivation of such studies is to make the final SOM more stable and less sensitive to the initialization [28].

Recall that when training the SOMs in the proposed SOMZSL, i.e., the input SOM and the attribute SOM, the weight vectors of one SOM are updated with respect to the BMU in the other SOM. From this point of view, SOMZSL and the Korresp algorithm have one thing in common [29]. Basically, Korresp associates each row of a given data matrix with a column of the matrix in such a way that it is the most probable given that row. Korresp then finds the BMUs by considering only the row vectors, but updates the weight vectors using the rows along with their associated columns. Korresp performs the same steps for the column vectors as well. As a result, Korresp yields a biclustering (coclustering) of the input data.

To sum up, the problem of ZSL has long been of interest to researchers because of its realistic scenario. Numerous studies on ZSL have been devoted to develop projection-based methods to make images represented by virtual features compatible with their classes represented by a set of semantic attributes. In these studies, either the feature space or the semantic space was usually preferred as the embedding space in which the projection takes place. However, image features may not be appropriate to represent classes that do not physically exist, and semantic attributes may not be appropriate to represent images because such attributes are derived based on human knowledge. Instead of preferring such a direct projection, one can first project the feature space and the semantic space onto some intermediate layers separately, making sure that these layers are compatible, and then try to establish a correspondence between these two layers. In this study, for the first time in the literature, we propose to use the well-known and well-appreciated SOM to construct these layers. In fact, we establish the required correspondence during the construction of the SOMs. In the following section, we explain the proposed idea in detail.

3 Methodology

3.1 Problem definition

In this study, we are concerned with constructing a ZSL classifier defined as a function

$$f : X \rightarrow \mathcal{U}, \tag{1}$$

where $X \subset \mathbb{R}^d$ is the input (feature) space, which can be extracted in various ways: for example, using the more traditional methods SURF and SIFT, or with newer methods that use convolutional neural networks (CNN) [30]. $\mathcal{U}$ is the set of unseen (test) classes. For this purpose, we have a training set $\mathcal{D}^{tr} = \{(\mathbf{x}_i, y_i) \in X \times \mathcal{S}\}_{i=1}^{N_{tr}}$, where $\mathcal{S}$ is the set of seen (training) classes. We assume that $\mathcal{S}$ and $\mathcal{U}$ are disjoint: $\mathcal{S} \cap \mathcal{U} = \emptyset$, meaning that the classes that are present during training are different from those to be classified. Using the classifier, the task we are interested in to classify the instances of test set $\mathcal{D}^{te} = \{\mathbf{x}_i \in X\}_{i=1}^{N_{te}}$.

3.1.1 Attribute space

As mentioned above, ZSL is a special kind of classification problem that involves classifying instances of unseen classes. Another special feature of ZSL is that there is usually an additional source of information regarding seen and unseen classes, which is encoded as a list of attributes. That is, we assume that each class is represented by a vector whose elements correspond to a semantic attribute. For example, in the case of animal recognition, an attribute may indicate whether an animal has a tail or a big nose. Alternatively, we may have textual features extracted from the texts describing classes. In either case, the attributes form a space called the attribute (semantic) space A in which both seen and unseen classes are represented. We think of A as a k-dimensional real-valued space $A \subset \mathbb{R}^k$, and denote the class representations in it by $\mathbf{m}_y$, where y can belong to $\mathcal{S}$ or $\mathcal{U}$.

3.2 SOM

A Self-Organizing Map (SOM) is a regular lattice of neurons organized to reflect the topology of the original space in order to preserve the local neighbourhoods within it [12]. In addition to location information, each neuron is associated with a weight vector (also known as a code vector) with dimension equal to the dimension of the original space. Training a SOM consists of learning these vectors, and after the training phase, each instance is mapped to a neuron whose weight vector is the nearest to it, which is referred to as Best Matching Unit (BMU). Starting from randomly initialized weight vectors, an instance is picked randomly at each iteration, making the SOM training stochastic, and then the corresponding BMU is calculated. Finally, the weight vectors of the neurons are updated depending on how close they are to the BMU.

Let $\mathbf{x}$ be the randomly picked instance at iteration $t+1$, then the update rule for the jth neuron is given as follows:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \alpha(t) h_{jc(\mathbf{x})}(\mathbf{x} - \mathbf{w}_j(t)), \tag{2}$$

where $\alpha(t)$ is the learning rate and $h_{jc(\mathbf{x})}$ is the neighborhood function. Specifically, $\alpha(t)$ controls the magnitude of the update and is a monotonically decreasing function of time. A common choice is to use exponential decay:

$$\alpha(t) = \alpha_0 e^{-t\times\zeta} \tag{3}$$

where $\alpha_0 > 0$ is the initial learning rate and $\zeta > 0$ is the decay rate. Getting back to (2), $h_{jc(\mathbf{x})}$ indicates how close the jth neuron is to $c(\mathbf{x})$, the BMU for $\mathbf{x}$, at time t. For this purpose, it is common to consider the standard Gaussian neighborhood function:

$$h_{jc(\mathbf{x})} = exp\left(-\frac{\|\mathbf{r}_j - \mathbf{r}_{c(\mathbf{x})}\|}{2\sigma(t)^2}\right), \tag{4}$$

where $\mathbf{r}_j$ and $\mathbf{r}_{c(\mathbf{x})}$ denote the coordinates of the j-th neuron and the BMU respectively. $\sigma(t)$ controls the width of the neighborhood and decreases over time, as in $\alpha(t)$. In fact, it can be updated using (3) replacing α_0 with σ_0, initial value for the width. When updating $\mathbf{w}_j$, the term $h_{jc(\mathbf{x})}$ becomes more important as the jth neuron approaches the BMU, i.e. $c(\mathbf{x})$. As a result, the neurons near $c(\mathbf{x})$, are most affected by $\mathbf{x}$, are, therefore, more strongly attracted to $\mathbf{x}$ than the others.

3.3 ZSL via SOM

We now proceed to the use of SOMs for the problem of ZSL. In essence, we propose to create two SOMs of the same shape and size, one for the input space and one for the attribute space. We refer to the former as the input SOM and the latter as the attribute SOM and aim to establish a correspondence between them. The initial advantage of this approach is that it ensures compatibility between the input space and the attribute space through the created SOMs, so that a connection between the two spaces is established. Unlike the other studies on ZSL in the literature that try to achieve this advantage, our proposed approach has nothing to do with an optimization problem as it relies entirely on the SOM learning. Moreover, we also leverage unlabeled test instances as well as semantic representations of unseen classes in the training phase, which makes the created SOM transductive, and improves generalisation accuracy.

For both SOMs, we assume without loss of generality that they have a rectangular shape of size $p \times q$. Thus, the training phase of the proposed ZSL method consists of learning the weight vectors, i.e., the code vectors, of both SOMs. For the j-th neuron, we denote the weight vector corresponding to the input space by $\mathbf{w}_j^{in} \in \mathbb{R}^d$ and the one corresponding to the attribute space by $\mathbf{w}_j^{att} \in \mathbb{R}^k$.

3.3.1 Phase - 1 : using training set

At first, we use the training set, i.e., $\mathcal{D}^{tr}$, in a supervised way to learn $\mathbf{w}_j^{in}$ and $\mathbf{w}_j^{att}$ $(j \in \{1, ..., p \times q\})$. To this end, at each iteration of the learning, we pick a random pair $(\mathbf{x}, y)$ from $\mathcal{D}^{tr}$, and then compute two BMUs, one for $\mathbf{x}$ and one for y:

$$c(\mathbf{x}) = \underset{j}{\text{argmin}} \|\mathbf{x} - \mathbf{w}_j^{in}\|, \tag{5}$$

$$c(y) = \underset{j}{\text{argmin}} \|\mathbf{m}_y - \mathbf{w}_j^{att}\|, \tag{6}$$

where $\mathbf{m}_y$ is the class representation (prototype) of class y. Note that $c(\mathbf{x})$ and $c(y)$ is not necessarily the same for pair $(\mathbf{x}, y)$. Here, we propose to update the weight vectors of the input SOM by considering the BMU for the attribute SOM, i.e., $c(y)$, as the winner neuron. Similarly, we update the weight vectors of the attribute SOM based on the BMU for the input space, i.e., $c(\mathbf{x})$. This results in the following update rules for the j-th neuron in the input SOM and in the attribute SOM:

$$\mathbf{w}_j^{in}(t+1) = \mathbf{w}_j^{in}(t) + \alpha(t) h_{jc(y)}(\mathbf{x} - \mathbf{w}_j^{in}(t)), \tag{7}$$

$$\mathbf{w}_j^{att}(t+1) = \mathbf{w}_j^{att}(t) + \alpha(t) h_{jc(\mathbf{x})}(\mathbf{m}_y - \mathbf{w}_j^{att}(t)). \tag{8}$$

This little trick makes it possible to order the neurons of the input SOM (resp. attribute SOM) with respect to the attribute space (resp. input space). In this way, the input SOM is started to reflect the attribute space, while the weight vectors of its neurons have the same dimension as the input space. Similarly, the attribute SOM is created based on the input space. Moreover, since the size and shape of the two SOMs are identical, there is a one-to-one correspondence between the SOMs. Consequently, instances of the input space can be first mapped to the input SOM, and then can directly pass to the attribute SOM to meet the class representations.

3.3.2 Phase - 2 : using test set and unseen classes

In the second phase of the proposed ZSL method, (unlabeled) test instances and the semantic representations of unseen classes are included in the training of the SOMs, which gives a transductive property to our proposed method. This phase is similar to the conventional SOM learning in that updating the weight vectors is performed with respect to the BMUs in the SOM of interest. That is, the input SOM is trained considering only the test instances from the input space; BMUs of the attribute SOM do not affect this training. The same holds for the training of the attribute SOM. However, we also require that the SOMs retain the properties of the opponent space, which is the acquisition of the first phase. To this end, we consider two regularization parameters λ_1 and λ_2, one for each SOM, leading to the following update rules:

$$\mathbf{w}_j^{in}(t+1) = \mathbf{w}_j^{in}(t) + \lambda_1 \left[\alpha(t) h_{jc(\mathbf{x})}(\mathbf{x} - \mathbf{w}_j^{in}(t))\right], \tag{9}$$

$$\mathbf{w}_j^{att}(t+1) = \mathbf{w}_j^{att}(t) + \lambda_2 \left[\alpha(t) h_{jc(y)}(\mathbf{m}_y - \mathbf{w}_j^{att}(t))\right]. \tag{10}$$

Here, we note that the number of test instances usually far exceeds the number of unseen classes, causing the weight vectors of the input SOM to be updated much more frequently than those of the attribute SOM. To solve this problem, we recommend setting λ_1 much smaller than λ_2.

Below is a figure illustrating the two phases of the proposed method just explained. Taking the second neuron in both SOMs as an example, Fig. 2a shows how the corresponding weight vectors, i.e., w_2^{in} and w_2^{att}, are updated. Specifically, w_2^{in} is pulled in the direction of $\mathbf{x}$ to the extent that the second neuron is close to the BMU of its class, i.e., $c(y)$, in the attribute SOM. Meanwhile, w_2^{att} is shifted towards $\mathbf{m}_y$, the semantic representation of class y, in proportion to the neighborhood between the location of the second neuron and the BMU of $\mathbf{x}$, i.e., $c(\mathbf{x})$, in the input SOM. On the other hand, Fig. 2 shows how w_2^{in} and w_2^{att}, are updated when test instances and the representations of unseen classes come into play. Here, the update of w_2^{in} (resp. w_2^{att}) is performed in a conventional way, i.e., with respect to the BMU of the SOM to which they belong.

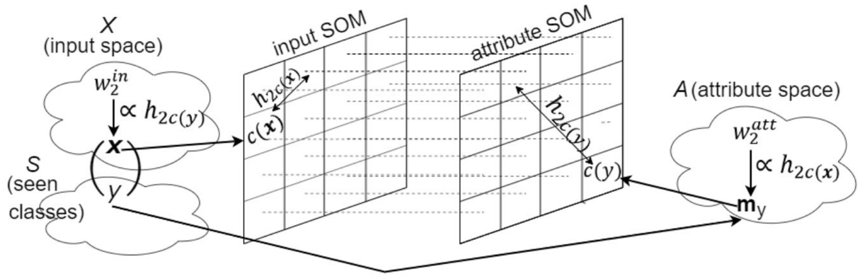
3.3.3 Classification

Once the SOMs are fully trained using the training set, the test set, and the representations of the seen and unseen classes, we can now classify test instances. To this end, we propose the following rule. A test instance is first mapped to its BMU in the input SOM using (5), and then passed to the corresponding BMU in the attribute SOM, which has the same coordinates as the one in the input SOM. Using its weight vector, the test instance can now be represented in the attribute space and compared to the semantic representations of the unseen classes. Finally, the class whose weight vector is the nearest can be considered the winning class. Formally, this classification rule can be described as follows:

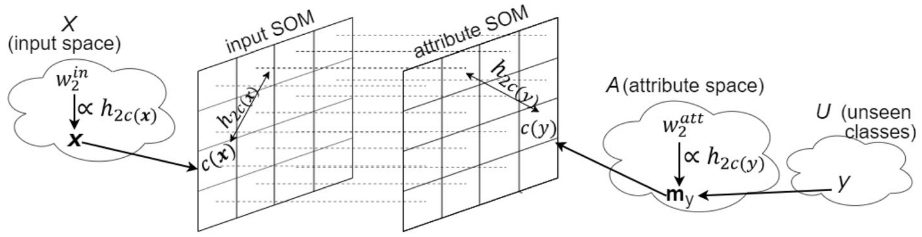
$$\operatorname*{argmin}_{y \in \mathcal{U}} \|\mathbf{w}_{c(\mathbf{x})}^{att} - \mathbf{m}_y\|. \tag{11}$$

Figure 3 illustrates the classification rule explained above. Finally, to provide a complete guide to SOMZSL, we

Fig. 2 The two phases of SOMZSL



(a) SOMZSL - Phase 1: Updating the weight vectors using training set.



(b) SOMZSL - Phase 2: Updating the weight vectors using test set and the representations of unseen classes.

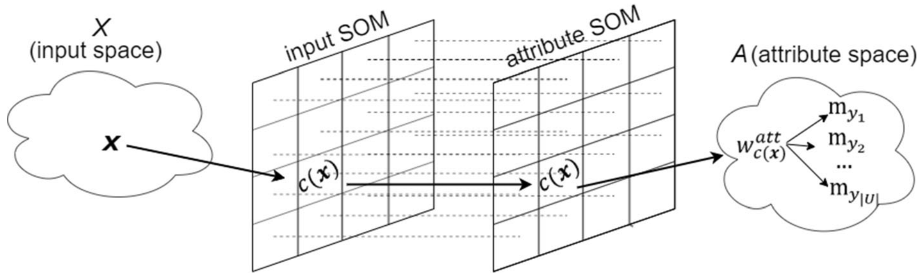


Fig. 3 Classification using SOMZSL

present its pseudocode in **Algorithm 1** that aggregates Phase-1 and Phase-2 as well as the classification rule therein.

```
Algorithm 1 SOMZSL: a SOM based Zero-Shot Learning method.
    Input: Training set: $\mathcal{D}^{tr} = \{(\mathbf{x}_i, y_i) \in X \times \mathcal{S}\}_{i=1}^{N_{tr}}$, test set $\mathcal{D}^{te} = \{\mathbf{x}_i \in X\}_{i=1}^{N_{te}}$,
    sets of seen and classes: $\mathcal{S}$, and $\mathcal{U}$, class representations $\mathbf{m}_y$ $(y \in \mathcal{S} \cup \mathcal{U})$, input and
    attribute SOMs of size $p \times q$, max. iteration number: $maxIter1$ and $maxIter2$,
    initial learn. rate $\alpha_0$, decay rate $\zeta$, init. neigh. width $\sigma_0$, reg. param. $\lambda$.
    Output: Predicted test instances.
    Phase - 1: Using Training Set
 1: for $t = 1 : maxIter1$ do
 2:     for $(\mathbf{x}, y) \in \mathcal{D}^{tr}$ do
 3:         Calculate $c(\mathbf{x})$ and $c(y)$, the BMUs for $\mathbf{x}$ and $y$, as in (5) and (6) resp.
 4:         for $j = 1 : p \times q$ do
 5:             wrt $c(\mathbf{x})$ and $c(y)$ update $w_j^{in}$ and $w_j^{att}$ as in (7) and (8) resp.
 6:         end for
 7:     end for
 8:     Update the learning rate $\alpha(t)$ and neighborhood width $\sigma(t)$ as in (3) using
    $\alpha_0, \sigma_0$, and $\zeta$.
 9: end for        ▷ now, input SOM and attribute SOM have been partially trained.
    Phase - 2: Using Test Set and Unseen Classes
10: for $t = 1 : maxIter2$ do
11:     for $\mathbf{x} \in \mathcal{D}^{te}$ do
12:         Calculate $c(\mathbf{x})$, the BMU for $\mathbf{x}$, as in (5).
13:         for $j = 1 : p \times q$ do
14:             wrt $c(\mathbf{x})$ update $w_j^{in}$ as in (9).
15:         end for
16:     end for
17:     for $y \in \mathcal{U}$ do
18:         Calculate $c(y)$, the BMU for $y$, as in (6).
19:         for $j = 1 : p \times q$ do
20:             wrt $c(y)$ update $w_j^{att}$ as in (10).
21:         end for
22:         Update the learning rate $\alpha(t)$ and neigh. width $\sigma(t)$ as in (3) using $\alpha_0, \sigma_0$,
    and $\zeta$.
23:     end for
24: end for     ▷ Input SOM and attribute SOM have been trained in a transductive
    manner.
    The Classification
25: for $\mathbf{x} \in \mathcal{D}^{te}$ do
26:     Calculate $c(\mathbf{x})$ as in (5).
27:     Compare $w^{att}_{c(\mathbf{x})}$ and the class representations $\mathbf{m}_y$ $(y \in \mathcal{U})$ as in (11).
28:     Assign $\mathbf{x}$ to the winner (unseen) class from the previous step.
29: end for
```

3.3.4 Space and computational complexity of SOMZSL

Space Complexity. SOMZSL has two SOMs: input SOM and attribute SOM. The space required to store the input SOM and attribute SOM is $\mathcal{O}(N \cdot d)$ and $\mathcal{O}(N \cdot k)$, respectively, where N denotes the number of neurons, which is the same in both SOMs. Remember also that d (or k) denotes the dimension of the input space (or attribute space). At each iteration of Phase-1, SOMZSL requires an array of the distances between an image and the neurons of the input SOM. This array consists of scalers, so the required space is $\mathcal{O}(N)$. In addition, the algorithm requires an array of neighborhoods between the BMU of that iteration and the neurons; which again has a space requirement of $\mathcal{O}(N)$. These space requirements are also necessary for the attribute SOM. Concretely, the required space is $\mathcal{O}(N)$ for storing the distances between class prototype in the semantic space and the neurons of the attribute SOM. Also, a memory space of $\mathcal{O}(N)$ is required to store the neighborhood measures to the BMU in the attribute SOM. Since Phase-2 of SOMZSL begins as soon as Phase-1 is completed, the memory used in Phase-1 can be reused; Phase-2 requires no additional memory.

Time Complexity. The time complexity of Phase-1 of SOMZSL is $\mathcal{O}(maxIter1 \cdot N \cdot N_{tr})$, and the time complexity of Phase-2 is $\mathcal{O}(maxIter2 \cdot N \cdot (N_{te} + |\mathcal{U}|)$. If we consider the maximum iteration numbers corresponding to the first and the second phases of the algorithms as the same, then the overall time complexity of SOMZSL

$$\mathcal{O}(maxIter \cdot N \cdot (N_{tr} + N_{te} + |\mathcal{U}|)),$$

where $N, maxIter, N_{tr}, N_{te}$ and $|\mathcal{U}|$ are the total number of neurons in each of the SOMs, the maximum iteration number, the number of training and test instances, and the cardinality of the set of unseen classes, respectively. This indicates that the time required to train SOMZSL is linear with the total number of neurons, the number of training and test instances; and not quadratic with any of the inputs.

4 Experiments

The purpose of this section is to show the effectiveness of the proposed SOMZSL and to explain which settings lead to maximizing the performance of it. To this end, we conduct a range of experiments on six benchmark datasets and five state-of-the-art ZSL methods. In the following subsections, we present the results of these experiments.

4.1 Datasets

We divide the datasets used in this study into two categories, depending on the semantic representation of the classes. In the first category, we consider visual attributes such as color, hasTail, hasFur, etc., while in the second category we use textual attributes based on TF-IDF scores extracted from the corresponding Wikipedia articles. Below, we describe the datasets of these two categories in detail.

Visual Attributes AWA1 [2], AWA2 [3], SUN [31] and aPY [32] are the datasets considered in this study where the classes have visual attributes. AWA1 and AWA2 are one of the most commonly used datasets in ZSL. They consist of images of 50 animals represented by 85 visual attributes. AWA2 differs from AWA1 in that the images are taken from public sources such as Flickr and Wikipedia. SUN contains images of scene classes and therefore addresses the problem of scene categorization in computer vision. aPY is a rather small dataset compared to the other datasets mentioned, containing object classes from aPascal and aYahoo. For extracting the features from the images of these datasets, we use CNNs trained on ImageNet-1K leading to 101-layered ResNet and then use 2048-dim top-layer pooling units of the ResNet.

Textual Attributes CUB [33] and NAB [34] are datasets in this category. CUB and NAB contain images of bird species that are similar to each other, making CUB and NAB fine-grained datasets. NAB is larger than CUB, both in terms of the number of classes and the number of images. For these datasets, we employ the Visual Part Detector/Encoder network (VPDE-net) to extract the features of the images, as explained in [35]. This results in 3583 and 3072 features for the images in CUB and NAB, respectively.

Splits We split the classes AWA1, AWA2, SUN and aPY into training, validation and testing as in a well-known survey on ZSL [3]. For CUB and NAB, we follow the settings proposed in [35]. That is, for every unseen class there is at least one seen class from the same super-category, which increases the relevance between seen and unseen classes. We further split the seen classes into training and validation sets.

Below is Table 1 which summarizes all the necessary information about the above datasets, including the number of attributes, the number of seen/unseen classes etc. Also, the horizontal line separates the datasets in the first category from those of the second category. Here $|\mathcal{S}|$ denotes the number of (seen) training and validation classes separated by the sum sign. The same is true for $|\mathcal{D}^{tr}|$, which denotes the total number of training and validation instances.

4.2 Benchmark methods

We compare the performance of the proposed SOMZSL with that of several benchmark methods used for the zero-shot learning problem: Deep Visual Semantic Embedding (DeViSE) [7], Attribute Label Embedding (ALE) [5], Structured Joint Embedding (SJE) [8] and Embarrassingly Simple Zero-Shot Learning (ESZSL) [9]. Commonly, they aim at learning a matrix, so-called projection matrix, that allows projection from the input(visual) space to the attribute (semantic) space. In doing so, they all aim to maximize the compatibility between the images represented in the input space and the corresponding classes represented

Table 1 Statistics of the benchmark datasets used for evaluation

Datasets	Size	Detail	# Atts	$\vert\mathcal{S}\vert$	$\vert\mathcal{U}\vert$	$\vert\mathcal{D}^{tr}\vert$	$\vert\mathcal{D}^{te}\vert$	Total Inst.
AWA1 [2]	Medium	Coarse	85	27 + 13	10	16864 + 7926	5685	30475
AWA2 [3]	Medium	Coarse	85	27 + 13	10	20218 + 9191	7913	30475
SUN [31]	Medium	Fine	102	580 + 65	72	11600 + 1300	1440	14340
aPY [32]	Small	Coarse	64	15 + 5	12	6086 + 1329	7924	15339
CUB [33]	Medium	Fine	7551	100 + 50	50	5875 + 2946	2967	11788
NAB [34]	Large	Fine	13217	200 + 123	81	25291 + 13939	9332	48562

by their prototypes in the attribute space. With this in mind, they deal with different optimization functions. Nevertheless, a survey [3] unifies these objective functions and gives an overview of these methods, so we omit the details here.

4.3 Implementation details

SOMZSL The proposed method SOMZSL is based on creating two SOMs, input SOM and attribute SOM, and then trying to establish a one-to-one correspondence between them. Thus, the use of SOMZSL starts with determining the shape (structure) and size of the SOMs. For the shape, a hexagonal lattice or rectangular lattice is usually considered [12, 36]. From the point of view of topology preservation, there is no significant difference between the hexagonal and the rectangular lattice, both fit the data in a similar way [37]. We also found no significant difference in terms of SOMZSL performance during the experiments, so we report here only the results obtained with rectangular lattices. For determining the size of the SOMs employed in SOMZSL, we leave the related discussion in Sect. 4.5.1.

For initializing the weight vectors (code vectors) associated with the neurons of a SOM, there are two leading approaches: random initialization and PCA-based initialization [12, 26]. In the case of SOMZSL, the first approach proposes to select a random image (either a training or a test image) for each neuron of the input SOM and then use its feature vector to initialize the neuron. Similarly, for each neuron of the attribute SOM, a class representation (either a seen or an unseen class) is randomly selected. Unlike the random initialization, PCA-based initialization is a deterministic approach that relies on linear combinations of the first two principal components of the space to be mapped [26]. In particular, in the case of SOMZSL, the PCA-based initialization requires the use of linear combinations of the first two principal components of feature (visual) space for the input SOM and those of the attribute space for the attribute SOM. During the experiments, we tried both initialization techniques but did not observe a notable difference in terms of the accuracy of SOMZSL, while the PCA-based initialization generally provided faster convergence. Also for the sake of reproducibility of the results, we present here the results obtained with the PCA-based initialization.

The proposed method SOMZSL has some parameters need to be tuned. Below we provide Table 2 that shows the parameters leading to the best performance based on validations sets for each dataset.

The learning rate (α) is an important parameter to set not only for SOMZSL but also for the benchmark ZSL methods considered in the experiments. Below, we give Table 3 that shows the learning rates leading to the highest performance for the ZSL methods based on the validation sets.

Finally, ESZSL has two regularization parameters: gamma and lambda. W set them 0, 3; 0,3; 1,3; −1, 3; 0,3 and 0,3 for AWA1, AWA2, SUN, aPY, CUB and NAB, respectively, based on its performance on the validation sets.

4.4 Evaluation criterion

For ZSL, the most common criterion to evaluate the performance of ZSL methods is the Top-1 accuracy [2, 3, 5]. Specifically, Top-1 accuracy differs from conventional accuracy in that it averages the accuracy achieved for each unseen class, preventing the evaluation measure from being biased towards densely populated classes. Top-1 accuracy is thus calculated using:

$$\frac{1}{|\mathcal{U}|}\sum_{z\in\mathcal{U}}\frac{\#\text{ correctly predicted test instances of } z}{\#\text{ test instances of } z}. \qquad (12)$$

4.5 Results

Table 4 shows the results of the benchmark ZSL methods and the proposed SOMZSL for the ZSL datasets. For datasets AWA1, AWA2, and aPY; SOMZSL achieves the highest Top-1 accuracy. For these datasets, the improvement in the accuracy is between 2% and 4% compared to the second best ZSL method for each dataset. On the other hand, for SUN, SOMZSL achieves the worst accuracy. For the datasets CUB and NAB, where the classes are represented with textual attributes, in contrast to the other

Table 2 The parameters of SOMZSL used in the experiments

Datasets	SOM size	Max iter. for the SOMs		Reg. params.		Init. learn. rate	Decay rate	Init. neigh. rate
		*maxIter*1	*maxIter*2	λ_1	λ_2	α_0	ζ	σ_0
AWA1 [2]	20 × 20	15	20	0.01	5	0.01	0.1	0.01
AWA2 [3]	15 × 15	10	10	0.01	5	0.01	0.1	0.001
SUN [31]	25 × 25	15	15	0.01	10	0.1	0.01	0.01
aPY [32]	25 × 25	5	10	0.001	5	0.01	0.01	0.001
CUB [33]	25 × 25	7	7	0.01	10	0.01	0.001	0.001
NAB [34]	20 × 20	15	15	0.01	5	0.1	0.001	0.01

Table 3 Learning rates that provide the best results

	AWA1 [2]	AWA2 [25]	SUN [31]	aPY[32]	CUB[33]	NAB[34]
DeViSE [7]	0.01	0.001	0.01	1	1	1
ALE [5]	0.01	0.01	0.2	0.04	0.5	0.5
SJE [8]	1	1	1	0.02	1	1

Table 4 The Top-1 scores of the benchmark ZSL methods and SOMZSL for the benchmark datasets

Datasets	DeViSE [7]	ALE [5]	SJE [8]	ESZSL [9]	SOMZSL (Ours)
AWA1 [2]	0.556	0.566	0.585	0.556	**0.628**
AWA2 [3]	0.576	0.537	0.583	0.537	**0.611**
SUN [31]	0.545	**0.591**	0.534	0.554	0.489
aPY [32]	0.332	0.345	0.315	0.355	**0.371**
CUB [33]	0.381	0.332	0.381	**0.391**	0.361
NAB [34]	0.361	0.328	0.381	**0.412**	0.381

datasets where the classes are represented with visual attributes, SOMZSL achieves comparable results. In fact, it performs better than ALE, where the difference in the accuracy is about 4% in favour of SOMZSL. Also for CUB and NAB, it can be clearly stated that ESZSL performs better than the other methods. Finally, for the purpose of visualizing the comparison between the performances of the ZSL methods including SOMZSL, we provide Fig. 4.

The datasets where SOMZSL comes first have in common that they are coarse-grained datasets. For such datasets, due to large inter-class variations the task of image classification is less challenging than for fine-grained datasets, where the classes have similar appearances. Thus, we can conclude that our proposed method SOMZSL exhibits superior performance in a zero-shot learning problem when the images in the problem are distinguishable to some extent. If this is not the case for a ZSL problem, one can still achieve satisfactory performance with SOMZSL as long as he/she is more careful in setting the parameters of SOMZSL.

4.5.1 The effect of SOM size on the performance of SOMZSL

For determining the ideal size of a SOM, several empirical rules have been suggested [12, 36]. Of them, the common practice suggested by Kohonen is to set the number of

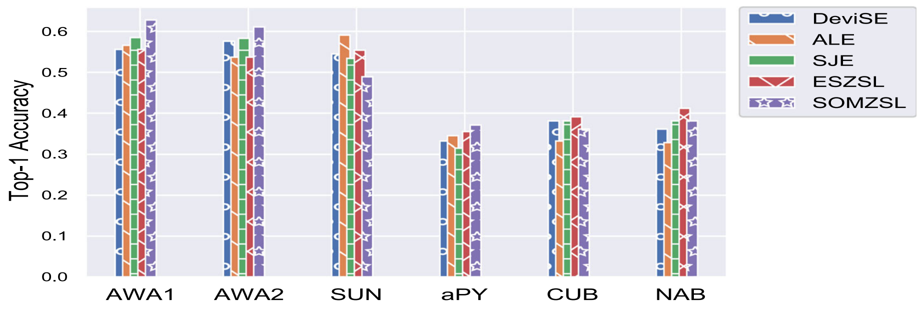


Fig. 4 A bar chart for illustrating the Top-1 scores of the ZSL methods

neurons to $5\sqrt{N}$, where N is the number of data points [12]. In the case of SOMZSL, however, there are two SOMs that must have the same number of neurons due to the one-to-one correspondence that is sought. We thereby have two candidates for N: the number of images and the number of classes based on the input SOM and attribute SOM, respectively. Considering that the number of images is much larger than the number of classes, we base the number of neurons common to both SOMs on the number of classes; otherwise, the class prototypes would be represented quite sparsely in the attribute SOM.

For the datasets used in the experiments, the total number of classes (seen and unseen) ranges from 32 to 717. This suggests that the upper bound for the number of neurons should be around $5\sqrt{717} \cong 134$, which can be achieved with a SOM of size 15×15. Nevertheless, we double this number to examine the effect of SOM size on SOMZSL on a larger scale. Therefore, we allow the size to vary between 5×5 and 30×30. Figure 5 shows the results in this regard.

Based on Fig. 5, we draw the following conclusions. The immediate result is that the size of SOMs affects the performance of SOMZSL. More specifically, increasing the SOM size initially improves the performance of SOMZSL, but further increase may lead to deterioration of the performance. For AWA2, the decrease in the accuracy due to the increase in the SOM size is up to 7%. Therefore, it is unrealistic to conclude that the larger the SOM size, the better the performance of SOMZSL. This may be due to the sparse representation of images and/or class prototypes in the SOMs. Moreover, the experiments suggest that both 20×20 and 25×25 are good choices for SOM size, regardless of the size of the data. Despite the fact that aPY is a rather small dataset compared to NAB (see Table 1, the ideal SOM size is identical, i.e., 25×25, for both datasets. Finally, it is also interesting to note that SOMZSL can perform well even when it uses very small SOMs inside. For AWA2,for example, SOMZSL can achieve an accuracy of 0.59 with SOMs of size 5×5, which is higher than the other ZSL methods tested. This is an important indication that SOMZSL can provide fast and accurate classification for the zero-shot learning problem.

4.6 Comparison of time complexities of the ZSL methods

We now compare the time complexity of the ZSL methods considered in the experiments with ours. As discussed in the Sect. 3.3.4, the time complexity of the proposed method SOMZSL is $\mathcal{O}(maxIter \cdot N \cdot (N_{tr} + N_{te} + |\mathcal{U}|))$, where N, N_{tr}, N_{te} and $|\mathcal{U}|$ are the total number of neurons, the number of training and test instances, and the cardinality of the set of unseen classes, respectively.

The objective function of the ZSL methods DeViSE [7], ALE [5] and SJE [8] is inspired by the SVM. In fact, they are all a variant of either ranking or the structured SVM, thus are very similar. These functions are all convex and are optimized by Stochastic Gradient Descent with the same time complexity: $\mathcal{O}(maxIter \cdot N_{tr} \cdot d \cdot k \cdot |\mathcal{S}|)$, where d, k and $|\mathcal{S}|$ are the dimension of the input space, the dimension of the attribute space and the cardinality of the set of unseen classes, respectively. It is typical that the

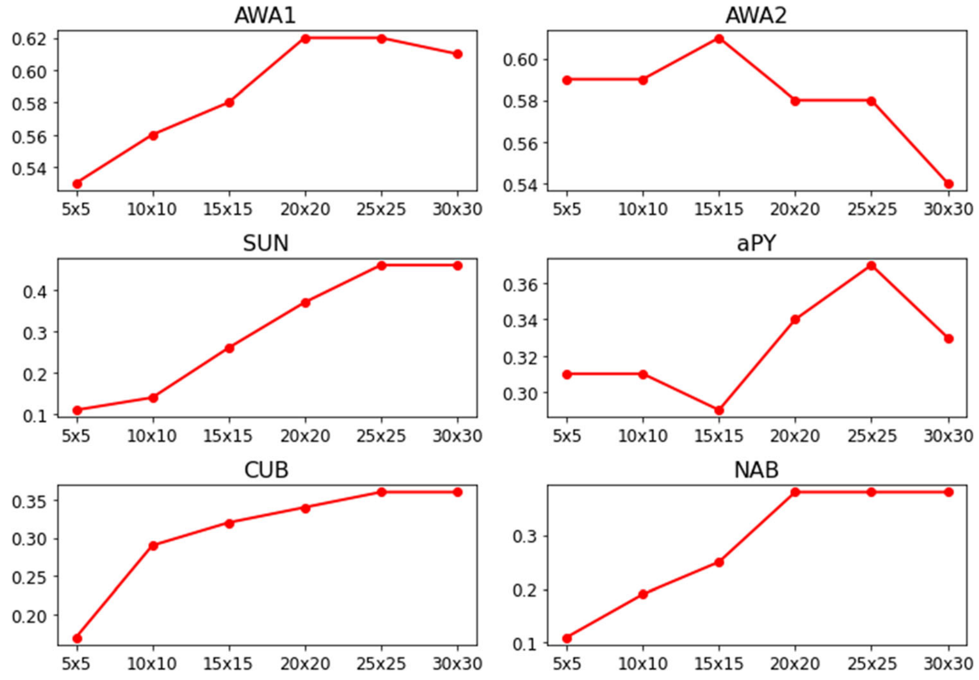


Fig. 5 The Effect of SOM Size on SOMZSL

number of neurons in a SOM far less than the number of instances, so $N << N_t r$ almost always holds. Also, $(N_{tr} + N_{te} + |\mathcal{U}|)$ is usually less than $d \cdot k \cdot |\mathcal{S}|)$; the statistics of the datasets shown in Table 1 support this assertion. Thus, we conclude that SOMZSL requires less time to train than DeViSE, ALE and SJE. As for ESZSL [9], the method involves two matrix inversions; and its time complexity is $\mathcal{O}(N_{tr}^3 + |\mathcal{S}|^3)$, thus is much less efficient than SOMZSL.

5 Conclusions and future research

5.1 The ZSL problem

In many cases, it may be may be problematic to obtain a certain number of labeled instances from each class of interest, before building a classifier. At the extreme, a classifier may be exposed to classes that are completely different from those on which it was trained. In the context of image classification, Zero-Shot Learning (ZSL) addresses this problem. To classify test classes to be classified, ZSL utilizes training classes whose labeled images are available during training, as well as high-level descriptions of the test and training classes. Usually, these descriptions are given as a vector of semantic attributes that form the attribute space. Many ZSL methods aim to learn a projection matrix/function to project images in feature (input) space (input) into attribute space. However, this usually involves an attempt to solve an optimization function without leveraging unlabeled test images, leading to the projection domain shift problem.

5.2 The Proposed ZSL method

In this study, we present a novel ZSL method based on the well known SOM algorithm, called SOMZSL. Briefly, SOMZSL creates two SOMs with the same size and dimension, one for the feature space and one for the attribute space. SOMZSL uses both labeled images of the training classes and unlabeled images of the test class to train the SOMs. When classifying an image, SOMZSL maps it to the SOM of the feature space and then forwards it to the SOM of the attribute space thanks to the one-to-one correspondence. This results in the image being represented with a vector of the same dimension as that of the attribute space, which in turn allows a direct comparison between the test image and the test classes.

5.3 The impact and insights of the proposed method

SOMZSL comes with several advantages. First, SOMZSL does not have to deal with a costly optimization problem thanks to the heuristic nature of the SOM algorithm on which SOMZSL is based. Also, the SOM algorithm allows SOMZSL to preserve the topology of the feature and the semantic space. What is more, SOMZSL also incorporates unlabeled test images in the training process, which helps mitigate the projection domain shift problem inherent in ZSL.

To see the above advantages of SOMZSL in practise, we conducted several experiments on six benchmark datasets. SOMZSL, achieves the highest (Top-1) accuracy for half of these datasets compared to four state-of-the-art ZSL methods, namely DeViSE [7], ALE [5], SJE [8], and ESZSL [9]. Specifically, it increases the accuracy by up to 4% compared to the second best ZSL method in each case. In addition, based on the time complexity analysis, we showed that SOMZSL requires less time to build a ZSL model than DeViSE , ALE, SJE , and ESZSL.

We observe that for fine-grained datasets, where classes have similar appearances, SOMZSL exhibits comparable performance though; it is is particularly successful on coarse-grained datasets, where the classes are relatively more distinguishable. As for the ideal SOM size that SOMZSL requires, we find that the ideal size is usually around 20×20, which varies slightly depending on the size of the dataset.

5.4 Limitations

The proposed method SOMZSL is mainly limited by the capabilities of the SOM algorithm. As is well known, the theoretical aspects of SOM still remain elusive 30 years after its invention. Therefore SOMZSL cannot provide theoretical guarantees for its performance.

5.5 Future research

Being a simple and robust method, SOM has long been studied by researchers, which has led to the development of various applications and extensions, such as growing SOM, 3D SOM. Considering these extensions for SOMZSL has the potential to improve the efficiency of SOMZSL, and thus can be a good direction for future research. It is also evident that the problem of not having labeled instances for certain classes is not limited to image classification, so SOMZSL cannot be restricted to ZSL only. Therefore, SOMZSL can be used in a wider range of object recognition tasks in the future.

Data availibility statement The data used in this research are public and can be accessed through the references provided.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Palatucci MM, Pomerleau DA, Hinton GE, Mitchell T (2009) Zero-shot learning with semantic output codes. In: Proceedings of advances in neural information processing systems, 22
2. Lampert CH, Nickisch H, Harmeling S (2013) Attribute-based classification for zero-shot visual object categorization. IEEE Transact Pattern Anal Mach Intell 36(3):453–465
3. Xian Y, Lampert CH, Schiele B, Akata Z (2018) Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. IEEE Transact Pattern Anal Mach Intell 41(9):2251–2265
4. Lampert, C.H., Nickisch, H., Harmeling, S (2009) Learning to detect unseen object classes by between-class attribute transfer. In: Proceedings of the IEEE conference on computer vision and pattern recognition, p 951–958 . IEEE
5. Akata Z, Perronnin F, Harchaoui Z, Schmid C (2015) Label-embedding for image classification. IEEE Transact Pattern Anal Mach Intell 38(7):1425–1438
6. Kodirov E, Xiang T, Gong S (2017) Semantic autoencoder for zero-shot learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, p 3174–3183
7. Frome A, Corrado GS, Shlens J, Bengio S, Dean J, Ranzato MA, Mikolov T (2013) Devise: a deep visual-semantic embedding model. In: Proceedings of advances in neural information processing systems, 26
8. Akata Z, Reed S, Walter D, Lee H, Schiele B (2015) Evaluation of output embeddings for fine-grained image classification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2927–2936
9. Romera-Paredes B, Torr P (2015) An embarrassingly simple approach to zero-shot learning. In: Proceedings of international conference on machine learning, pp 2152–2161. PMLR
10. Li K, Min MR, Fu Y (2019) Rethinking zero-shot learning: A conditional visual classification perspective. In: Proceedings of the IEEE/CVF international conference on computer vision, p 3583–3592
11. Fu Y, Hospedales TM, Xiang T, Gong S (2015) Transductive multi-view zero-shot learning. IEEE Transact Pattern Anal Mach Intell 37(11):2332–2345
12. Kohonen T (2000) Self-Organizing Maps. Springer, Berlin
13. Chen Z, Liu B (2018) Lifelong machine learning. Synth Lect Artif Intell Mach Learn 12(3):1–207
14. Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Transact Knowl Data Eng 22(10):1345–1359
15. Sariyildiz MB, Cinbis RG (2019) Gradient matching generative networks for zero-shot learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, p 2168–2178
16. Annadani Y, Biswas S (2018) Preserving semantic relations for zero-shot learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, p 7603–7612
17. Changpinyo S, Chao WL, Gong B, Sha F (2016) Synthesized classifiers for zero-shot learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, p 5327–5336
18. Liu S, Long M, Wang J, Jordan MI (2018) Generalized zero-shot learning with deep calibration network, 31
19. Lei Ba J, Swersky K, Fidler S (2015) Predicting deep zero-shot convolutional neural networks using textual descriptions. In: Proceedings of the IEEE international conference on computer vision, p 4247–4255
20. Zhang L, Wang P, Liu L, Shen C, Wei W, Zhang Y, Van Den Hengel A (2020) Towards effective deep embedding for zero-shot learning. IEEE Transact Circuits Syst Video Technol 30(9):2843–2852
21. Mishra A, Krishna Reddy S, Mittal A, Murthy HA(2018) A generative model for zero shot learning using conditional variational autoencoders. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, p 2188–2196
22. Chalasani R, Principe JC (2015) Self-organizing maps with information theoretic learning. Neurocomputing 147:3–14
23. Laerhoven, K.V (2001) Combining the self-organizing map and k-means clustering for on-line classification of sensor data. In: Proceedings of the international conference on artificial neural networks, p 464–469
24. Liu Y, Weisberg RH (2011) A review of self-organizing map applications in meteorology and oceanography. Self-Organ Maps: Appl Novel Algorithm Des 1:253–272
25. Alahakoon D, Halgamuge SK (2000) Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Transact Neural Netw 11:601–614
26. Akinduko AA, Mirkes EM, Gorban AN (2016) Som: Stochastic initialization versus principal components. Inform Sci 364:213–221
27. Cottrell M, Olteanu M, Rossi F, Villa-Vialaneix N (2018) Self-organizing maps, theory and applications. Revista de Investigacion Operacional 39(1):1–22
28. Mariette J, Villa-Vialaneix N (2016) Aggregating self-organizing maps with topology preservation. Adv Self-Organ Maps Learn Vector Quant. Springer, Berlin, pp 27–37
29. Bourgeois N, Cottrell M, Déruelle B, Lamassé S, Letrémy P (2015) How to improve robustness in kohonen maps and display additional information in factorial analysis: application to text mining. Neurocomputing 147:120–135
30. Zheng L, Yang Y, Tian Q (2017) Sift meets cnn: a decade survey of instance retrieval. IEEE Transact Pattern Anal Mach Intell 40(5):1224–1244
31. Patterson G, Hays J (2012) Sun attribute database: Discovering, annotating, and recognizing scene attributes. In: Proceedings of the IEEE conference on computer vision and pattern recognition, p 2751–2758
32. Farhadi A, Endres I, Hoiem D, Forsyth D (2009) Describing objects by their attributes. In: Proceedings of the IEEE conference on computer vision and pattern recognition, p 1778–1785
33. Welinder P, Branson S, Mita T, Wah C, Schroff F, Belongie S, Perona P (2010) Caltech-ucsd birds 200
34. Van Horn G, Branson S, Farrell R, Haber S, Barry J, Ipeirotis P, Perona P, Belongie S (2015) Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, p 595–604
35. Elhoseiny M, Elfeki M (2019) Creativity inspired zero-shot learning. In: Proceedings of the IEEE/CVF international conference on computer vision, p 5784–5793

36. Kohonen T (2013) Essentials of the self-organizing map. Neural Netw 37:52–65
37. Cottrell M, Olteanu M, Rossi F, Villa-Vialaneix N (2016) Theoretical and applied aspects of the self-organizing maps. Adv Self-Org Maps Learn Vector Quant. Springer, Berlin, pp 3–26