arXiv:1603.00550v3 [cs.CV] 27 May 2016

Synthesized Classifiers for Zero-Shot Learning

Soravit Changpinyo*, Wei-Lun Chao*
U. of Southern California
Los Angeles, CA
schangpi, weilunc@usc.edu

Boqing Gong
U. of Central Florida
Orlando, FL
bgong@crcv.ucf.edu

Fei Sha
U. of California
Los Angeles, CA
feisha@cs.ucla.edu

Abstract

Given semantic descriptions of object classes, zero-shot learning aims to accurately recognize objects of the unseen *classes, from which no examples are available at the training stage, by associating them to the* seen *classes, from which labeled examples are provided. We propose to tackle this problem from the perspective of manifold learning. Our main idea is to align the semantic space that is derived from external information to the model space that concerns itself with recognizing visual features. To this end, we introduce a set of "phantom" object classes whose coordinates live in both the semantic space and the model space. Serving as bases in a dictionary, they can be optimized from labeled data such that the* synthesized *real object classifiers achieve optimal discriminative performance. We demonstrate superior accuracy of our approach over the state of the art on four benchmark datasets for zero-shot learning, including the full ImageNet Fall 2011 dataset with more than 20,000 unseen classes.*

1. Introduction

Visual recognition has made significant progress due to the widespread use of deep learning architectures [23, 45] that are optimized on large-scale datasets of human-labeled images [40]. Despite the exciting advances, to recognize objects "in the wild" remains a daunting challenge. Many objects follow a long-tailed distribution: in contrast to common objects such as household items, they do not occur frequently enough for us to collect and label a large set of representative exemplar images.

For example, this challenge is especially crippling for fine-grained object recognition (classifying species of birds, designer products, etc.). Suppose we want to carry a visual search of "Chanel Tweed Fantasy Flap Handbag". While handbag, flap, tweed, and Chanel are popular accessory, style, fabric, and brand, respectively, the combination of them is rare — the query generates about 55,000 results on Google search with a small number of images. The amount of labeled images is thus far from enough for directly building a high-quality classifier, unless we treat this category as a composition of attributes, for each of which more training data can be easily acquired [25].

It is thus imperative to develop methods for zero-shot learning, namely, to expand classifiers and the space of possible labels beyond *seen* objects, of which we have access to the labeled images for training, to *unseen* ones, of which no labeled images are available [25, 34]. To this end, we need to address two key interwoven challenges [34]: (1) how to relate unseen classes to seen ones and (2) how to attain optimal discriminative performance on the unseen classes even though we do not have their labeled data.

To address the first challenge, researchers have been using visual attributes [12, 24, 35] and word vectors [13, 31, 44] to associate seen and unseen classes. We call them the semantic embeddings of objects. Much work takes advantage of such embeddings directly as middle layers between input images and output class labels [1, 13, 19, 25, 27, 28, 33, 39, 44], whereas others derive new representations from the embeddings using, for example, Canonical Correlation Analysis (CCA) [14, 15, 29] or sparse coding [22, 50, 51]. For the second challenge, the hand-designed probabilistic models in [25] have been competitive baselines. More recent studies show that nearest neighbor classifiers in the semantic space are very effective [13, 14, 15, 16, 33]. Finally, classifiers for the unseen classes can directly be constructed in the input feature space [1, 10, 26, 30, 48, 51].

In this paper, we tackle these two challenges with ideas from manifold learning [5, 18], converging to a two-pronged approach. We view object classes in a se-

* Equal contributions

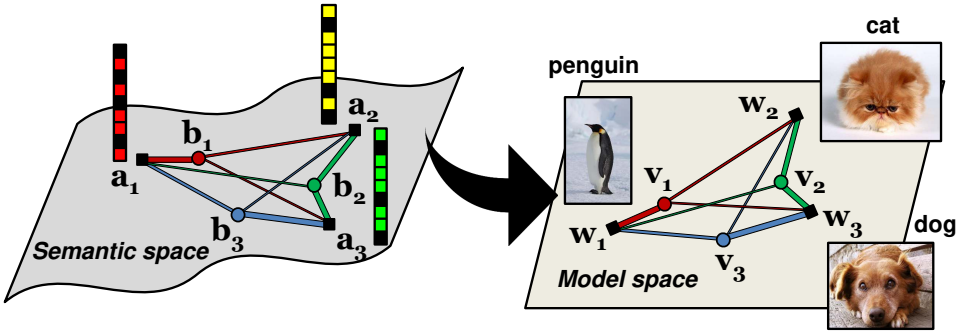

Figure 1: Illustration of our method for zero-shot learning. Object classes live in two spaces. They are characterized in the semantic space with semantic embeddings ($\boldsymbol{a}$s) such as attributes and word vectors of their names. They are also represented as models for visual recognition ($\boldsymbol{w}$s) in the model space. In both spaces, those classes form weighted graphs. The main idea behind our approach is that these two spaces should be aligned. In particular, the coordinates in the model space should be the projection of the graph vertices from the semantic space to the model space — preserving class relatedness encoded in the graph. We introduce adaptable phantom classes ($\boldsymbol{b}$ and $\boldsymbol{v}$) to connect seen and unseen classes — classifiers for the phantom classes are bases for synthesizing classifiers for real classes. In particular, the synthesis takes the form of convex combination.

mantic space as a weighted graph where the nodes correspond to object class names and the weights of the edges represent how they are related. Various information sources can be used to infer the weights — human-defined attributes or word vectors learnt from language corpora. On the other end, we view models for recognizing visual images of those classes as if they live in a space of models. In particular, the parameters for each object model are nothing but coordinates in this model space whose geometric configuration also reflects the relatedness among objects. Fig. 1 illustrates this idea conceptually.

But *how do we align the semantic space and the model space*? The semantic space coordinates of objects are designated or derived based on external information (such as textual data) that do not directly examine visual appearances at the lowest level, while the model space concerns itself largely for recognizing low-level visual features. To align them, we view the coordinates in the model space as the projection of the vertices on the graph from the semantic space — there is a wealth of literature on manifold learning for computing (low-dimensional) Euclidean space embeddings from the weighted graph, for example, the well-known algorithm of Laplacian eigenmaps [5].

To adapt the embeddings (or the coordinates in the model space) to data, we introduce a set of *phantom* object classes — the coordinates of these classes in both the semantic space and the model space are adjustable and optimized such that the resulting model for the real object classes achieve the best performance in discriminative tasks. However, as their names imply, those phantom classes do not correspond to and are not optimized to recognize any real classes directly. For mathematical convenience, we parameterize the weighted graph in the semantic space with the phantom classes in such a way that the model for any real class is a convex combinations of the coordinates of those phantom classes. In other words, the "models" for the phantom classes can also be interpreted as bases (classifiers) in a dictionary from which a large number of classifiers for real classes can be synthesized via convex combinations. In particular, when we need to construct a classifier for an unseen class, we will compute the convex combination coefficients from this class's semantic space coordinates and use them to form the corresponding classifier.

To summarize, our main contribution is a novel idea to cast the challenging problem of recognizing unseen classes as learning manifold embeddings from graphs composed of object classes. As a concrete realization of this idea, we show how to parameterize the graph with the locations of the phantom classes, and how to derive embeddings (i.e., recognition models) as convex combinations of base classifiers. Our empirical studies extensively test our synthesized classifiers on four benchmark datasets for zero-shot learning, including the full ImageNet Fall 2011 release [8] with 20,345 unseen classes. The experimental results are very encouraging; the synthesized classifiers outperform several state-of-the-art methods, including attaining better or matching performance of Google's ConSE algorithm [33] in the large-scale setting.

The rest of the paper is organized as follows. We give an overview of relevant literature in Section 2, describe our approach in detail in Section 3, demonstrate its effectiveness in Section 4, and conclude in Section 5.

2. Related Work

In order to transfer knowledge between classes, zero-shot learning relies on semantic embeddings of class labels, including attributes (both manually defined [1, 25, 47] and discriminatively learned [3, 49]), word vectors [13, 33, 38, 44], knowledge mined from the Web [10, 30, 37, 38], or a combination of several embeddings [2, 14, 16].

Given semantic embeddings, existing approaches to zero-shot learning mostly fall into embedding-based and similarity-based methods. In the embedding-based approaches, one first maps the input image representations to the semantic space, and then determines the class labels in this space by various relatedness measures implied by the class embeddings [1, 2, 13, 14, 16, 22, 25, 28, 33, 44, 47]. Our work as well as some recent work combine these two stages [1, 2, 13, 39, 47, 50, 51], leading to a unified framework empirically shown to have, in general, more accurate predictions. In addition to directly using fixed semantic embeddings, some work maps them into a different space through CCA [14, 15, 29] and sparse coding [22, 50, 51].

In the similarity-based approaches, in contrast, one builds the classifiers for unseen classes by relating them to seen ones via class-wise similarities [10, 16, 17, 30, 37, 38]. Our approach shares a similar spirit to these models but offers richer modeling flexibilities thanks to the introduction of phantom classes.

Finally, our convex combination of base classifiers for synthesizing real classifiers can also be motivated from multi-task learning with shared representations [4]. While labeled examples of each task are required in [4], our method has no access to *data* of the unseen classes.

3. Approach

We describe our methods for addressing zero-shot learning where the task is to classify images from unseen classes into the label space of unseen classes.

Notations Suppose we have training data $\mathcal{D} = \{(\boldsymbol{x}_n \in \mathbb{R}^{\mathsf{D}}, y_n)\}_{n=1}^{\mathsf{N}}$ with the labels coming from the label space of *seen* classes $\mathcal{S} = \{1, 2, \cdots, \mathsf{S}\}$. Denote by $\mathcal{U} = \{\mathsf{S}+1, \cdots, \mathsf{S}+\mathsf{U}\}$ the label space of *unseen* classes.

We focus on linear classifiers in the visual feature space $\mathbb{R}^{\mathsf{D}}$ that assign a label $\hat{y}$ to a data point $\boldsymbol{x}$ by

$$\hat{y} = \arg\max_{c} \quad \boldsymbol{w}_c^{\mathsf{T}} \boldsymbol{x}, \tag{1}$$

where $\boldsymbol{w}_c \in \mathbb{R}^{\mathsf{D}}$, although our approach can be readily extended to nonlinear settings by the kernel trick [41].

3.1. Main idea

Manifold learning The main idea behind our approach is shown by the conceptual diagram in Fig. 1. Each class c has a coordinate $\boldsymbol{a}_c$ and they live on a manifold in the semantic embedding space. In this paper, we explore two types of such spaces: attributes [25, 46] and class name embeddings via word vectors [32]. We use attributes in this text to illustrate the idea and in the experiments we test our approach on both types.

Additionally, we introduce a set of *phantom* classes associated with semantic embeddings $\boldsymbol{b}_r, r = 1, 2, \ldots, \mathsf{R}$. We stress that they are phantom as they themselves do **not** correspond to any real objects — they are introduced to increase the modeling flexibility, as shown below.

The real and phantom classes form a weighted bipartite graph, with the weights defined as

$$s_{cr} = \frac{\exp\{-d(\boldsymbol{a}_c, \boldsymbol{b}_r)\}}{\sum_{r=1}^{\mathsf{R}} \exp\{-d(\boldsymbol{a}_c, \boldsymbol{b}_r)\}} \tag{2}$$

to correlate a real class c and a phantom class r, where

$$d(\boldsymbol{a}_c, \boldsymbol{b}_r) = (\boldsymbol{a}_c - \boldsymbol{b}_r)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{a}_c - \boldsymbol{b}_r), \tag{3}$$

and $\boldsymbol{\Sigma}^{-1}$ is a parameter that can be learned from data, modeling the correlation among attributes. For simplicity, we set $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$ and tune the scalar free hyper-parameter σ by cross-validation. The more general Mahalanobis metric can be used and we propose one way of learning such metric as well as demonstrate its effectiveness in the Suppl.

The specific form of defining the weights is motivated by several manifold learning methods such as SNE [18]. In particular, s_{cr} can be interpreted as the conditional probability of observing class r in the neighborhood of class c. However, other forms can be explored and are left for future work.

In the model space, each real class is associated with a classifier $\boldsymbol{w}_c$ and the phantom class r is associated with a virtual classifier $\boldsymbol{v}_r$. We align the semantic and the model spaces by viewing $\boldsymbol{w}_c$ (or $\boldsymbol{v}_r$) as the embedding of the weighted graph. In particular, we appeal to the idea behind Laplacian eigenmaps [5], which seeks the embedding that maintains the graph structure as much as possible; equally, the distortion error

$$\min_{\boldsymbol{w}_c, \boldsymbol{v}_r} \|\boldsymbol{w}_c - \sum_{r=1}^{\mathsf{R}} s_{cr} \boldsymbol{v}_r\|_2^2$$

is minimized. This objective has an analytical solution

$$\boldsymbol{w}_c = \sum_{r=1}^{\mathsf{R}} s_{cr} \boldsymbol{v}_r, \quad \forall c \in \mathcal{T} = \{1, 2, \cdots, \mathsf{S}+\mathsf{U}\} \tag{4}$$

In other words, the solution gives rise to the idea of *synthesizing classifiers* from those virtual classifiers $\boldsymbol{v}_r$. For conceptual clarity, from now on we refer to $\boldsymbol{v}_r$ as base classifiers in a dictionary from which new classifiers can be synthesized. We identify several advantages. First, we could construct an infinite number of classifiers as long as we know how to compute s_{cr}. Second, by making $\mathsf{R} \ll \mathsf{S}$, the formulation can significantly reduce the learning cost as we only need to learn R base classifiers.

3.2. Learning phantom classes

Learning base classifiers We learn the base classifiers $\{\boldsymbol{v}_r\}_{r=1}^{\mathsf{R}}$ from the training data (of the seen classes only). We experiment with two settings. To learn one-versus-other classifiers, we optimize,

$$\min_{\boldsymbol{v}_1,\cdots,\boldsymbol{v}_{\mathsf{R}}} \sum_{c=1}^{\mathsf{S}} \sum_{n=1}^{\mathsf{N}} \ell(\boldsymbol{x}_n, \mathbb{I}_{y_n,c}; \boldsymbol{w}_c) + \frac{\lambda}{2} \sum_{c=1}^{\mathsf{S}} \|\boldsymbol{w}_c\|_2^2, \quad (5)$$
$$\text{s.t.} \quad \boldsymbol{w}_c = \sum_{r=1}^{\mathsf{R}} s_{cr} \boldsymbol{v}_r, \quad \forall c \in \mathcal{T} = \{1, \cdots, \mathsf{S}\}$$

where $\ell(\boldsymbol{x}, y; \boldsymbol{w}) = \max(0, 1 - y\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x})^2$ is the squared hinge loss. The indicator $\mathbb{I}_{y_n,c} \in \{-1, 1\}$ denotes whether or not $y_n = c$. Alternatively, we apply the Crammer-Singer multi-class SVM loss [7], given by

$$\begin{aligned}&\ell_{\text{cs}}(\boldsymbol{x}_n,\ y_n; \{\boldsymbol{w}_c\}_{c=1}^{\mathsf{S}}) \\ &= \max(0, \max_{c \in \mathcal{S} - \{y_n\}} \Delta(c, y_n) + \boldsymbol{w}_c{}^{\mathsf{T}}\boldsymbol{x}_n - \boldsymbol{w}_{y_n}{}^{\mathsf{T}}\boldsymbol{x}_n),\end{aligned}$$

We have the standard Crammer-Singer loss when the structured loss $\Delta(c, y_n) = 1$ if $c \neq y_n$, which, however, ignores the semantic relatedness between classes. We additionally use the ℓ_2 distance for the structured loss $\Delta(c, y_n) = \|\boldsymbol{a}_c - \boldsymbol{a}_{y_n}\|_2^2$ to exploit the class relatedness in our experiments. These two learning settings have separate strengths and weaknesses in empirical studies.

Learning semantic embeddings The weighted graph eq. (2) is also parameterized by adaptable embeddings of the phantom classes $\boldsymbol{b}_r$. For this work, however, for simplicity, we assume that each of them is a sparse linear combination of the seen classes' attribute vectors:

$$\boldsymbol{b}_r = \sum_{c=1}^{\mathsf{S}} \beta_{rc} \boldsymbol{a}_c, \forall r \in \{1, \cdots, \mathsf{R}\},$$

Thus, to optimize those embeddings, we solve the following optimization problem

$$\begin{aligned}&\min_{\{\boldsymbol{v}_r\}_{r=1}^{\mathsf{R}}, \{\beta_{rc}\}_{r,c=1}^{\mathsf{R},\mathsf{S}}} \sum_{c=1}^{\mathsf{S}} \sum_{n=1}^{\mathsf{N}} \ell(\boldsymbol{x}_n, \mathbb{I}_{y_n,c}; \boldsymbol{w}_c) \\ &+ \frac{\lambda}{2} \sum_{c=1}^{\mathsf{S}} \|\boldsymbol{w}_c\|_2^2 + \eta \sum_{r,c=1}^{\mathsf{R},\mathsf{S}} |\beta_{rc}| + \frac{\gamma}{2} \sum_{r=1}^{\mathsf{R}} (\|\boldsymbol{b}_r\|_2^2 - h^2)^2, \\ &\text{s.t.} \quad \boldsymbol{w}_c = \sum_{r=1}^{\mathsf{R}} s_{cr} \boldsymbol{v}_r, \quad \forall c \in \mathcal{T} = \{1, \cdots, \mathsf{S}\},\end{aligned}$$

where h is a predefined scalar equal to the norm of real attribute vectors (i.e., 1 in our experiments since we perform ℓ_2 normalization). Note that in addition to learning $\{\boldsymbol{v}_r\}_{r=1}^{\mathsf{R}}$, we learn combination weights $\{\beta_{rc}\}_{r,c=1}^{\mathsf{R},\mathsf{S}}$. Clearly, the constraint together with the third term in the objective encourages the sparse linear combination of the seen classes' attribute vectors. The last term in the objective demands that the norm of $\boldsymbol{b}_r$ is not too far from the norm of $\boldsymbol{a}_c$.

We perform alternating optimization for minimizing the objective function with respect to $\{\boldsymbol{v}_r\}_{r=1}^{\mathsf{R}}$ and $\{\beta_{rc}\}_{r,c=1}^{\mathsf{R},\mathsf{S}}$. While this process is nonconvex, there are useful heuristics to initialize the optimization routine. For example, if $\mathsf{R} = \mathsf{S}$, then the simplest setting is to let $\boldsymbol{b}_r = \boldsymbol{a}_r$ for $r = 1, \ldots, \mathsf{R}$. If $\mathsf{R} \leq \mathsf{S}$, we can let them be (randomly) selected from the seen classes' attribute vectors $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_{\mathsf{R}}\} \subseteq \{\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_{\mathsf{S}}\}$, or first perform clustering on $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_{\mathsf{S}}\}$ and then let each $\boldsymbol{b}_r$ be a combination of the seen classes' attribute vectors in cluster r. If $\mathsf{R} > \mathsf{S}$, we could use a combination of the above two strategies. We describe in more detail how to optimize and cross-validate hyperparameters in the Suppl.

3.3. Comparison to several existing methods

We contrast our approach to some existing methods. [30] combines **pre-trained** classifiers of seen classes to construct new classifiers. To estimate the semantic embedding (e.g., word vector) of a test image, [33] uses the decision values of pre-trained classifiers of seen objects to weighted average the corresponding semantic embeddings. Neither of them has the notion of base classifiers, which we introduce for constructing the classifiers and nothing else. We thus expect them to be more effective in transferring knowledge between seen and unseen classes than overloading the pretrained and fixed classifiers of the seen classes for dual duties. We note that [1] can be considered as a special case of our method. In [1], each attribute corresponds to a base and each "real" classifier corresponding to an actual object

is represented as a linear combination of those bases, where the weights are the real objects' "descriptions" in the form of attributes. This modeling is limiting as the number of bases is fundamentally limited by the number of attributes. Moreover, the model is strictly a subset of our model.[1] Recently, [50, 51] propose similar ideas of aligning the visual and semantic spaces but take different approaches from ours.

4. Experiments

We evaluate our methods and compare to existing state-of-the-art models on several benchmark datasets. While there is a large degree of variations in the current empirical studies in terms of datasets, evaluation protocols, experimental settings, and implementation details, we strive to provide a comprehensive comparison to as many methods as possible, not only citing the published results but also reimplementing some of those methods to exploit several crucial insights we have discovered in studying our methods.

We summarize our main results in this section. More extensive details are reported in the Suppl. We provide not only comparison in recognition accuracy but also analysis in an effort to understand the sources of better performance.

4.1. Setup

Datasets We use four benchmark datasets in our experiments: the **Animals with Attributes (AwA)** [25], **CUB-200-2011 Birds (CUB)** [46], **SUN Attribute (SUN)** [36], and the **ImageNet** (with full 21,841 classes) [40]. Table 1 summarizes their key characteristics. The Suppl. provides more details.

Semantic spaces For the classes in **AwA**, we use 85-dimensional binary or continuous attributes [25], as well as the 100 and 1,000 dimensional word vectors [31], derived from their class names and extracted by Fu et al. [14, 15]. For **CUB** and **SUN**, we use 312 and 102 dimensional continuous-valued attributes, respectively. We also thresh them at the global means to obtain binary-valued attributes, as suggested in [25]. Neither datasets have word vectors for their class names. For **ImageNet**, we train a skip-gram language model [31, 32] on the latest Wikipedia dump corpus[2] (with more than 3 billion words) to extract a 500-dimensional word vector for each class. Details of this training are in the Suppl. We ignore classes without word vectors in the experiments, resulting in 20,345 (out of 20,842) unseen classes. For both the continuous attribute vectors and the word vector embeddings of the class names, we normalize them to have unit ℓ_2 norms unless stated otherwise.

Table 1: Key characteristics of studied datasets

Dataset name	# of seen classes	# of unseen classes	Total # of images
AwA†	40	10	30,475
CUB‡	150	50	11,788
SUN‡	645/646	72/71	14,340
ImageNet§	1,000	20,842	14,197,122

†: Following the prescribed split in [25].
‡: 4 (or 10, respectively) random splits, reporting average.
§: Seen and unseen classes from ImageNet ILSVRC 2012 1K [40] and Fall 2011 release [8, 13, 33].

Visual features Due to variations in features being used in literature, it is impractical to try all possible combinations of features and methods. Thus, we make a major distinction in using shallow features (such as color histograms, SIFT, PHOG, Fisher vectors) [1, 2, 20, 25, 38, 47] and deep learning features in several recent studies of zero-shot learning. Whenever possible, we use (shallow) features provided by those datasets or prior studies. For comparative studies, we also extract the following deep features: AlexNet [23] for **AwA** and **CUB** and GoogLeNet [45] for all datasets (all extracted with the Caffe package [21]). For AlexNet, we use the 4,096-dimensional activations of the penultimate layer (fc7) as features. For GoogLeNet, we take the 1,024-dimensional activations of the pooling units, as in [2]. Details on how to extract those features are in the Suppl.

Evaluation protocols For **AwA**, **CUB**, and **SUN**, we use the (normalized, by class-size) multi-way classification accuracy, as in previous work. Note that the accuracy is always computed on images from unseen classes.

Evaluating zero-shot learning on the large-scale **ImageNet** requires substantially different components from evaluating on the other three datasets. First, two evaluation metrics are used, as in [13]: Flat hit@K (F@K) and Hierarchical precision@K (HP@K).

[1] For interested readers, if we set the number of attributes as the number of phantom classes (each $\boldsymbol{b}_r$ is the one-hot representation of an attribute), and use Gaussian kernel with anisotropically diagonal covariance matrix in eq. (3) with properly set bandwidths (either very small or very large) for each attribute, we will recover the formulation in [1] when the bandwidths tend to zero or infinity.

[2] http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2 on September 1, 2015

F@K is defined as the percentage of test images for which the model returns the true label in its top K predictions. Note that, F@1 is the multi-way classification accuracy. HP@K takes into account the hierarchical organization of object categories. For each true label, we generate a ground-truth list of K closest categories in the hierarchy and compute the degree of overlapping (i.e., precision) between the ground-truth and the model's top K predictions. For the detailed description of this metric, please see the Appendix of [13] and the Suppl.

Secondly, following the procedure in [13, 33], we evaluate on three scenarios of increasing difficulty:

- *2-hop* contains 1,509 unseen classes that are within two tree hops of the seen 1K classes according to the ImageNet label hierarchy[3].
- *3-hop* contains 7,678 unseen classes that are within three tree hops of seen classes.
- *All* contains all 20,345 unseen classes in the ImageNet 2011 21K dataset that are not in the ILSVRC 2012 1K dataset.

The numbers of unseen classes are slightly different from what are used in [13, 33] due to the missing semantic embeddings (i.e., word vectors) for certain class names.

In addition to reporting published results, we have also reimplemented the state-of-the-art method ConSE [33] on this dataset, introducing a few improvements. Details are in the Suppl.

Implementation details We cross-validate all hyperparameters — details are in the Suppl. For convenience, we set the number of phantom classes R to be the same as the number of seen classes S, and set $\boldsymbol{b}_r = \boldsymbol{a}_c$ for $r = c$. We also experiment setting different R and learning $\boldsymbol{b}_r$. Our study (cf. Fig. 2) shows that when R is about 60% of S, the performance saturates. We denote the three variants of our methods in constructing classifiers (Section 3.2) by Ours$^{\text{o-vs-o}}$ (one-versus-other), Ours^{cs} (Crammer-Singer) and Ours^{struct} (Crammer-Singer with structured loss).

4.2. Experimental results

4.2.1 Main results

Table 2 compares the proposed methods to the state-of-the-art from the previously published results on benchmark datasets. While there is a large degree of variations in implementation details, the main observation is that our methods attain the best performance in most scenarios. In what follows, we analyze those results in detail.

We also point out that the settings in some existing work are highly different from ours; we do not include their results in the main text for fair comparison [3, 14, 15, 16, 19, 22, 27, 49] — but we include them in the Suppl. In some cases, even with additional data and attributes, those methods underperform ours.

Table 2: Comparison between our results and the previously *published* results in multi-way classification accuracies (in %) on the task of zero-shot learning. For each dataset, the best is in red and the 2nd best is in blue.

Methods	**AwA**	**CUB**	**SUN**	**ImageNet**
DAP [25]	***41.4***	-	***22.2***	-
IAP [25]	***42.2***	-	***18.0***	-
BN [47]	***43.4***	-	-	-
ALE [1]	***37.4***	***18.0***†	-	-
SJE [2]	***66.7***	***50.1***†	-	-
ESZSL [39]	***49.3***	-	-	-
ConSE[33]	-	-	-	***1.4***
SSE-ReLU [51]*	***76.3***	***30.4***†	-	-
[50]*	***80.5***	***42.1***†	-	-
Ours$^{\text{o-vs-o}}$	69.7	53.4	62.8	1.4
Ours^{cs}	68.4	51.6	52.9	-
Ours^{struct}	72.9	54.7	62.7	1.5

†: Results reported on a particular seen-unseen split.
*: Results were just brought to our attention. Note that VGG [43] instead of GoogLeNet features were used, improving on **AwA** but worsening on **CUB**. Our results using VGG will appear in a longer version of this paper.

4.2.2 Large-scale zero-shot learning

One major limitation of most existing work on zero-shot learning is that the number of unseen classes is often small, dwarfed by the number of seen classes. However, real-world computer vision systems need to face a very large number of unseen objects. To this end, we evaluate our methods on the large-scale **ImageNet** dataset.

Table 3 summarizes our results and compares to the ConSE method [33], which is, to the best of our knowledge, the state-of-the-art method on this dataset.[4] Note that in some cases, our own implementation ("ConSE by us" in the table) performs slightly worse than the reported results, possibly attributed to differences in visual features, word vector embeddings, and other implementation details. Nonetheless, the proposed methods (using

[3] http://www.image-net.org/api/xml/structure_released.xml

[4] We are aware of recent work by Lu [29] that introduces a novel form of semantic embeddings.

Table 3: Comparison between results by ConSE and our method on **ImageNet**. For both types of metrics, the higher the better.

Scenarios	Methods	Flat Hit@K					Hierarchical precision@K			
	K=	1	2	5	10	20	2	5	10	20
2-hop	ConSE [33]	9.4	15.1	24.7	32.7	41.8	21.4	24.7	26.9	28.4
	ConSE by us	8.3	12.9	21.8	30.9	41.7	21.5	23.8	27.5	31.3
	Ours$^{\text{o-vs-o}}$	10.5	16.7	28.6	40.1	52.0	25.1	27.7	30.3	32.1
	Ours^{struct}	9.8	15.3	25.8	35.8	46.5	23.8	25.8	28.2	29.6
3-hop	ConSE [33]	2.7	4.4	7.8	11.5	16.1	5.3	20.2	22.4	24.7
	ConSE by us	2.6	4.1	7.3	11.1	16.4	6.7	21.4	23.8	26.3
	Ours$^{\text{o-vs-o}}$	2.9	4.9	9.2	14.2	20.9	7.4	23.7	26.4	28.6
	Ours^{struct}	2.9	4.7	8.7	13.0	18.6	8.0	22.8	25.0	26.7
All	ConSE [33]	1.4	2.2	3.9	5.8	8.3	2.5	7.8	9.2	10.4
	ConSE by us	1.3	2.1	3.8	5.8	8.7	3.2	9.2	10.7	12.0
	Ours$^{\text{o-vs-o}}$	1.4	2.4	4.5	7.1	10.9	3.1	9.0	10.9	12.5
	Ours^{struct}	1.5	2.4	4.4	6.7	10.0	3.6	9.6	11.0	12.2

the same setting as "ConSE by us") always outperform both, especially in the very challenging scenario of *All* where the number of unseen classes is 20,345, significantly larger than the number of seen classes. Note also that, for both types of metrics, when K is larger, the improvement over the existing approaches is more pronounced. It is also not surprising to notice that as the number of unseen classes increases from the setting *2-hop* to *All*, the performance of both our methods and ConSE degrade.

4.2.3 Detailed analysis

We experiment extensively to understand the benefits of many factors in our and other algorithms. While trying all possible combinations is prohibitively expensive, we have provided a comprehensive set of results for comparison and drawing conclusions.

Advantage of continuous attributes It is clear from Table 4 that, in general, continuous attributes as semantic embeddings for classes attain better performance than binary attributes. This is especially true when deep learning features are used to construct classifiers. It is somewhat expected that continuous attributes provide a more accurate real-valued similarity measure among classes. This presumably is exploited further by more powerful classifiers.

Advantage of deep features It is also clear from Table 4 that, across all methods, deep features significantly boost the performance based on shallow features. We use GoogLeNet and AlexNet (numbers in parentheses) and GoogLeNet generally outperforms AlexNet. It is worthwhile to point out that the reported results under deep features columns are obtained using linear classifiers, which outperform several nonlinear classifiers that use shallow features. This seems to suggest that deep features, often thought to be specifically adapted to seen training images, still work well when transferred to unseen images [13].

Table 5: Effect of types of semantic embeddings on **AwA**.

Semantic embeddings	Dimensions	Accuracy (%)
word vectors	100	42.2
word vectors	1000	57.5
attributes	85	69.7
attributes + word vectors	185	73.2
attributes + word vectors	1085	**76.3**

Which types of semantic space? In Table 5, we show how effective our proposed method (Ours$^{\text{o-vs-o}}$) exploits the two types of semantic spaces: (continuous) attributes and word-vector embeddings on **AwA** (the only dataset with both embedding types). We find that attributes yield better performance than word-vector embeddings. However, combining the two gives the best result, suggesting that these two semantic spaces could be complementary and further investigation is ensured.

Table 6 takes a different view on identifying the best semantic space. We study whether we can learn optimally the semantic embeddings for the phantom classes that correspond to base classifiers. These preliminary studies seem to suggest that learning attributes could have a positive effect, though it is difficult to improve over word-vector embeddings. We plan to study this issue more thoroughly in the future.

How many base classifiers are necessary? In Fig. 2, we investigate how many base classifiers are needed —

Table 4: Detailed analysis of various methods: the effect of feature and attribute types on multi-way classification accuracies (in %). Within each column, the best is in red and the 2nd best is in blue. We cite both previously published results (numbers in bold italics) and results from our implementations of those competing methods (numbers in normal font) to enhance comparability and to ease analysis (see texts for details). We use the shallow features provided by [25], [20], [36] for **AwA**, **CUB**, **SUN**, respectively.

Methods	Attribute type	Shallow features			Deep features		
		AwA	**CUB**	**SUN**	**AwA**	**CUB**	**SUN**
DAP [25]	binary	***41.4***	28.3	***22.2***	60.5 (50.0)	39.1 (34.8)	44.5
IAP [25]	binary	***42.2***	24.4	***18.0***	57.2 (53.2)	36.7 (32.7)	40.8
BN [47]	binary	***43.4***	-	-	-	-	-
ALE [1]‡	binary	***37.4***	***18.0***†	-	-	-	-
ALE	binary	34.8	27.8	-	53.8 (48.8)	40.8 (35.3)	53.8
SJE [2]	continuous	***42.3***‡	***19.0***†‡	-	***66.7 (61.9)***	***50.1 (40.3)***†	-
SJE	continuous	36.2	34.6	-	66.3 (63.3)	46.5 (42.8)	56.1
ESZSL [39]§	continuous	***49.3***	37.0	-	59.6 (53.2)	44.0 (37.2)	8.7
ESZSL	continuous	44.1	38.3	-	64.5 (59.4)	34.5 (28.0)	18.7
ConSE [33]	continuous	36.5	23.7	-	63.3 (56.5)	36.2 (32.6)	51.9
COSTA [30]$^{\sharp}$	continuous	38.9	28.3	-	61.8 (55.2)	40.8 (36.9)	47.9
Ours$^{\text{o-vs-o}}$	continuous	42.6	35.0	-	69.7 (64.0)	53.4 (46.6)	62.8
Ours^{cs}	continuous	42.1	34.7	-	68.4 (64.8)	51.6 (45.7)	52.9
Ours^{struct}	continuous	41.5	36.4	-	72.9 (62.8)	54.5 (47.1)	62.7

†: Results reported by the authors on a particular seen-unseen split.
‡: Based on Fisher vectors as shallow features, different from those provided in [20, 25, 36].
§: On the attribute vectors without ℓ_2 normalization, while our own implementation shows that normalization helps in some cases.
$^{\sharp}$: As co-occurrence statistics are not available, we combine pre-trained classifiers with the weights defined in eq. (2).

Table 6: Effect of learning semantic embeddings

Datasets	Types of embeddings	w/o learning	w/ learning
AwA	attributes	69.7%	71.1%
	100-d word vectors	42.2%	42.5%
	1000-d word vectors	57.6%	56.6%
CUB	attributes	53.4%	54.2%
SUN	attributes	62.8%	63.3%

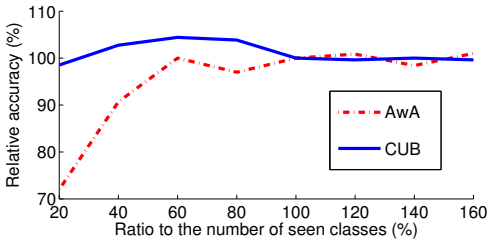

Figure 2: We vary the number of phantom classes R as a percentage of the number of seen classes S and investigate how much that will affect classification accuracy (the vertical axis corresponds to the ratio with respect to the accuracy when R = S). The base classifiers are learned with Ours$^{\text{o-vs-o}}$.

so far, we have set that number to be the number of seen classes out of convenience. The plot shows that in fact, a smaller number (about 60% -70%) is enough for our algorithm to reach the plateau of the performance curve. Moreover, increasing the number of base classifiers does not seem to have an overwhelming effect. Further details and analysis are in the Suppl.

5. Conclusion

We have developed a novel classifier synthesis mechanism for zero-shot learning by introducing the notion of "phantom" classes. The phantom classes connect the dots between the seen and unseen classes — the classifiers of the seen and unseen classes are constructed from the same base classifiers for the phantom classes and with the same coefficient functions. As a result, we can conveniently learn the classifier synthesis mechanism leveraging labeled data of the seen classes and then readily apply it to the unseen classes. Our approach outperforms the state-of-the-art methods on four benchmark datasets in most scenarios.

Acknowledgments

B.G. is partially supported by NSF IIS-1566511. Others are partially supported by USC Annenberg Graduate Fellowship, NSF IIS-1065243, 1451412, 1513966, 1208500, CCF-1139148, a Google Research Award, an Alfred. P. Sloan Research Fellowship and ARO# W911NF-12-1-0241 and W911NF-15-1-0484.

References

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for attribute-based classification. In *CVPR*, 2013. 1, 3, 4, 5, 6, 8, 13

[2] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, 2015. 3, 5, 6, 8, 11, 13, 15

[3] Z. Al-Halah and R. Stiefelhagen. How to transfer? zero-shot object recognition via hierarchical transfer of semantic attributes. In *WACV*, 2015. 3, 6, 14, 15

[4] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008. 3

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. 1, 2, 3

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 14

[7] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2002. 4

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 5, 12, 13

[9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 14

[10] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *ICCV*, 2013. 1, 3, 11

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008. 13

[12] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 1

[13] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 1, 3, 5, 6, 7, 12, 13

[14] Y. Fu, T. M. Hospedales, T. Xiang, Z. Fu, and S. Gong. Transductive multi-view embedding for zero-shot recognition and annotation. In *ECCV*, 2014. 1, 3, 5, 6, 14, 15

[15] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong. Transductive multi-view zero-shot learning. *TPAMI*, 37(11):2332–2345, 2015. 1, 3, 5, 6, 14, 15

[16] Z. Fu, T. Xiang, E. Kodirov, and S. Gong. Zero-shot object recognition by semantic manifold distance. In *CVPR*, 2015. 1, 3, 6, 14, 15

[17] E. Gavves, T. Mensink, T. Tommasi, C. G. M. Snoek, and T. Tuytelaars. Active transfer learning with zero-shot priors: Reusing past datasets for future tasks. In *ICCV*, December 2015. 3

[18] G. E. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *NIPS*, 2002. 1, 3

[19] D. Jayaraman and K. Grauman. Zero-shot recognition with unreliable attributes. In *NIPS*, 2014. 1, 6, 12, 13, 14, 15

[20] D. Jayaraman, F. Sha, and K. Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *CVPR*, 2014. 5, 8, 13

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 2014. 5, 13

[22] E. Kodirov, T. Xiang, Z. Fu, and S. Gong. Unsupervised domain adaptation for zero-shot learning. In *ICCV*, 2015. 1, 3, 6, 14, 15

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 5, 13

[24] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 1

[25] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *TPAMI*, 36(3):453–465, 2014. 1, 3, 5, 6, 8, 11, 12, 13

[26] J. Lei Ba, K. Swersky, S. Fidler, and R. salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *ICCV*, December 2015. 1

[27] X. Li, Y. Guo, and D. Schuurmans. Semi-supervised zero-shot classification with label representation learning. In *ICCV*, 2015. 1, 6, 14, 15

[28] Z. Li, E. Gavves, T. Mensink, and C. G. Snoek. Attributes make sense on segmented objects. In *ECCV*, 2014. 1, 3

[29] Y. Lu. Unsupervised learning of neural network outputs. *arXiv preprint arXiv:1506.00990*, 2015. 1, 3, 6

[30] T. Mensink, E. Gavves, and C. G. Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *CVPR*, 2014. 1, 3, 4, 8

[31] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR Workshops*, 2013. 1, 5, 12

[32] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 3, 5, 12

[33] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014. 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 16, 17

[34] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009. 1

[35] D. Parikh and K. Grauman. Relative attributes. In *ICCV*, 2011. 1

[36] G. Patterson, C. Xu, H. Su, and J. Hays. The sun attribute database: Beyond categories for deeper scene understanding. *IJCV*, 108(1-2):59–81, 2014. 5, 8, 11, 13

[37] M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR*, 2011. 3

[38] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele. What helps where–and why? semantic relatedness for knowledge transfer. In *CVPR*, 2010. 3, 5, 12

[39] B. Romera-Paredes and P. H. S. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, 2015. 1, 3, 6, 8, 11, 12, 13, 14, 15

[40] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 1, 5, 12, 13

[41] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 3

[42] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 14

[43] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6, 14

[44] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013. 1, 3

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 5, 13

[46] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 3, 5, 11

[47] X. Wang and Q. Ji. A unified probabilistic approach modeling relationships between attributes and objects. In *ICCV*, 2013. 3, 5, 6, 8, 12

[48] Y. Yang and T. M. Hospedales. A unified perspective on multi-domain and multi-task learning. In *ICLR*, 2015. 1

[49] F. X. Yu, L. Cao, R. S. Feris, J. R. Smith, and S.-F. Chang. Designing category-level attributes for discriminative visual recognition. In *CVPR*, 2013. 3, 6, 14, 15

[50] Z. Zhang and V. Saligrama. Classifying unseen instances by learning class-independent similarity functions. *arXiv preprint arXiv:1511.04512*, 2015. 1, 3, 5, 6, 12, 14, 15

[51] Z. Zhang and V. Saligrama. Zero-shot learning via semantic similarity embedding. In *ICCV*, 2015. 1, 3, 5, 6, 11, 12, 14, 15

[52] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 13

Supplementary Material: Synthesized Classifiers for Zero-Shot Learning

In this Supplementary Material, we provide details omitted in the main text.

- Section A: cross-validation strategies (Section 3.2 of the main paper).
- Section B: learning metrics for semantic similarity (Section 3.1 of the main paper).
- Section C: details on experimental setup (Section 4.1 of the main paper).
- Section D: implementation details (Section 4.1 and 4.2.3 of the main paper).
- Section E: additional experimental results and analyses (Section 4.2 of the main paper).

A. Cross-validation (CV) strategies

There are a few free hyper-parameters in our approach (cf. Section 3.2 of the main text). To choose the hyper-parameters in the conventional cross-validation (CV) for multi-way classification, one splits the training data into several folds such that they share the same set of class labels with one another. Clearly, this strategy is not sensible for zero-shot learning as it does not imitate what actually happens at the test stage. We thus introduce a new strategy for performing CV, inspired by the hyper-parameter tuning in [39]. The key difference of the new scheme to the conventional CV is that we split the data into several folds such that the class labels of these folds are disjoint. For clarity, we denote the conventional CV as *sample*-wise CV and our scheme as *class*-wise CV. Figure 3(b) and 3(c) illustrate the two scenarios, respectively. We empirically compare them in Section E.1. Note that several existing models [2, 10, 39, 51] also follow similar hyper-prameter tuning procedures.

A.1. Learning semantic embeddings

We propose an optimization problem for learning semantic embeddings in Section 3.2 of the main text. There are four hyper-parameters λ, σ, η, and γ to be tuned. To reduce the search space during cross-validation, we first fix $\boldsymbol{b}_r = \boldsymbol{a}_r$ for $r = 1, \ldots, \mathsf{R}$ and tune λ, σ. Then we fix λ and σ and tune η and γ.

B. Learning metrics for computing similarities between semantic embeddings

Recall that, in Section 3.1 of the main text, the weights in the bipartite graph are defined based on the distance $d(\boldsymbol{a}_c, \boldsymbol{b}_r) = (\boldsymbol{a}_c - \boldsymbol{b}_r)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{a}_c - \boldsymbol{b}_r)$. In this section, we describe an objective for learning a more general Mahalnobis metric than $\Sigma^{-1} = \sigma^2 \boldsymbol{I}$. We focus on the case when $\mathsf{R} = \mathsf{S}$ and on learning a diagonal metric $\Sigma^{-1} = \boldsymbol{M}^T \boldsymbol{M}$, where $\boldsymbol{M}$ is also diagonal. We solve the following optimization problem.

$$\min_{\boldsymbol{M}, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_\mathsf{R}} \sum_{c=1}^{\mathsf{S}} \sum_{n=1}^{\mathsf{N}} \ell(\boldsymbol{x}_n, \mathbb{I}_{y_n, c}; \boldsymbol{w}_c) \tag{6}$$

$$+ \frac{\lambda}{2} \sum_{r=1}^{\mathsf{R}} \|\boldsymbol{v}_r\|_2^2 + \frac{\gamma}{2} \|\boldsymbol{M} - \sigma \boldsymbol{I}\|_F^2 , \tag{7}$$

$$\text{s.t.} \quad \boldsymbol{w}_c = \sum_{r=1}^{\mathsf{R}} s_{cr} \boldsymbol{v}_r, \quad \forall c \in \mathcal{T} = \{1, \cdots, \mathsf{S}\}$$

where $\ell(\boldsymbol{x}, y; \boldsymbol{w}) = \max(0, 1 - y \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x})^2$ is the squared hinge loss. The indicator $\mathbb{I}_{y_n, c} \in \{-1, 1\}$ denotes whether or not $y_n = c$.

Again, we perform alternating optimization for minimizing the above objective function. At first, we fix $\boldsymbol{M} = \sigma \boldsymbol{I}$ and optimize $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_\mathsf{R}\}$ to obtain a reasonable initialization. Then we perform alternating optimization. To further prevent over-fitting, we alternately optimize $\boldsymbol{M}$ and $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_\mathsf{R}\}$ on different but overlapping subsets of training data. In particular, we split data into 5 folds and optimize $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_\mathsf{R}\}$ on the first 4 folds and $\boldsymbol{M}$ on the last 4 folds. We report results in Section E.5.

C. Details on the experimental setup

We present details on the experimental setup in this section and additional results in Section E.

C.1. Datasets

We use four benchmark datasets in our experiments. The **Animals with Attributes (AwA)** dataset [25] consists of 30,475 images of 50 animal classes. Along with the dataset, a standard data split is released for zero-shot learning: 40 seen classes (for training) and 10 unseen classes. The second dataset is the **CUB-200-2011 Birds (CUB)** [46]. It has 200 bird classes and 11,788 images. We randomly split the 200 classes into 4 disjoint sets (each with 50 classes) and treat each of them as the unseen classes in turn. We report the average results from the four splits. The **SUN Attribute (SUN)** dataset [36]

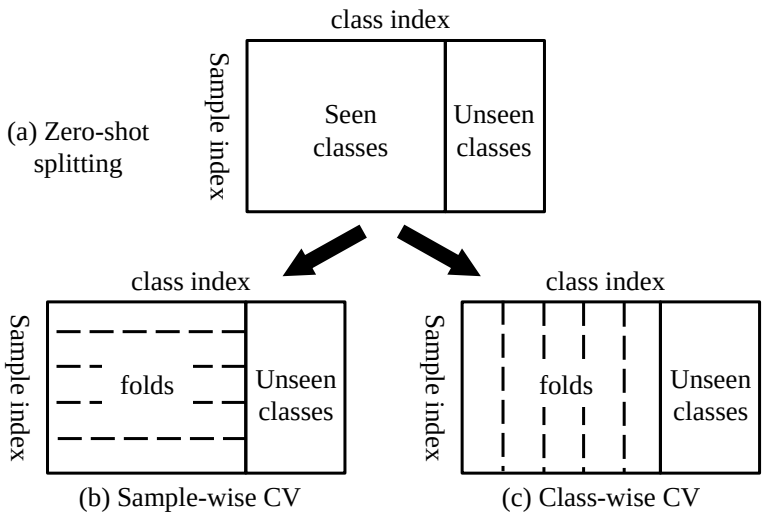

Figure 3: Data splitting for different cross-validation (CV) strategies: (a) the seen-unseen class splitting for zero-shot learning, (b) the sample-wise CV, (c) the class-wise CV (cf. Section 3.2 of the main paper).

contains 14,340 images of 717 scene categories (20 images from each category). Following [25], we randomly split the 717 classes into 10 disjoint sets (each with 71 or 72 classes) in a similar manner to the class splitting on CUB. We note that some previous published results [19, 39, 50, 51] are based on a simpler setting with 707 seen and 10 unseen classes. For comprehensive experimental comparison, we also report our results on this setting in Table 9.

For the large-scale zero-shot experiment on the **ImageNet** dataset [8], we follow the setting in [13, 33]. The ILSVRC 2012 1K dataset [40] contains 1,281,167 training and 50,000 validation images from 1,000 categories and is treated as the seen-class data. Images of unseen classes come from the rest of the ImageNet Fall 2011 release dataset [8] that do not overlap with any of the 1,000 categories. We will call this release the ImageNet 2011 21K dataset (as in [13, 33]). Overall, this dataset contains 14,197,122 images from 21,841 classes, and we conduct our experiment on **20,842 unseen classes**[5].

C.2. Semantic spaces

SUN Each image is annotated with 102 continuous-valued attributes. For each class, we average attribute vectors over all images belonging to that class to obtain a class-level attribute vector.

ImageNet We train a skip-gram language model [31, 32] on the latest Wikipedia dump corpus[6] (with more than 3 billion words) to extract a 500-dimensional word vector for each class. In particular, we train the model using the word2vec package[7] — we preprocess the corpus with the word2phrase function so that we can directly obtain word vectors for both single-word and multiple-word terms, including those terms in the ImageNet *synsets*[8]. We impose no restriction on the vocabulary size. Following [13], we use a window size of 20, apply the hierarchical softmax for predicting adjacent terms, and train the model with a single pass through the corpus. As one class may correspond to multiple word vectors by the nature of synsets, we simply average them to form a single word vector for each class. We obtain word vectors for all the 1,000 seen classes and for **20,345** (out of 20,842) unseen classes. We ignore classes without word vectors in the experiments.

C.3. Visual features

We denote features that are not extracted by deep learning as shallow features.

Shallow features On **AwA**, many existing approaches take traditional features such as color histograms, SIFT, and PHOG that come with the dataset [25, 38, 47],

[5] There is one class in the ILSVRC 2012 1K dataset that does not appear in the ImageNet 2011 21K dataset. Thus, we have a total of 20,842 unseen classes to evaluate.

[6] http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2 on September 1st, 2015

[7] https://code.google.com/p/word2vec/

[8] Each class of **ImageNet** is a *synset*: a set of synonymous terms, where each term is a word or phrase.

while others use the Fisher vectors [1, 2]. The **SUN** dataset also comes with several traditional shallow features, which are used in [19, 25, 39].

In our experiments, we use the shallow features provided by [25] , [20], and [36] for **AwA**, **CUB**, and **SUN**, respectively, unless stated otherwise.

Deep features Given the recent impressive success of deep Convolutional Neural Networks (CNNs) [23] on image classification, we conduct experiments with deep features on all datasets. We use the Caffe package [21] to extract AlexNet [23] and GoogLeNet [45] features for images from **AwA** and **CUB**. Observing that GoogLeNet give superior results over AlexNet on **AwA** and **CUB**, we focus on GoogLeNet features on large datasets: **SUN** and **ImageNet**. These networks are pre-trained on the ILSVRC 2012 1K dataset [8, 40] for **AwA**, **CUB**, and **ImageNet**. For **SUN**, the networks are pre-trained on the Places database [52], which was shown to outperform the networks pre-trained on ImageNet on scene classification tasks. For AlexNet, we use the 4,096-dimensional activations of the penultimate layer (fc7) as features, and for GoogLeNet we extract features by the 1,024-dimensional activations of the pooling units following the suggestion by [2].

For **CUB**, we crop all images with the provided bounding boxes. For **ImageNet**, we center-crop all images and do not perform any data augmentation or other preprocessing.

C.4. Evaluation protocols on ImageNet

When computing Hierarchical precision@K (HP@K), we use the algorithm in the Appendix of [13] to compute a set of at least K classes that are considered to be correct. This set is called $hCorrectSet$ and it is computed for each K and class c. See Algorithm 1 for more details. The main idea is to expand the radius around the true class c until the set has at least K classes.

Note that $validRadiusSet$ depends on which classes are in the label space to be predicted (i.e., depending on whether we consider *2-hop*, *3-hop*, or *All*. We obtain the label sets for *2-hop* and *3-hop* from the authors of [13, 33]. We implement Algorithm 1 to derive $hCorrectSet$ ourselves.

Algorithm 1 Algorithm for computing $hCorrectSet$ for $H@K$ [13]

Input: K, class c, ImageNet hierarchy
$hCorrectSet \leftarrow \emptyset$
$R \leftarrow 0$
while NumberElements($hCorrectSet$) $< K$ **do**
  $radiusSet \leftarrow$ all nodes in the hierarchy which are R hops from c
  $validRadiusSet \leftarrow$ ValidLabelNodes($radiusSet$)
  $hCorrectSet \leftarrow hCorrectSet \cup validRadiusSet$
  $R \leftarrow R + 1$
end while
return $hCorrectSet$

D. Implementation details

D.1. How to avoid over-fitting?

Since during training we have access only to data from the seen classes, it is important to avoid over-fitting to those seen classes. We apply the *class-wise* cross-validation strategy (Section A), and restrict the semantic embeddings of phantom classes to be close to the semantic embeddings of seen classes (Section 3.2 of the main text).

D.2. Combination of attributes and word vectors

In Table 5 of the main text and Section E.2 of this material, we combine attributes and word vectors to improve the semantic embeddings. We do so by first computing s_{rc} in eq. (2) of the main text for different semantic sources, and then perform convex combination on s_{rc} of different sources to obtain the final s_{rc}. The combining weights are determined via cross-validation.

D.3. Initialization

All variables are randomly initialized, unless stated otherwise. Other details on initialization can be found in Section 3.2 of the main text and Section B and D.5 of this material.

D.4. ConSE [33]

Instead of using the CNN 1K-class classifiers directly, we train (regularized) logistic regression classifiers using recently released multi-core version of LIBLINEAR [11]. Furthermore, in [33], the authors use the averaged word vectors for seen classes, but keep for each unseen class the word vectors of all synonyms. In other words, each unseen class can be represented by multiple word vectors. In our implementation, we use averaged word vectors for both seen and unseen classes for fair comparison.

D.5. Varying the number of base classifiers

In Section 4.2.3 and Figure 2 of the main text, we examine the use of different numbers of base classifiers

Table 7: Comparison between sample- and class-wise cross-validation for hyper-parameter tuning on **CUB** (learning with the one-versus-other loss).

CV Scenarios	**CUB** (AlexNet)	**CUB** (GoogLeNet)
Sample-wise	44.7	52.0
Class-wise	46.6	53.4

(i.e., R). The semantic embedding $\boldsymbol{b}_r$ of the phantom classes are set equal to $\boldsymbol{a}_r, \forall r \in \{1, \cdots, \mathsf{R}\}$ at 100% (i.e., $\mathsf{R} = \mathsf{S}$). For percentages smaller than 100%, we perform K-means and set $\boldsymbol{b}_r$ to be the cluster centroids after ℓ_2 normalization (in this case, $\mathsf{R} = K$). For percentages larger than 100%, we set the first S $\boldsymbol{b}_r$ to be $\boldsymbol{a}_r$, and the remaining $\boldsymbol{b}_r$ as the random combinations of $\boldsymbol{a}_r$ (also with ℓ_2 normalization on $\boldsymbol{b}_r$).

E. Additional experimental results and analyses

We present in this section some additional experimental results on zero-shot learning. *Unless stated otherwise, we focus on learning with the one-versus-other loss (cf. eq. (5) of the main text).*

E.1. Cross-validation (CV) strategies

Table 7 shows the results on **CUB** (averaged over four splits) using the hyper-parameters tuned by class-wise CV and sample-wise CV, respectively. The results based on class-wise CV are about 2% better than those of sample-wise CV, verifying the necessity of simulating the zero-shot learning scenario while we tune the hyper-parameters at the training stage.

E.2. Additional comparison of different semantic spaces for embedding classes

Our method for synthesizing classifiers accepts different semantic embedding spaces. We expand our results on Table 5 of the main text to include AlexNet features as well. The results are in Table 8. We use word vectors provided by Fu et al. [14, 15], which are of 100 and 1000 dimensions per class, respectively. We see that the two types of features, AlexNet and GoogLeNet, demonstrate very similar trends. First, higher-dimensional word vectors often give rise to better performance. Second, human-annotated attributes outperform automatically-learned word vectors. Third, combining the word vectors and the attributes leads to better results than separately using either one of them.

E.3. Comparison to other state-of-the-art methods

In Table 9, we contrast our methods to several other state-of-the-art methods, in addition to Table 2 of the main text. We note subtle differences in the experiment setup of some of these methods from ours:

- TMV-BLP and TMV-HLP [14, 15]. These methods focus on the transductive setting, where they have access to unlabeled test data from unseen classes during the training stage. Additionally, they use OverFeat [42] features for **CUB**, OverFeat+DeCAF [9] for **AwA**, and both attributes and word vectors for class embeddings.

- [22]. This method works on the transductive setting. It uses OverFeat features for both **AwA** and **CUB**, and combines attributes and word vectors for class embeddings.

- [27] This method works on the semi-supervised setting, where a portion of unlabeled data (not used for testing) from unseen classes are available at training.

- HAT-n [3]. This method uses extra semantic information (WordNet class hiearchy). It uses CNN-M2K features [6] and extra cropped images.

- AMP (SR+SE) [16]. This method uses attributes and (100-dimensional) word vectors and OverFeat features in their experiments.

- [49]. This method focuses on mining/discovering new (category-level) attributes. It requires extra human efforts to annotate the new attributes for unseen classes.

- [19]. The best result on **AwA** presented in this paper uses the discovered attributes in [49].

As shown in Table 9, our method outperforms all of them on the dataset **CUB** despite the fact that they employ extra images or semantic embedding information.

E.3.1 SUN-10

Some existing work [19, 39, 50, 51] considers another setting for **SUN** dataset — with 707 seen classes and 10 unseen classes. Moreover, [39, 50, 51] use the VGG-verydeep-19 [43] CNN features. In Table 9, we provide results of our approach based on this splitting. Compared to previously published results, our method again clearly shows superior performance.

Table 8: Comparison between different semantic embedding spaces for our approach.

Methods	Semantic embeddings	Dimensions	Features	**AwA**
Ours$^{\text{o-vs-o}}$	word vectors	100	AlexNet	37.6
Ours$^{\text{o-vs-o}}$	word vectors	1000	AlexNet	52.4
Ours$^{\text{o-vs-o}}$	attributes	85	AlexNet	64.0
Ours$^{\text{o-vs-o}}$	attributes + word vectors	85 + 100	AlexNet	65.6
Ours$^{\text{o-vs-o}}$	attributes + word vectors	85 + 1000	AlexNet	**68.0**
SJE[2]	word vectors	400	GoogLeNet	51.2
Ours$^{\text{o-vs-o}}$	word vectors	100	GoogLeNet	42.2
Ours$^{\text{o-vs-o}}$	word vectors	1000	GoogLeNet	57.5
Ours$^{\text{o-vs-o}}$	attributes	85	GoogLeNet	69.7
Ours$^{\text{o-vs-o}}$	attributes + word vectors	85 + 100	GoogLeNet	73.2
Ours$^{\text{o-vs-o}}$	attributes + word vectors	85 + 1000	GoogLeNet	**76.3**

Table 9: Comparison between our method and other state-of-the-art methods for zero-shot learning. Our results are based on the GoogLeNet features and attributes or attributes + word vectors as semantic embeddings. See Section E.3 for the details of other methods and **SUN-10**.

Methods	Shallow features			Deep features			
	AwA	**CUB**	**SUN-10**	**AwA**	**CUB**	**SUN**	**SUN-10**
TMV-BLP [14]	***47.7***	***16.3***†‡	-	***77.8***	***45.2***†	-	-
TMV-HLP [15]	***49.0***	***19.5***†‡	-	***80.5***	***47.9***†	-	-
[22]	***49.7***	***28.1***†‡	-	***75.6***	***40.6***†	-	-
[27]	***40.0***	-	-	-	-	-	-
HAT-n [3]	-	-	-	***68.8***	***48.6***†	-	-
AMP (SR+SE) [16]	-	-	-	***66.0***	-	-	-
[49]	***48.3***	-	-	-	-	-	-
[19]	***48.7***	-	56.2	-	-	-	-
ESZSL [39]	-	-	65.8	-	-	-	***82.1***
SSE-ReLU [51]	-	-	-	-	-	-	***82.5***
[50]	-	-	-	-	-	-	***83.8***
SJE[2]	-	-	-	-	-	-	87.0
Ours$^{\text{o-vs-o}}$	42.6	35.0	-	69.7	53.4	62.8	90.0
Ours^{cs}	42.1	34.7	-	68.4	51.6	52.9	87.0
Ours^{struct}	41.5	36.4	-	72.9	54.5	62.7	85.0
§Ours$^{\text{o-vs-o}}$	-	-	-	76.3	-	-	-

†: Results reported by the authors on a particular seen-unseen split.
‡: Based on Fisher vectors as shallow features.
§: Ours with attributes + (1000-dimensional) word vectors.

E.4. Discussion on the numbers of seen and unseen classes

In this subsection, we analyze the results under different numbers of seen/unseen classes in performing zero-shot learning using the **CUB** dataset.

E.4.1 Varying the number of seen classes

We first examine the performance of zero-shot learning under different numbers of seen classes (e.g., 50, 100, and 150) while fixing the number of unseen classes to be 50. We perform 20 random selections of seen/unseen classes. Unsurprisingly, Table 10 shows that increasing the number of seen classes in training leads to improved performance on zero-shot learning.

Table 10: Performance of our method under different number S of seen classes on **CUB**. The number of unseen classes U is fixed to be 50.

U and S	S = 50	S = 100	S = 150
U = 50	38.4	49.9	53.8

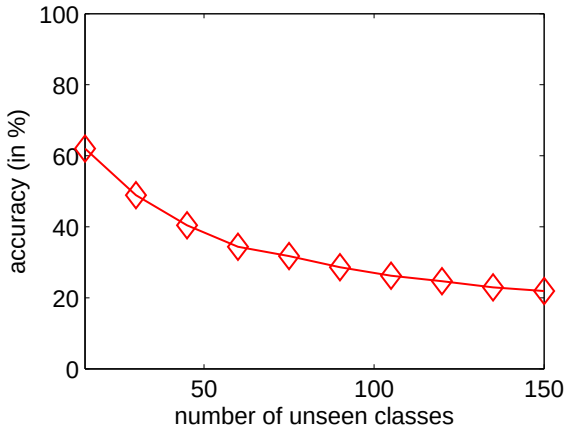

Figure 4: Performance of our method under different numbers of unseen classes on **CUB**. The number of seen classes is fixed to be 50.

Table 11: Effect of learning metrics for computing semantic similarity on **AwA**.

Dataset	Type of embeddings	w/o learning	w/ learning
AwA	attributes	69.7%	73.4%

E.4.2 Varying the number of unseen classes

We then examine the performance of our approach to zero-shot learning under different numbers of unseen classes (e.g., within [0, 150]), with the number of seen classes fixed to be 50 during training. We again conduct experiments on **CUB**, and perform 20 random selections of seen/unseen classes. The results are presented in Figure 4. We see that the accuracy drops as the number of unseen classes increases.

E.5. Results on learning metrics for computing semantic similarities

We improve our method by also learning metrics for computing semantic similarity. Please see Section B for more details. Preliminary results on **AwA** in Table 11 suggest that learning metrics can further improve upon our current one-vs-other formulation.

E.6. Detailed results and analysis of experiments on ImageNet

Table 12 provides expanded zero-shot learning results on **ImageNet** (cf. Table 3 of the main text). Note that ConSE [33] has a free parameter T, corresponding to how many nearest seen classes to use for convex combination. In our implementation, we follow the paper to test on $T = 1$, 10, and 1,000. We further apply the class-wise cross-validation (cf. Section A of this material) to automatically set T. We also report the published best result in [33]. Our methods (Ours$^{\text{o-vs-o}}$ and Ours^{struct}) achieve the highest accuracy in most cases.

As mentioned in the main text, the three sets of unseen classes, *2-hop*, *3-hop*, and *All* are built according to the ImageNet label hierarchy. Note that they are not mutually exclusive. Indeed, *3-hop* contains all the classes in *2-hop*, and *All* contains all in *3-hop*. To examine if the semantic similarity/dissimilarity to the 1K seen classes (according to the label hierarchy) would affect the classification accuracy, we split *All* into three *disjoint* sets, *2-hop*, *pure 3-hop*, and *others*, which contain 1,509, 6,169, and 12,667 classes, respectively (totally 20,345). We then test on *All*, but report accuracies of images belonging to different *disjoint* sets separately. Figure 5 summarizes the results. Our method outperforms ConSE in almost all cases. The decreasing accuracies from *2-hop*, *pure 3-hop*, to *others* (by both methods) verify the high correlation of the semantic relationship to the classification accuracy in zero-shot learning. This observation suggests an obvious potential limitation: it is unrealistic to expect good performance on unseen classes that are semantically too dissimilar to seen classes.

E.7. Qualitative results

In this subsection, we present qualitative results of our method. We first illustrate what visual information the models (classifiers) for unseen classes capture, when provided with only semantic embeddings (no example images). In Figure 6, we list (on top) the 10 unseen class labels of **AwA**, and show (in the middle) the top-5 images classified into each class c, according to the decision values $\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{x}$ (cf. eq. (1) and (4) of the main text). Misclassified images are marked with red boundaries. At the bottom, we show the first (highest score) misclassified image (according to the decision value) into each class and its ground-truth class label. According to the top images, our method reasonably captures discriminative visual properties of each unseen class based solely on its semantic embedding. We can also see that the misclassified images are with appearance so similar to that of predicted class that even humans cannot easily distin-

Table 12: Flat Hit@K and Hierarchial precision@K performance (in %) on the task of zero-shot learning on **ImageNet**. We mainly compare it with ConSE(T) [33], where T is the number of classifiers to be combined in their paper. For ConSE(CV), T is obtained by class-wise CV. Lastly, the best published results in [33] are also reported, corresponding to ConSE(10) [33]. For both types of metrics, the higher the better.

Scenarios	Methods	Flat Hit@K					Hierarchical precision@K			
	K=	1	2	5	10	20	2	5	10	20
2-hop	ConSE(1)	9.0	12.9	20.8	28.3	38.1	21.1	22.2	24.8	28.1
	ConSE(10)	9.2	13.7	22.4	31.0	41.4	22.5	24.1	27.3	30.8
	ConSE(10) [33]	9.4	15.1	24.7	32.7	41.8	21.4	24.7	26.9	28.4
	ConSE(1000)	8.9	13.3	21.8	30.1	40.3	22.0	23.7	27.0	30.4
	ConSE(CV)	8.3	12.9	21.8	30.9	41.7	21.5	23.8	27.5	31.3
	$\text{Ours}^{\text{o-vs-o}}$	**10.5**	**16.7**	**28.6**	**40.1**	**52.0**	**25.1**	**27.7**	**30.3**	**32.1**
	$\text{Ours}^{\text{struct}}$	9.8	15.3	25.8	35.8	46.5	23.8	25.8	28.2	29.6
3-hop	ConSE(1)	2.8	4.2	7.2	10.1	14.3	6.2	18.4	20.4	22.1
	ConSE(10)	**2.9**	4.5	7.7	11.3	16.1	6.9	20.9	23.1	25.2
	ConSE(10) [33]	2.7	4.4	7.8	11.5	16.1	5.3	20.2	22.4	24.7
	ConSE(1000)	2.8	4.3	7.4	10.9	15.6	6.8	20.7	22.9	25.1
	ConSE(CV)	2.6	4.1	7.3	11.1	16.4	6.7	21.4	23.8	26.3
	$\text{Ours}^{\text{o-vs-o}}$	**2.9**	**4.9**	**9.2**	**14.2**	**20.9**	7.4	**23.7**	**26.4**	**28.6**
	$\text{Ours}^{\text{struct}}$	**2.9**	4.7	8.7	13.0	18.6	**8.0**	22.8	25.0	26.7
All	ConSE(1)	1.4	2.3	3.8	5.6	7.8	3.0	7.6	8.7	9.6
	ConSE(10)	**1.5**	2.3	4.0	6.0	8.7	3.3	8.9	10.2	11.4
	ConSE(10) [33]	1.4	2.2	3.9	5.8	8.3	2.5	7.8	9.2	10.4
	ConSE(1000)	**1.5**	2.3	3.9	5.8	8.4	3.2	8.8	10.2	11.3
	ConSE(CV)	1.3	2.1	3.8	5.8	8.7	3.2	9.2	10.7	12.0
	$\text{Ours}^{\text{o-vs-o}}$	1.4	**2.4**	**4.5**	**7.1**	**10.9**	3.1	9.0	10.9	**12.5**
	$\text{Ours}^{\text{struct}}$	**1.5**	**2.4**	4.4	6.7	10.0	**3.6**	**9.6**	**11.0**	12.2

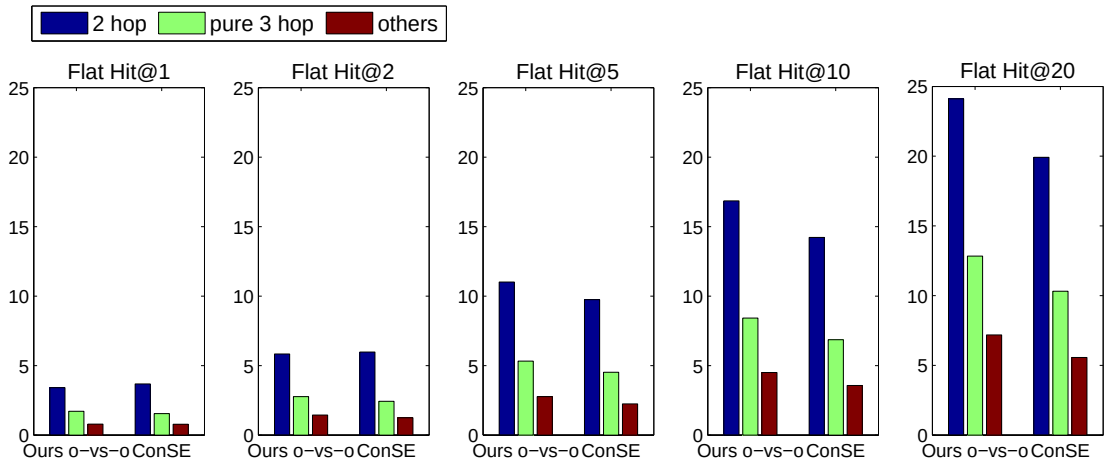

Figure 5: On the **ImageNet** dataset, we outperform ConSE (i.e., ConSE(10) of our implementation) on different *disjoint* sets of categories in the scenario *All* in almost all cases. See Section E.6 of this material for details.

guish between the two. For example, the pig image at the bottom of the second column looks very similar to the image of hippos. Figure 7 and Figure 8 present the results in the same format on **CUB** and **SUN**, respectively (both on a subset of unseen class labels).

We further analyze the success and failure cases; i.e., why an image from unseen classes is misclassified. The illustrations are in Figure 9, 10, and 11 for **AwA**, **CUB**, and **SUN**, respectively. In each figure, we consider **(Left)** one unseen class and show its convex combination weights $\boldsymbol{s}_c = \{s_{c1}, \cdots, s_{c\mathsf{R}}\}$ as a histogram. We then present **(Middle-Left)** the top-3 semantically simi-

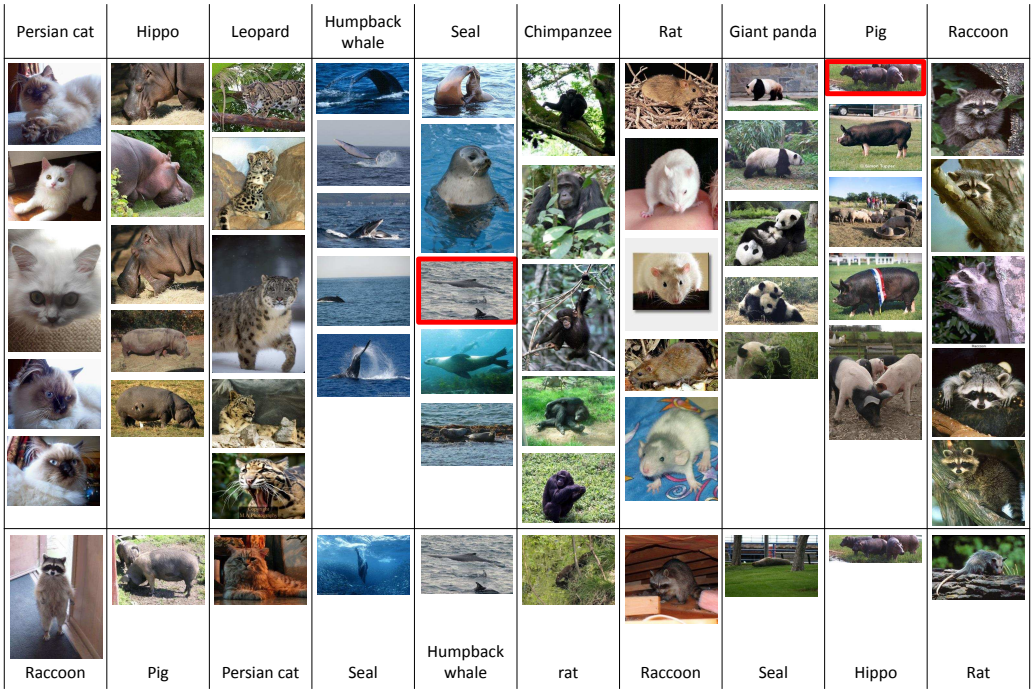

Figure 6: Qualitative results of our method (Ours^{struct}) on **AwA**. **(Top)** We list the 10 unseen class labels. **(Middle)** We show the top-5 images classified into each class, according to the decision values. *Misclassified images are marked with red boundaries*. **(Bottom)** We show the first misclassified image (according to the decision value) into each class and its ground-truth class label.

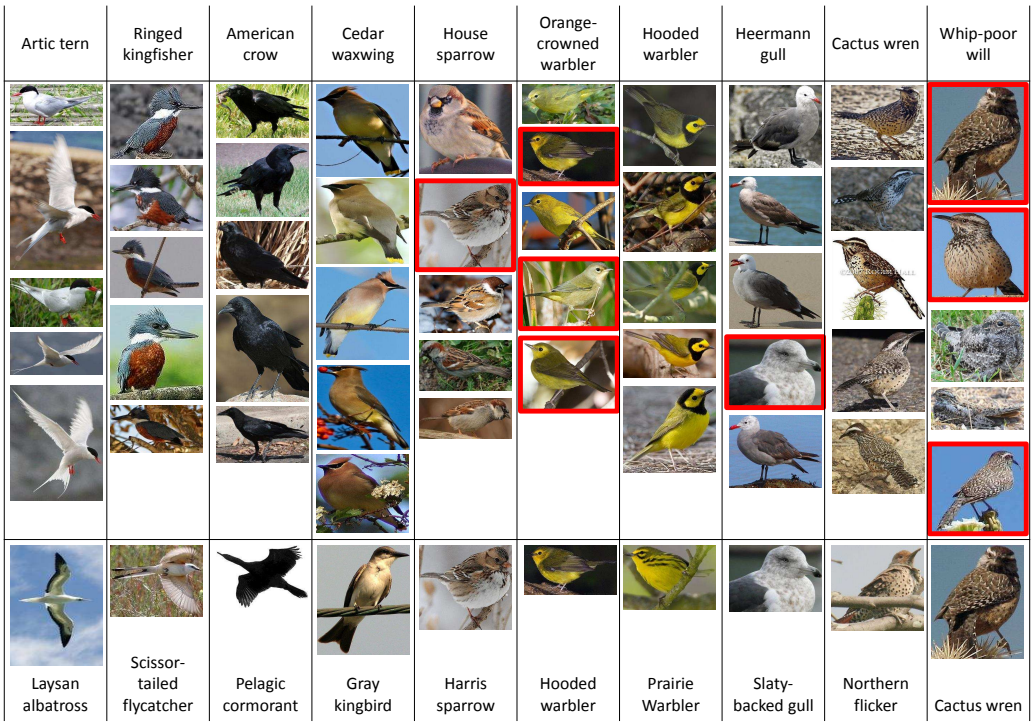

Figure 7: Qualitative results of our method (Ours^{struct}) on **CUB**. **(Top)** We list a subset of unseen class labels. **(Middle)** We show the top-5 images classified into each class, according to the decision values. *Misclassified images are marked with red boundaries*. **(Bottom)** We show the first misclassified image (according to the decision value) into each class and its ground-truth class label.

Figure 8: Qualitative results of our method (Ours$^{\text{o-vs-o}}$) on **SUN**. **(Top)** We list a subset of unseen class labels. **(Middle)** We show the top-5 images classified into each class, according to the decision values. *Misclassified images are marked with red boundaries*. **(Bottom)** We show the first misclassified image (according to the decision value) into each class and its ground-truth class label.

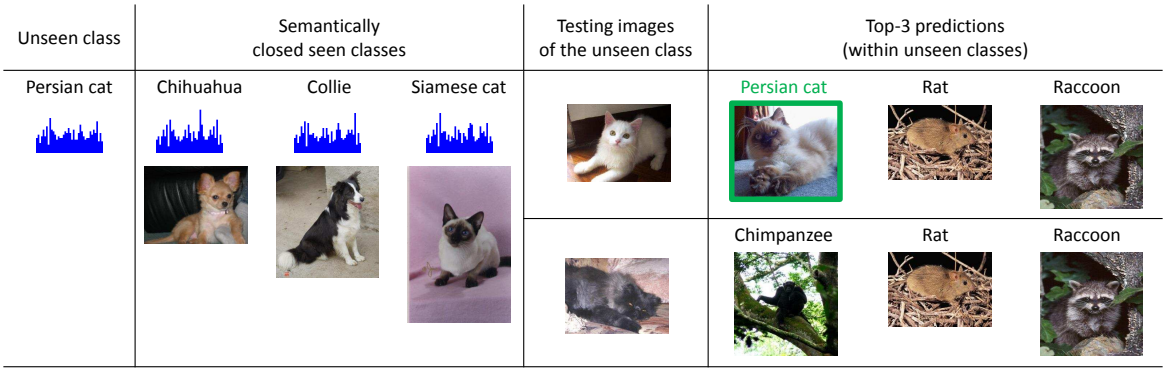

Figure 9: Success/failure case analysis of our method (Ours^{struct}) on **AwA**: (Left) an unseen class label, (Middle-Left) the top-3 semantically similar seen classes to that unseen class, (Middle-Right) two test images of such unseen class, and (Right) the top-3 predicted unseen classes. The green text corresponds to the correct label.

lar (in terms of $\boldsymbol{s}_c$) seen classes and their most representative images. As our model exploits phantom classes to connect seen and unseen classes in both semantic and model spaces, we expect that the model (classifier) for such unseen class captures similar visual information as those for semantically similar seen classes do. **(Middle-Right)** We examine two images of such unseen class, where the top one is correctly classified; the bottom one, misclassified. We also list **(Right)** the top-3 predicted labels (within the pool of unseen classes) and their most representative images. Green corresponds to correct labels. We see that, in the misclassified cases, the test images are visually dissimilar to those of the semantically similar seen classes. The synthesized unseen classifiers, therefore, cannot correctly recognize them, leading to incorrect predictions.

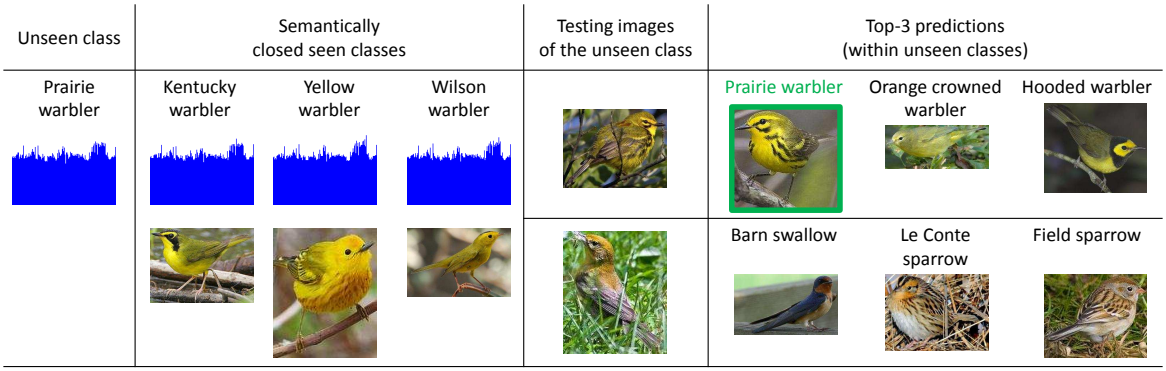

Figure 10: Success/failure case analysis of our method (Ours^{struct}) on **CUB**. (Left) an unseen class label, (Middle-Left) the top-3 semantically similar seen classes to that unseen class, (Middle-Right) two test images of such unseen classes, and (Right) the top-3 predicted unseen class. The green text corresponds to the correct label.

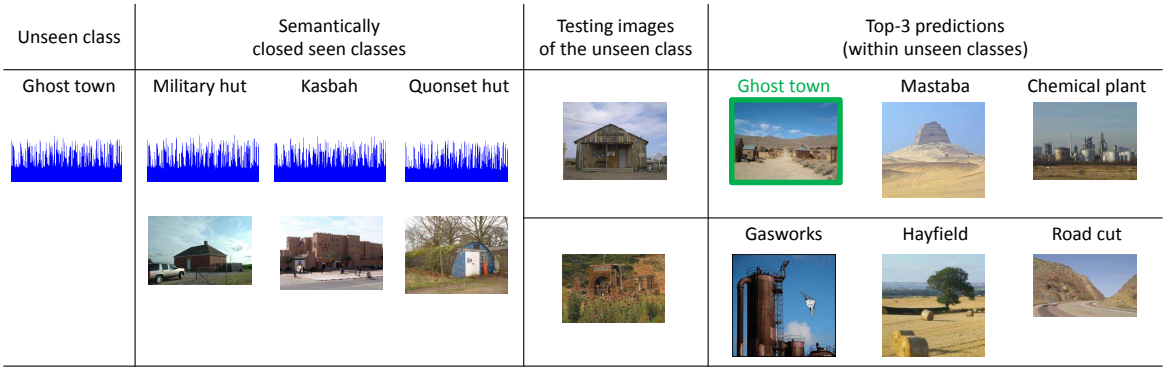

Figure 11: Success/failure case analysis of our method (Ours$^{\text{o-vs-o}}$) on **SUN**. (Left) an unseen class label, (Middle-Left) the top-3 semantically similar seen classes to that unseen class, (Middle-Right) two test images of such unseen classes, and (Right) the top-3 predicted unseen class. The green text corresponds to the correct label.

E.8. Comparison between shallow and deep features of our approach

In Table 4 of the main text, our approach performs better with deep features than with shallow features compared to other methods. We propose explanations for this phenomenal. Deep features are learned hierarchically and expected to be more abstract and semantically meaningful. Arguably, similarities between them (measured in inner products between classifiers) might be more congruent with similarities computed in the semantic embedding space for combining classifiers. Additionally, shallow features have higher dimensions (around 10,000) than deep features (e.g., 1024 for GoogLeNet) so they might require more phantom classes to synthesize classifiers.

E.9. Analysis on the number of base classifiers

In Fig. 2 of the main text, we show that even by using fewer base (phantom) classifiers than the number of seen classes (e.g., around 60 %), we get comparable or even better results, especially for **CUB**. We surmise that this is because **CUB** is a fine-grained recognition benchmark and has higher correlations among classes, and provide analysis in Fig. 12 to justify this.

We train one-versus-other classifiers for each value of the regularization parameter (i.e., λ in eq. (5) of the main text) on both **AwA** and **CUB**, and then perform PCA on the resulting classifier matrices. We then plot the required number (in percentage) of PCA components to capture 95% of variance in the classifiers. Clearly, **AwA** requires more. This explains why we see the drop in accuracy for **AwA** but not **CUB** in Fig. 2 of the main text when using even fewer base classifiers. Particularly, the low percentage for **CUB** in Fig. 12 implies that fewer

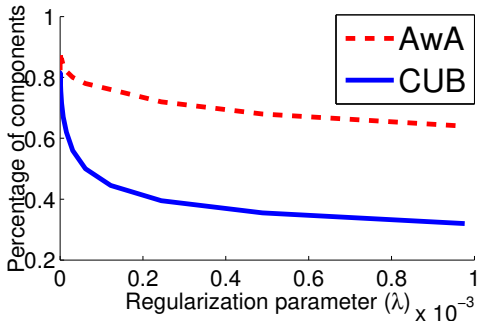

Figure 12: Percentages of basis components required to capture 95% of variance in classifier matrices for **AwA** and **CUB**.

base classifiers are possible.