Supplementary Material: Synthesized Classifiers for Zero-Shot Learning

Soravit Changpinyo*, Wei-Lun Chao*, Boqing Gong
U. of Southern California U. of Central Florida
Los Angeles, CA Orlando, FL
schangpi, weilunc@usc.edu bgong@crcv.ucf.edu
Fei Sha
U. of California
Los Angeles, CA
feisha@cs.ucla.edu

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

• Section 1: cross-validation strategies (Section 3.2 of the main paper).
• Section 2: learning metrics for semantic similarity (Section 3.1 of the main paper).
• Section 3: details on experimental setup (Section 4.1 of the main paper).
• Section 4: implementation details (Section 4.1 and 4.2.3 of the main paper).
• Section 5: additional experimental results and analyses (Section 4.2 of the main paper).

1. 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 [25]. 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 1(b) and 1(c) illustrate the two scenarios, respectively. We empirically compare them in Section 5.1. Note that several existing models [2, 7, 25, 34] also follow similar hyper-parameter tuning procedures.

1.1. 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 $\lambda, \sigma, \eta,$ and $\gamma$ to be tuned. To reduce the search space during cross-validation, we first fix $b_r = a_r$ for $r = 1, \ldots, R$ and tune $\lambda, \sigma$. Then we fix $\lambda$ and $\sigma$ and tune $\eta$ and $\gamma$.

2. 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 $d(a_c, b_r) = (a_c - b_r)^T \Sigma^{-1} (a_c - b_r)$. In this section, we describe an objective for learning a more general Mahalanobis metric than $\Sigma^{-1} = \sigma^2 I$. We focus on the case when $R = S$ and on learning a diagonal metric $\Sigma^{-1} = M^T M$, where $M$ is also diagonal. We solve the following optimization problem.

$$\begin{align*}
\min_{M, v_1, \ldots, v_R} & \sum_{c=1}^{S} \sum_{n=1}^{N} \ell(x_n, y_{n,c}; w_c) \\
& + \lambda \frac{1}{2} \sum_{r=1}^{R} \|v_r\|^2 + \gamma \frac{1}{2} \|M - \sigma I\|_F^2,
\end{align*}$$

subject to $w_c = \sum_{r=1}^{R} s_{cr} v_r, \quad \forall c \in T = \{1, \ldots, S\}$

where $\ell(x, y; w) = \max(0, 1 - yw^T x)^2$ 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 $M = \sigma I$ and optimize $\{v_1, \ldots, v_R\}$ to obtain a reasonable initialization. Then we perform alternating optimization. To further prevent over-fitting, we alternately
optimize $M$ and $\{v_1, \cdots, v_R\}$ on different but overlapping subsets of training data. In particular, we split data into 5 folds and optimize $\{v_1, \cdots, v_R\}$ on the first 4 folds and $M$ on the last 4 folds. We report results in Section 5.5.

### 3. Details on the experimental setup

We present details on the experimental setup in this section and additional results in Section 5.

#### 3.1. Datasets

We use four benchmark datasets in our experiments. The **Animals with Attributes (AwA)** dataset [18] 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)** [30]. 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 [23] contains 14,340 images of 717 scene categories (20 images from each category). Following [18], 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 [13, 25, 33, 34] 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 3.

For the large-scale zero-shot experiment on the **ImageNet** dataset [5], we follow the setting in [9, 22]. The ILSVRC 2012 1K dataset [26] 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 [5] that do not overlap with any of the 1,000 categories. We will call this release the ImageNet 2011 21K dataset (as in [9, 22]). Overall, this dataset contains 14,197,122 images from 21,841 classes, and we conduct our experiment on 20,842 unseen classes.

#### 3.2. 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 [20, 21] on the latest Wikipedia dump corpus (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 — 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\footnote{Each class of ImageNet is a synset: a set of synonymous terms, where each term is a word or phrase.}. We impose no restriction on the vocabulary size. Following [9], 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.

3.3. 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 [18, 24, 31], while others use the Fisher vectors [1, 2]. The SUN dataset also comes with several traditional shallow features, which are used in [13, 18, 25].

In our experiments, we use the shallow features provided by [18, 14, and 23] for AwA, CUB, and SUN, respectively, unless stated otherwise.

Deep features Given the recent impressive success of deep Convolutional Neural Networks (CNNs) [17] on image classification, we conduct experiments with deep features on all datasets. We use the Caffe package [15] to extract AlexNet [17] and GoogLeNet [29] 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 [5, 26] for AwA, CUB, and ImageNet. For SUN, the networks are pre-trained on the Places database [35], 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.

3.4. Evaluation protocols on ImageNet

When computing Hierarchical precision@K (HP@K), we use the algorithm in the Appendix of [9] 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.

Algorithm 1 Algorithm for computing \( hCorrectSet \) for \( H@K \) [9]

\begin{verbatim}
1: Input: \( K \), class \( c \), ImageNet hierarchy
2: \( hCorrectSet \leftarrow \emptyset \)
3: \( R \leftarrow 0 \)
4: \( \text{while } \) \( \text{NumberElements}(hCorrectSet) < K \) \( \text{do} \)
5: \( \quad \text{radiusSet} \leftarrow \text{all nodes in the hierarchy which are} \ R \ \text{hops from } c \)
6: \( \quad \text{validRadiusSet} \leftarrow \text{ValidLabelNodes}(\text{radiusSet}) \)
7: \( \quad hCorrectSet \leftarrow hCorrectSet \cup \text{validRadiusSet} \)
8: \( \quad R \leftarrow R + 1 \)
9: \( \text{end while} \)
10: \( \text{return } hCorrectSet \)
\end{verbatim}

Note that \( \text{validRadiusSet} \) 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 [9, 22]. We implement Algorithm 1 to derive \( hCorrectSet \) ourselves.

4. Implementation details

4.1. 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 1), 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).

4.2. Combination of attributes and word vectors

In Table 5 of the main text and Section 5.2 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.
4.3. 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 2 and 4.5 of this material.

4.4. ConSE [22]

Instead of using the CNN 1K-class classifiers directly, we train (regularized) logistic regression classifiers using recently released multi-core version of LIBLINEAR [8]. Furthermore, in [22], 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.

4.5. 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 (i.e., $R$). The semantic embedding $b_r$ of the phantom classes are set equal to $a_r$, $\forall r \in \{1, \cdots, R\}$ at 100% (i.e., $R = S$). For percentages smaller than 100%, we perform $K$-means and set $b_r$ to be the cluster centroids after $\ell_2$ normalization (in this case, $R = K$). For percentages larger than 100%, we set the first $S b_r$ to be $a_r$, and the remaining $b_r$ as the random combinations of $a_r$ (also with $\ell_2$ normalization on $b_r$).

5. 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).

5.1. Cross-validation (CV) strategies

Table 1 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.

<table>
<thead>
<tr>
<th>CV Scenarios</th>
<th>CUB (AlexNet)</th>
<th>CUB (GoogLeNet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample-wise</td>
<td>44.7</td>
<td>52.0</td>
</tr>
<tr>
<td>Class-wise</td>
<td>46.6</td>
<td>53.4</td>
</tr>
</tbody>
</table>

5.2. 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 2. We use word vectors provided by Fu et al. [10, 11], 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.

5.3. Comparison to other state-of-the-art methods

In Table 3, 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 [10, 11]. 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 [27] features for CUB, OverFeat+DeCAF [6] for AwA, and both attributes and word vectors for class embeddings.
- [16]. 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.
- [19] 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 hierarchy). It uses CNN-M2K features [4] and extra cropped images.
- AMP (SR+SE) [12]. This method uses attributes.
Table 2: Comparison between different semantic embedding spaces for our approach.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Semantic embeddings</th>
<th>Dimensions</th>
<th>Features</th>
<th>AwA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours o-vs-o</td>
<td>word vectors</td>
<td>100</td>
<td>AlexNet</td>
<td>37.6</td>
</tr>
<tr>
<td>Ours o-vs-o</td>
<td>word vectors</td>
<td>1000</td>
<td>AlexNet</td>
<td>52.4</td>
</tr>
<tr>
<td>Ours o-vs-o</td>
<td>attributes</td>
<td>85</td>
<td>AlexNet</td>
<td>64.0</td>
</tr>
<tr>
<td>Ours o-vs-o</td>
<td>attributes + word vectors</td>
<td>85 + 100</td>
<td>AlexNet</td>
<td>65.6</td>
</tr>
<tr>
<td>Ours</td>
<td>attributes + word vectors</td>
<td>85 + 1000</td>
<td>AlexNet</td>
<td>68.0</td>
</tr>
<tr>
<td>SJE[2]</td>
<td>word vectors</td>
<td>400</td>
<td>GoogLeNet</td>
<td>51.2</td>
</tr>
<tr>
<td>Ours o-vs-o</td>
<td>word vectors</td>
<td>100</td>
<td>GoogLeNet</td>
<td>42.2</td>
</tr>
<tr>
<td>Ours o-vs-o</td>
<td>word vectors</td>
<td>1000</td>
<td>GoogLeNet</td>
<td>57.5</td>
</tr>
<tr>
<td>Ours o-vs-o</td>
<td>attributes</td>
<td>85</td>
<td>GoogLeNet</td>
<td>69.7</td>
</tr>
<tr>
<td>Ours o-vs-o</td>
<td>attributes + word vectors</td>
<td>85 + 100</td>
<td>GoogLeNet</td>
<td>73.2</td>
</tr>
<tr>
<td>Ours</td>
<td>attributes + word vectors</td>
<td>85 + 1000</td>
<td>GoogLeNet</td>
<td>76.3</td>
</tr>
</tbody>
</table>

and (100-dimensional) word vectors and OverFeat features in their experiments.

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

- [13]. The best result on AwA presented in this paper uses the discovered attributes in [32].

As shown in Table 3, our method outperforms all of them on the dataset CUB despite the fact that they employ extra images or semantic embedding information.

5.3.1 SUN-10

Some existing work [13, 25, 33, 34] considers another setting for SUN dataset — with 707 seen classes and 10 unseen classes. Moreover, [25, 33, 34] use the VGG-verydeep-19 [28] CNN features. In Table 3, we provide results of our approach based on this splitting. Compared to previously published results, our method again clearly shows superior performance.

5.4. 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.

5.4.1 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 4 shows that increasing the number of seen classes in training leads to improved performance on zero-shot learning.

5.4.2 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 2. We see that the accuracy drops as the number of unseen classes increases.

5.5. Results on learning metrics for computing semantic similarities

We improve our method by also learning metrics for computing semantic similarity. Please see Section 2 for more details. Preliminary results on AwA in Table 5 suggest that learning metrics can further improve upon our current one-vs-other formulation.
Table 3: 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 5.3 for the details of other methods and SUN-10.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Shallow features</th>
<th>Deep features</th>
<th>AwA</th>
<th>CUB</th>
<th>SUN-10</th>
<th>AwA</th>
<th>CUB</th>
<th>SUN</th>
<th>SUN-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMV-BLP [10]</td>
<td>47.7</td>
<td>16.3†‡</td>
<td>77.8</td>
<td>45.2</td>
<td>-</td>
<td>77.8</td>
<td>45.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TMV-HLP [11]</td>
<td>49.0</td>
<td>19.5†‡</td>
<td>80.5</td>
<td>47.9</td>
<td>-</td>
<td>80.5</td>
<td>47.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[16]</td>
<td>49.7</td>
<td>28.1†‡</td>
<td>75.6</td>
<td>40.6</td>
<td>-</td>
<td>75.6</td>
<td>40.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[19]</td>
<td>40.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HAT-n [3]</td>
<td>-</td>
<td>-</td>
<td>68.8</td>
<td>48.6</td>
<td>-</td>
<td>68.8</td>
<td>48.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AMP (SR+SE) [12]</td>
<td>-</td>
<td>-</td>
<td>66.0</td>
<td>-</td>
<td>-</td>
<td>66.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[32]</td>
<td>48.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[13]</td>
<td>48.7</td>
<td>56.2</td>
<td>-</td>
<td>-</td>
<td>82.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>82.1</td>
</tr>
<tr>
<td>ESZSL [25]</td>
<td>-</td>
<td>65.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SSE-ReLU [34]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>82.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>82.5</td>
</tr>
<tr>
<td>SJE [2]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours o-vs-o</td>
<td>42.6</td>
<td>35.0</td>
<td>69.7</td>
<td>53.4</td>
<td>62.8</td>
<td>90.0</td>
<td>87.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours CS</td>
<td>42.1</td>
<td>34.7</td>
<td>68.4</td>
<td>51.6</td>
<td>52.9</td>
<td>87.0</td>
<td>87.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours struct</td>
<td>41.5</td>
<td>36.4</td>
<td>72.9</td>
<td>54.5</td>
<td>62.7</td>
<td>85.0</td>
<td>85.0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

†: 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.

5.6. Detailed results and analysis of experiments on ImageNet

Table 6 provides expanded zero-shot learning results on ImageNet (cf. Table 3 of the main text). Note that ConSE [22] 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 $T = 1, 10, 1,000$. We further apply the class-wise cross-validation (cf. Section 1 of this material) to automatically set $T$. We also report the published best result in [22]. Our methods (Ours 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 3 sum-

![Figure 2: Performance of our method under different numbers of unseen classes on CUB. The number of seen classes is fixed to be 50.](image-url)
marizes 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.

5.7. 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 4, 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 \( w_c^T x \) (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 distinguish between the two. For example, the pig image at the bottom of the second column looks very similar to the image of hippos. Figure 5 and Figure 6 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 7, 8, and 9 for AwA, CUB, and SUN, respectively. In each figure, we consider (Left) one unseen class and show its convex combination weights \( s_c = \{s_{c1}, \ldots, s_{cR}\} \) as a histogram. We then present (Middle-Left) the top-3 semantically similar (in terms of \( s_c \)) 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.

5.8. 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.

5.9. 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. 10 to justify this.

We train one-versus-other classifiers for each value of the regularization parameter (i.e., \( \lambda \) 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. 10 implies that fewer base classifiers are possible.

References

Table 6: Flat Hit@K and Hierachial precision@K performance (in %) on the task of zero-shot learning on ImageNet. We mainly compare it with ConSE($T$) [22], 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 [22] are also reported, corresponding to ConSE(10) [22]. For both types of metrics, the higher the better.

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Methods</th>
<th>Flat Hit@K</th>
<th>Hierarchical precision@K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K=1$</td>
<td>2 5 10 20</td>
<td>2 5 10 20</td>
</tr>
<tr>
<td>2-hop</td>
<td>ConSE(1)</td>
<td>9.0 12.9 20.8 28.3 38.1</td>
<td>21.1 22.2 24.8 28.1</td>
</tr>
<tr>
<td></td>
<td>ConSE(10)</td>
<td>9.2 13.7 22.4 31.0 41.4</td>
<td>22.5 24.1 27.3 30.8</td>
</tr>
<tr>
<td></td>
<td>ConSE(10) [22]</td>
<td>9.4 15.1 24.7 32.7 41.8</td>
<td>21.4 24.7 26.9 28.4</td>
</tr>
<tr>
<td></td>
<td>ConSE(1000)</td>
<td>8.9 13.3 21.8 30.1 40.3</td>
<td>22.0 23.7 27.0 30.4</td>
</tr>
<tr>
<td></td>
<td>ConSE(CV)</td>
<td>8.3 12.9 21.8 30.9 41.7</td>
<td>21.5 23.8 27.5 31.3</td>
</tr>
<tr>
<td></td>
<td>Ours o-vs-o</td>
<td><strong>10.5 16.7 28.6 40.1 52.0</strong></td>
<td><strong>25.1 27.7 30.3 32.1</strong></td>
</tr>
<tr>
<td></td>
<td>Ours struct</td>
<td>9.8 15.3 25.8 35.8 46.5</td>
<td>23.8 25.8 28.2 29.6</td>
</tr>
<tr>
<td>3-hop</td>
<td>ConSE(1)</td>
<td>2.8 4.2 7.2 10.1 14.3</td>
<td>6.2 18.4 20.4 22.1</td>
</tr>
<tr>
<td></td>
<td>ConSE(10)</td>
<td><strong>2.9 4.5 7.7 11.3 16.1</strong></td>
<td>6.9 20.9 23.1 25.2</td>
</tr>
<tr>
<td></td>
<td>ConSE(10) [22]</td>
<td>2.7 4.4 7.8 11.5 16.1</td>
<td>5.3 20.2 22.4 24.7</td>
</tr>
<tr>
<td></td>
<td>ConSE(1000)</td>
<td>2.8 4.3 7.4 10.9 15.6</td>
<td>6.8 20.7 22.9 25.1</td>
</tr>
<tr>
<td></td>
<td>ConSE(CV)</td>
<td>2.6 4.1 7.3 11.1 16.4</td>
<td>6.7 21.4 23.8 26.3</td>
</tr>
<tr>
<td></td>
<td>Ours o-vs-o</td>
<td><strong>2.9 4.9 9.2 14.2 20.9</strong></td>
<td>7.4 23.7 26.4 28.6</td>
</tr>
<tr>
<td></td>
<td>Ours struct</td>
<td><strong>2.9 4.7 8.7 13.0 18.6</strong></td>
<td><strong>8.0 22.8 25.0 26.7</strong></td>
</tr>
<tr>
<td>All</td>
<td>ConSE(1)</td>
<td>1.4 2.3 3.8 5.6 7.8</td>
<td>3.0 7.6 8.7 9.6</td>
</tr>
<tr>
<td></td>
<td>ConSE(10)</td>
<td><strong>1.5 2.3 4.0 6.0 8.7</strong></td>
<td><strong>3.3 8.9 10.2 11.4</strong></td>
</tr>
<tr>
<td></td>
<td>ConSE(10) [22]</td>
<td>1.4 2.2 3.9 5.8 8.3</td>
<td>2.5 7.8 9.2 10.4</td>
</tr>
<tr>
<td></td>
<td>ConSE(1000)</td>
<td><strong>1.5 2.3 3.9 5.8 8.4</strong></td>
<td>3.2 8.8 10.2 11.3</td>
</tr>
<tr>
<td></td>
<td>ConSE(CV)</td>
<td>1.3 2.1 3.8 5.8 8.7</td>
<td>3.2 9.2 10.7 12.0</td>
</tr>
<tr>
<td></td>
<td>Ours o-vs-o</td>
<td><strong>1.4 2.4 4.5 7.1 10.9</strong></td>
<td>3.1 9.0 10.9 <strong>12.5</strong></td>
</tr>
<tr>
<td></td>
<td>Ours struct</td>
<td><strong>1.5 2.4 4.4 4.4 6.7 10.0</strong></td>
<td><strong>3.6 9.6 11.0 12.2</strong></td>
</tr>
</tbody>
</table>

Figure 3: 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 5.6 of this material for details.


<table>
<thead>
<tr>
<th>Persian cat</th>
<th>Hippo</th>
<th>Leopard</th>
<th>Humpback whale</th>
<th>Seal</th>
<th>Chimpanzee</th>
<th>Rat</th>
<th>Giant panda</th>
<th>Pig</th>
<th>Raccoon</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
</tbody>
</table>

**Figure 4:** Qualitative results of our method (Ours\textsuperscript{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.

<table>
<thead>
<tr>
<th>Artic tern</th>
<th>Ringed kingfisher</th>
<th>American crow</th>
<th>Cedar waxwing</th>
<th>House sparrow</th>
<th>Orange-crowned warbler</th>
<th>Hooded warbler</th>
<th>Heermann gull</th>
<th>Cactus wren</th>
<th>Whip-poor will</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
</tbody>
</table>

**Figure 5:** Qualitative results of our method (Ours\textsuperscript{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 6: Qualitative results of our method (Ours$^{o-v-o}$) 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 7: Success/failure case analysis of our method (Ours$^{struc}$) 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.

Figure 8: Success/failure case analysis of our method (Ours\textsuperscript{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 9: Success/failure case analysis of our method (Ours\textsuperscript{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.

Figure 10: Percentages of basis components required to capture 95% of variance in classifier matrices for AwA and CUB.


[34] Z. Zhang and V. Saligrama. Zero-shot learning via semantic similarity embedding. In *ICCV*, 2015. 1, 2, 5, 6