An Empirical Study and Analysis of Generalized Zero-Shot Learning for Object Recognition in the Wild

Wei-Lun Chao¹*, Soravit Changpinyo¹*, Boqing Gong², and Fei Sha³

¹ Department of Computer Science, University of Southern California
weilunc, schangpi@usc.edu
² Center for Research in Computer Vision, University of Central Florida
bgong@crcv.ucf.edu
³ Department of Computer Science, University of California, Los Angeles
feisha@cs.ucla.edu

Abstract. We investigate the problem of generalized zero-shot learning (GZSL). GZSL relaxes the unrealistic assumption in conventional ZSL that test data belong only to unseen novel classes. In GZSL, test data might also come from seen classes and the labeling space is the union of both types of classes. We show empirically that a straightforward application of the classifiers provided by existing ZSL approaches does not perform well in the setting of GZSL. Motivated by this, we propose a surprisingly simple but effective method to adapt ZSL approaches for GZSL. The main idea is to introduce a calibration factor to calibrate the classifiers for both seen and unseen classes so as to balance two conflicting forces: recognizing data from seen classes and those from unseen ones. We develop a new performance metric called the Area Under Seen-Unseen accuracy Curve to characterize this tradeoff. We demonstrate the utility of this metric by analyzing existing ZSL approaches applied to the generalized setting. Extensive empirical studies reveal strengths and weaknesses of those approaches on three well-studied benchmark datasets, including the large-scale ImageNet Full 2011 with 21,000 unseen categories. We complement our comparative studies in learning methods by further establishing an upper-bound on the performance limit of GZSL. There, our idea is to use class-representative visual features as the idealized semantic embeddings. We show that there is a large gap between the performance of existing approaches and the performance limit, suggesting that improving the quality of class semantic embeddings is vital to improving zero-shot learning.

Keywords: zero-shot learning, object recognition

1 Introduction

The availability of large-scale labeled training images is one of the key factors that contribute to recent successes in visual object recognition and classification.

* Equal contributions
It is well-known, however, that object frequencies in natural images follow long-tailed distributions \[1,2,3\]. For example, some animal or plant species are simply rare by nature — it is uncommon to find alpacas wandering around the streets. Furthermore, brand new categories could just emerge with zero or little labeled images; newly defined visual concepts or products are introduced everyday. In this real-world setting, it would be desirable for computer vision systems to be able to recognize instances of those rare classes, while demanding minimum human efforts and labeled examples.

Zero-shot learning (ZSL) has long been believed to hold the key to the above problem of recognition in the wild. ZSL differentiates two types of classes: seen and unseen, where labeled examples are available for seen classes only. Without labeled data, models for unseen classes are learned by relating them to seen ones. This is often achieved by embedding both seen and unseen classes into a common semantic space, such as visual attributes \[4,5,6\] or \textsc{word2vec} representations of the class names \[7,8,9\]. This common semantic space enables transferring models for the seen classes to unseen ones \[10\].

The setup for ZSL is that once models for unseen classes are learned, they are judged based on their ability to discriminate among unseen classes, assuming the absence of seen objects during the test phase. Since proposed in the seminal work of Lampert et al. \[4\], this setting has almost always been adopted for evaluating ZSL methods \[10,11,12,13,14,15,8,16,17,18,19,20,21,22,23,24,25,26,27,28\].

But, does this problem setting truly reflect what recognition in the wild entails? While the ability to learn novel concepts is by all means a trait that any zero-shot learning systems should possess, it is merely one side of the coin. The other important — yet so far under-studied — trait is the ability to remember past experiences, i.e., the seen classes.

Why is this trait desirable? Consider how data are distributed in the real world. The seen classes are often the most common; it is therefore unrealistic to assume that we will never encounter them during the test stage. For models generated by ZSL to be truly useful, they should not only accurately discriminate among either seen or unseen classes themselves but also accurately discriminate between the seen and unseen ones.

Thus, to understand better how existing ZSL approaches will perform in the real world, we advocate evaluating them in the setting of generalized zero-shot learning (GZSL), where test data are from both seen and unseen classes and we need to classify them in the joint labeling space of both types of classes. Previous work in this direction is scarce. See related work for more details.

Our contributions include an extensive empirical study of several existing ZSL approaches in this new setting. We show that a straightforward application of classifiers constructed by those approaches performs poorly. In particular, test data from unseen classes are almost always classified as a class from the “seen” ones. We propose a surprisingly simple yet very effective method called calibrated stacking to address this problem. This method is mindful of the two conflicting forces: recognizing data from seen classes and recognizing data from unseen ones. We introduce a new performance metric called Area Under Seen-Unseen
accuracy Curve (AUSUC) that can evaluate zero-shot learning approaches on how well they can tradeoff among the two. We demonstrate the utility of this metric by evaluating several representative ZSL approaches under this metric on three benchmark datasets, including the large-scale full ImageNet Fall 2011 release dataset [29] that contains 21,000 unseen categories.

We complement our comparative studies in learning methods by further establishing an upper-bound on the performance limit of zero-shot learning. There, our idea is to use class-representative visual features as the ideal semantic embeddings to construct zero-shot learning classifiers. We demonstrate that there is a large gap between existing approaches and this ideal performance limit, suggesting that improving class semantic embedding is vital to achieving true recognition in the wild through zero-shot learning.

The rest of the paper is organized as follows. Section 2 reviews relevant literature on generalized zero-shot learning. We define this setting formally and shed lights on its difficulty in Section 3. In Section 4, we propose a method to remedy the observed issues in the previous section and compare it to related approaches. Experimental results, detailed analysis, and discussions are provided in Section 5, 6, and 7, respectively.

2 Related Work

There has been very little work in generalized zero-shot learning. [30], [31], [8] and [17] allow the label space of their classifiers to include seen classes but they only test on data from the unseen class. [9] proposes a two-stage approach that first determines whether a test data point is from seen or unseen classes, then use the corresponding classifiers accordingly. However, their experiments are limited to only 2 or 6 unseen classes. We describe their methods in Section 4.3 with details in the Supplementary Material, and compare to those approaches in Section 5. [32] and [33] focus on training a zero-shot binary classifier for each unseen class that is able to classify it against seen classes — it is not clear how to distinguish multiple unseen classes from seen ones.

3 Generalized Zero-Shot Learning

In this section, we describe formally the setting of generalized zero-shot learning. We then present empirical evidence to illustrate the difficulty of this problem.

3.1 Conventional and Generalized Zero-shot Learning

Suppose we are given the training data \( \mathcal{D} = \{(x_n \in \mathbb{R}^D, y_n)\}_{n=1}^N \) with the labels \( y_n \) from the label space of seen classes \( \mathcal{S} = \{1, 2, \cdots, S\} \). Denote by \( \mathcal{U} = \{S + 1, \cdots, S + U\} \) the label space of unseen classes. We use \( \mathcal{T} = \mathcal{S} \cup \mathcal{U} \) to represent the union of the two sets of classes.

In the (conventional) zero-shot learning (ZSL) setting, the main goal is to classify test data into unseen classes, assuming the absence of seen classes in the
test phase. In other words, each unclassified data point is assumed to come from
and will be assigned to one of the labels in $U$.

Existing research on ZSL has been almost entirely focusing on this setting
$[4, 10, 11, 12, 13, 14, 15, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28]$. However, in real-
applications, the assumption of encountering data only from the unseen classes is
hardly realistic. The seen classes are often the most common objects we see in
the real world. Thus, the objective in the conventional ZSL does not truly reflect
how the classifiers will perform recognition in the wild.

Motivated by this shortcoming of the conventional ZSL, we advocate studying
the more general setting of generalized zero-shot learning (GZSL) where we no
longer limit the possible class memberships of test data — each of them belongs
to one of the classes in $T$.

3.2 Classifiers

Without the loss of generality, we assume that for each class $c \in T$, we have
a discriminant scoring function $f_c(x)$, from which we would be able to derive
the label for $x$. For instance, for an unseen class $u$, the method of synthesized
classifiers [28] defines $f_u(x) = w_u^T x$, where $w_u$ is the model parameter vector
for the class $u$, constructed from its semantic embedding $a_u$ (such as its attribute
vector or the word vector embedding associated with the name of the class). In
ConSE [17], $f_u(x) = \cos(s(x), a_u)$, where $s(x)$ is the predicted embedding
of the data sample $x$. In DAP/IAP [34], $f_u(x)$ is a probabilistic model of attribute
vectors. We assume that similar discriminant functions for seen classes can be
constructed in the same manner given their corresponding semantic embeddings.

How to assess an algorithm for GZSL? We define and differentiate the follow-
ing performance metrics: $A_{U\rightarrow U}$ the accuracy of classifying test data from $U$ into
$U$, $A_{S\rightarrow S}$ the accuracy of classifying test data from $S$ into $S$, and finally $A_{S\rightarrow T}$
and $A_{U\rightarrow T}$ the accuracies of classifying test data from either seen or unseen
classes into the joint label space. Note that $A_{U\rightarrow U}$ is the standard performance
metric used for conventional ZSL and $A_{S\rightarrow S}$ is the standard metric for multi-
class classification. Furthermore, note that we do not report $A_{T\rightarrow T}$ as simply
averaging $A_{S\rightarrow T}$ and $A_{U\rightarrow S}$ to compute $A_{T\rightarrow T}$ might be misleading when the
two metrics are not balanced, as shown below.

3.3 Generalized ZSL is hard

To demonstrate the difficulty of GZSL, we report empirical results of using a
simple but intuitive algorithm for GZSL. Given the discriminant functions, we
adopt the following classification rule

$$\hat{y} = \arg \max_{c \in T} f_c(x)$$

which we refer to as direct stacking.

We use the rule on “stacking” classifiers from the following zero-shot learning
approaches: DAP and IAP [34], ConSE [17], and Synthesized Classifiers (SynC)
We tune the hyper-parameters for each approach based on class-wise cross validation [28,26,32]. We test GZSL on two datasets AwA [34] and CUB [35] — details about those datasets can be found in Section 5.

Table 1. Classification accuracies (in %) on conventional ZSL ($A_{U \rightarrow U}$), multi-class classification for seen classes ($A_{S \rightarrow S}$), and GZSL ($A_{S \rightarrow T}$ and $A_{U \rightarrow T}$), on AwA and CUB. Significant drops are observed from $A_{U \rightarrow U}$ to $A_{U \rightarrow T}$.

| Method       | AwA          | CUB          |
|--------------|--------------|--------------|
|              | $A_{U \rightarrow U}$ | $A_{S \rightarrow S}$ | $A_{U \rightarrow T}$ | $A_{S \rightarrow T}$ | $A_{U \rightarrow T}$ | $A_{S \rightarrow T}$ |
| DAP [34]     | 51.1         | 78.5         | 2.4          | 77.9         | 38.8         | 56.0         | 4.0          | 55.1         |
| IAP [34]     | 56.3         | 77.3         | 1.7          | 76.8         | 36.5         | 69.6         | 1.0          | 69.4         |
| ConSE [17]   | 63.7         | 76.9         | 9.5          | 75.9         | 35.8         | 70.5         | 1.8          | 69.9         |
| SynC$^{o \text{-} vs-o}$ [28] | 70.1 | 67.3 | 0.3 | 67.3 | 53.0 | 67.2 | 8.4 | 66.5 |
| SynC$^{\text{struct}}$ [28] | 73.4 | 81.0 | 0.4 | 81.0 | 54.4 | 73.0 | 13.2 | 72.0 |

Table 1 reports experimental results based on the 4 performance metrics we have described previously. Our goal here is not to compare between methods. Instead, we examine the impact of relaxing the assumption of the prior knowledge of whether data are from seen or unseen classes.

We observe that, in this setting of GZSL, the classification performance for unseen classes ($A_{U \rightarrow T}$) drops significantly from the performance in conventional ZSL ($A_{U \rightarrow U}$), while that of seen ones ($A_{S \rightarrow T}$) remains roughly the same as in the multi-class task ($A_{S \rightarrow S}$). That is, nearly all test data from unseen classes are misclassified into the seen classes. This unusual degradation in performance highlights the challenges of GZSL; as we only see labeled data from seen classes during training, the scoring functions of seen classes tend to dominate those of unseen classes, leading to biased predictions in GZSL. In particular, the classification rule will aggressively classify a new data point into the label space of $S$ because classifiers for the seen classes do not get trained on “negative” examples from the unseen classes.

In what follows, we consider several approaches to curbing this aggressiveness.

## 4 Approach for GZSL

The previous example shows that the classifiers for unseen classes constructed by conventional ZSL methods should not be naively combined with models for seen classes to expand the labeling space required by the generalized ZSL.

In what follows, we propose a simple variant to the naive approach of direct stacking. We also develop a metric that measures the performance of GZSL, by acknowledging that there is an inherent tradeoff between recognizing seen classes and recognizing unseen classes. This metric, referred to as the Area Under Seen-Unseen accuracy Curve (AUSUC), balances the two conflicting forces. We conclude this section by describing two related approaches: despite their sophistication, they do not perform well empirically.
4.1 Calibrated stacking

Our approach stems from the observation that the scores of the discriminant functions for the seen classes are often greater than the scores for the unseen classes. Thus, intuitively, we would like to reduce the scores for the seen classes. This leads to the following classification rule:

\[ \hat{y} = \arg \max_{c \in T} f_c(x) - \gamma \mathbb{I}[c \in S], \] (2)

where the indicator \( \mathbb{I}[:] \in \{0, 1\} \) indicates whether or not \( c \) is a seen class and \( \gamma \) is a calibration factor. We term this adjustable rule as calibrated stacking.

Another way to interpret \( \gamma \) is to regard it as the prior likelihood of a data point coming from unseen classes. When \( \gamma = 0 \), the calibrated stacking rule reverts back to the direct stacking rule, described previously.

It is also instructive to consider the two extreme cases of \( \gamma \). When \( \gamma \to +\infty \), the classification rule will ignore all seen classes and classify all data points into one of the unseen classes. When there is no new data point coming from seen classes, this classification rule essentially implements what one would do in the setting of conventional ZSL. On the other hand, when \( \gamma \to -\infty \), the classification rule only considers the label space of seen classes as in standard multi-way classification. The calibrated stacking rule thus represents a middle ground between aggressively classifying every data point into seen classes and conservatively classifying every data point into unseen classes. Adjusting this hyperparameter thus gives a tradeoff, which we exploit to define a new performance metric.

4.2 Area Under Seen-Unseen Accuracy Curve (AUSUC)

Varying the calibration factor \( \gamma \), we can compute a series of classification accuracies \( (A_{U \to T}, A_{S \to T}) \). Fig. 1 plots those points for the dataset AwA using the classifiers generated by the method in [28] based on class-wise cross validation. We call such a plot the Seen-Unseen accuracy Curve (SUC).

The two ends of the curve represent the extreme cases \( (\gamma \to -\infty \text{ and } \gamma \to +\infty) \). \( \gamma = 0 \) corresponds to direct stacking, denoted by a cross. The curve
is similar to many familiar curves in machine learning for representing conflicting goals, such as the Precision-Recall (PR) curve and the Receiving Operator Characteristic (ROC) curve.

A convenient way to summarize the plot with one number is to use the Area Under SUC (AUSUC). The higher the area is, the better an algorithm is able to balance $A_{U\rightarrow T}$ and $A_{S\rightarrow T}$. In Section 5, Section 6, and the Supplementary Material, we evaluate the performance of existing zero-shot learning methods under this metric, as well as provide further insights and analyses.

An immediate and important use of the metric AUSUC is for model selection. Many ZSL learning methods require tuning hyperparameters — previous work tune them based on the accuracy $A_{U\rightarrow U}$. The selected model, however, does not necessarily balance optimally between $A_{U\rightarrow T}$ and $A_{S\rightarrow T}$. Instead, we advocate using AUSUC for model selection and hyperparameter tuning. Models with higher values of AUSUC are likely to perform in balance for the task of GZSL. For detailed discussions, see the Supplementary Material.

### 4.3 Alternative approaches

Socher et al. [9] propose a two-stage zero-shot learning approach that first predicts whether an image is of seen or unseen classes and then accordingly applies the corresponding classifiers. The first stage is based on the idea of novelty detection and assigns a high novelty score if it is unlikely for the data points to come from seen classes. They experiment with two novelty detection strategies: Gaussian and LoOP models [36]. We briefly describe and contrast them to our approach below. The details are in the Supplementary Material.

**Novelty detection** The main idea is to assign a novelty score $N(x)$ to each sample $x$. With this novelty score, the final prediction rule becomes

$$
\hat{y} = \begin{cases} 
\arg \max_{c \in S} f_c(x), & \text{if } N(x) \leq -\gamma. \\
\arg \max_{c \in U} f_c(x), & \text{if } N(x) > -\gamma.
\end{cases}
$$

where $-\gamma$ is the novelty threshold. The scores above this threshold indicate belonging to unseen classes. To estimate $N(x)$, for the Gaussian model, data points in seen classes are first modeled with a Gaussian mixture model. The novelty score of a data point is then its negative log probability value under this mixture model. Alternatively, the novelty score can be estimated using the Local Outlier Probabilities (LoOP) model [36]. The idea there is to compute the distances of $x$ to its nearest seen classes. Those distances are then converted into an outlier probability, interpreted as the likelihood of $x$ being from unseen classes.

**Relation to Calibrated Stacking** If we define a new form of novelty score $N(x) = \max_{u \in U} f_u(x) - \max_{s \in S} f_s(x)$ in eq. (3), we recover the prediction rule in eq. (2). However, this relation holds only if we are interested in predicting one label $\hat{y}$. When we are interested in predicting a set of labels (for example, hoping that the correct labels are in the top $K$ predicted labels, (i.e., the Flat hit@K metric, cf. Section 5), the two prediction rules will give different results.
### Table 2. Key characteristics of studied datasets

| Dataset name | Number of seen classes | Number of unseen classes | Total number of images |
|--------------|------------------------|--------------------------|------------------------|
| AwA†         | 40                     | 10                       | 30,475                 |
| CUB‡         | 150                    | 50                       | 11,788                 |
| ImageNet§    | 1000                   | 20,842                   | 14,197,122             |

†: following the prescribed split in [34]. ‡: following [28] to report the average over 4 random splits. §: seen and unseen classes from ImageNet ILSVRC 2012 1K [37] and Fall 2011 release [29], respectively.

## 5 Experimental Results

### 5.1 Setup

**Datasets** We mainly use three benchmark datasets: the Animals with Attributes (AwA) [34], CUB-200-2011 Birds (CUB) [35], and ImageNet (with full 21,841 classes) [37]. Table 2 summarizes their key characteristics.

**Semantic spaces** For the classes in AwA and CUB, we use 85-dimensional or 312-dimensional binary or continuous-valued attributes, respectively [34,35]. For ImageNet, we use 500-dimensional word vectors (word2vec) trained by a skip-gram language model [7,38] provided by Changpinyo et al. [28]. We ignore those classes without word vectors, which results in 20,345 (out of 20,842) unseen classes. We follow [28] to normalize all but binary embeddings to have unit \( \ell_2 \) norms.

**Visual features** We use the GoogLeNet deep features [39] for all datasets (all extracted with the Caffe package [40]). Extracted features come from the 1,024-dimensional activations of the pooling units, as in [20,28].

**Zero-shot learning methods** We examine several representative conventional zero-shot learning approaches, described briefly below. Direct Attribute Prediction (DAP) and Indirect Attribute Prediction (IAP) [34] are probabilistic models that perform attribute predictions as an intermediate step and then use them to compute MAP predictions of unseen class labels. ConSE [17] makes use of pre-trained classifiers for seen classes and their probabilistic outputs to infer the semantic embeddings of each test example, and then classifies it into the unseen class with the most similar semantic embedding. SynC [28] is a recently proposed multi-task learning approach that synthesizes a novel classifier based on semantic embeddings and base classifiers that are learned with labeled data from the seen classes. Two versions of this approach — SynC\textsuperscript{o-v-o} and SynC\textsuperscript{struct} — use one-versus-other and Crammer-Singer style [41] loss functions to train classifiers. We use binary attributes for DAP and IAP, and continuous attributes and word2vec for ConSE and SynC, following [34,17,28].
The Generalized Zero-Shot Learning Tasks There are no previously established benchmark tasks for GZSL. We thus define a set of tasks that reflects more closely how data are distributed in real-world applications.

We construct the GZSL tasks by composing test data as a combination of images from both seen and unseen classes. We follow existing splits of the datasets for the conventional ZSL to separate seen and unseen classes. Moreover, for the datasets AwA and CUB, we hold out 20% of the data points from the seen classes (previously, all of them are used for training in the conventional zero-shot setting) and merge them with the data from the unseen classes to form the test set; for ImageNet, we combine its validation set (having the same classes as its training set) and the 21K classes that are not in the ILSVRC 2012 1K dataset.

Evaluation Metrics While we will primarily report the performance of ZSL approaches under the metric Area Under Seen-Unseen accuracy Curve (AUSUC) developed in Section 4.1, we explain how its two accuracy components $A_{S \rightarrow T}$ and $A_{U \rightarrow T}$ are computed below.

For AwA and CUB, seen and unseen accuracies correspond to (normalized-by-class-size) multi-way classification accuracy, where the seen accuracy is computed on the 20% images from the seen classes and the unseen accuracy is computed on images from unseen classes.

For ImageNet, seen and unseen accuracies correspond to Flat hit@K (F@K), 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 unnormalized multi-way classification accuracy. Moreover, following the procedure in [8,17,28], we evaluate on three scenarios of increasing difficulty: (1) 2-hop contains 1,509 unseen classes that are within two tree hops of the seen 1K classes according to the ImageNet label hierarchy. (2) 3-hop contains 7,678 unseen classes that are within three tree hops of seen classes. (3) All contains all 20,345 unseen classes.

5.2 Which method to use to perform GZSL?

In Section 4, we describe several ways of utilizing seen and unseen classifiers for generalized ZSL. Table 3 provides a experimental comparison between those methods, with hyperparameters cross-validated to maximize AUSUC.

The results show that, irrespective of which ZSL methods are used to generate models for seen and unseen classes, our method of calibrated stacking for generalized ZSL outperforms other methods. In particular, despite their probabilistic justification, the two novelty detection methods do not perform well. We believe that this is because most existing zero-shot learning methods are discriminative and optimized to take full advantage of class labels and semantic information. In contrast, either Gaussian or LoOP approach models all the seen classes as a whole, possibly at the cost of modeling inter-class differences.

1 http://www.image-net.org/api/xml/structure_released.xml
Table 3. Performance measured in AUSUC of several methods for Generalized Zero-Shot Learning on AwA and CUB. The higher the better (the upper-bound is 1).

| Method           | AwA                        | CUB                        |
|------------------|----------------------------|----------------------------|
|                  | Novelty detection [9]      | Novelty detection [9]      |
|                  | Calibrated                 | Calibrated                 |
|                  | Gaussian                   | LoOP                       | Stacking                   |
|                  | Gaussian                   | LoOP                       | Stacking                   |
|                  | Stacking                   |                             |                            |
| DAP              | 0.302                      | 0.366                      | 0.122                      | 0.194                      |
| IAP              | 0.307                      | 0.287                      | 0.394                      | 0.129                      | 0.137                      | 0.212                      |
| ConSE            | 0.342                      | 0.300                      | 0.428                      | 0.130                      | 0.136                      | 0.212                      |
| SynC_{o-vs-o}    | 0.420                      | 0.378                      | 0.568                      | 0.191                      | 0.209                      | 0.336                      |
| SynC_{struct}    | 0.424                      | 0.373                      | 0.583                      | 0.199                      | 0.224                      | 0.356                      |

Fig. 2. Comparison between several ZSL approaches of the performance on the task of GZSL for AwA and CUB.

Fig. 3. Comparison between ConSE and SynC of their performance on the task of GZSL for ImageNet where the unseen classes are within 2 tree-hops from seen classes.
Table 4. Performance measured in AUSUC by different zero-shot learning approaches on GZSL on ImageNet, using our method of calibrated stacking.

| Unseen classes | Method       | Flat hit@K |
|----------------|--------------|------------|
|                |              | 1 | 5 | 10 | 20   |
| 2-hop          | ConSE        | 0.042 | 0.168 | 0.247 | 0.347 |
|                | SynC^o-vs-o  | 0.044 | 0.218 | 0.338 | 0.466 |
|                | SynC^struct  | 0.043 | 0.199 | 0.308 | 0.433 |
| 3-hop          | ConSE        | 0.013 | 0.057 | 0.090 | 0.135 |
|                | SynC^o-vs-o  | 0.012 | 0.070 | 0.119 | 0.186 |
|                | SynC^struct  | 0.013 | 0.066 | 0.110 | 0.170 |
| All            | ConSE        | 0.007 | 0.030 | 0.048 | 0.073 |
|                | SynC^o-vs-o  | 0.006 | 0.034 | 0.059 | 0.097 |
|                | SynC^struct  | 0.007 | 0.033 | 0.056 | 0.090 |

5.3 Which Zero-shot Learning approach is more robust to GZSL?

Fig. 2 contrasts in detail several ZSL approaches when tested on the task of GZSL, using the method of calibrated stacking. Clearly, the SynC method dominates all other methods in the whole ranges. The crosses on the plots mark the results of direct stacking (Section 3).

Fig. 3 contrasts in detail ConSE to SynC, the two known methods for large-scale ZSL. When the accuracies measured in Flat hit@1 (i.e., multi-class classification accuracy), neither method dominates the other, suggesting the different tradeoffs by the two methods. However, when we measure hit rates in the top \( K > 1 \), SynC dominates ConSE. Table 4 gives summarized comparison in AUSUC between the two methods on the ImageNet dataset. We observe that SynC in general outperforms ConSE except when Flat hit@1 is used, in which case the two methods’ performances are nearly indistinguishable. Additional plots can be found in the Supplementary Material.

6 Analysis on (Generalized) Zero-shot Learning

Zero-shot learning, either in their conventional setting or generalized setting, is a challenging problem as there is no labeled data for the unseen classes. The performance of ZSL methods depends on at least two factors: (1) how seen and unseen classes are related; (2) how effectively the relation can be exploited by learning algorithms to generate models for the unseen classes. For generalized zero-shot learning, the performance further depends on how classifiers for seen and unseen classes are combined to classify new data into the joint label space.

Despite extensive study in ZSL, several questions remain understudied. For example, given a dataset and a split of seen and unseen classes, what is the best possible performance of any ZSL method? How far are we from there? What is the most crucial component we can improve in order to reduce the gap between the state-of-the-art and the ideal performances?

In this section, we empirically analyze ZSL methods in detail and shed light on some of those questions.
Fig. 4. We contrast the performance of GZSL to multi-class classifiers trained with labeled data from both seen and unseen classes on the dataset ImageNet-2K. GZSL uses word2vector (in red color) and the idealized visual features (G-attr) as semantic embeddings (in black color).

**Setup** As ZSL methods do not use labeled data from unseen classes for training classifiers, one reasonable estimate of their best possible performance is to measure the performance on a multi-class classification task where annotated data on the unseen classes are provided.

Concretely, to construct the multi-class classification task, on AwA and CUB, we randomly select 80% of the data along with their labels from all classes (seen and unseen) to train classifiers. The remaining 20% will be used to assess both the multi-class classifiers and the classifiers from ZSL. Note that, for ZSL, only the seen classes from the 80% are used for training — the portion belonging to the unseen classes are not used.

On ImageNet, to reduce the computational cost (of constructing multi-class classifiers which would involve 21,000-way classification), we subsample another 1,000 unseen classes from its original 21,000 unseen classes. We call this new dataset ImageNet-2K (including the 1K seen classes from ImageNet). The subsampling procedure is described in the Supplementary Material and the main goal is to maintain the proportions of difficult unseen classes unchanged. Out of those 1,000 unseen classes, we randomly select 50 samples per class and reserve them for testing and use the remaining examples (along with their labels) to train 2000-way classifiers. As in AwA and CUB, for ZSL methods, those unseen examples are never used.

For ZSL methods, we use either attribute vectors or word vectors (word2vec) as semantic embeddings. Since SynCo-vs-o [28] performs well on a range of datasets and settings, we focus on this method. For multi-class classification, we train one-versus-other SVMs. Once we obtain the classifiers for both seen and unseen classes, we use the calibrated stacking decision rule to combine (as in generalized ZSL) and vary the calibration factor $\gamma$ to obtain the Seen-Unseen accuracy Curve, exemplified in Fig. 1.

**How far are we from the ideal performance?** Fig. 5 displays the Seen-Unseen accuracy Curves for ImageNet-2K — additional plots on ImageNet-2K and similar ones on AwA and CUB are in the Supplementary Material.
Clearly, there is a large gap between the performance of GZSL using the default \textit{word2vec} semantic embeddings and the ideal performance indicated by the multi-class classifiers. Note that the cross marks indicate the results of \textit{direct stacking}. The multi-class classifiers not only dominate GZSL in the whole ranges (thus, with very high AUSUCs) but also are capable of learning classifiers that are well-balanced (such that \textit{direct stacking} works well).

\textit{How much can idealized semantic embeddings help?} We hypothesize that a large portion of the gap between GZSL and multi-class classification can be attributed to the weak semantic embeddings used by the GZSL approach.

We investigate this by using a form of \textit{idealized} semantic embeddings. As the success of zero-shot learning relies heavily on how accurate semantic embeddings represent visual similarity among classes, we examine the idea of \textit{visual features as semantic embeddings}. Concretely, for each class, semantic embeddings can be obtained by averaging visual features of images belonging to that class. We call them \textbf{G-attr} as we derive the visual features from GoogLeNet. Note that, for unseen classes, we only use the reserved training examples to derive the semantic embeddings; we do not use their labels to train classifiers.

Fig. 5 shows the performance of GZSL using \textbf{G-attr} — the gaps to the multi-class classification performance are significantly reduced from those made by GZSL using \textit{word2vec}. In some cases (see the Supplementary Material for more comprehensive experiments), GZSL can almost match the performance of multi-class classifiers without even using labels from the unseen classes!

\textit{How much labeled data do we need to improve GZSL’s performance?}

Imagine we are given a budget to label data from unseen classes, how much those labels can improve GZSL’s performance?

Table 5 contrasts the AUSUCs obtained by GZSL to those from mutli-class classification on \textbf{ImageNet-2K}, where GZSL is allowed to use visual features as embeddings — those features can be computed from a few labeled images from the unseen classes, a scenario we can refer to as “few-shot” learning. Using about (randomly sampled) 100 labeled images per class, GZSL can quickly approach the performance of multi-class classifiers, which use about 1,000 labeled images per class. Moreover, those G-attr visual features as semantic embeddings improve upon \textit{word2vec} more significantly under Flat hit@K = 1 than when K > 1.

We further examine on the whole \textbf{ImageNet} with 21,000 unseen classes in Table 6, where we keep 80\% of the unseen classes’ examples to derive \textbf{G-attr} and test on the rest, and observe similar trends. Specifically on Flat hit@1, the performance of G-attr from merely 1 image is boosted \textit{threefold} of that by \textit{word2vec}, while G-attr from 100 images achieves over tenfold. See the Supplementary Material for details, including results on \textbf{AwA} and \textbf{CUB}. (Due to the constraint on computational resources, we are not able to complete the construction of 22,000-way multi-class classifiers on the full \textbf{ImageNet}, as we did for \textbf{ImageNet-2K} in Table 5.)
Table 5. Comparison of performance measured in AUSUC between GZSL (using word2vec and G-attr) and multi-class classification on ImageNet-2K. Few-shot results are averaged over 100 rounds. GZSL with G-attr improves upon GZSL with word2vec significantly and quickly approaches multi-class classification performance.

| Method | Flat hit@K |
|--------|------------|
|        | 1          | 5          | 10         | 20         |
| WORD2VEC | 0.04       | 0.17       | 0.27       | 0.38       |
| G-attr from 1 image | 0.08±0.003 | 0.25±0.005 | 0.33±0.005 | 0.42±0.005 |
| G-attr from 2 images | 0.12±0.002 | 0.33±0.004 | 0.44±0.005 | 0.54±0.005 |
| G-attr from 5 images | 0.17±0.002 | 0.44±0.003 | 0.56±0.003 | 0.66±0.003 |
| G-attr from 10 images | 0.20±0.002 | 0.50±0.002 | 0.62±0.002 | 0.72±0.002 |
| G-attr from 100 images | 0.25±0.001 | 0.57±0.001 | 0.69±0.001 | 0.78±0.001 |
| G-attr from all images | 0.25       | 0.58       | 0.69       | 0.79       |
| Multi-class classification | 0.35       | 0.66       | 0.75       | 0.82       |

Table 6. Comparison of performance measured in AUSUC between GZSL with word2vec and GZSL with G-attr on the full ImageNet with 21,000 unseen classes. Few-shot results are averaged over 20 rounds.

| Method | Flat hit@K |
|--------|------------|
|        | 1          | 5          | 10         | 20         |
| WORD2VEC | 0.006      | 0.034      | 0.059      | 0.096      |
| G-attr from 1 image | 0.018±0.0002 | 0.071±0.0007 | 0.106±0.0009 | 0.150±0.0011 |
| G-attr from 2 images | 0.027±0.0002 | 0.103±0.0007 | 0.153±0.0009 | 0.212±0.0012 |
| G-attr from 5 images | 0.041±0.0002 | 0.152±0.0006 | 0.221±0.0007 | 0.300±0.0008 |
| G-attr from 10 images | 0.050±0.0002 | 0.184±0.0003 | 0.263±0.0004 | 0.352±0.0005 |
| G-attr from 100 images | 0.065±0.0001 | 0.230±0.0002 | 0.322±0.0002 | 0.421±0.0002 |
| G-attr from all images | 0.067      | 0.236      | 0.329      | 0.429      |

7 Discussion

Zero-shot learning methods have been studied and evaluated in the unrealistic setting where the test data are assumed to come from unseen classes only. In this paper, we take a first step toward making zero-shot learning closer to the goal of object recognition in the wild. We advocate studying the problem of generalized zero-shot learning where test data can come from both seen and unseen classes. We show that naively using the classifiers constructed by zero-shot learning approaches does not perform well in this generalized setting. Instead, we propose a simple but effective method that can be used to balance two conflicting forces: recognizing data from seen classes versus unseen ones. We develop a performance metric to characterize the tradeoff and examine the utility of this metric in evaluating various zero-shot learning approaches. Our analysis also leads us to investigate the best possible performance of any zero-shot learning methods. We show that there is a large gap between existing approaches and the best possible. Furthermore, we show that this gap can be reduced significantly if idealized semantic embeddings are used. Thus, an important direction for future research is to improve the quality of semantic embeddings of seen and unseen classes.
References

1. Sudderth, E.B., Jordan, M.I.: Shared segmentation of natural scenes using dependent pitman-yor processes. In: NIPS. (2008)
2. Salakhutdinov, R., Torralba, A., Tenenbaum, J.: Learning to share visual appearance for multiclass object detection. In: CVPR. (2011)
3. Zhu, X., Anguelov, D., Ramanan, D.: Capturing long-tail distributions of object subcategories. In: CVPR. (2014)
4. Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: CVPR. (2009)
5. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. In: CVPR. (2009)
6. Parikh, D., Grauman, K.: Relative attributes. In: ICCV. (2011)
7. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR Workshops. (2013)
8. Frome, A., Corrado, G.S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., Mikolov, T.: Devise: A deep visual-semantic embedding model. In: NIPS. (2013)
9. Socher, R., Ganjoo, M., Manning, C.D., Ng, A.: Zero-shot learning through cross-modal transfer. In: NIPS. (2013)
10. Palatucci, M., Pomerleau, D., Hinton, G.E., Mitchell, T.M.: Zero-shot learning with semantic output codes. In: NIPS. (2009)
11. Yu, X., Aloimonos, Y.: Attribute-based transfer learning for object categorization with zero/one training example. In: ECCV. (2010)
12. Rohrbach, M., Stark, M., Schiele, B.: Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In: CVPR. (2011)
13. Kankuekul, P., Kawewong, A., Tangruamsub, S., Hasegawa, O.: Online incremental attribute-based zero-shot learning. In: CVPR. (2012)
14. Akata, Z., Perronnin, F., Harchaoui, Z., Schmid, C.: Label-embedding for attribute-based classification. In: CVPR. (2013)
15. Yu, F.X., Cao, L., Feris, R.S., Smith, J.R., Chang, S.F.: Designing category-level attributes for discriminative visual recognition. In: CVPR. (2013)
16. Mensink, T., Gavves, E., Snoek, C.G.: Costa: Co-occurrence statistics for zero-shot classification. In: CVPR. (2014)
17. Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G.S., Dean, J.: Zero-shot learning by convex combination of semantic embeddings. In: ICLR. (2014)
18. Jayaraman, D., Grauman, K.: Zero-shot recognition with unreliable attributes. In: NIPS. (2014)
19. Al-Halah, Z., Stiefelhagen, R.: How to transfer? zero-shot object recognition via hierarchical transfer of semantic attributes. In: WACV. (2015)
20. Akata, Z., Reed, S., Walter, D., Lee, H., Schiele, B.: Evaluation of output embeddings for fine-grained image classification. In: CVPR. (2015)
21. Fu, Y., Hospedales, T.M., Xiang, T., Gong, S.: Transductive multi-view zero-shot learning. TPAMI (2015)
22. Fu, Z., Xiang, T., Kodirov, E., Gong, S.: Zero-shot object recognition by semantic manifold distance. In: CVPR. (2015)
23. Li, X., Guo, Y., Schuurmans, D.: Semi-supervised zero-shot classification with label representation learning. In: ICCV. (2015)
24. Romera-Paredes, B., Torr, P.H.S.: An embarrassingly simple approach to zero-shot learning. In: ICML. (2015)
25. Kodirov, E., Xiang, T., Fu, Z., Gong, S.: Unsupervised domain adaptation for zero-shot learning. In: ICCV. (2015) 2, 4
26. Zhang, Z., Saligrama, V.: Zero-shot learning via semantic similarity embedding. In: ICCV. (2015) 2, 4, 5, 17
27. Zhang, Z., Saligrama, V.: Classifying unseen instances by learning class-independent similarity functions. arXiv preprint arXiv:1511.04512 (2015) 2, 4
28. Changpinyo, S., Chao, W.L., Gong, B., Sha, F.: Synthesized classifiers for zero-shot learning. arXiv preprint arXiv:1603.00550 (2016) 2, 4, 5, 6, 8, 9, 12, 17
29. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. (2009) 3, 8
30. Tang, K.D., Tappen, M.F., Sukthankar, R., Lampert, C.H.: Optimizing one-shot recognition with micro-set learning. In: CVPR. (2010) 3
31. Mensink, T., Verbeek, J., Perronnin, F., Csurka, G.: Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In: ECCV. (2012) 3
32. Elhoseiny, M., Saleh, B., Elgammal, A.: Write a classifier: Zero-shot learning using purely textual descriptions. In: ICCV. (2013) 3, 5, 17
33. Lei Ba, J., Swersky, K., Fidler, S., Salakhutdinov, R.: Predicting deep zero-shot convolutional neural networks using textual descriptions. In: ICCV. (2015) 3
34. Lampert, C.H., Nickisch, H., Harmeling, S.: Attribute-based classification for zero-shot visual object categorization. TPAMI 36(3) (2014) 453–465 4, 5, 8
35. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology (2011) 5, 8
36. Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: LoOP: local outlier probabilities. In: CIKM. (2009) 7, 18, 19
37. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV (2015) 8
38. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS. (2013) 8
39. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. CVPR (2015) 8
40. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: ACM Multimedia. (2014) 8
41. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. JMLR 2 (2002) 265–292 8
A Hyper-parameter tuning strategies

A.1 Cross-validation with AUSUC

In Section 4.2 of the main text, we introduce the Area Under Seen-Unseen accuracy curve (AUSUC). To tune the hyper-parameters based on this metric, we simulate the generalized zero-shot learning setting during cross-validation.

Concretely, we split the training data into 5 folds A1, A2, A3, A4 and A5 so that the class labels of these folds are disjoint. We further split 80% and 20% of data from each fold (A1-A5, respectively) into pseudo-train and pseudo-test sets, respectively. We then combine the pseudo-train sets of four folds (for example, A1-A4) for training, and validate on (i) the pseudo-test sets of such four folds (i.e., A1-A4) and (ii) the pseudo-train set of the remaining fold (i.e., A5). That is, the remaining fold serves as the pseudo-unseen classes in cross-validation. We repeat this process for 5 rounds — each round selects a fold as the “remaining” fold, and computes AUSUC on the corresponding validation set. Finally, the average of AUSUCs over all rounds is used to select hyper-parameters.

A.2 Comparison to an alternative strategy

Another strategy for hyper-parameter tuning is to find two sets of hyper-parameters: one optimized for seen classes and the other for unseen classes. The standard cross-validation technique, where \( A_{S \rightarrow S} \) is optimized, can be used for the former. For the latter, it has been shown that the class-wise cross-validation technique \([28,26,32]\), where the conventional zero-shot learning task is simulated, outperforms the standard technique \([28]\). In this case, \( A_{U \rightarrow U} \) is optimized. We thus use the first set of hyper-parameters to construct the scoring functions for
the seen classes, and use the second set for the unseen classes (cf. Section 3.2 and 3.3 of the main text).

In this subsection, we show that the strategy that jointly optimizes hyper-parameters based on AUSUC in most cases leads to better models for GZSL than the strategy that optimizes seen and unseen classifiers’ performances separately. On AwA and CUB, we perform 5-fold cross-validation based on the two strategies and compare the performance of those selected models in Table 1. In general, cross-validation based on AUSUC leads to better models for GZSL. The exceptions are ConSE on AwA and DAP on CUB.

Table 1. Comparison of performance measured in AUSUC between two cross-validation strategies on AwA and CUB. One strategy is based on accuracies ($A_{S\rightarrow S}$ and $A_{U\rightarrow U}$) and the other is based on AUSUC. See text for details.

| Method | AwA CV strategies | CUB CV strategies |
|--------|------------------|------------------|
|        | Accuracies | AUSUC | Accuracies | AUSUC |
| DAP    | 0.341 | 0.366 | 0.202 | 0.194 |
| IAP    | 0.366 | 0.394 | 0.194 | 0.199 |
| ConSE  | 0.443 | 0.428 | 0.190 | 0.212 |
| SynCo-vs-o | 0.539 | 0.568 | 0.324 | 0.336 |
| SynCstruct | 0.551 | 0.583 | 0.356 | 0.356 |

### B Novelty detection approaches: details and additional results

In Section 4.3, we describe alternative approaches to calibrated stacking that are based on the idea of novelty detection. The two novelty detection approaches — Gaussian and LoOP [36] — have been explored by [9]. In this section, we provide details and additional results on these approaches.

#### B.1 Algorithms

In [9], Socher et al. first learn a mapping from the visual feature space to the semantic embedding space. The novelty detection is then performed in this semantic space. Below we describe how to compute novelty scores under Gaussian and LoOP models.

**Gaussian** Training examples of seen classes are first mapped into the semantic space and modeled by a Gaussian mixture model — each class is parameterized by a mean vector and an isometric covariance matrix. The mean is set to be the class’ semantic embedding and the covariance matrix is set to be the covariance of all mapped training examples of that class. The novelty score of a test data point is then its negative log probability value under this mixture model.
LoOP  Let $X_S$ be the set of all the mapped training examples from seen classes. For a test sample $x$ (also mapped into the semantic space), a context set $C(x) \subseteq X_S$ of $k$ nearest neighbors is first defined. The probabilistic set distance $pdist$ from $x$ to all the points in $C(x)$ is then computed as follows

$$pdist_\lambda(x, C(x)) = \lambda \sqrt{\frac{\sum_{x' \in C(x)} d(x, x')^2}{|C(x)|}},$$  \hspace{1cm} (1)$$

where $d(x, x')$ is chosen to be the Euclidean distance function. Such a distance is then used to define the local outlier factor

$$lof_\lambda(x) = \frac{pdist_\lambda(x, C(x))}{\mathbb{E}_{x' \in C(x)}[pdist_\lambda(x', C(x'))]} - 1.$$  \hspace{1cm} (2)$$

Finally, the Local Outlier Probability (LoOP), which can be viewed as the novelty score, is computed as

$$\text{LoOP}(x) = \max \left\{ 0, \text{erf} \left( \frac{lof_\lambda(x)}{Z_\lambda(X_S)} \right) \right\},$$  \hspace{1cm} (3)$$

where erf is the Gauss error function and $Z_\lambda(X_S) = \lambda \sqrt{\mathbb{E}_{x' \in X_S}[(lof_\lambda(x'))^2]}$ is the normalization constant.

**B.2 Implementation details**

We use the code provided by Socher et al. [9] and follow their settings. In particular, we train a two-layer neural networks with the same loss function as in [9] to learn a mapping from the visual feature space to the semantic embedding space. We tune the hyper-parameter $\lambda$ (a multiplier on the standard deviation) in LoOP jointly with other hyper-parameters of zero-shot learning approaches — although we empirically observe that $\lambda$ does not significantly affect the novelty detection rankings, consistent with the observations made by [36]. Following [9], we set the number of neighbors (from the seen classes’ examples) $k$ in LoOP to be 20.

**B.3 Additional results**

In Section 5.2 and Table 3 of the main text, we compare Gaussian and LoOP to calibrated stacking, with hyper-parameters cross-validated to maximize AUSUC. In Table 2, we show that calibrated stacking outperforms Gaussian and LoOP as well when hyper-parameters are cross-validated to maximize accuracies (cf. Section A.2). We further show the SUCs of Gaussian, LoOP, and calibrated stacking on AwA in Fig. 1. We observe the superior performance of calibrated stacking over Gaussian and LoOP across all zero-shot learning approaches, regardless of cross-validation strategies. Moreover, interestingly, we see that the curves for Gaussian and LoOP cross each other in such a way that implies that Gaussian has a tendency to classifying more data into “unseen” categories (consistent with the observations reported by [9]).
Table 2. Performance measured in AUSUC for novelty detection (Gaussian and LoOP) and calibrated stacking on AwA and CUB. Hyper-parameters are cross-validated to maximize accuracies. Calibrated stacking outperforms Gaussian and LoOP in all cases. Also, see Table 3 of the main text for the performance when hyper-parameters are cross-validated to directly maximize AUSUC.

| Method | Novelty detection | Calibrated Stacking | Novelty detection | Calibrated Stacking |
|--------|-------------------|---------------------|-------------------|---------------------|
|        | Gaussian | LoOP | Stack | Gaussian | LoOP | Stack |
| DAP    | 0.280    | 0.250 | **0.341** | 0.126    | 0.142 | **0.202** |
| IAP    | 0.319    | 0.289 | **0.366** | 0.132    | 0.149 | **0.194** |
| ConSE  | 0.364    | 0.331 | **0.443** | 0.131    | 0.141 | **0.190** |
| SynC$^{o-v-o}$ | 0.411 | 0.387 | **0.539** | 0.195    | 0.219 | **0.324** |
| SynC$^{\text{struct}}$ | 0.424 | 0.380 | **0.551** | 0.199    | 0.225 | **0.356** |

Fig. 1. Seen-Unseen accuracy Curves (SUC) for Gaussian (Gauss), LoOP, and calibrated stacking (Cal Stack) for all zero-shot learning approaches on AwA. Hyper-parameters are cross-validated based on accuracies (top) and AUSUC (bottom). Calibrated stacking outperforms both Gaussian and LoOP in all cases.
C Comparison between zero-shot learning approaches: Additional results

We provide additional SUC plots for comparing different zero-shot learning approaches, which complements Fig. 2 and Fig. 3 of Section 5.3 of the main text.

In particular, Fig. 2 provides SUCs for all splits of CUB and Fig. 3 provides SUCs for ImageNet 3-hop and All. As before, we observe the superior performance of the method of SynC over other approaches in most cases.

![SUC plots for CUB splits](image)

**Fig. 2.** Comparison of performance measured in AUSUC between different zero-shot learning approaches on the four splits of CUB.

D Analysis on (generalized) zero-shot learning: details and additional results

D.1 Subsampling procedure for constructing ImageNet-2K

In Section 6 of the main text, we construct ImageNet-2K by subsampling 1,000 unseen classes from the original 21,000 unseen classes of ImageNet to reduce computational cost. We pick 74 classes from 2-hop, 303 from “pure” 3-hop (that is, the set of 3-hop classes that are not in the set of 2-hop classes), and 623 from the rest of the classes. These numbers are picked to maintain the proportions of the three types of unseen classes in the original ImageNet (see Evaluation Metrics in Section 5.1 of the main text for more details). Each of these classes has between 1,050-1,550 examples.
Fig. 3. Comparison of performance measured in AUSUC between different zero-shot learning approaches on **ImageNet-3hop** (left) and **ImageNet-All** (right).
D.2 Additional results

We provide additional SUC plots on the performance of GZSL with idealized semantic embeddings \textbf{G-attr} in comparison to the performance of multi-class classification: Fig. 4 for AwA and CUB and Fig. 5 for ImageNet-2K. As observed in Fig. 4 of Section 6 of the main text, the gaps between GZSL with visual attributes/\textsc{Word2Vec} and multi-class classifiers are reduced significantly. The effect of \textbf{G-attr} is particularly immense on the CUB dataset, where GZSL almost matches the performance of multi-class classifiers without using labels from the unseen classes (0.54 vs. 0.59).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig4}
\caption{Comparison between GZSL and multi-class classifiers trained with labeled data from both seen and unseen classes on the datasets \textsc{AwA} and \textsc{CUB}. GZSL uses visual attributes (in red color) or \textbf{G-attr} (in black color) as semantic embeddings.}
\end{figure}

Finally, we provide additional results on the analysis of how much labeled data needed to improve GZSL’s performance on \textsc{AwA} and \textsc{CUB}. In Table 3, we see the same trend as in Table 5 and 6 of the main text; GZSL with \textbf{G-attr} quickly approaches the performance of multi-class classifiers and large improvements from GZSL with visual attributes are observed — even though these attributes are defined and annotated by human experts in this case.
**Fig. 5.** Comparison between GZSL and multi-class classifiers trained with labeled data from both seen and unseen classes on the dataset ImageNet-2K. GZSL uses **word2vec** (in red color) or **G-attr** (in black color) as semantic embeddings.

**Table 3.** Comparison of performance measured in AUSUC between GZSL (using (human-defined) visual attributes and **G-attr**) and multi-class classification on **AwA** and **CUB**. Few-shot results are averaged over 1,000 rounds. GZSL with **G-attr** improves upon GZSL with visual attributes significantly. On **CUB**, the performance of GZSL with visual attributes almost matches that of multi-class classification.

| Method            | AwA     | CUB     |
|-------------------|---------|---------|
| **VISUAL ATTRIBUTES** |         |         |
| G-attr (1-shot)   | 0.55±0.04 | 0.26±0.02 |
| G-attr (2-shot)   | 0.61±0.03 | 0.34±0.02 |
| G-attr (5-shot)   | 0.66±0.02 | 0.44±0.01 |
| G-attr (10-shot)  | 0.69±0.02 | 0.49±0.01 |
| G-attr (100-shot) | 0.71±0.003 | -†      |
| G-attr (all images) | 0.71     | 0.54     |
| **GZSL**          |         |         |
| G-attr (1-shot)   | 0.55±0.04 | 0.26±0.02 |
| G-attr (2-shot)   | 0.61±0.03 | 0.34±0.02 |
| G-attr (5-shot)   | 0.66±0.02 | 0.44±0.01 |
| G-attr (10-shot)  | 0.69±0.02 | 0.49±0.01 |
| G-attr (100-shot) | 0.71±0.003 | -†      |
| G-attr (all images) | 0.71     | 0.54     |
| **Multi-class classification** | 0.81     | 0.59     |

†: We omit this setting as no class in CUB has more than 100 labeled images.