

Supplementary Material: Predicting Visual Exemplars of Unseen Classes for Zero-Shot Learning

Soravit Changpinyo
U. of Southern California
Los Angeles, CA
schangpi@usc.edu

Wei-Lun Chao
U. of Southern California
Los Angeles, CA
weilun@usc.edu

Fei Sha
U. of Southern California
Los Angeles, CA
feisha@usc.edu

This supplementary material provides the following details omitted in the main text.

- Sect. **A**: Details on our proposed zero-shot learning method (Sect. 2.1 of the main text)
- Sect. **B**: Details on the experimental setup, including details on datasets, details on baselines, and hyper-parameter tuning (Sect. 3.1 of the main text)
- Sect. **C**: Expanded and additional results on the predicted exemplars, including another metric for evaluating the quality of predicted exemplars, and larger visualization (Sect. 3.2 of the main text).
- Sect. **D**: Additional experimental results on ZSL, including expanded Table 3 and Table 4, ZSL with word vectors as semantic representations, and qualitative results (Sect. 3.3.1 and 3.3.2 of the main text).
- Sect. **E**: Generalized zero-shot learning results
- Sect. **F**: Details and additional results on zero-shot to few-shot learning experiments, including details on how to select a subset of peeked unseen classes. (Sect. 3.3.3 of the main text)
- Sect. **G**: Additional analysis on dimension for PCA (Sect. 3.3.4 of the main text)
- Sect. **H**: Details on multi-layer perceptron (Sect. 3.3.4 of the main text)

A. Details on our proposed zero-shot learning method

SVR formulation for predicting visual exemplars In Sect. 2.1 of the main text, given semantic representation-visual exemplar pairs of the seen classes, we learn d support vector regressors (SVR) with RBF kernel. Specifically, for

each dimension $d = 1, \dots, d$ of $\mathbf{v}_c$, SVR is learned based on the ν -SVR formulation [24]:

$$\begin{aligned} \min_{\mathbf{w}, \xi, \xi', \epsilon} \quad & \frac{1}{2} \mathbf{w}^T \mathbf{w} + \lambda(\nu \epsilon + \frac{1}{S} \sum_{c=1}^S (\xi_c + \xi'_c)) \\ \text{s.t.} \quad & \mathbf{w}^T \boldsymbol{\theta}^{\text{rbf}}(\mathbf{a}_c) - \mathbf{v}_c \leq \epsilon + \xi_c \\ & \mathbf{v}_c - \mathbf{w}^T \boldsymbol{\theta}^{\text{rbf}}(\mathbf{a}_c) \leq \epsilon + \xi'_c \\ & \xi_c \geq 0, \xi'_c \geq 0, \end{aligned} \quad (1)$$

where $\boldsymbol{\theta}^{\text{rbf}}$ is an implicit nonlinear mapping based on our kernel. We have dropped the subscript d for aesthetic reasons but readers are reminded that each regressor is trained independently with its own target values (i.e., $\mathbf{v}_{cd}$) and parameters (i.e., $\mathbf{w}_d$). We found that the regression error is not sensitive to λ and set it to 1 in all experiments except for zero-shot to few-shot learning. We jointly tune $\nu \in (0, 1]$ and the kernel bandwidth and finally apply the same set of hyper-parameters for all the d regressors. Details on hyper-parameter tuning can be found in Sect. B.3. The resulting $\boldsymbol{\psi}(\cdot) = [\mathbf{w}_1^T \boldsymbol{\theta}^{\text{rbf}}(\cdot), \dots, \mathbf{w}_d^T \boldsymbol{\theta}^{\text{rbf}}(\cdot)]^T$, where $\mathbf{w}_d$ is from the d -th regressor.

B. Details on the experimental setup

B.1. Additional information on datasets

We experiment on four benchmark datasets. The **Animals with Attributes (AwA)** dataset [17] consists of 30,475 images of 50 animal classes, along with a standard data split for zero-shot learning — 40 seen classes (for training) and 10 unseen classes. The **CUB-200-2011 Birds (CUB)** [28] has 200 bird classes and 11,788 images, while the **SUN Attribute (SUN)** dataset [21] contains 14,340 images of 717 scene categories (20 images from each category). We follow seen/unseen splits in [17] for AwA, and [4] for CUB (4 splits) and SUN (10 splits). We report average results from all the splits.

On **ImageNet** [6], we follow the setting in [9, 20, 4]. We use the ILSVRC 2012 1K dataset [23], which contains

1,281,167 training and 50,000 validation images from 1,000 categories, as data from seen classes. Images of unseen classes come from the rest of the ImageNet Fall 2011 release dataset [6] that do not overlap with any of those 1,000 categories. In total, this dataset consists of 14,197,122 images from 21,841 classes, **20,842 unseen classes** of which are unseen ones. Note that, as mentioned in [4], there is one class in the ILSVRC 2012 1K dataset that does not appear in the ImageNet 2011 21K dataset. Thus, we have a total of 20,842 unseen classes to evaluate.

B.2. Details on ZSL baselines

We focus on comparing our method with a recent state-of-the-art baseline SYNC [4]. Specifically, we adopt the version that sets the number of base classifiers to be S (the number of seen classes), and sets $\mathbf{b}_r = \mathbf{a}_c$ for $r = c$ (cf. Sec. 2.2.2 of the main text). Note that this is the version that has reported results on all four datasets.

SYNC has been shown to outperform multiple strong baselines under the same setting. In particular, under the setting of [4] which we adopt in this paper, SYNC outperforms SJE [3], ESZSL [22], COSTA [19], and CONSE [20]. For more details, see Table 3 and Table 4 in [4]. Under the setting of [30] (with ResNet deep features [11] and standard dataset splits), SYNC is the best performing method among [17, 2, 9, 26, 20, 3, 22, 33, 31] on average on **AwA**, **CUB**, **SUN**, and additionally **aPY** [8] (See Fig. 1 in [30]), and by far the best performing method on **ImageNet** among [2, 9, 26, 20, 3, 22, 31] (See Table 4 in [30]). Note that Xian et al. [30] recently proposes alternative splits of the datasets. In some scenarios, SYNC may not perform best on these splits. We leave further investigation of the performance of our ZSL method on these newly proposed splits for future work.

Besides SYNC, in Table 3 and Table 5 of the main text, we also include ZSL recognition accuracies of *recent* ZSL methods that have not been compared in [4], including BiDiLEL [29], LATeM [31], and CCA [18]. For each of these methods, we strive to ensure fair comparison in terms of semantic representations, visual features, and evaluation metrics.

B.3. Hyper-parameter tuning

There are several hyper-parameters to be tuned in our experiments: (a) projected dimensionality d for PCA and (b) λ , ν , and the RBF-kernel bandwidth in SVR. For (a), we found that the ZSL performance is not sensitive to d and thus set $d = 500$ for all experiments. For (b), we perform *class-wise* cross-validation (CV), following previous work [4, 7, 33], with two exceptions. First, we found $\lambda = 1$ works robustly on all datasets for zero-shot learning. Second, we fix all the hyper-parameters when we increase the number of peeked unseen classes (c.f. Sect. 3.3.3 of the

main text) in the case of EXEM (1NN)¹.

The *class-wise* CV can be done as follows. We hold out data from a subset of seen classes as pseudo-unseen classes, train our models on the remaining folds (which belong to the remaining classes), and tune hyper-parameters based on a certain performance metric on the held-out fold. This scenario simulates the ZSL setting and has been shown to outperform the conventional CV in which each fold contains a portion of training examples from all classes [4].

We consider the following two performance metrics. The first one minimizes the distance between the predicted exemplars and the ground-truth (average of PCA-projected validation data of each class) in $\mathbb{R}^d$. We use the Euclidean distance in this case. We term this measure **CV-distance**. This approach does not assume the downstream task at training and aims to measure the quality of predicted exemplars by its *faithfulness*.

The other approach maximizes the zero-shot classification accuracy on the validation set. This measure can easily be obtained for EXEM (1NN) and EXEM (1NNs), which use simple decision rules that have no further hyper-parameters to tune. Empirically, we found that **CV-accuracy** generally leads to slightly better performance. The results reported in the main text for these two approaches are thus based on this measure.

On the other hand, EXEM (SYNC^{0-vs-0}), EXEM (SYNC^{STRUCT}), EXEM (CONSE), and EXEM (LATeM) require further hyper-parameter tuning. For computational purposes, we use **CV-distance** for tuning hyper-parameters of the regressors, followed by the hyper-parameter tuning for SYNC and CONSE using the predicted exemplars. Since SYNC and CONSE construct their classifiers based on the distance values between class semantic representations, we do not expect a significant performance drop in this case. (We remind the reader that, in EXEM (SYNC^{0-vs-0}), EXEM (SYNC^{STRUCT}), EXEM (CONSE), and EXEM (LATeM), the predicted exemplars are used as semantic representations.)

C. Expanded and additional results on the predicted exemplars

C.1. Another metric for evaluating the quality of predicted visual exemplars

Besides the Pearson correlation coefficient used in Table 2 of the main text², we provide another evidence that predicted exemplars better reflect visual similarities (as de-

¹In the experiments where we peek into some unseen classes' examples, we find that, for EXEM (1NN), fixing the hyper-parameters tuned on ZSL (with 0 peeked unseen classes) works robustly for other numbers of peeked unseen classes. However, this is not the case for SYNC, in which case we tune the hyper-parameters for different numbers of peeked unseen classes.

²We treat rows of each distance matrix as data points and compute the Pearson correlation coefficients between matrices.

Table 1: Overlap of k-nearest classes (in %) on **AwA**, **CUB**, **SUN**. We measure the overlap between those searched by real exemplars and those searched by semantic representations (i.e., attributes) or predicted exemplars. We set k to be 40 % of the number of unseen classes. See text for more details.

Distances for kNN using	AwA (k=4)	CUB (k=20)	SUN (k=29)
Semantic representations	57.5	68.9	75.2
Predicted exemplars	67.5	80.0	82.1

finer by real exemplars) than semantic representations. Let $\%kNN\text{overlap}(D)$ be the percentage of k-nearest neighbors (neighboring classes) using distances D that overlap with k-nearest neighbors using real exemplar distances. In Table 1, we report $\%kNN\text{overlap}$ (semantic representation distances) and $\%kNN\text{overlap}$ (predicted exemplar distances). We set k to be 40% of the number of unseen classes, but we note that the trends are consistent for different ks. Similar to the results in the main text, we observe clear improvement in all cases.

C.2. Larger visualization of the predicted exemplars

We provide the t-SNE visualization [27] of the predicted visual exemplars of the *unseen* classes for **AwA**, **CUB**, **SUN**, and **ImageNet** in Fig. 1, 2, 3, and 4, respectively — each class is designated a color, with its corresponding real images/predicted exemplar marked with crosses/circle. Note that these figures are larger-size versions of Fig. 2 of the main text. For many of the unseen classes, the predicted exemplars are well aligned with their corresponding real images, explaining the superior performance of applying them for ZSL even though a simple nearest neighbor classification is used.

Note that it is the *relative* distance that is important. Even when the predicted exemplars are not well aligned with their corresponding images, they are in many cases closer to those images than the predicted exemplars of other classes are. For example, on **AwA**, we would be able to predict test images from “orange” class correctly as the closest exemplar is orange (but the images and the exemplar are not exactly aligned).

D. Expanded zero-shot learning results

D.1. Expanded main results on small datasets

Table 2 expands Table 3 of the main text to include additional baselines. First, we include results of additional baselines [3, 22, 19] reported in [4]. Second, we report results of very recently proposed methods that use the more optimistic metric *per-sample* accuracy as well as different types of deep visual features.

Per-sample accuracy is computed by averaging over accuracy of each sample. This is different from *per-class* accuracy that is computed by averaging over accuracy of each unseen class. It is likely that *per-sample* accuracy is the more optimistic metric of the two, as [30] reports that they are unable to reproduce results of SSE [33], which uses *per-sample* accuracy, with *per-class* accuracy.

We also note that visual features can affect the performance greatly. For example, VGG features [25] of **AwA** used in [33, 34, 29] are likely more discriminative than GoogLeNet features. In particular, BiDILEL [29] reports results on both features with VGG outperforming GoogLeNet by an absolute 5.8%. This could explain strong results on **AwA** reported in [33, 34, 29]. It would also be interesting to investigate how GoogLeNet V2 [12] (in addition to *per-sample* evaluation metric) used by DEM [32] contributes to their superior performance on **AwA**.

Finally, despite the variations in experimental settings, our method still outperforms all baselines on **CUB**.

D.2. Expanded ImageNet results

Table 3 expands the results of Table 4 in the main text to include other previously *published* results that use AlexNet features [14] and evaluate on all unseen classes. In all cases, our method outperforms the baseline approaches.

D.3. Additional ZSL results with word vectors as semantic representations

In Table 4, we show that we can improve the quality of word vectors on **AwA** as well. We use the 1,000-dimensional word vectors in [10] and follow the same evaluation protocol as before. For other specific details, please refer to [4].

D.4. Qualitative results

Finally, we provide qualitative results on the zero-shot learning task on **AwA** and **SUN** in Fig. 5. For each row, we provide a class name, three attributes with the highest strength, and the nearest image to the predicted exemplar (projected back to the original visual feature space). We stress that each class that we show here is an *unseen* class, and the images are from unseen classes as well. Generally, the results are reasonable; class names, attributes, and images generally correspond well. Even when the image is from the wrong class, the appearance of the nearest image is reasonable. For example, we predict a hippopotamus exemplar from the pig attributes, but the image does not look too far from pigs. This could also be due to the fact that many of these attributes are not *visual* and thus our regressors are prone to learning the wrong thing [13].

Table 2: Expanded comparison (cf. Table 3 of the main text) to existing ZSL approaches in the multi-way classification accuracies (in %) on **AwA**, **CUB**, and **SUN**. For each dataset, we mark the best in red and the second best in blue. We include results of recent ZSL methods with other types of deep features (VGG by [25] and GoogLeNet V2 by [12]) and/or different evaluation metrics. See text for details on how to interpret these results.

Approach	Visual features	Evaluation metric	AwA	CUB	SUN
SSE [33]	VGG	per-sample	76.3	30.4 [§]	-
JLSE [34]	VGG	per-sample	80.5	42.1 [§]	-
BiDiLEL [29]	VGG	per-sample	79.1	47.6 [§]	-
DEM [32]	GoogLeNet V2	per-sample	86.7	58.3 [§]	-
SJE [3]	GoogLeNet	per-class	66.3	46.5	56.1
ESZSL [22]	GoogLeNet	per-class	64.5	34.5	18.7
COSTA [19]	GoogLeNet	per-class	61.8	40.8	47.9
CONSE [†] [20]	GoogLeNet	per-class	63.3	36.2	51.9
BiDiLEL [29]	GoogLeNet	per-class	72.4	49.7 [§]	-
LATEM [‡] [31]	GoogLeNet	per-class	72.1	48.0	64.5
SYNC ^{O-VS-O} [4]	GoogLeNet	per-class	69.7	53.4	62.8
SYNC ^{CS} [4]	GoogLeNet	per-class	68.4	51.6	52.9
SYNC ^{STRUCT} [4]	GoogLeNet	per-class	72.9	54.5	62.7
EXEM (CONSE)	GoogLeNet	per-class	70.5	46.2	60.0
EXEM (LATEM) [‡]	GoogLeNet	per-class	72.9	56.2	67.4
EXEM (SYNC ^{O-VS-O})	GoogLeNet	per-class	73.8	56.2	66.5
EXEM (SYNC ^{STRUCT})	GoogLeNet	per-class	77.2	59.8	66.1
EXEM (1NN)	GoogLeNet	per-class	76.2	56.3	69.6
EXEM (1NNs)	GoogLeNet	per-class	76.5	58.5	67.3

[§]: on a particular split of seen/unseen classes. [†]: reported in [4]. [‡]: based on the code of [31], averaged over 5 different initializations.

Table 3: Expanded comparison (cf. Table 4 of the main text) to existing ZSL approaches on **ImageNet** using **word vectors** of the class names as semantic representations. For both types of metrics (in %), the higher the better. The best is in red. AlexNet is by [14]. The number of actual unseen classes are given in parentheses. [†]: reported in [4].

Test data	Approach K=	Visual features	Flat Hit@K					Hierarchical precision@K			
			1	2	5	10	20	2	5	10	20
<i>2-hop</i> (1,549)	DEViSE [9]	AlexNet	6.0	10.1	18.1	26.4	36.4	15.2	19.2	21.7	23.3
	CONSE [20]	AlexNet	9.4	15.1	24.7	32.7	41.8	21.4	24.7	26.9	28.4
<i>2-hop</i> (1,509)	CONSE [†] [20]	GoogLeNet	8.3	12.9	21.8	30.9	41.7	21.5	23.8	27.5	31.3
	SYNC ^{O-VS-O} [4]	GoogLeNet	10.5	16.7	28.6	40.1	52.0	25.1	27.7	30.3	32.1
	SYNC ^{STRUCT} [4]	GoogLeNet	9.8	15.3	25.8	35.8	46.5	23.8	25.8	28.2	29.6
	EXEM (SYNC ^{O-VS-O})	GoogLeNet	11.8	18.9	31.8	43.2	54.8	25.6	28.1	30.2	31.6
	EXEM (1NN)	GoogLeNet	11.7	18.3	30.9	42.7	54.8	25.9	28.5	31.2	33.3
	EXEM (1NNs)	GoogLeNet	12.5	19.5	32.3	43.7	55.2	26.9	29.1	31.1	32.0
<i>3-hop</i> (7,860)	DEViSE [9]	AlexNet	1.7	2.9	5.3	8.2	12.5	3.7	19.1	21.4	23.6
	CONSE [20]	AlexNet	2.7	4.4	7.8	11.5	16.1	5.3	20.2	22.4	24.7
<i>3-hop</i> (7,678)	CONSE [†] [20]	GoogLeNet	2.6	4.1	7.3	11.1	16.4	6.7	21.4	23.8	26.3
	SYNC ^{O-VS-O} [4]	GoogLeNet	2.9	4.9	9.2	14.2	20.9	7.4	23.7	26.4	28.6
	SYNC ^{STRUCT} [4]	GoogLeNet	2.9	4.7	8.7	13.0	18.6	8.0	22.8	25.0	26.7
	EXEM (SYNC ^{O-VS-O})	GoogLeNet	3.4	5.6	10.3	15.7	22.8	7.5	24.7	27.3	29.5
	EXEM (1NN)	GoogLeNet	3.4	5.7	10.3	15.6	22.7	8.1	25.3	27.8	30.1
	EXEM (1NNs)	GoogLeNet	3.6	5.9	10.7	16.1	23.1	8.2	25.2	27.7	29.9
<i>All</i> (20,842)	DEViSE [9]	AlexNet	0.8	1.4	2.5	3.9	6.0	1.7	7.2	8.5	9.6
	CONSE [20]	AlexNet	1.4	2.2	3.9	5.8	8.3	2.5	7.8	9.2	10.4
<i>All</i> (20,345)	CONSE [†] [20]	GoogLeNet	1.3	2.1	3.8	5.8	8.7	3.2	9.2	10.7	12.0
	SYNC ^{O-VS-O} [4]	GoogLeNet	1.4	2.4	4.5	7.1	10.9	3.1	9.0	10.9	12.5
	SYNC ^{STRUCT} [4]	GoogLeNet	1.5	2.4	4.4	6.7	10.0	3.6	9.6	11.0	12.2
	EXEM (SYNC ^{O-VS-O})	GoogLeNet	1.6	2.7	5.0	7.8	11.8	3.2	9.3	11.0	12.5
	EXEM (1NN)	GoogLeNet	1.7	2.8	5.2	8.1	12.1	3.7	10.4	12.1	13.5
	EXEM (1NNs)	GoogLeNet	1.8	2.9	5.3	8.2	12.2	3.6	10.2	11.8	13.2

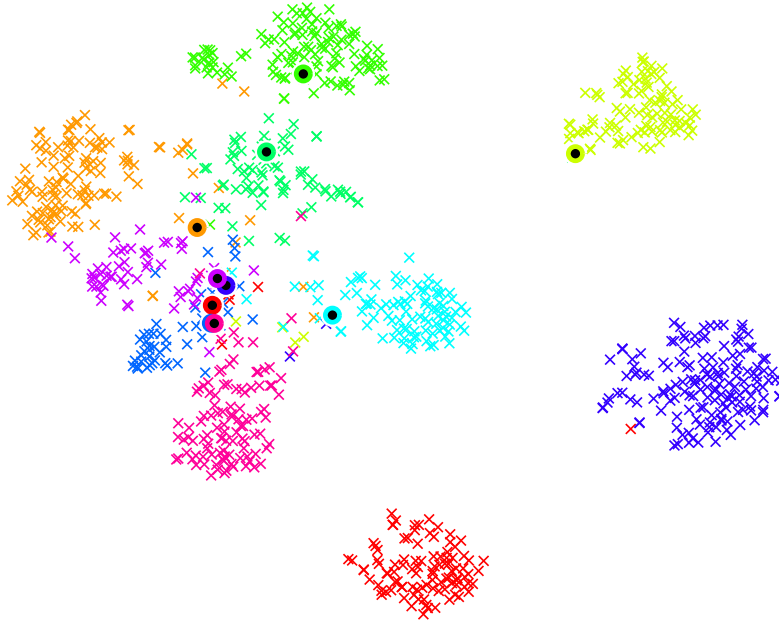


Figure 1: t-SNE [27] visualization of real images (crosses) and predicted visual exemplars (circles) for 10 *unseen* classes on **AWA**. Different colors of symbols denote different unseen classes. Perfect predictions of visual features/exemplars would result in well-aligned crosses and circles of the same color. Best viewed in color.

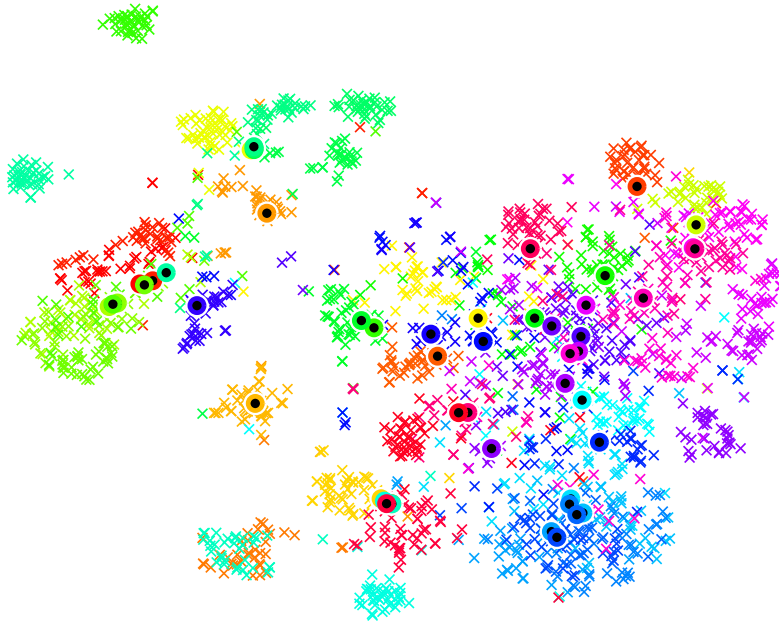


Figure 2: t-SNE [27] visualization of real images (crosses) and predicted visual exemplars (circles) for 50 *unseen* classes on **CUB** (first split). Different colors of symbols denote different unseen classes. Perfect predictions of visual features/exemplars would result in well-aligned crosses and circles of the same color. Best viewed in color.

E. Generalized zero-shot learning results

Conventional zero-shot learning setting unrealistically assumes that test data always come from the unseen classes.

Motivated by this, recent work proposes to evaluate zero-shot learning methods in the more practical setting called generalized zero-shot learning (GZSL). In GZSL, instances from both seen and unseen classes are present at test time,

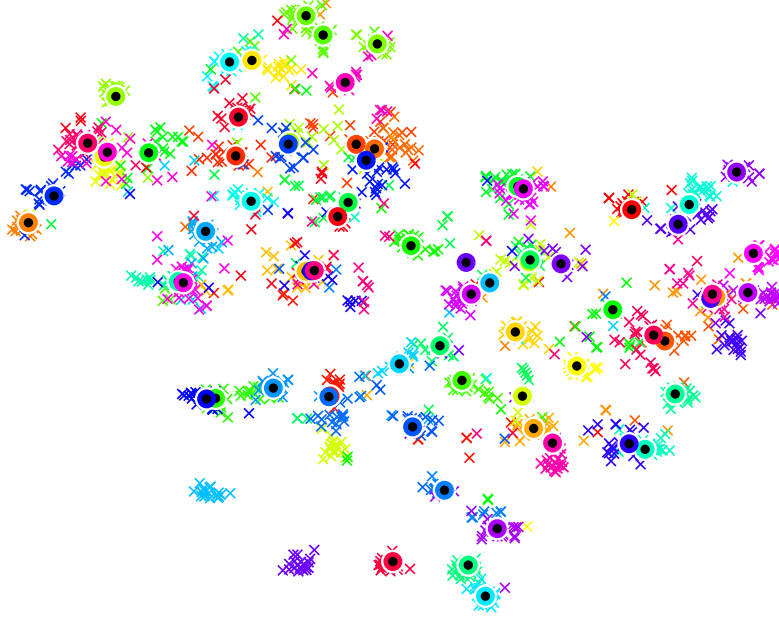


Figure 3: t-SNE [27] visualization of real images (crosses) and predicted visual exemplars (circles) for 72 *unseen* classes on **SUN** (first split). Different colors of symbols denote different unseen classes. Perfect predictions of visual features/exemplars would result in well-aligned crosses and circles of the same color. Best viewed in color.

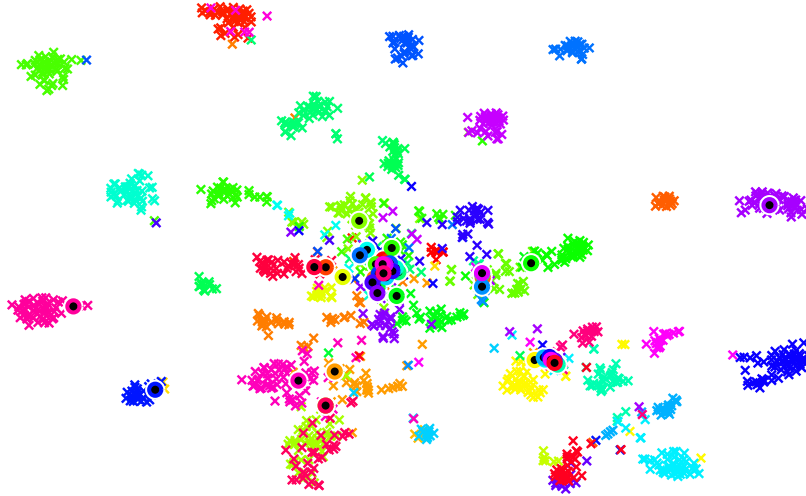


Figure 4: t-SNE [27] visualization of real images (crosses) and predicted visual exemplars (circles) for 48 randomly sampled *unseen* classes from 2-hop on **ImageNet**. Different colors of symbols denote different unseen classes. Perfect predictions of visual features/exemplars would result in well-aligned crosses and circles of the same color. Best viewed in color.

and the label space is the union of both types of classes. We refer the reader for more discussions regarding GZSL and related settings in [5, 30].

We evaluate our methods and baselines using the Area Under Seen-Unseen accuracy Curve (AUSUC) [5] and report the results in Table 5. Following the same evaluation procedure as before, our approach again outperforms the

baselines on all datasets.

Recently, Xian et al. [30] proposes to unify the evaluation protocol in terms of image features, class semantic embeddings, data splits, and evaluation criteria for conventional and generalized zero-shot learning. In their protocol, GZSL is evaluated by the harmonic mean of seen and unseen classes’ accuracies. Technically, AUSUC provides a

Class name	Top 3 attributes	Nearest image to predicted exemplar	Class name	Top 3 attributes	Nearest image to predicted exemplar
Persian cat	Furry, domestic, paws		Airport	open area, manmade, far-away horizon	
Humpback whale	big, water, swims		Art studio	enclosed area, no horizon, manmade	
Leopard	fast, meat, stalker		Botanical garden	foliage, vegetation, trees	
Pig	quadrupedal, bulbous, newworld		Clothing store	cloth, enclosed area, shopping	

Figure 5: Qualitative zero-shot learning results on **AwA** (left) and **SUN** (right). For each row, we provide a class name, three attributes with the highest strength, and the nearest image to the predicted exemplar (projected back to the original visual feature space).

Table 4: ZSL results in the per-class multi-way classification accuracies (in %) on **AwA** using word vectors as semantic representations. We use the 1,000-dimensional word vectors in [10]. All approaches use GoogLeNet as the visual features.

Approach	AwA
SYNCO-VS-O [4]	57.5
EXEM (SYNCO-VS-O)	61.7
EXEM (1NN)	63.5

Table 5: Generalized ZSL results in Area Under Seen-Unseen accuracy Curve (AUSUC) [5] on **AwA**, **CUB**, and **SUN**. For each dataset, we mark the best in red and the second best in blue. All approaches use GoogLeNet as the visual features and calibrated stacking [5] to combine the scores for seen and unseen classes.

Approach	AwA	CUB	SUN
DAP [†] [17]	0.366	0.194	0.096
IAP [†] [17]	0.394	0.199	0.145
CONSE [†] [20]	0.428	0.212	0.200
ESZSL [†] [22]	0.449	0.243	0.026
SYNCO-VS-O [†] [4]	0.568	0.336	0.242
SYNCTRICT [†] [4]	0.583	0.356	0.260
EXEM (SYNCO-VS-O)	0.553	0.365	0.265
EXEM (SYNCTRICT)	0.587	0.397	0.288
EXEM (1NN)	0.570	0.318	0.284
EXEM (1NNs)	0.584	0.373	0.287

[†]: results are reported in [5].

more complete picture of zero-shot learning method’s performance, but it is less simpler than the harmonic mean. As in Sect. B.2, further investigation under this newly proposed evaluation protocol (both in conventional and generalized zero-shot learning) is left for future work.

F. Additional details on zero-shot to few-shot learning experiments

F.1. Details on how to select peeked unseen classes

Denote by B the number of peeked unseen classes whose labeled data will be revealed. In what follows, we provide detailed descriptions of how we select a subset of peeked unseen classes of size B .

Uniform random and heavy (light)-toward-seen random

As mentioned in Sect. 3.1 of the main text, there are different subsets of unseen classes on **ImageNet** according to the WordNet hierarchy. Each subset contains unseen classes with a certain range of tree-hop distance from the 1K seen classes. The smaller the distance is, the higher the semantic similarity between unseen classes and seen classes. Here, we consider the following three *disjoint* subsets:

- *2-hop*: 1,509 (out of 1,549) unseen classes that are within 2 tree-hop distance from the 1K seen classes.
- *Pure 3-hop*: 6,169 (out of 6,311) unseen classes that are with exactly 3 tree-hop distance from the 1K seen classes.
- *Rest*: 12,667 (out of 12,982) unseen classes that are with more than 3 tree-hop distance from the 1K seen classes.

Note that *3-hop* defined in Sect. 3.1 of the main text is exactly $2\text{-hop} \cup \text{Pure } 3\text{-hop}$, and *All* is $2\text{-hop} \cup \text{Pure } 3\text{-hop} \cup \text{Rest}$.

For uniform random, we pick from *2-hop*/*Pure 3-hop*/*Rest* the number of peeked unseen classes proportional to their set size (i.e., 1,509/6,169/12,667). That

is, we do not bias the selected classes towards any subset. For heavy-toward-seen random, we pick from *2-hop/Pure 3-hop/Rest* the number of peeked unseen classes proportional to $(16 \times 1,509)/(4 \times 6,169)/(1 \times 12,667)$. For light-toward-seen random, we pick from *2-hop/Pure 3-hop/Rest* the number of peeked unseen classes proportional to $(1 \times 1,509)/(4 \times 6,169)/(16 \times 12,667)$. Given the number of peeked unseen classes for each subset, we then perform uniform sampling (without replacement) within each subset to select the peeked unseen classes. If the number of peeked unseen classes to select from a subset exceeds the number of classes of that subset, we split the exceeding budget to other subsets following the proportion.

DPP Given a ground set of N items (e.g., classes) and the corresponding N -by- N kernel matrix $\mathbf{L}$ that encodes the pair-wise item similarity, a DPP [16] defines the probability of any subset sampled from the ground set. The probability of a specific subset is proportional to the determinant of the principal minor of $\mathbf{L}$ indexed by the subset. A diverse subset is thus with a higher probability to be sampled.

For zero-shot to few-shot learning experiments, we construct $\mathbf{L}$ with the RBF kernel computed on semantic representations (e.g, word vectors) of all the seen and unseen classes (i.e., $S + U$ classes). We then compute the U -by- U kernel matrix $\mathbf{L}_U$ conditional on that all the S seen classes are already included in the subset. Please refer to [16] for details on conditioning in DPPs. With $\mathbf{L}_U$, we would like to select additional B classes that are diverse from each other and from the seen classes to be the peeked unseen classes.

Since finding the most diverse subset (either fixed-size or not) is an NP-hard problem [15, 16], we apply a simple greedy algorithm to sequentially select classes. Denote Q_t as the set of peeked unseen classes with size t and U_t as the remaining unseen classes, we enumerate all possible subset of size $t + 1$ (i.e., $Q_t \cup \{c \in U_t\}$). We then include c^* that leads to the largest probability into Q_{t+1} (i.e., $Q_{t+1} = Q_t \cup \{c^*\}$ and $U_{t+1} = U_t - \{c^*\}$). We iteratively perform the update until $t = B$.

F.2. Additional results on zero-shot to few-shot learning results

In this section, we analyze experimental results for EXEM (1NN) in detail. We refer the reader to the setup described in Sect. 3.3.3 of the main text, as well as additional setup below.

F.2.1 Additional setup

We will consider several fine-grained evaluation metrics. We denote by $\mathcal{A}_{\mathcal{X} \rightarrow \mathcal{Y}}^K$ the Flat Hit@K of classifying test instances from $\mathcal{X}$ to the label space of $\mathcal{Y}$. Since there are

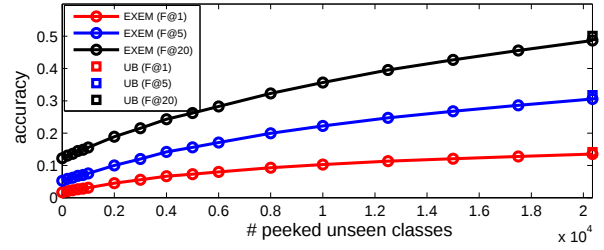


Figure 6: Combined accuracy $\mathcal{A}_{\mathcal{U} \rightarrow \mathcal{U}}^K$ vs. the number of peeked unseen classes for EXEM (1NN). The “squares” correspond to the **upperbound (UB)** obtained by EXEM (1NN) on *real* exemplars. $F@K=1, 5$, and 20.

two types of test data, we will have two types of accuracy: the peeked unseen accuracy $\mathcal{A}_{\mathcal{P} \rightarrow \mathcal{U}}^K$ and the remaining unseen accuracy $\mathcal{A}_{\mathcal{R} \rightarrow \mathcal{U}}^K$, where $\mathcal{P}$ is the peeked unseen set, $\mathcal{R}$ is the remaining unseen set, and $\mathcal{U} = \mathcal{P} \cup \mathcal{R}$ is the unseen set. Then, the combined accuracy $\mathcal{A}_{\mathcal{U} \rightarrow \mathcal{U}}^K = w_{\mathcal{P}} \mathcal{A}_{\mathcal{P} \rightarrow \mathcal{U}}^K + w_{\mathcal{R}} \mathcal{A}_{\mathcal{R} \rightarrow \mathcal{U}}^K$, where $w_{\mathcal{P}}$ ($w_{\mathcal{R}}$) is the proportion of test instances in peeked unseen (remaining unseen) classes to the total number of test instances. Note that the combined accuracy is the one we use in the main text.

Note also that we follow the evaluation protocol for **ImageNet** in previous literature by using “per-image” accuracy. We will also explore “per-class” accuracy and show that we reach the same conclusion. See Sect F.2.4.

F.2.2 Full curves for EXEM (1NN)

Fig. 6 shows $\mathcal{A}_{\mathcal{U} \rightarrow \mathcal{U}}^K$ when the number of peeked unseen classes keeps increasing. We observe this leads to improved overall accuracy, although the gain eventually is flat. We also show the upperbound: EXEM (1NN) with *real exemplars* instead of predicted ones for all the unseen classes. Though small, the gap to the upperbound could potentially be improved with a more accurate prediction method of visual exemplars, in comparison to SVR (Sect. A).

F.2.3 Detailed analysis of the effect of labeled data from peeked unseen classes

Fig. 7 expands the results in Fig. 6 by providing the weighed peeked unseen accuracy $w_{\mathcal{P}} \mathcal{A}_{\mathcal{P} \rightarrow \mathcal{U}}^K$ and the weighed remaining unseen accuracy $w_{\mathcal{R}} \mathcal{A}_{\mathcal{R} \rightarrow \mathcal{U}}^K$. We note that, as the number of peeked unseen classes increases, $w_{\mathcal{P}}$ goes up while $w_{\mathcal{R}}$ goes down, roughly linearly in both cases. Thus, the curves go up for the top row and go down for the bottom row.

As we observe additional labeled data from more peeked unseen classes, the weighed peeked unseen accuracy improves roughly linearly as well. On the other hand, the weighed remaining unseen accuracy degrades very quickly

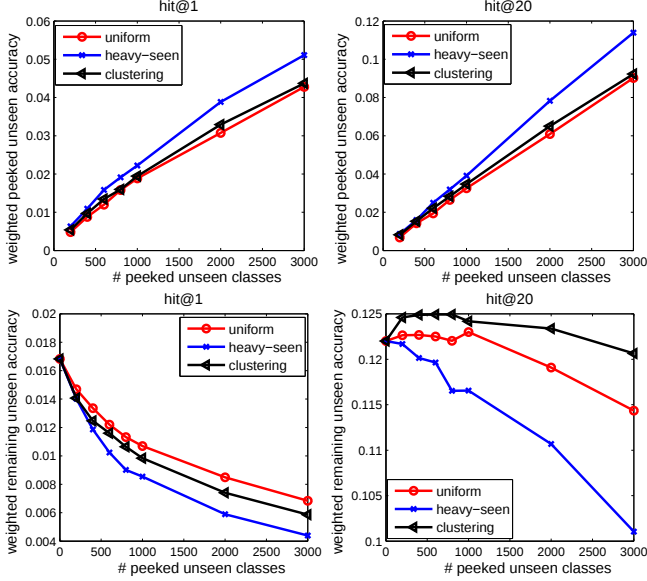


Figure 7: (Top) Weighted **peeked unseen** accuracy $w_{\mathcal{P}} \mathcal{A}_{\mathcal{P} \rightarrow \mathcal{U}}^K$ vs. the number of peeked unseen classes. (Bottom) Weighted **remaining unseen** $w_{\mathcal{R}} \mathcal{A}_{\mathcal{R} \rightarrow \mathcal{U}}^K$ accuracy vs. the number of peeked unseen classes. The weight $w_{\mathcal{P}}$ ($w_{\mathcal{R}}$) is the number of test instances belonging to $\mathcal{P}$ ($\mathcal{R}$) divided by the total number of test instances. The evaluation metrics are F@1 (left) and F@20 (right).

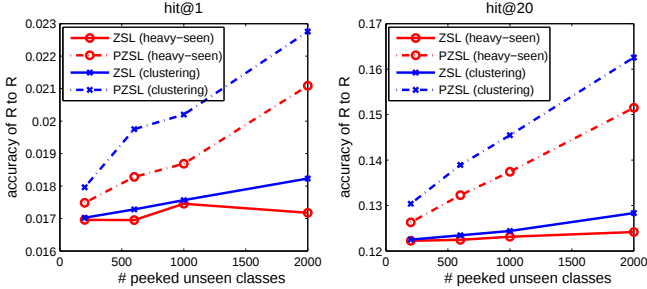


Figure 8: Accuracy on test instances from the **remaining unseen classes when classifying into the label space of remaining unseen classes only** $\mathcal{A}_{\mathcal{R} \rightarrow \mathcal{R}}^K$ vs. the number of peeked unseen classes. ZSL trains on labeled data from the seen classes only while PZSL (ZSL with peeked unseen classes) trains on the the labeled data from both seen and peeked unseen classes. The evaluation metrics are F@1 (left) and F@20 (right).

for F@1 but slower for F@20. This suggests that the improvement we see (over ZSL performance) in Fig. 4 of the main text and Fig. 6 is contributed largely by the fact that peeked unseen classes benefit themselves. *But how do peeked unseen classes exactly affect the remaining unseen classes?*

The above question is tricky to answer. There are two main factors that contribute to the performance on remaining unseen test instances. The first factor is the confusion among remaining classes themselves, and the second one

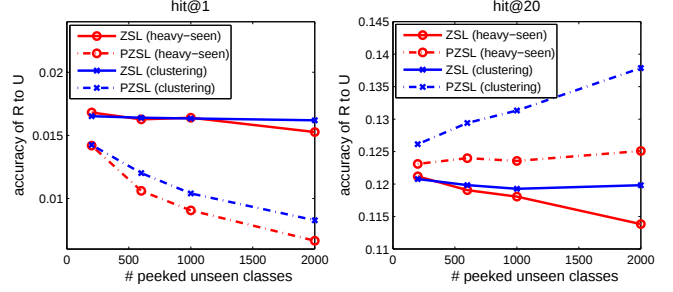


Figure 9: Accuracy on test instances from the **remaining unseen classes when classifying into the label space of unseen classes** $\mathcal{A}_{\mathcal{R} \rightarrow \mathcal{U}}^K$ vs. the number of peeked unseen classes. ZSL trains on labeled data from the seen classes only while PZSL (ZSL with peeked unseen classes) trains on the the labeled data from both seen and peeked unseen classes. Note that these plots are the un-weighted version of those at the bottom row of Fig. 7. The evaluation metrics are F@1 (left) and F@20 (right).

is the confusion with peeked unseen classes. We perform more analysis to understand the effect of each factor when classifying test instances from the remaining unseen set $\mathcal{R}$.

To remove the confusion with peeked unseen classes, we first restrict the label space to only the remaining unseen classes $\mathcal{R}$. In particular, we consider $\mathcal{A}_{\mathcal{R} \rightarrow \mathcal{R}}^K$ and compare the method in two settings: ZSL and PZSL (ZSL with peeked unseen classes). ZSL uses only the training data from seen classes while PZSL uses the training data from both seen and peeked unseen classes. In Fig. 8, we see that adding labeled data from peeked unseen classes *does help* by resolving confusion among remaining unseen classes *themselves*, suggesting that peeked unseen classes inform other remaining unseen classes about visual information.

In Fig. 9, we add the confusion introduced by peeked unseen classes back by letting the label space consist of both $\mathcal{P}$ and $\mathcal{R}$. That is, we consider $\mathcal{A}_{\mathcal{R} \rightarrow \mathcal{U}}^K$. For Flat Hit@1, the accuracy is hurt so badly that it goes down below ZSL baselines. However, for Flat Hit@20, the accuracy drops but still higher than ZSL baselines.

Thus, to summarize, adding peeked unseen classes has two benefits: it improves the accuracies on the peeked unseen classes $\mathcal{P}$ (Fig. 7 (Top)), as well as reduces the confusion among remaining unseen classes $\mathcal{R}$ themselves (Fig. 8). It biases the resulting classifiers towards the peeked unseen classes, hence causing confusion between $\mathcal{P}$ and $\mathcal{R}$ (Fig. 9). When the pros outweigh the cons, we observe overall improvement (Fig. 6). Additionally, when we use less strict metrics, peeked unseen classes always help (Fig. 9).

F.2.4 Results on additional metric, additional method, and additional rounds

We further provide experimental results on additional metric: per-class accuracy and on multiple values of K in Flat

Hit @K (i.e., $K \in \{1, 2, 5, 10, 20\}$); additional method: EXEM (1NNs). We also provide results for EXEM (1NN) averaged over multiple rounds using heavy-toward-seen random, light-toward-seen random, and uniform random to select peeked unseen classes to illustrate the stability of these methods.

Fig. 10 and 11 summarize the results for per-image and per-class accuracy, respectively. For both figures, each row corresponds to a ZSL method and each column corresponds to a specific value of K in Flat Hit@K. In particular, from top to bottom, ZSL methods are EXEM (1NN), EXEM (1NNs), and SYNC^{O-vs-O}. From left to right, Flat Hit@K = 1, 2, 5, 10, and 20.

Where to peek? No matter which type of accuracy is considered, we observe similar trends previously seen in the main text. *heavy-toward-seen* is preferable for strict metrics (i.e., small K) while *clustering* is preferable for flexible metrics (i.e., large K), for all zero-shot learning algorithms.

Comparison between ZSL methods No matter which type of accuracy is considered, we observe similar trends seen in the main text. EXEM (1NNs), under the same Flat Hit@K and subset selection method and number of peeked unseen classes, slightly outperforms EXEM (1NN). Both EXEM (1NN) and EXEM (1NNs) outperform SYNC^{O-vs-O}.

Per-class accuracy vs. Per-image accuracy Per-class accuracy is generally lower than per-image accuracy. This can be attributed to two factors. First, the average number of instances per class in *2-hop* is larger than that in *Pure 3-hop* and *Rest* (see Sect. F.1 for the definition)³. Second, the per-class accuracy in *2-hop* is higher than that in *Pure 3-hop* and *Rest*⁴. That is, when we compute the per-image accuracy, we emphasize the accuracy from *2-hop*. The first factor indicates the long-tail phenomena in **ImageNet**, and the second factor indicates the nature of zero-shot learning — unseen classes that are semantically more similar to the seen ones perform better than those that are less similar.

Stability of peeked unseen class random selection For all experimental results on PZSL above and in the main text, we use a single round of randomness for heavy-toward-seen, light-toward-seen, and uniform random (see Sect. F.1 for details). That is, given budget B , we apply a fixed set of peeked unseen classes sampled according to a particular random strategy to all ZSL methods. To illustrate the stability of these random methods, we consider EXEM (1NN)

³On average, *2-hop* has 696 instances/class, *Pure 3-hop* has 584 instances/class, and *Rest* has 452 instances/class

⁴For example, the per-class accuracy of EXEM (1NN) in *2-hop/Pure 3-hop/Rest* is 12.1/3.0/0.8 (%) at Flat Hit@1 under 1,000 peeked unseen classes selected by heavy-toward-seen random.

Table 6: Comparison on using predicted and real exemplars for the peeked classes for few-shot learning (in %). EXEM (1NN) with heavy-toward-seen random for peeking 1,000 classes is used.

Exemplar type	Flat Hit@K				
	1	2	5	10	20
Predicted	3.1	4.6	7.5	10.8	15.6
Real	3.1	4.6	7.5	10.8	15.6

with 10 rounds of randomness and provide the mean and standard deviation of accuracy in Fig. 12.

The results follow the same trends as shown in Fig. 10 and Fig. 4 of the main text; the standard deviation is small when compared to the gap among different random selection methods. This observation suggests that, for random subset selection, the distribution of peeked unseen classes is more important than specific peeked unseen classes being selected.

Real or predicted exemplars for peeked classes What should we use as visual exemplars for peeked classes? There are two options. The first option is to use their real exemplars based on (a few) peeked instances. The second option is to use their predicted visual exemplars (where real exemplars are used to learn the predictor). As the training error of the (non-linear) regressor is quite low, we observe an unnoticeable difference in zero-shot performance between the two options. For instance, in Table 6, the two set of numbers match when the number of peeked classes is 1,000.

G. Additional analysis on dimension for PCA

To better understand a trade-off between running time and ZSL performance, we expand Table 7 of the main text with more values for projected PCA dimensions d . In Table 7, we see that our approach is extremely robust. With $d=50$, it still outperforms all baselines (cf. Table 3) on **AwA** and **SUN**; with $d=100$, on all 3 datasets. Moreover, our method works reasonably over a wide range of (large enough) d on all datasets.

H. Details on multi-layer perceptron

We follow the notations defined at the beginning of Sect. 2 and Sect. 2.1 of the main text. Similar to [32], our multi-layer perceptron is of the form:

$$\frac{1}{S} \sum_{c=1}^S \|v_c - W_2 \cdot \text{ReLU}(W_1 \cdot a_c)\|_2^2 + \lambda \cdot R(W_1, W_2), \quad (2)$$

where R denotes the ℓ_2 regularization, S is the number of seen classes, v_c is the visual exemplar of class c , a_c is the semantic representation of class c , and the weights W_1 and W_2 are parameters to be optimized.

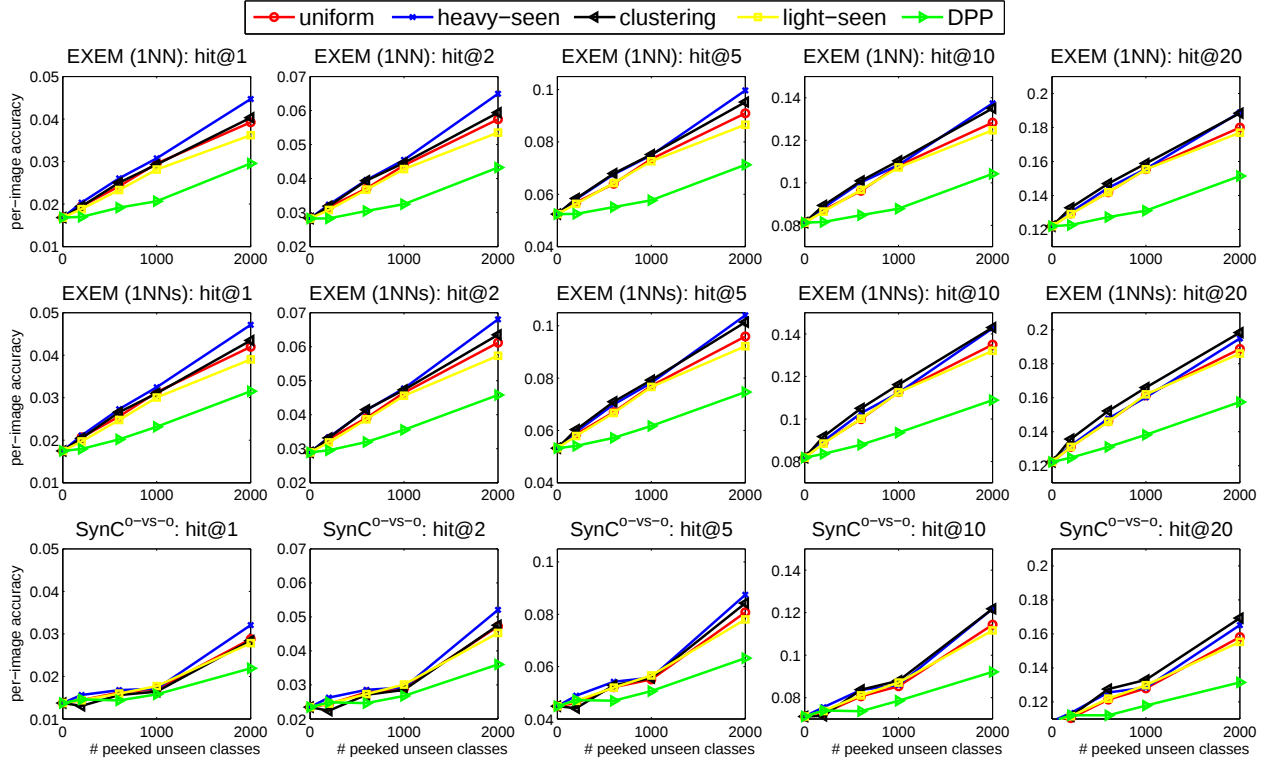


Figure 10: Combined per-image accuracy vs. the number of peeked unseen classes for EXEM (1NN), EXEM (1NNs), and SYNC. The evaluation metrics are, from left to right, Flat Hit@1, 2, 5, 10, and 20. Five subset selection approaches are considered.

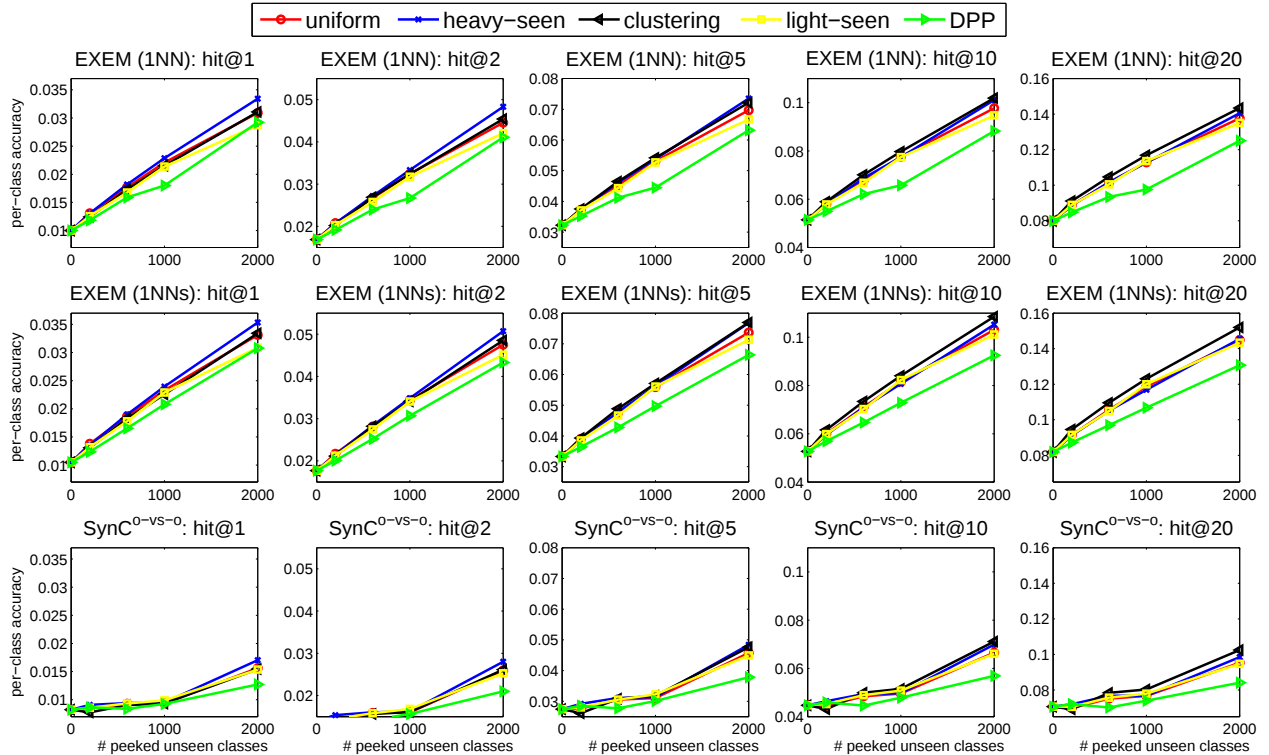


Figure 11: Combined per-class accuracy vs. the number of peeked unseen classes for EXEM (1NN), EXEM (1NNs), and SYNC. The evaluation metrics are, from left to right, Flat Hit@1, 2, 5, 10, and 20. Five subset selection approaches are considered.

Table 7: Accuracy of EXEM (1NN) on **AwA**, **CUB**, and **SUN** when predicted exemplars are from original visual features (No PCA) and PCA-projected features (PCA with $d = 1024, 500, 200, 100, 50, 10$).

Dataset name	No PCA $d = 1024$	PCA $d = 1024$	PCA $d = 500$	PCA $d = 200$	PCA $d = 100$	PCA $d = 50$	PCA $d = 10$
AwA	77.8	76.2	76.2	76.0	75.8	76.5	73.4
CUB	55.1	56.3	56.3	58.2	54.7	54.1	38.4
SUN	69.2	69.6	69.6	69.6	69.3	68.3	55.3

Following [32], we randomly initialize the weights W_1 and W_2 , and set the number of hidden units for **AwA** and **CUB** to be 300 and 700, respectively. We use Adam optimizer with a learning rate 0.0001 and minibatch size of 5. We tune λ on the same splits of data as in other experiments with class-wise CV (Sect. B.3). Our code is implemented in TensorFlow [1].

References

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 12
- [2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for attribute-based classification. In *CVPR*, 2013. 2
- [3] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, 2015. 2, 3, 4
- [4] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Synthesized classifiers for zero-shot learning. In *CVPR*, 2016. 1, 2, 3, 4, 7
- [5] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *ECCV*, 2016. 6, 7
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 2
- [7] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *ICCV*, 2013. 2
- [8] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 2
- [9] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 1, 2, 4
- [10] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong. Transductive multi-view zero-shot learning. *TPAMI*, 2015. 3, 7
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2
- [12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3, 4
- [13] D. Jayaraman, F. Sha, and K. Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *CVPR*, 2014. 3
- [14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3, 4
- [15] A. Kulesza and B. Taskar. k-dpps: Fixed-size determinantal point processes. In *ICML*, 2011. 8
- [16] A. Kulesza and B. Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3), 2012. 8
- [17] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *TPAMI*, 36(3):453–465, 2014. 1, 2, 7
- [18] Y. Lu. Unsupervised learning of neural network outputs: with application in zero-shot learning. In *IJCAI*, 2016. 2
- [19] T. Mensink, E. Gavves, and C. G. Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *CVPR*, 2014. 2, 3, 4
- [20] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014. 1, 2, 4, 7
- [21] G. Patterson, C. Xu, H. Su, and J. Hays. The sun attribute database: Beyond categories for deeper scene understanding. *IJCV*, 108(1-2):59–81, 2014. 1
- [22] B. Romera-Paredes and P. H. S. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, 2015. 2, 3, 4, 7
- [23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 1
- [24] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000. 1
- [25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3, 4
- [26] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013. 2
- [27] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(2579-2605):85, 2008. 3, 5, 6
- [28] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 1
- [29] Q. Wang and K. Chen. Zero-shot visual recognition via bidirectional latent embedding. *arXiv preprint arXiv:1607.02104*, 2016. 2, 3, 4

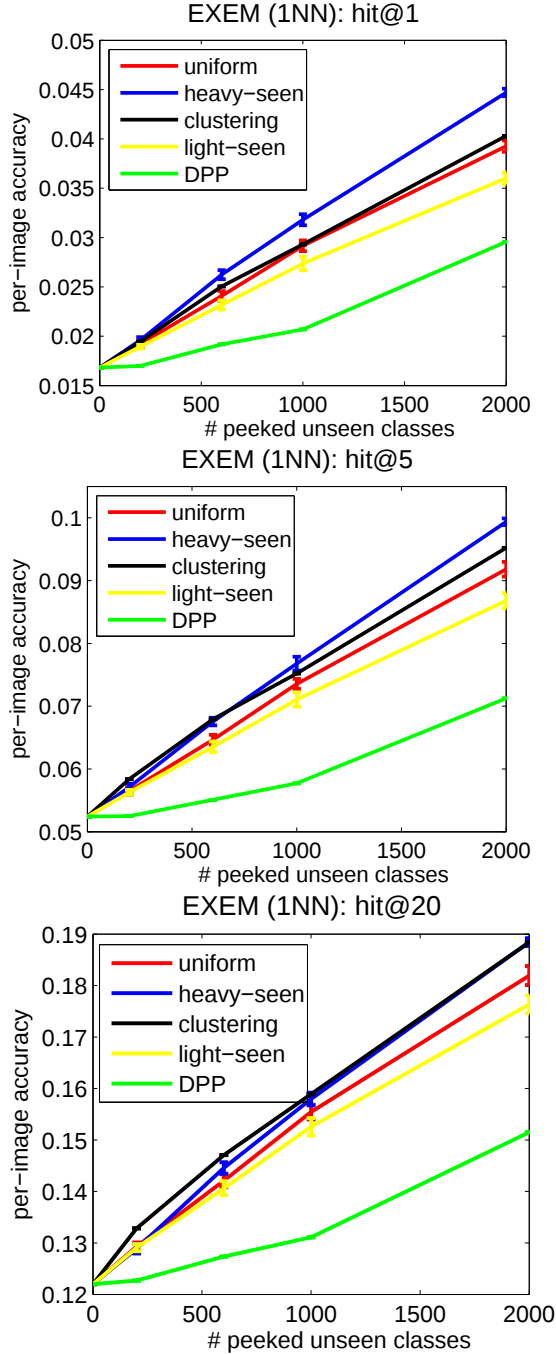


Figure 12: Stability of random subset selection strategies. Combined per-image accuracy vs. the number of peeked unseen classes for EXEM (1NN). Five subset selection approaches are compared. The results of heavy-toward-seen, light-toward-seen, and uniform random are **averaged over 10 random rounds** along with the error bars showing the standard deviation. The evaluation metrics are, from top to bottom, Flat Hit@1, 5, and 20.

B. Schiele. Latent embeddings for zero-shot classification. In *CVPR*, 2016. 2, 4

[32] L. Zhang, T. Xiang, and S. Gong. Learning a deep embedding model for zero-shot learning. In *CVPR*, 2017. 3, 4, 10, 12

[33] Z. Zhang and V. Saligrama. Zero-shot learning via semantic similarity embedding. In *ICCV*, 2015. 2, 3, 4

[34] Z. Zhang and V. Saligrama. Zero-shot learning via joint latent similarity embedding. In *CVPR*, 2016. 3, 4

[30] Y. Xian, Z. Akata, and B. Schiele. Zero-shot learning – the Good, the Bad and the Ugly. In *CVPR*, 2017. 2, 3, 6

[31] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and