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

Wei-Lun Chao*[1], Soravit Changpinyo*[1], Boqing Gong[2], and Fei Sha[3]

[1]Dept. of Computer Science, U. of Southern California, United States
[2]Center for Research in Computer Vision, U. of Central Florida, United States
[3]Dept. of Computer Science, U. of California, Los Angeles, United States
{weilunc, schangpi}@usc.edu, bgong@crcv.ucf.edu, feisha@cs.ucla.edu

This Supplementary Material provides the following additional details, results, and analysis (along with their corresponding sections of the main text).

- Section 1: Hyper-parameter tuning strategies (Section 4.2 of the main text).
- Section 2: Novelty detection approaches: details and additional results (Section 4.3 and 5.2 of the main text).
- Section 3: Comparison between zero-shot learning approaches: additional ZSL algorithm, dataset, and results (Section 5 of the main text).
- Section 4: Analysis on (generalized) zero-shot learning: details and additional results (Section 6 of the main text).

1 Hyper-parameter tuning strategies

1.1 Cross-validation with AUSUC

In Section 4.2 of the main text, we introduce the Area Under Seen-Unseen accuracy Curve (AUSUC), which is analogous to many metrics in computer vision and machine learning that balance two conflicting (sub)metrics, such as area under ROC. To tune the hyper-parameters based on this metric[1], 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

* Equal contributions

[1] AUSUC is computed by varying the γ factor within a range. If a single γ is desired, another measure such as "F-score" balancing $A_{\mathcal{U}\to\mathcal{T}}$ and $A_{\mathcal{S}\to\mathcal{T}}$ can be used. One can also assume a prior probability of whether any instance is seen or unseen to select the factor.

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.

1.2 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 $A_{\mathcal{S}\to\mathcal{S}}$ is optimized, can be used for the former. For the latter, it has been shown that the class-wise cross-validation technique [1,2,3], where the conventional zero-shot learning task is simulated, outperforms the standard technique [1]. In this case, $A_{\mathcal{U}\to\mathcal{U}}$ 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 ($A_{\mathcal{S}\to\mathcal{S}}$ and $A_{\mathcal{U}\to\mathcal{U}}$) and the other is based on AUSUC. See text for details.

Method	**AwA**		**CUB**	
	CV strategies		CV strategies	
	Accuracies	AUSUC	Accuracies	AUSUC
DAP [4]	0.341	**0.366**	**0.202**	0.194
IAP [4]	0.366	**0.394**	0.194	**0.199**
ConSE [5]	**0.443**	0.428	0.190	**0.212**
SynC$^{\text{o-vs-o}}$ [1]	0.539	**0.568**	0.324	**0.336**
SynC$^{\text{struct}}$ [1]	0.551	**0.583**	0.356	0.356

2 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 [6] — have been explored by [7]. In this section, we provide details and additional results on these approaches.

2.1 Algorithms

In [7], 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 vector 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 $X_{\mathcal{S}}$ be the set of all the mapped training examples from seen classes. For a test sample $\boldsymbol{x}$ (also mapped into the semantic space), a context set $C(\boldsymbol{x}) \subseteq X_{\mathcal{S}}$ of k nearest neighbors is first defined. The probabilistic set distance *pdist* from $\boldsymbol{x}$ to all the points in $C(\boldsymbol{x})$ is then computed as follows

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

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

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

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

$$LoOP(\boldsymbol{x}) = \max\left\{0, \operatorname{erf}\left(\frac{lof_{\lambda}(\boldsymbol{x})}{Z_{\lambda}(X_{\mathcal{S}})}\right)\right\}, \tag{3}$$

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

2.2 Implementation details

We use the code provided by Socher et al. [7] and follow their settings. In particular, we train a two-layer neural network with the same loss function as in [7] to learn a mapping from the visual feature space to the semantic embedding space. We tune the hyper-parameter λ (a multiplier on the standard deviation) in LoOP jointly with other hyper-parameters of zero-shot learning approaches — although we empirically observe that λ does not significantly affect the novelty detection rankings, consistent with the observations made by [6]. Following [7], we set the number of neighbors (from the seen classes' examples) k in LoOP to be 20.

2.3 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 1.2). 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 [7]).

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	**AwA**			**CUB**		
	Novelty detection [7]		Calibrated	Novelty detection [7]		Calibrated
	Gaussian	LoOP	Stacking	Gaussian	LoOP	Stacking
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$^{\text{o-vs-o}}$	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**

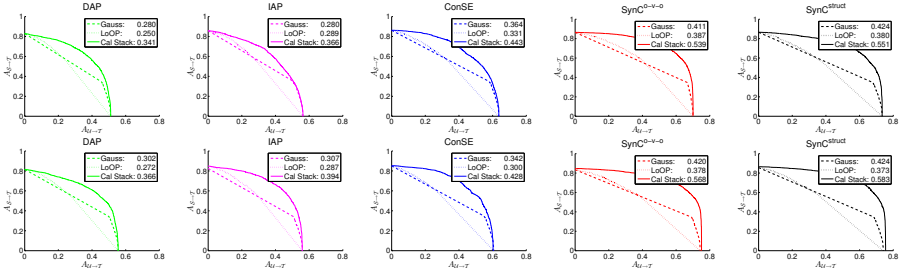

Fig. 1. Seen-Unseen accuracy Curves (SUC) for Gaussian (Gauss), LoOP, and calibrated stacking (Cal Stack) for all zero-shot learning approaches on **AwA**. Hyper-parameters are cross-validated based on accuracies (top) and AUSUC (bottom). Calibrated stacking outperforms both Gaussian and LoOP in all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

3 Comparison between zero-shot learning approaches: additional algorithm, dataset, and results

3.1 Additional results for Section 5.3 of the main text

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.

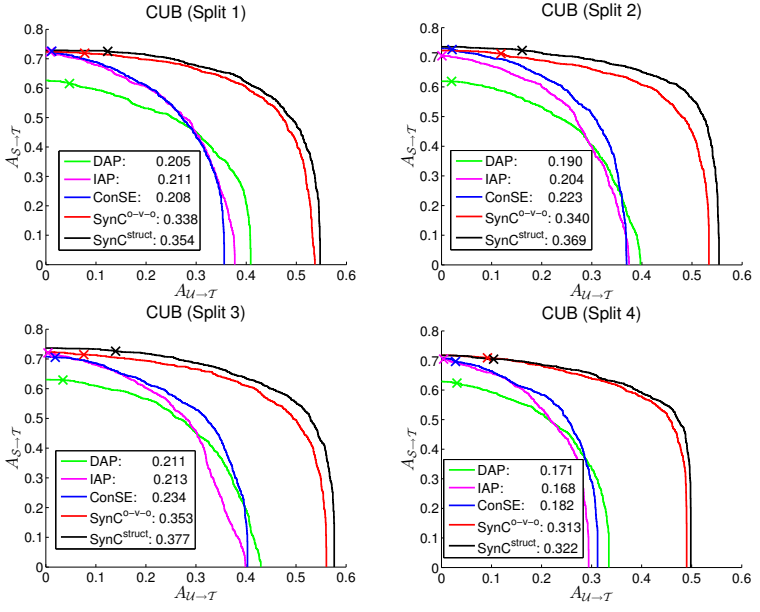

Fig. 2. Comparison of performance measured in AUSUC between different zero-shot learning approaches on the four splits of **CUB**.

3.2 Additional algorithm

We examine an additional zero-shot learning algorithm ESZSL [8] on the GZSL task. ESZSL learns a (linear) mapping from visual features to the semantic space (e.g., attributes or WORD2VEC) with well-designed regularization terms. The mapped feature vector of a test instance is compared to semantic representations of classes (e.g., via dot products) for classification. Contrast to DAP/IAP [4], and similar to [9,10], ESZSL can easily incorporate continuous semantic representations and directly optimize the classification performance on seen classes.

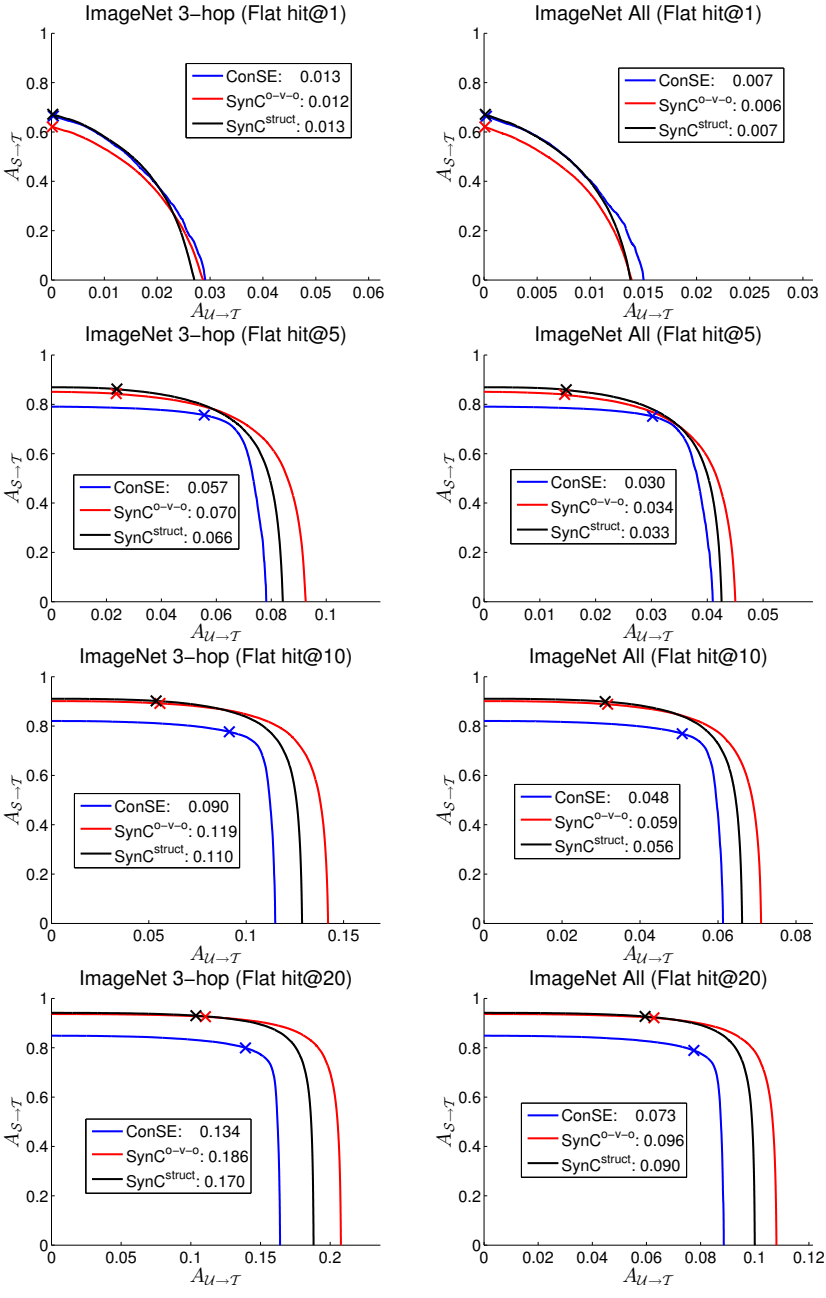

Fig. 3. Comparison of performance measured in AUSUC between different zero-shot learning approaches on **ImageNet-*3hop*** (left) and **ImageNet-*All*** (right).

Table 3 shows the results on **AwA** and **CUB**, with the hyper-parameters cross-validated to maximize AUSUC. Compared to other zero-shot learning algorithms (cf. Table 3 in the main text), ESZSL outperforms DAP/IAP/ConSE, but is still worse than both versions of SynC. Moreover, the proposed *calibrated stacking* clearly leads to better results than both novelty detection methods for ESZSL.

Table 3. Performance of ESZSL (measured in AUSUC) on **AwA** and **CUB**. Hyper-parameters are cross-validated to maximize AUSUC.

Method	**AwA**			**CUB**		
	Novelty detection [7]		Calibrated	Novelty detection [7]		Calibrated
	Gaussian	LoOP	Stacking	Gaussian	LoOP	Stacking
ESZSL [8]	0.358	0.331	0.449	0.146	0.166	0.243

3.3 Additional dataset

We further experiment on another benchmark dataset, the **SUN attribute dataset (SUN)** [11], which contains 14,340 images of 717 scene categories. We extract the 1,024 dimensional GoogLeNet features [12] pre-trained on the ILSVRC 2012 1K training set [13], and use the provided 102 continuous- and binary-valued attributes. Following [1,4], we split the dataset into 10 folds with disjoint classes (each with 71/72 classes), and in turn treat each fold as a test set of unseen classes. We then report the average results over 10 rounds. Similar to the setting on **AwA** and **CUB**, we hold out 20% of the data points from seen classes and merge them with those of the unseen classes to form the test set. The results are summarized in Table 4, where calibrated stacking outperforms Gaussian and LoOP for all ZSL algorithms. Besides, both versions of SynC still outperform the other ZSL algorithms.

Table 4. Performance measured in AUSUC of several methods for Generalized Zero-Shot Learning on **SUN**. Hyper-parameters are cross-validated to maximize AUSUC.

Method	**SUN**		
	Novelty detection [7]		Calibrated
	Gaussian	LoOP	Stacking
DAP	0.061	0.062	0.096
IAP	0.093	0.095	0.145
ConSE	0.120	0.122	0.200
ESZSL	0.017	0.018	0.026
$\text{SynC}^{\text{o-vs-o}}$	0.144	0.146	0.242
$\text{SynC}^{\text{struct}}$	0.151	0.153	0.260

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

4.1 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 20,345 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.

4.2 Additional results

We provide additional SUC plots on the performance of GZSL with idealized semantic embeddings **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/WORD2VEC and multi-class classifiers are reduced significantly. The effect of **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).

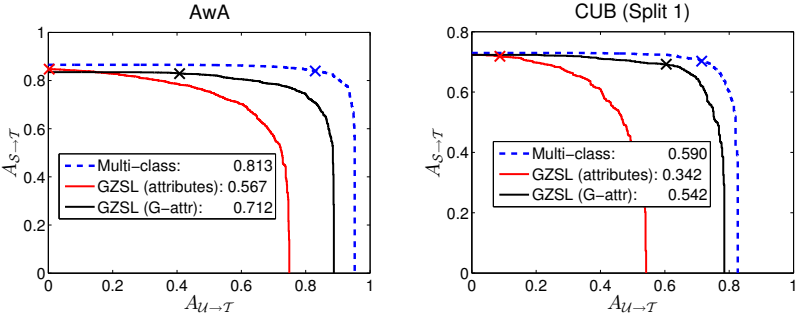

Fig. 4. Comparison between GZSL and multi-class classifiers trained with labeled data from both seen and unseen classes on the datasets **AwA** and **CUB**. GZSL uses visual attributes (in red color) or **G-attr** (in black color) as semantic embeddings.

Finally, we provide additional results on the analysis of how much labeled data needed to improve GZSL's performance on **AwA** and **CUB**. In Table 5, we see the same trend as in Table 5 and 6 of the main text; GZSL with **G-attr** quickly approaches the performance of multi-class classifiers and large improvements from GZSL with visual attributes are observed — even though these

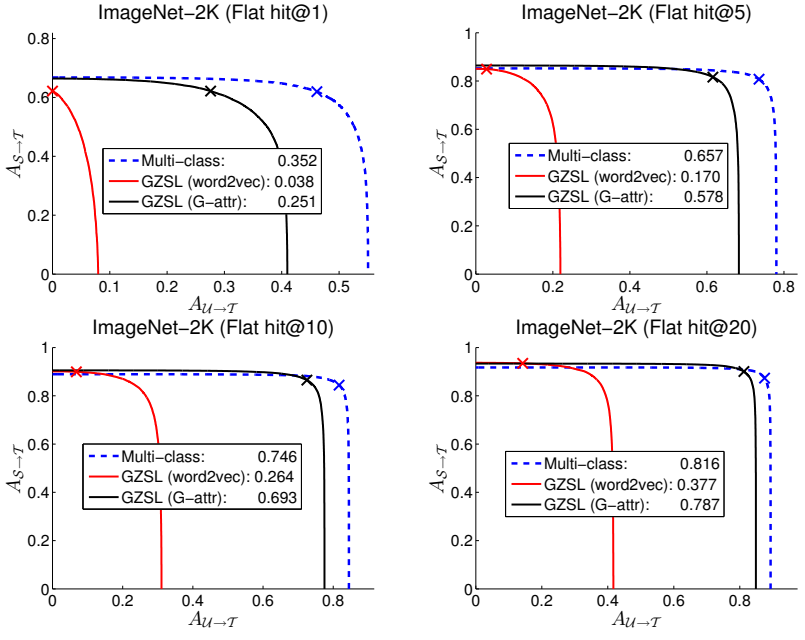

Fig. 5. Comparison between GZSL and multi-class classifiers trained with labeled data from both seen and unseen classes on the dataset **ImageNet-2K**. GZSL uses WORD2VEC (in red color) or **G-attr** (in black color) as semantic embeddings.

attributes are defined and annotated by human experts in this case. We also provide in Table 6 and 7 the full versions of Table 5 and 6 of the main text, with 2-shot and 5-shot results included.

Table 5. 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**
GZSL	VISUAL ATTRIBUTES	0.57	0.34
	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.

Table 6. 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
GZSL	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

References

1. Changpinyo, S., Chao, W.L., Gong, B., Sha, F.: Synthesized classifiers for zero-shot learning. In: CVPR. (2016) 2, 7

Table 7. 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

2. Zhang, Z., Saligrama, V.: Zero-shot learning via semantic similarity embedding. In: ICCV. (2015) 2
3. Elhoseiny, M., Saleh, B., Elgammal, A.: Write a classifier: Zero-shot learning using purely textual descriptions. In: ICCV. (2013) 2
4. Lampert, C.H., Nickisch, H., Harmeling, S.: Attribute-based classification for zero-shot visual object categorization. TPAMI **36**(3) (2014) 453–465 2, 5, 7
5. 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) 2
6. Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: LoOP: local outlier probabilities. In: CIKM. (2009) 2, 3
7. Socher, R., Ganjoo, M., Manning, C.D., Ng, A.: Zero-shot learning through cross-modal transfer. In: NIPS. (2013) 2, 3, 4, 7
8. Romera-Paredes, B., Torr, P.H.S.: An embarrassingly simple approach to zero-shot learning. In: ICML. (2015) 5, 7
9. Akata, Z., Perronnin, F., Harchaoui, Z., Schmid, C.: Label-embedding for attribute-based classification. In: CVPR. (2013) 5
10. Akata, Z., Reed, S., Walter, D., Lee, H., Schiele, B.: Evaluation of output embeddings for fine-grained image classification. In: CVPR. (2015) 5
11. Patterson, G., Xu, C., Su, H., Hays, J.: The sun attribute database: Beyond categories for deeper scene understanding. IJCV **108**(1-2) (2014) 59–81 7
12. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. (2015) 7
13. 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) 7