Supplementary material for "Field-Guide-Inspired Zero-Shot Learning"

1. Overview

In this supplementary material we look at some more results that could not be presented in the main paper. We show attributes queried by *Sibling-variance* on SUN and AWA2 in Sec. 2. Sec. 3 shows the performance of the two acquisition functions on AWA2 and SUN. In Sec. 4 we present results for when the interactive learner uses a single image for SUN and AWA2. In Sec. 5 we compare the performance of our method with TF-VAEGAN on the AWA2 and SUN. Sec. 6 shows the effect of not using a taxonomy in *Sibling-variance* for AWA2 and SUN. In Sec. 7 we show the learner's behavior when classes other than the annotators first choice are chosen for SUN and AWA2. Sec. 8 presents more t-SNE visualization examples of the learning progression for novel class descriptors on all three datasets. We also **strongly** encourage the reader to refer to the **supplementary video** for better visualizations of the t-SNE progression.

2. More Qualitative Evaluation

Figure 1, shows the attributes queried first by the *Sibling-variance* method for 2 supercategories of SUN and AWA2. It also shows the attributes picked by measuring variance over all the classes. For SUN, attributes like "enclosed/open area" or "man-made/natural" may help in disambiguating between very different classes, but within a supercategory they do not help. For example, for the superclass "Indoor sports and leisure", all the classes are closed and man-made. But attributes like "competing", "spectating" are more informative. Similarly for indoor workplaces, attributes like "using tools" and "studying/learning" are very informative. Similar patterns can be seen on AWA2. *Sibling-variance* asks for attributes informative within the superclass.

Figure 1. Attributes selected by *Sibling-variance* for a parent class in the taxonomy and the attributes selected by measuring variance over all classes (top) for SUN and AWA2.

3. Comparison of Acquisition Functions on AWA2 and CUB

Figure 2 shows the performance of the two attribute querying acquisition functions with the CADA-VAE model on AWA2 and SUN. Our acquisition functions perform significantly better than a random acquisition function, showing the value of our field-guide annotation.

While the results for fine-grained dataset such as SUN are similar to that on CUB, for AWA2 *Representation-change* performs better than *Sibling-variance* in the later stages. This might be because the AWA2 model is trained for fewer coarse-grained classes with thousands of images and has a better representation and understanding of changing representation.

4. *Image-based* Results for AWA2 and SUN

Figure 3 shows the performance of our approach when one image is given by the annotator along with the interactive attribute annotations for AWA2 and SUN. As for

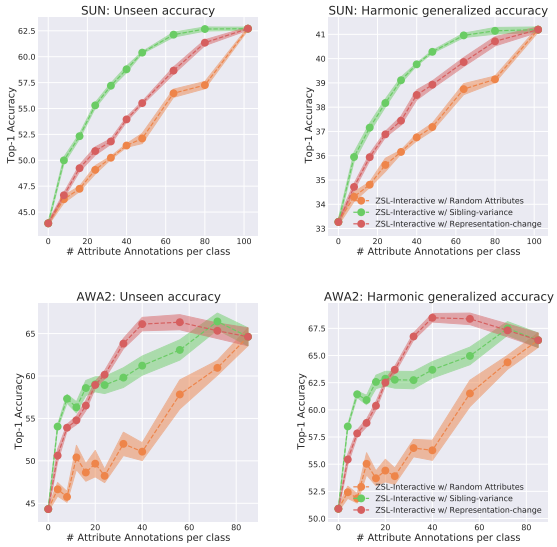

Figure 2. Performance of the two acquisition functions with CADA-VAE on AWA2 and SUN. Both functions perform better than the random acquisition function. On SUN, *Sibling-variance* performs better than *Representation-change*, but the latter does not require taxonomy information. Results on AWA2 are different, in the earlier stages *Sibling-variance* is better than *Representation-change*, but in the later stages *Representation-change* is better.

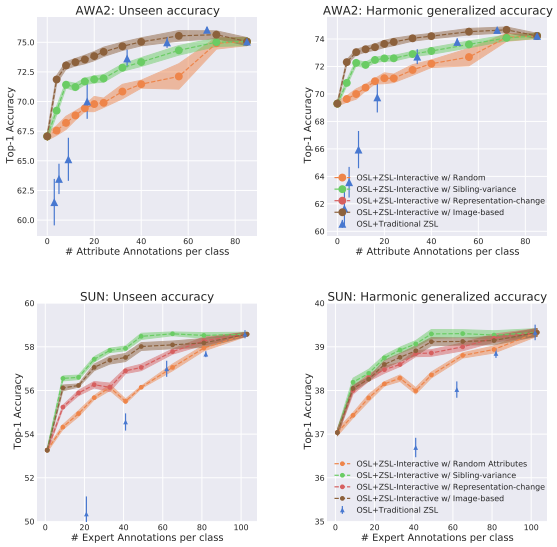

Figure 3. Performance of our method under the zero+one-shot setting when the annotator provides a single image for novel class along with the interactive attribute values for SUN and AWA2.

CUB, all the methods perform better than the baselines. For SUN, the *Image-based* function performs on par with *Sibling-variance* without requiring an additional taxonomy. For AWA2, the *Image-based* function performs better than all the methods we propose and the baselines. AWA2 has more training images and the classes are not very fine-grained. This might be the reason why *Image-based* acquisition functions work better for AWA2.

5. Performance of TF-VAEGAN on AWA2 and SUN

Figure 4 show the performance of our approach when the base model is TF-VAEGAN on SUN and AWA2. Our field-guide way of annotation works better then traditional ZSL baselines for both the dataset, proving the effectiveness and generalization of our method. For fine-grained classes such as SUN and CUB, our method is .

6. Performance on AWA2 and SUN Without Taxonomy

Figure 5 compares the method without taxonomy information against the model where the taxonomy is known. The results follow the conclusion from the main paper. When there is no taxonomy information available, this method loses performance because the local variation of a class cannot be measured, and hence those attributes can-

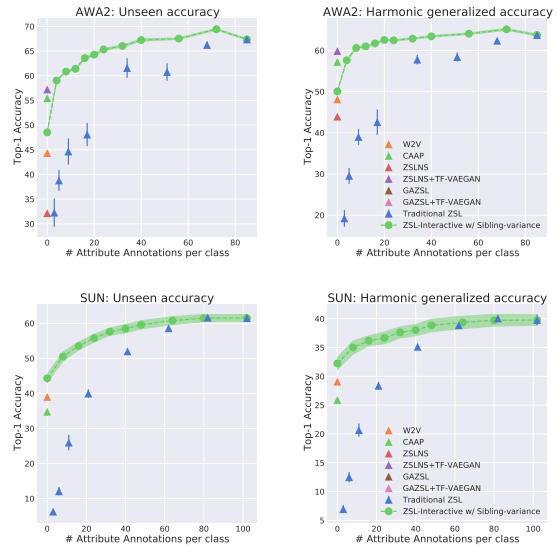

Figure 4. Comparison of our method against unsupervised and traditional ZSL baselines with TF-VAEGAN as the base model. Our method performs better than tradtional ZSL at the same attribute annotation cost for both AWA2 and SUN. Similar to results for CADA-VAE in the main paper, our method works better than the unsupervised baselines for SUN.

not be selected. But even without the taxonomy the method performs better than ZSL and is useful for cases when the taxonomy is not known or difficult to acquire.

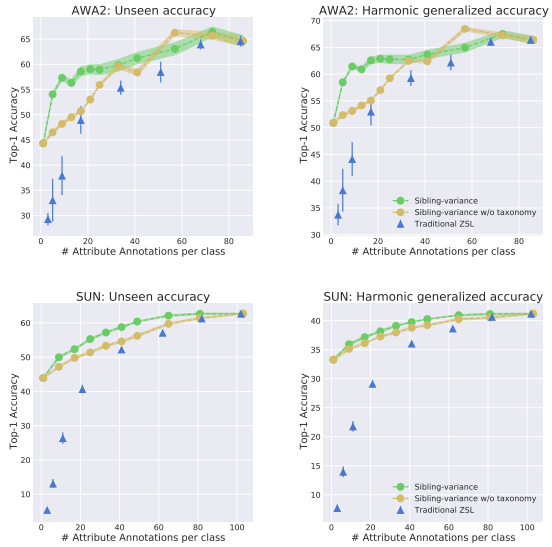

Figure 5. Comparing *Sibling-variance* against a variant where the taxonomy is unknown on AWA2 and SUN. The model loses accuracy if sibling classes are not used to measure *Sibling-variance*.

7. Effect on Performance When Changing Similar Base Class

The similar class given by the annotator is certainly more important than each attribute annotation. We look at the effect of choosing another class: either a random class from the full set, or a sibling of the expert selection that is closest to the expert selected sibling in word2vec embedding space.

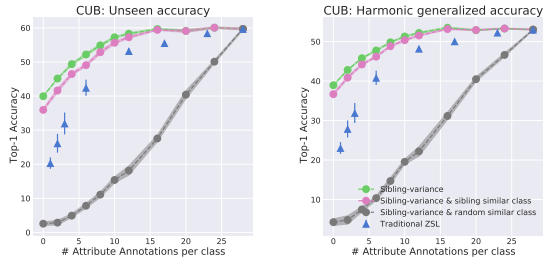

Figure 6. Comparing different variants for selecting the similar class $S(y)$ on CUB dataset. The model is not sensitive to the similar class as long as the selected class is not wildly different.

Figure 6 shows *Sibling-variance* with these annotations along with the ZSL baseline on CUB. The similar class chosen by annotators performs best. When we choose a class that is close to this (sibling), the method performs slightly worse. This shows that although our method performance is affected if a non-optimal nearest class is chosen, it is not very sensitive to it. Both these variants do significantly better than randomly selecting a similar class. This suggests that the interactive model will perform well as long as annotators do not provide a wildly different looking similar class.

Figure 7 shows the results for *Sibling-variance* on AWA2 and SUN. The similar class chosen by annotators performs similar to when a sibling base class is chosen for AWA2 and SUN. The performance is again not very sensitive to choosing the similar class as long as they are not very different.

Along with the sensitivity to the choice of the similar class, we also evaluated our method with sensitivety to attribute values. Note that incorrect or noisy attribute values will affect not just our proposed active ZSL but also the traditional non-interactive ZSL. With 10% noise in the novel attributes, when all attributes are provided, both our method and traditional ZSL see a $\sim 3\%$ drop in performance. With partial attribute annotations (5 per class), our proposed annotation strategy ($\sim 1\%$ drop) fares much better than traditional ZSL annotation ($\sim 5\%$ drop).

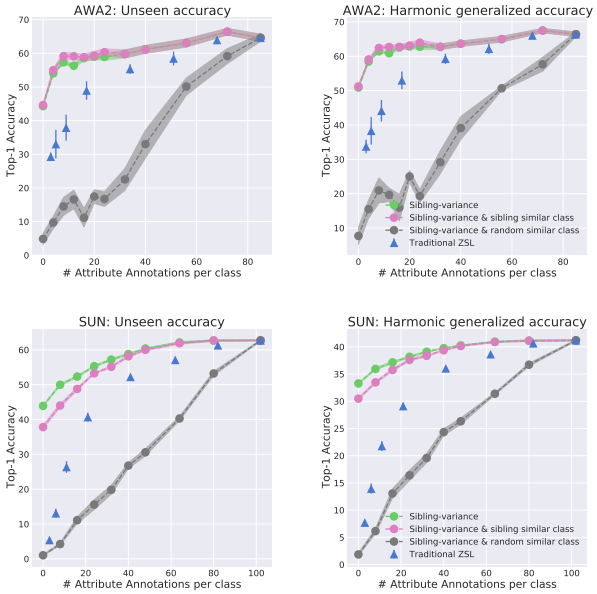

Figure 7. Comparing different variants for selecting the similar class $S(y)$ on AWA2 and SUN dataset. The model is not sensitive to the similar class as long as the selected class is not wildly different.

8. t-SNE Visualizations for More Classes and Dataset

Figure 8 show the progression of *Sibling-variance* and random attributes for all 10 AWA2 novel classes. Note that in the standard split of AWA2, the classes are split in a way that sometimes no good similar classes could be found. For example, both seal and walrus are in the test split and hence the annotators chose beaver and walrus as similar classes. Similarly no good base class is there for giraffe and bat so the annotators had to chose zebra and squirrel. Nonetheless

the faster progression towards novel classes' images and attributes can be seen for the classes when using *Sibling-variance* over random attributes.

Figure 9 and 10 show the progression of *Sibling-variance* and random attributes for all 20 novel classes of CUB (out of 50) and SUN (out of 70). Faster progression can be seen for *Sibling-variance* over these classes as well.

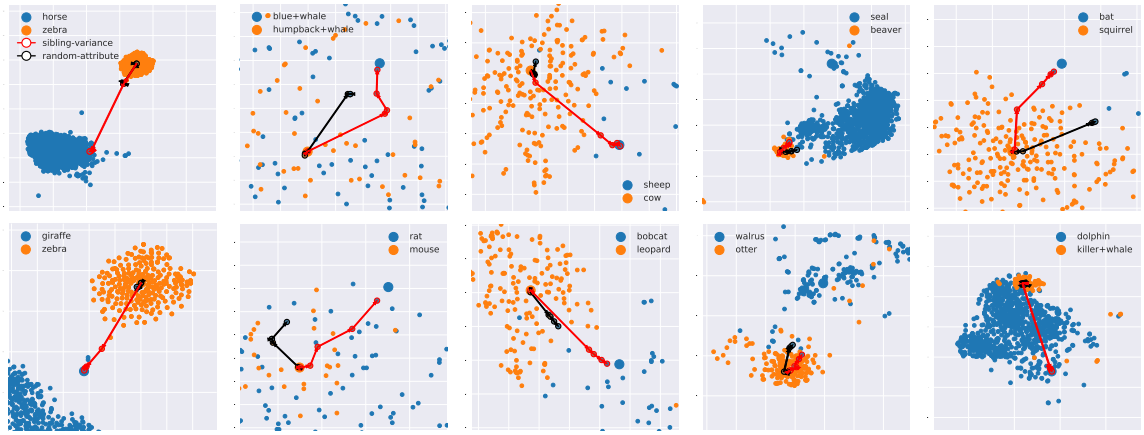

Figure 8. t-SNE visualizations. For all 10 AWA2 novel classes and corresponding similar base classes. Smaller dots represent test images and larger dots represent class attribute embeddings. Red edges show the progression of novel class attributes as learners interact using *Sibling-variance*. Dots with black edges show the progression with the *random* function. Both methods start at the base class attribute descriptor, and aim to reach to the novel class descriptor with as few interactions as possible. In most cases *Sibling-variance* reaches closer to the novel class descriptor quicker in contrast to *random*.

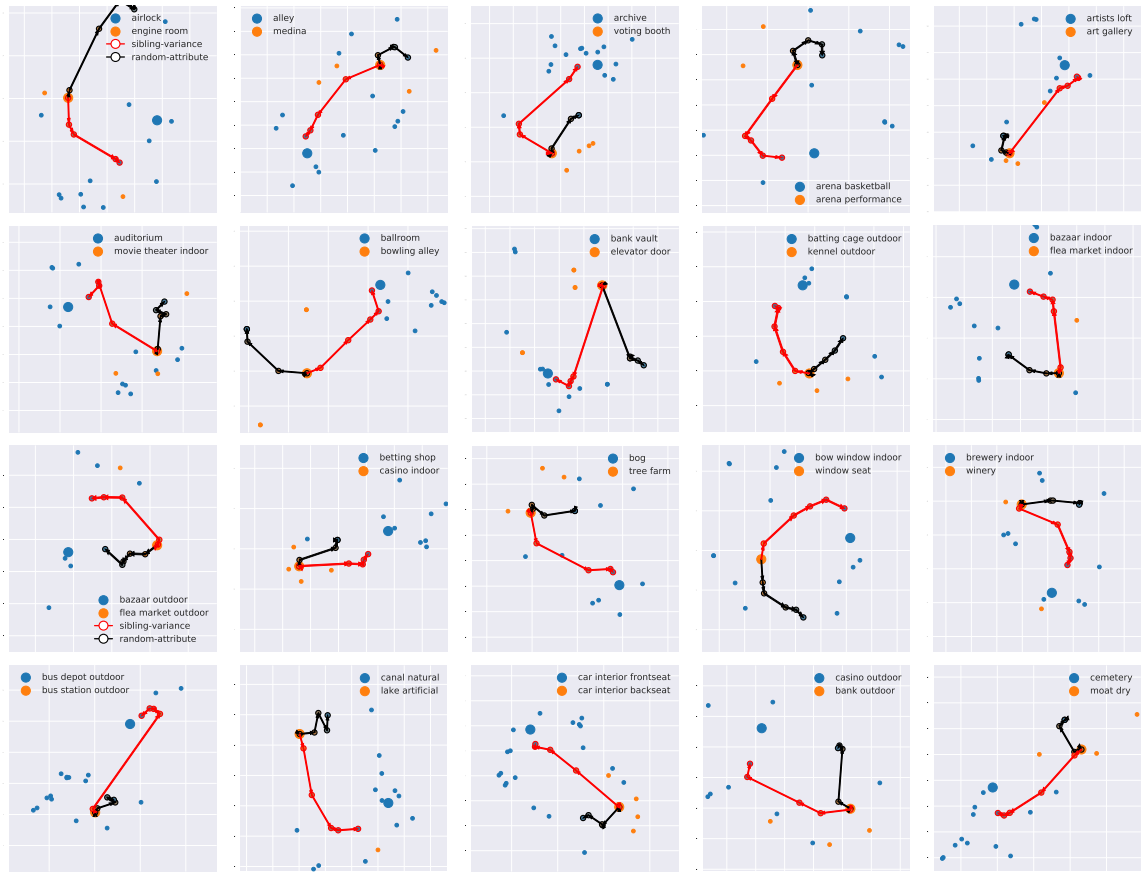

Figure 9. t-SNE visualizations. For 20 SUN novel classes and corresponding similar base classes. Smaller dots represent test images and larger dots represent class attribute embeddings. Red edges show the progression of novel class attributes as learners interact using *Sibling-variance*. Dots with black edges show the progression with the *random* function. Both methods start at the base class attribute descriptor, and aim to reach the novel class descriptor with as few interactions as possible. In most cases *Sibling-variance* reaches closer to the novel class descriptor quicker in contrast to *random*.

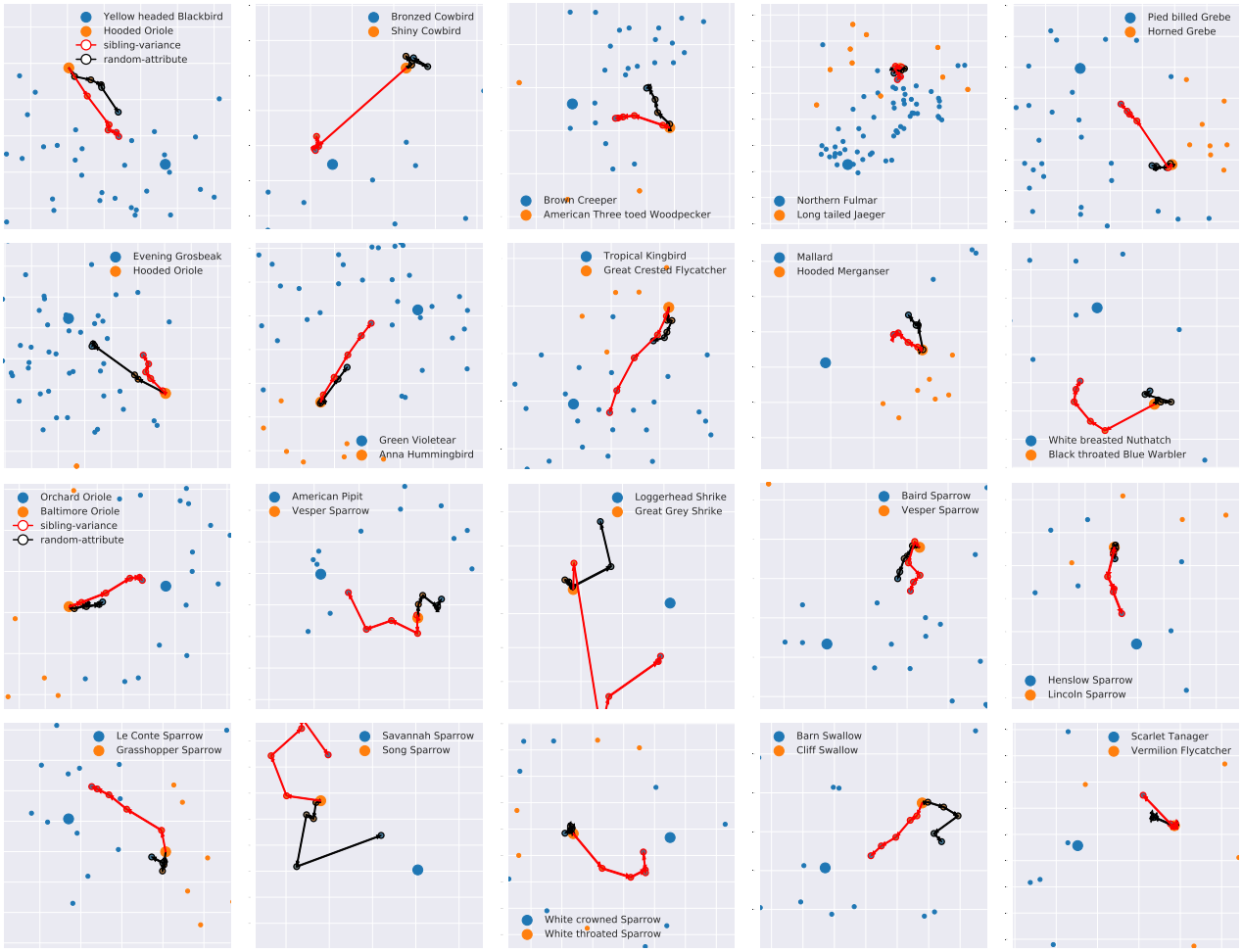

Figure 10. t-SNE visualizations. For 20 CUB novel classes and corresponding similar base classes. Smaller dots represent test images and larger dots represent class attribute embeddings. Red edges show the progression of novel class attributes as learners interact using *Sibling-variance*. Dots with black edges show the progression with the *random* function. Both methods start at the base class attribute descriptor, and aim to reach the novel class descriptor with as fewer interactions as possible. In most cases *Sibling-variance* reaches closer to the novel class descriptor quicker in contrast to *random*.