Supplementary Material For Visually Grounded Comparative Language Generation

Anonymous ACL submission

A Algorithmic Approach to Dataset Construction

We present here an algorithmic approach to collecting a dataset of image pairs with natural language text describing their differences. The central challenge is to balance empirical desiderata—mainly, sample coverage and model relevance—with practical constraints of data quality and cost. This algorithmic approach underpins the dataset collection we outline in the paper body.

A.1 Goals

Our goal is to collect a dataset of tuples (i_1, i_2, t), where i_1 and i_2 are images, and t is a textual comparison of them. We can consider each image i as drawn from some domain $\mathcal{D} \in \{$*furniture, trees, ...*$\}$, or a completely open domain of all concepts. There are several criteria we would like to balance:

1. **Coverage** A dataset should sufficiently cover $\mathcal{D}$ so that generalization across the space is possible.

2. **Relevance** Given the capabilities for models to distinguish i_1 and i_2, t should provide value.

3. **Comparability** Each pair (i_1, i_2) must have sufficient structural similarities that a human annotator can reasonably write t comparing them. Pairs that are too different will yield lengthy and uninteresting descriptions without direct contrasting statements. Pairs that are too similar for human perception may yield *"I can't see any difference."*

4. **Efficiency** Image judgements and textual annotations require human labor. With a fixed budget, we would like to yield a dataset of the largest size possible.

We describe sampling algorithms for addressing these issues given the choice of a domain.

A.2 Pivot-Branch Sampling

Drawing a single image i from domain $\mathcal{D}$, there is a chance $p \in [0, 1]$ that each image is ill-suited for comparisons. For example, i might be out-of-focus or contain multiple instances.

If a pair of images is drawn, and each has probability p of being discarded, then $\frac{1}{(1-p)^2}$ times more pairs must be selected and annotated. For example, if $p = \frac{2}{3}$, then the annotation cost is scaled by 2.25. This severely impacts annotation *efficiency*.

To combat this, we employ *pivot-branch sampling*. Each image on one side of the comparison (say, i_{pivot}) is vetted individually, and k images on the other side (say, i_{branch}) are sampled to produce pairs. With k-times fewer i_{pivot} images, it is feasible to check each instance for usability. This lowers the annotation cost scale to $\frac{1}{1-p}$ (e.g., with $p = \frac{2}{3}$, this is 1.5).

Splitting our selection from $\mathcal{D}$ into two parts allows us to define two distinct sampling strategies. One choice is for $s_{\text{pivot}}(\mathcal{D})$ to select pivot images. The second is for $s_{\text{branch}}(\mathcal{D}, i_{\text{pivot}}, k)$ to sample k images given a single pivot image.

A.3 Designing $s_{\textbf{pivot}}(\mathcal{D})$

Selecting i_{pivot} are important because each will contribute to k image pairs in a dataset. Here we consider the case where there are class labels $c \in \mathcal{C}$ available for each image in the domain. We propose selecting s_{pivot} to sample uniformly over $\mathcal{C}$. This strategy attempts to provide coverage over $\mathcal{D}$ using class labels as a coarse measure of diversity. It accounts for category-level dataset bias (e.g., where most images belong to only a few classes). This pushes the need to address *relevance* and *comparability* to the sampling procedure for branched images.

A.4 Designing $s_{\textbf{branch}}(\mathcal{D}, i_{\textbf{pivot}}, k)$

Given each pivot image i_{pivot}, we will choose k images from $\mathcal{D}$ for comparison. We can make use of additional functions and structure available on $\mathcal{D}$:

$\mathcal{V}(i_1, i_2) \rightarrow [0, 1]$
A function that measures the visual similarity between any two images.

$\mathcal{T}(\mathcal{D})$
A taxonomy over $\mathcal{D}$, with image class labels $c \in \mathcal{C}$ as leaves.

We can partition $k = k_v + k_t$ to sample k_v visually-similar images using and k_t taxonomically related images. A simple strategy for visually similar images is to pick

$$\underset{i' \in \mathcal{D}, i' \neq i_{\text{pivot}}}{\operatorname{argmin}} \quad \mathcal{V}(i_{\text{pivot}}, i')$$

k_v times without replacement. This samples the k_v most visually similar images to i_{pivot}, excluding the image itself.

To employ taxonomic information, we propose a walk over mutually exclusive subsets of $\mathcal{T}(\mathcal{D})$. We define a function $a_{\mathcal{T}(\mathcal{D})}(c, \ell)$ that gives the set of other taxonomic leaves that share a common ancestor exactly ℓ taxonomic levels above c, and no levels lower. More formally, if we use $p(c, c', \ell)$ to express that c and c' share a parent ℓ taxonomic levels above c, then we can define:

$$a_{\mathcal{T}(\mathcal{D})}(c, \ell) = \{c' : p(c, c', \ell) \wedge \nexists_{\ell' < \ell}\, p(c, c', \ell')\}$$

The function $a_{\mathcal{T}(\mathcal{D})}(c, \ell)$ partitions the taxonomy $\mathcal{T}(\mathcal{D})$ into disjoint subtrees. For example, $a_{\mathcal{T}(\mathcal{D})}(c, 1)$ are the set of sibling classes to c which share its direct parent; $a_{\mathcal{T}(\mathcal{D})}(c, 2)$ are the set of cousin classes to c which share its grandparent, but *not* its parent.

We can employ $a_{\mathcal{T}(\mathcal{D})}(c, \ell)$ by choosing class c from our pivot image i_{pivot} and varying ℓ. As we increase ℓ, we define mutually exclusive sets of classes with greater taxonomic distance from c.

To sample images using this scheme, we can further split our k_t budget for taxonomically sampled images into $k_t = k_{t_1} + k_{t_2} + \cdots + k_{t_\ell}$ for ℓ different levels. Then, if we write the set of classes $\mathcal{C}_\ell = a_{\mathcal{T}(\mathcal{D})}(c, \ell)$, we can sample k_{t_ℓ} images from $\mathcal{C}$. One scheme is to perform round-robin sampling: rotate through each class $c_\ell \in \mathcal{C}_\ell$ and sample sample one image from each until k_{t_ℓ} are chosen.

A.5 Analyzing $s_{\textbf{branch}}(\mathcal{D}, i_{\textbf{pivot}}, k)$

Given a good visual similarity function $\mathcal{V}$, image pairs will exhibit enough similarity to satisfy requirement that they be semantically close enough to be *comparable*. They may also be so visually similar that comparability is difficult. However, this aspect counter-balances with *relevance*: if $\mathcal{V}(i_1, i_2)$ is small under a visual model, but their differences are describable by humans, their difference description has high value because it distinguishes two points with high similarity in visual embeddings space.

The use of the taxonomy $\mathcal{T}(\mathcal{D})$ complements $\mathcal{V}$ by providing controllable *coverage* over $\mathcal{D}$ while maintaining *relevance* and *comparability*. Tuning the range of ℓ values used in the taxonomic splits $a_{\mathcal{T}(\mathcal{D})}(c, \ell)$ ensures comparability is maintained. Clamping ℓ below a threshold ensures images have sufficient similarity, and controlling the proportion of k_{t_ℓ} for small values of ℓ mitigates the risk of too-similar image pairs.

Similarly, we can adjust the relevance of taxonomic sampling by controlling the distribution of $k_{t_1} \ldots k_{t_\ell}$ with respect to the particular structure of the taxonomy $\mathcal{T}(\mathcal{D})$. If the taxonomy is well-balanced, then fixing a constant k_{t_ℓ} will draw proportionally more samples from subtrees close to c. This can be seen by considering that $a_{\mathcal{T}(\mathcal{D})}(c, \ell)$ defines exponentially larger subsets of $\mathcal{T}(\mathcal{D})$ as ℓ increases. Drawing the same number of samples from each subset biases the collection towards relevant pairs (which should be more difficult to distinguish) while maintaining sparse coverage over the entirety of $\mathcal{D}$.

B Model Optimization

For the CNN (both Inception-v4 and ResNet), we start with a model pretrained on ImageNet classification, and then finetune it to perform fine-grained species classification across a full export of iNaturalist (all taxonomic classes). After finetuning, we fix the CNN weights before training on our task.

We train with Adagrad for 700k steps using a learning rate of .01 and batch size of 2048. We decay the learning rate after 20k steps by a factor of 0.9. Gradients are clipped at a magnitude of 5.