Interactive Visual Feature Search

Supplementary Material

Code Our source code and sample Jupyter Notebooks that showcase VFS are available at https://github.com/lookingglasslab/VisualFeatureSearch. The notebooks are designed to run in Google Colab, and the repository includes additional instructions for running VFS locally with Jupyter Notebooks.

Additional Figures In this section, we provide additional figures for the domain generalization, editing classifiers, and ImageNet vs. PASS experiments.

Figure 5 includes additional queries and results for the ImageNet (in-domain), ImageNet-A, and ImageNet-Sketch (o.o.d.) experiments. In Fig. 5a, all queries are pictures of unicycles; when the unicycle wheels are highlighted in each query, both the ImageNet and ImageNet-Sketch queries are successfully matched to other unicycle wheels, while the ImageNet-A query is matched with various unrelated nearest neighbors. Similarly, in Fig. 5b, all three queries are images of bell peppers; however, only the in-domain query yields nearest neighbors that are also bell peppers. For both the unicycle and the bell pepper queries, the resulting similarity scores are highest for the in-domain queries (i.e. > 0.9) when compared to the scores for the o.o.d. queries (i.e. < 0.8).

Similarly, Figure 6 includes additional domain generalization visuals for the iWildCam dataset. Two sets of queries and search results are shown: one is of a cow during the day, while the other is of a deer at night. The results for the deer query are similar to those in Figure 2, as the features appear to be highly generalizable and have nearest neighbors of other deer across a variety of domains. However, the encoded features for the cow are less generalizable and simultaneously less accurate, as the nearest neighbors in other domains have lower similarity scores (0.94, 0.93) than those from the same domain (0.96), and the nearest neighbors from other domains contain horses, not cows. This particular query image is misclassified by the model as containing a horse, so the search results help visualize the features associated with this misclassification.

Figure 7 contains an additional example of the Editing Classifiers visualization, as well as an additional visual for the ImageNet vs. PASS experiment. Fig. 7a includes a query of a scooter on snow-covered ground; when the ground is highlighted, the original model's VFS results contain no other images of scooters or cars. However, when the edited model is used, four of the top-5 results contain cars, and the instance of a bobsled on ice from the original results is no longer included. Thus, this example provides further evidence that the model edit was successful.

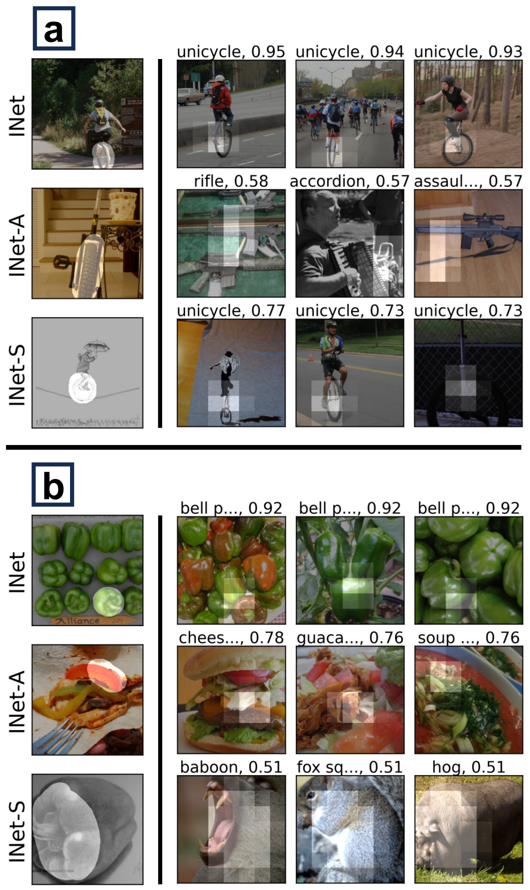

Figure 5. **Additional ImageNet Generalization.** Three queries of images from ImageNet, ImageNet-A [17], and ImageNet-Sketch [33] on an ImageNet model. **a**: All queries are of unicycle wheels. The ImageNet-Sketch unicycle is matched with other unicycle wheels in the dataset, but the similarity scores are lower than those from the in-domain query. **b**: All queries are of bell peppers, but only the in-domain search contains bell peppers in the results. Such visuals may provide insights into why a particular model misclassified a challenging example.

Figure 7b shows an additional example of a query containing a face with two sets of results for the ImageNet and PASS models, respectively. Although several of the highlighted regions in the PASS results contain no faces, two such results contain faces elsewhere in the image. Thus, the PASS-trained model is again able to successfully encode human faces and retrieve other images containing faces via VFS.

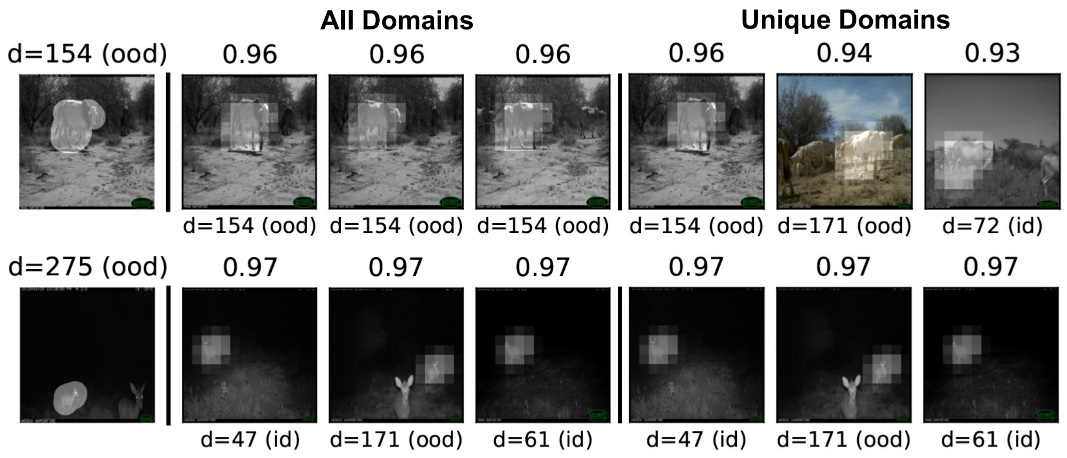

Figure 6. **Additional iWildCam Generalization.** Two sets of queries and results for a cow (top) and a deer (bottom). While the deer features appear to be highly generalizable with all top-3 results originating from different camera locations, the cow has a comparatively worse feature representation since its nearest neighbors from other domains have slightly lower similarity scores (0.94, 0.93) than inner-domain results (0.96); additionally, its nearest neighbors from other domains are images of horses, not cattle.

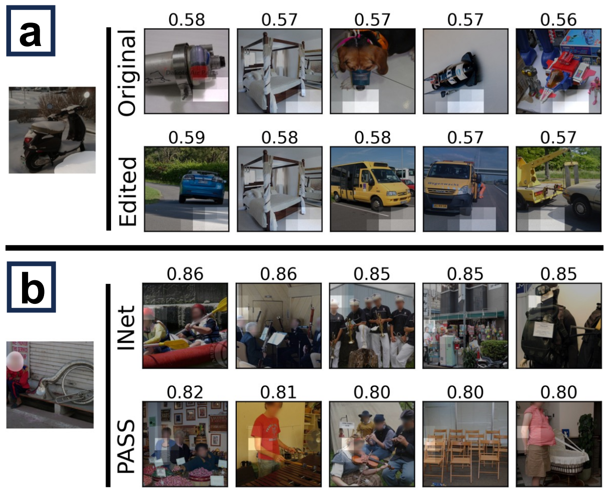

Figure 7. **Additional Results for Edited Classifier and PASS Training.** **a**: An additional visualization of the "vehicles on snow" classifier edit. The original model's VFS results consist entirely of unrelated objects such as ice, floors, and a tabletop, whereas the edited model's results contain several cars on asphalt. **b**: An additional search query and results for ImageNet- vs. PASS-trained models. While the localization of nearest neighbors is relatively poor for PASS (i.e. the highlighted regions in some search results do not contain faces), the PASS model is able to successfully able to match the query to other images that contain faces.

Implementation Details In Section 2, we described how to perform the region-based similarity search for VFS. However, in order to perform the search efficiently, we use a modified version of this algorithm that substitutes the sliding window approach with a convolution operation. This allows us to compute the searches on a GPU via PyTorch, which allows for extensive parallelization and enables each search to be much faster than if it were run on a CPU.

To implement the convolution-based searches, we first take the 3D tensor $\mathbf{z} \in \mathbb{R}^{H \times W \times C}$ from Section 2 and apply the mask $\mathbf{m}_l$ on it once more. We then crop the tensor, which we define as removing all rows/columns on the exterior of the feature map which only contains zero-valued elements. Let the resulting tensor be $\mathbf{z}'$, where:

$$\mathbf{z}'_{(i',j',k')} = \text{Crop}(\mathbf{z}_{(i,j,k)} \cdot \mathbf{m}_{l(i,j)}) \tag{1}$$

$\mathbf{z}'$ has dimensions $H' \times W' \times C$, where $H' \leq H$ and $W' \leq W$. We use $\mathbf{z}'$ as a 2D convolutional filter and apply it to a feature map $f_l(\mathbf{s})$ from the search database.

$$\mathbf{c} := f_l(\mathbf{s}) * \mathbf{z}' \tag{2}$$

Each element $\mathbf{c}_{(a,b)}$ is equivalent to the inner product $\vec{q} \cdot \vec{d_i}$, for a unique region vector $\vec{d_i}$ within the search image's feature map. We can perform a similar convolution to obtain the magnitudes of each $\vec{d_i}$, so we can thus compute the cosine similarities for all searchable regions within the image s without iteratively computing results with a sliding window.

Runtime Performance We measured the runtime performance of our VFS implementation by running multiple sets of searches over varying dataset sizes. We used subsets of the ImageNet dataset with n = 25,000, 50,000, 75,000, and 100,000 images; the similarity searches were computed for a ResNet50 model's conv5 features with dimensions $7 \times 7 \times 512$. In order to understand the impact of memory bandwidth, we performed two sets of experiments: the first set used a cache that was pre-loaded directly into a

GPU's VRAM, while the second set loaded the feature data in batches from a cache in regular RAM. For each dataset size and cache location, we computed search results for 20 hand-drawn queries; the mean runtimes and standard errors are included in Table 1. All searches were computed on a Google Colab VM with a 16 GB NVIDIA T4 GPU.

n	Cache Size	VRAM Time (s)	RAM Time (s)
$25k$	2.34 GB	0.139 ± 0.004	0.820 ± 0.009
$50k$	4.67 GB	0.263 ± 0.006	1.668 ± 0.021
$75k$	7.01 GB	0.420 ± 0.010	2.542 ± 0.018
$100k$	9.35 GB	0.564 ± 0.008	3.435 ± 0.023

Table 1. **Runtime Performance.** The mean runtime (and SEM) of VFS was measured over a set of 20 query regions, with a search dataset of size $n = 25,000$ through $100,000$. Searches were computed for ResNet50 conv5 features with dimensions $7 \times 7 \times 512$ and 32-bit precision; the resulting cache sizes for all feature data are included for reference. Results are reported for two cache locations: one set of experiments pre-loaded the cache entirely onto a GPU's VRAM, while the other set loaded features from a cache in regular RAM in batches. The results indicate that VFS is most efficient when the feature cache is able to be loaded directly into VRAM; loading from RAM may be necessary for especially large datasets, but this increases runtime significantly.

The results in Table 1 show that VFS is most efficient when the dataset's feature cache can be loaded entirely in a GPU's VRAM. The mean runtime for searching through a set of 50,000 images is 0.263 seconds, which enables VFS to be a fast and highly interactive tool. Computing search results with a cache in regular RAM is much slower, mainly due to the bandwidth required to move each batch of feature data to the GPU for similarity search. However, if a user's dataset features are too large to fit in their GPU VRAM, then loading from batches in RAM is a viable (albeit slower) method when using VFS.