Supplementary Material

A. Training details

Framework We use the pytorch-lightning [15] framework for training our models. We use backbone ResNet-50 and ViT models implemented in timm [63]. We adapt PyTorch public implementations of the FGVC models (PMG, WSDAN, SIMTrans) to work in our framework and with these backbones.

Hyperparameters We train for 50 epochs using a batch size of 16 (Aircraft, Cars, CUB) or 64 (NABirds, Fungi) on a single A100 GPU with automatic mixed precision. We use a cosine decay learning rate schedule without any warmup. We use an initial learning rate of 10^{-2} for all randomly initialized layers and $10^{-}3$ for pretrained layers; we found this to work well for all models with the exception of WSDAN, for which we use 10^{-3} for all layers. We apply weight decay to convolution and linear layers, but not to bias or normalization parameters (batch norm, layer norm). We set weight decay to 5×10^{-4} for models with a ResNet-50 backbone and 10^{-5} for models with a ViT-base backbone. For ViT models, we use a high dropout rate of 0.7 before the classifier to help with overfitting.

Backbones We use backbone models pretrained for classification on ImageNet-1k [47]. We choose not to use ImageNet-21k pretrained weights for ViT in order to keep the pretraining as uniform as possible across all models. This does have the effect of lowering the ViT performance, but it still performs well. For ResNet-50 we use the torchvision weights `IMAGENET1K_V2`. For ViT, we use the weights provided by timm [63] as `vit_base_patch16_224.augreg_in1k`.

Augmentation and preprocessing During training, random crops are taken from the images with area drawn uniformly between 8–100% of the image area, and with aspect ratio between $\frac{3}{4}$ and $\frac{4}{3}$, and then resized to 448×448. This is followed by scaling the brightness, saturation and contrast with values drawn randomly between 0.9 and 1.1, and applying a horizontal flip with probability 0.5. Pixels are then normalized by the ImageNet pixel mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$.

Evaluation At evaluation, images are resized to 512×512, center cropped to 448×448, and normalized by the ImageNet mean and standard deviation.

B. Dataset details

Table 1 gives the statistics of the datasets we use in our evaluation. We use five common FGVC datasets, as well as the proposed iCub dataset. All datasets have bounding box annotations except for Fungi. Fig. 12 shows distributions of object spatial properties (bounding box area, aspect ratio, and distance from image center) for CUB and iCub; iCub has a very different distribution of object size. Fig. 13 shows additional example images chosen randomly from iCub.

Dataset	#Cls	#Train	#Test	Boxes
Aircraft [41]	100	6,667	3,333	✓
Cars [33]	196	8,144	8,041	✓
CUB [57]	200	5,994	5,794	✓
NABirds [55]	555	23,929	24,633	✓
Fungi [44]	183	32,753	3,640	
iCub	200	n/a	16,876	✓

Table 1. Number of classes, training images, and test images for each of the datasets considered in this paper. Aircraft, Cars, CUB, and NABirds are well-established FGVC datasets, and Fungi is a more recent addition. We also introduce the iCub dataset to aid in our analysis.

C. Incorrect labels

Label errors in validation set	262
Out-of-distribution label errors	133
In *easy* set	27
In *elusive* set	129
In-distribution label errors	129
In *easy* set	1
In *elusive* set	72
... but actually easy (corrected)	49
... still elusive (corrected)	8

Table 2. **Label errors in CUB**. The CUB validation set has 262 incorrect labels (out of 5794). About half are out-of-distribution; the true class isn't in the dataset. Of the in-distribution errors, more than half are in the elusive set; but many of those images are actually easy given the correct labels (all models predict the true correct label).

Some degree of label error exists in many datasets, which often rely on crowd-sourcing to obtain or verify labels. Van Horn *et al*. [55] estimated around 4% label error in CUB using domain-expert opinions. They experimentally verified that up to 10% label error in the training set has little effect on the test performance; but label error in the test set is problematic, since correct predictions might be counted as incorrect.

We don't know the level of label error in the other datasets, but we did obtain the label errors and corrected labels for CUB from the authors of [55]. We evaluate the errors and their corrections using prediction overlap to see where label error shows up. We summarize our findings in

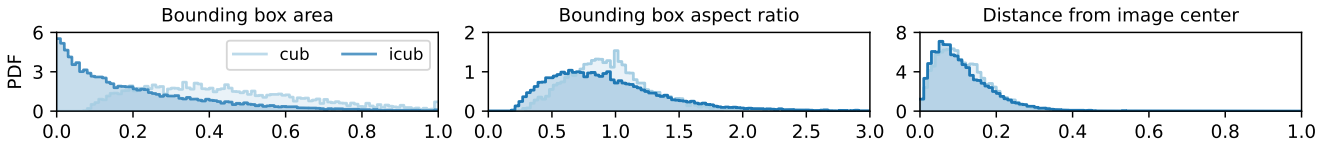

Figure 12. **CUB vs iCub**. Distribution of CUB and iCub images with respect to object size, aspect ratio, and distance from center (which are defined using the bounding box). iCub contains many more images with small (or distant) birds than CUB.

Figure 13. Additional iCub samples. Zoom in for detail.

Table 2. About a fifth of the out-of-distribution errors (actual ground-truth class isn't in the dataset) fall in the easy set, probably because they are similar to the class they are erroneously assigned to. On the other hand, more than half of the in-distribution errors show up in the elusive set—mostly, it turns out, because the models are actually predicting the correct class. If we assign the true correct label, 49 out of 72 of the "elusive" images are actually trivial, and only 8 remain elusive. Applying the corrected labels (for the in-distribution errors) reduces the size of the elusive subset of CUB from 3.8% to 2.7%.

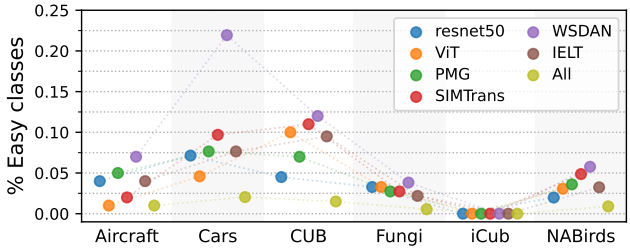

Figure 14. **Easy classes**. Percent of classes for each method and dataset that are easy (classes for which the model misclassifies 0 images across all runs). The easy classes for the "All" method are classified correctly across all 30 runs. Notably, iCub has no "easy" classes.

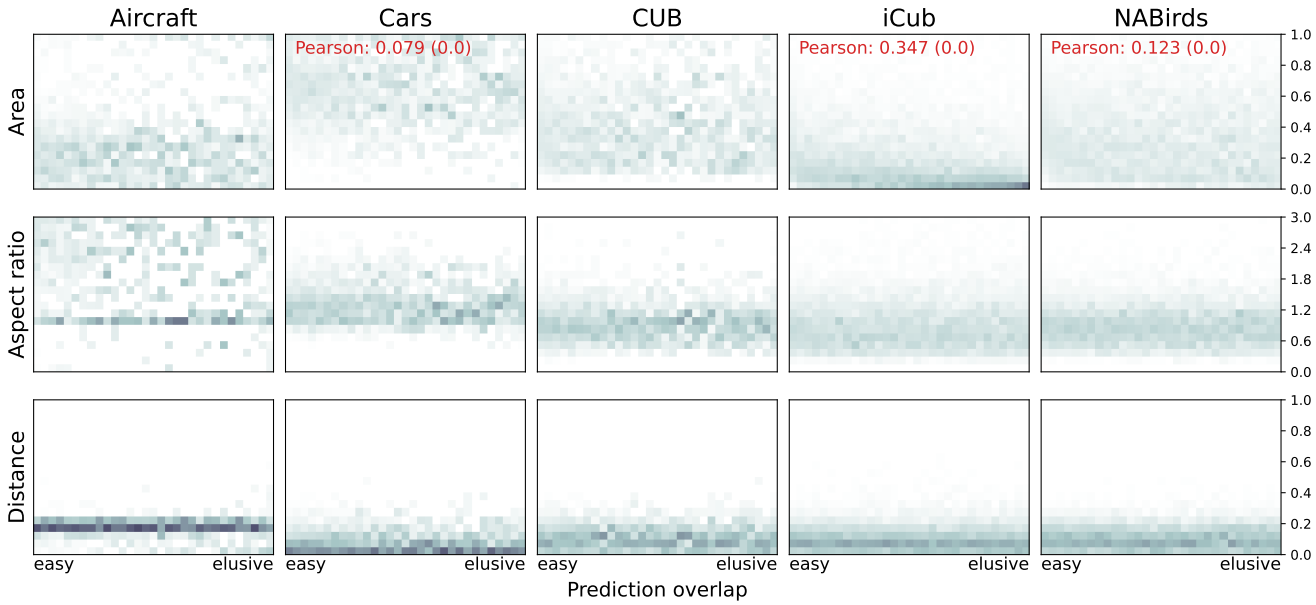

Figure 15. **Correlation with spatial properties**. We show correlation between object spatial properties (bounding box area, aspect ratio, and distance from center of image) and prediction overlap. In most cases, the correlation is negligible; however, iCub and NABirds both show correlation between object size and image difficulty (low prediction overlap).

D. Additional results

D.1. Easy classes

In Fig. 14, we show the percentage of easy classes, divided by model. An "easy" class is one for which all images of that class are always predicted correctly. For a given model, the easy classes are those for which all images are predicted correctly across the 5 different trials. Cars has the most easy classes when considering all models together, but CUB has the most on average when considering each model individually—except for WSDAN, which has a large number of easy classes for Cars. Notably, iCub has no easy classes.

D.2. Spatial properties

In Fig. 15, we show correlation between prediction overlap and three different spatial properties of images for each dataset. In most cases, there is not significant correlation. The exception is with object size, measured by bounding box area, on a few of the datasets; in particular, iCub and NABirds, with a small correlation observable for Cars as well. Aspect ratio and "centeredness" don't appear to contribute to image difficulty, but object size does.

D.3. Pairwise class confusion

We show additional visualizations of pairwise class confusion in Figs. 16 to 18. Fig. 16 shows the distributions of pairwise similarity between classes (Eq. (4)). The distributions are fairly similar between datasets, with the exception of iCub. In Fig. 17, we show a KL divergence confusion matrix for each dataset. These matrices were created by averaging the KL divergence scores across all models for each pair of classes. Fig. 18 shows a combination of a traditional confusion matrix thresholded according to the KL divergence matrix. The red cells (i, j) show similar class pairs (KL divergence more than 3 standard deviations below the mean), with darker color showing more predictions for class j on images that belong to class i. Blue cells show predictions belonging to classes that aren't considered "similar" according to our definition. Predictions are aggregated across all models.

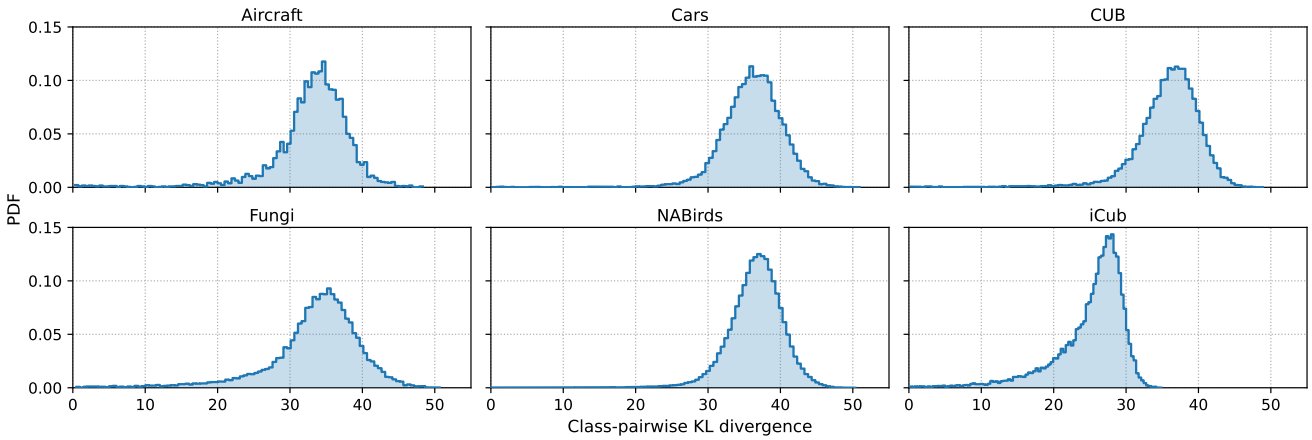

Figure 16. **Distributions of pairwise similarity**. These plots show the distribution of the proposed KL divergence-based class confusion measure. The x-axis is histogram bins of symmetric KL divergence values and the y-axis shows the density of class pairs that fall into that bin. iCub has a very heavy "leading tail" of confusion.

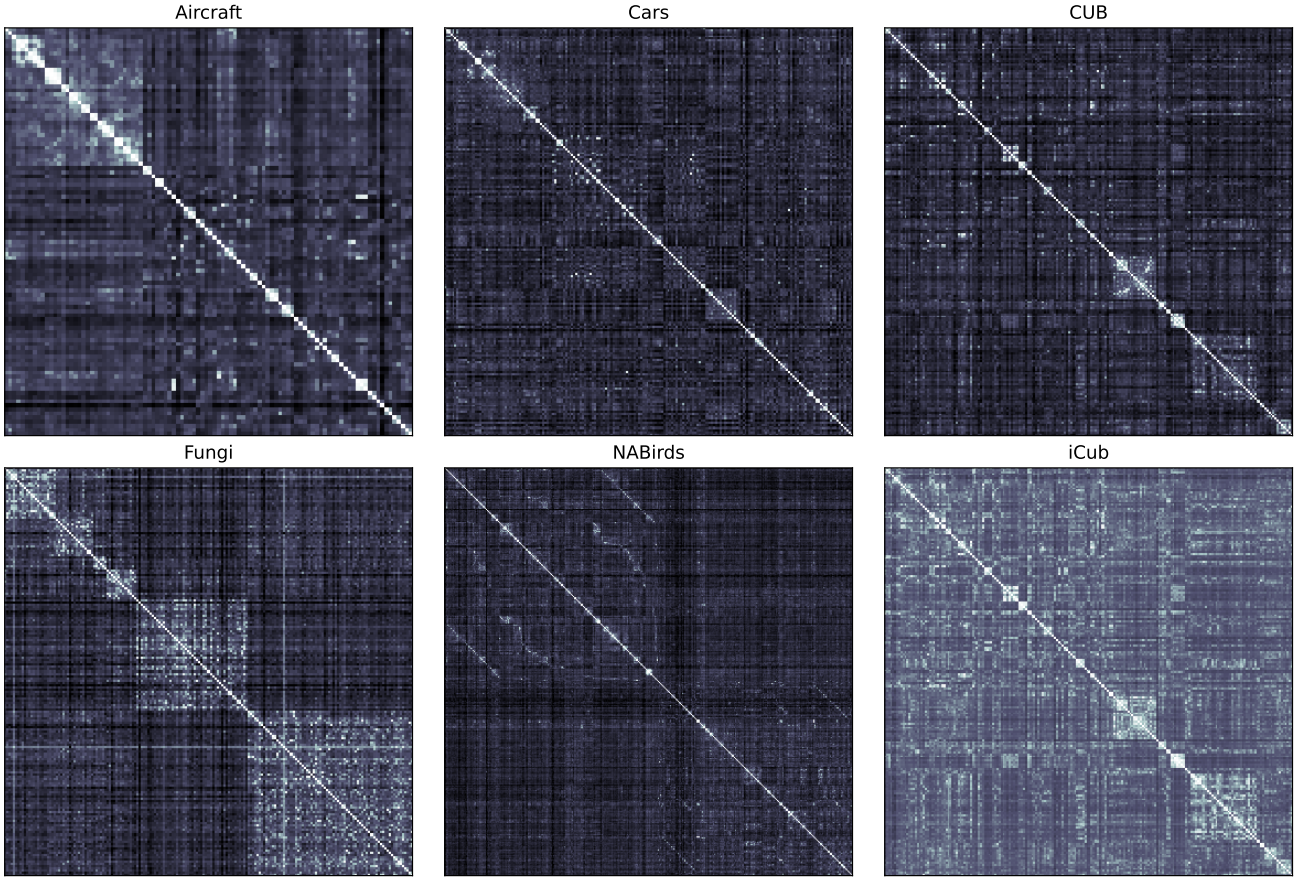

Figure 17. **Pairwise KL divergence**. We show pairwise KL divergence matrices averaged across all models; they are similar to confusion matrices but take into account the full predictive distribution rather than counting predictions. Lighter color indicates smaller KL divergence (higher class similarity).

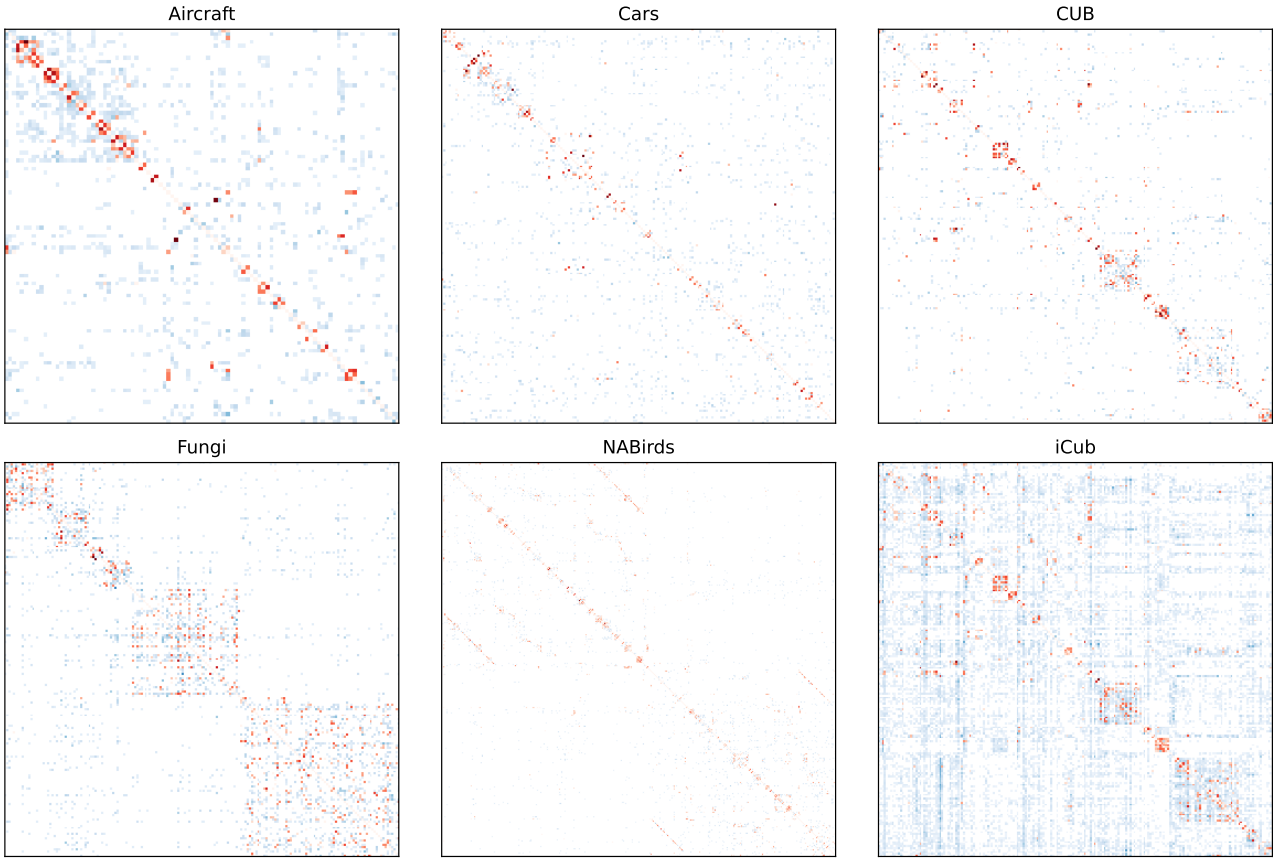

Figure 18. **Confusion matrices**. We show confusion matrices aggregating predictions across all models. Darker color indicates a larger number of predictions. Red cells indicate similar-class pairs, based on thresholding the pairwise KL divergence matrices at 3 standard deviations below the mean. Blue cells indicate predictions in classes that aren't considered "similar". We ignore the main diagonal (correct predictions) for this visualization.