Intelligent Image Collection: Building the Optimal Dataset

Matthew Gwilliam and Ryan Farrell
Brigham Young University
{mattgwilliamjr@gmail.com,farrell@cs.byu.edu}

Abstract

Key recognition tasks such as fine-grained visual categorization (FGVC) have benefited from increasing attention among computer vision researchers. The development and evaluation of new approaches relies heavily on benchmark datasets; such datasets are generally built primarily with categories that have images readily available, omitting categories with insufficient data. This paper takes a step back and rethinks dataset construction, focusing on intelligent image collection driven by: (i) the inclusion of all desired categories, and, (ii) the recognition performance on those categories. Based on a small, author-provided initial dataset, the proposed system recommends which categories the authors should prioritize collecting additional images for, with the intent of optimizing overall categorization accuracy. We show that mock datasets built using this method outperform datasets built without such a guiding framework. Additional experiments give prospective dataset creators intuition into how, based on their circumstances and goals, a dataset should be constructed.

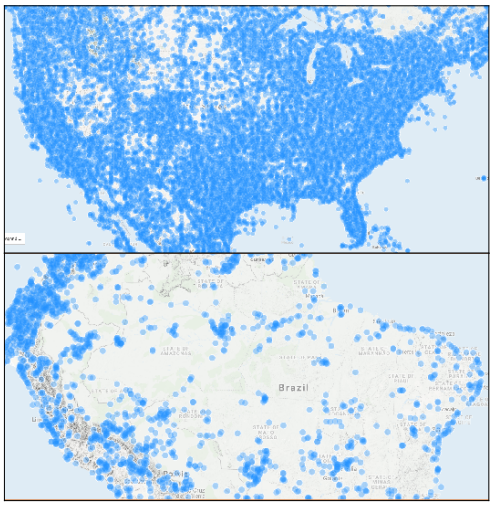

Figure 1. Bird observation distributions, from iNaturalist [28], over two different regions, the US (top) and the northern portion of South America (bottom). Darker shades of blue indicate higher observation counts, showing the observation density is much higher in the US than in the Amazon Rainforest.

1. Introduction

Exciting new developments in fine-grained visual categorization (FGVC) are being unveiled frequently [6, 11, 12, 23, 42, 48]. Researchers aim to find methods that will yield the highest accuracy, and every tenth of a percent counts. Performance of these efforts is measured primarily on a few key benchmark sets, including FGVC-Aircraft (Aircraft) [26], Caltech-UCSD Birds (CUB) [4, 41], Stanford Cars (Cars) [22], Stanford Dogs (Dogs) [19], and Oxford Flowers (Flowers) [27] datasets.

While standard dataset creation approaches (see Section 2) work fairly well for images collected from areas like North America and Western Europe, where an abundance of image data is accessible and available, they do not work as well in other parts of the world. Consider iNaturalist.org (iNat) [28], a web application where users (citizen scientists) post images (observations) of various wildlife which other citizen scientists with domain expertise then work together to identify (label). Figure 1 clearly demonstrates that iNat has large amounts of bird data for the United States (US), but virtually no data for the more biodiverse Amazon Rainforest.

The numbers provided in Table 1 help put this difference into perspective. Almost half of the 1800 bird species in Peru and Brazil, the countries containing the majority of the Amazon, have no iNat observations at all. The US meanwhile, home to less than half as many species as Peru and Brazil, has approximately 200× and 70× as many observations, respectively. It's obvious then, that for birds from these South American countries and from African nations like the Democratic Republic of Congo and Kenya, there isn't sufficient citizen-curated data that could be used to build a dataset with expert labels and annotations.

In situations like these, data augmentation methods [7, 10, 17, 35] and few-shot learning approaches [18, 33, 40, 43] can be helpful. However, even these techniques are no substitute for additional data. The only way to build

Country	Rank	iNat Obs	iNat Species	Species
Peru	2	8470	**997**	**1858**
Brazil	4	24786	**1149**	**1813**
Ecuador	5	20700	1157	1622
DR Congo	10	2522	**491**	**1107**
Tanzania	12	10548	676	1075
Kenya	13	8945	**698**	**1058**
United States	*23*	*1.748M*	*1070*[1]	*860*
Spain	89	17914	384	382
France	98	23794	384	357
Italy	99	36792	414	355

Table 1. The number of iNaturalist observations and species by Country; all verifiable, research-grade observations are included. The right-most column indicates the total number of bird species found in each country, as determined by Birdlife International [5]. The United States is highlighted (in italic) because it receives a disproportionate amount of attention – over 52.5% of iNaturalist's global bird observations despite being home to only 7.7% of the world's bird species. Additionally, the USA/North America is the focus of all three of the bird datasets commonly used for computer vision, CUB [4, 41] Birdsnap [2], and NABirds [37] datasets.

a sufficiently large dataset of Amazonian or Kenyan birds would be to organize an expedition to one of those areas to take photographs, gathering brand-new data. This would, no doubt, be incredibly time-consuming and expensive.

In light of the great expense associated with collecting images from the wild and labeling them, it is essential for the dataset creator to maximize the value added by each new image. Rather than collecting as many images as possible without a backward glance at which categories they come from, we theorize that it would instead be helpful to gather more images associated with the most difficult categories. It would thus be critical, for the sake of maximizing classification accuracy and minimizing the costs associated with the dataset's construction, to have a framework to guide decision-making, as the dataset is built, on how many additional images would be needed to represent each species.

As an example, suppose we have two images from each of the species represented in Figure 2. If we had a budget that allowed us to acquire just 6 more images, then our approach would be to distribute those 6 new images in the manner shown in Figure 2, where the "harder" categories to differentiate (the terns) receive the new images and the easy category (the violetear) receives none. A photographic exploration into the Amazon could take a similar approach, using a small initial dataset to determine which species are most difficult to classify, then selecting locations where those species are known to be found. In Section 4, we

[1]For the United States and others near the bottom of the table, iNaturalist has more species observed than total species present in the country, due to the observation of vagrant species, e.g. those blown in by storms.

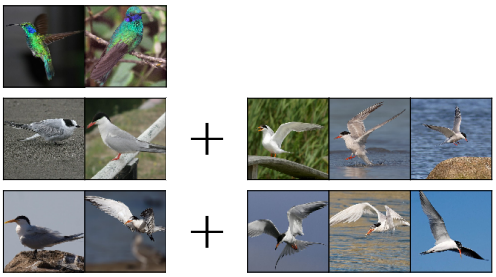

Figure 2. This sample dataset illustrates how if we can only acquire a fixed number of additional images, we would rather add images for birds with high visual similarity like the Common tern and Elegant tern, shown in the bottom two rows, than for readily-distinguished birds like the Green violetear, shown at the top.

demonstrate that when done properly, this kind of approach maximizes overall categorization accuracy.

In this paper, we explore how the creators of a fine-grained dataset should approach image collection after they've curated an initial dataset. With the intent of maximizing categorization accuracy, we propose the idea of intelligently determining which categories a dataset creator should prioritize adding images to. We describe a method that achieves higher accuracy than random selection. We demonstrate the method's robustness, power, and flexibility through a series of experiments simulating how it would apply in the real world. We run additional experiments to analyze the behaviors of our method across datasets, both at the per-class and overall level. We thus frame dataset construction as a compelling problem to solve and present an original method for addressing it, whose success we prove with overall categorization accuracy improvements.

2. Background and Related Work

Computer vision datasets are normally created in 3 main steps. While this pattern is near universal, it is not without its difficulties and shortcomings. The **first** step, obtaining the images, is straightforward – large quantities of images in the chosen fine-grained domain are gathered from Google, Flickr, iNaturalist, or another online image source. However, the set of classes used in a dataset is almost always driven by the classes for which images are readily available, *not by the set of classes that one might actually want to recognize*. Some researchers have explored augmenting these datasets with additional weakly/noisily labeled images from online sources [21, 45].

The **second** step, assigning accurate category labels, is done in one of two ways. Crowdsourcing – where each image is labeled by a small number of non-expert workers (*e.g.* five) via a service such as Amazon's Mechanical Turk [1] (see Sorokin, *et al.* [34]) – is by far the most common approach. However, the use of non-expert workers

leads to subtle classification errors. For example, Van Horn, *et al*. [37] found class label errors of at least 4% for CUB-200 [41], the most widely-used FGVC dataset. Since SOTA accuracies on CUB are now over 89%, this 4% error is quite significant. Therefore, Van Horn, *et al*. instead advocate for the use of subject domain experts for assigning category labels. However, this comes at the high cost of finding and compensating these experts.

The **third** step, obtaining detailed annotations such as bounding boxes, part keypoints, or object silhouettes, is typically done via crowdsourcing. This can be costly, and unfortunately, as workers are paid by work unit, they often rush to complete tasks, reducing annotation quality. Several papers, *e.g*. [13, 25, 37, 38, 44], have proposed models to improve annotator consistency and reduce cost.

Berg, *et al*. [3] identify properties of a good dataset including variety, scale, precision, suitability, cost and representativeness. While properties such as scale, precision and cost are addressed above, criteria such as variety, suitability and representativeness deserve additional attention. We contend that existing datasets for natural domains lack the sufficient variety (diversity) of suitabable, representative data needed for successful recognition in most locations worldwide, particularly for species of interest (*e.g*. rare and endangered species). Available data often has a long-tailed distribution, whereas few datasets (iNaturalist [39] is one) directly address this.

One important aspect for many domains is taxonomic or hierarchical structure. The hierarchical structure may not be critical to accuracy, in and of itself, but non-uniform intercategory distances are inherent and fundamentally important – some categories are unique or highly distinctive while others are nearly identical differing only by small and/or subtle markings. Even datasets such as ImageNet [8] that structure the data hierarchically, typically measure error with a flat loss across the leaf nodes. This implies that we are equally willing to misclassify a queried image of a hummingbird as an eagle as we are to mistake it for another species of hummingbird with very subtle differences. These errors are tremendously different and, as a community, we should be very unhappy about calling them equivalent. Two notable exceptions to the flat loss are the works of Deng, *et al*. [9], and Ordonez, *et al*. [29, 30]. Both focus on the non-leaf nodes in the hierarchy, respectively trying to select the level in the hierarchy with the greatest confidence and to predict the word that humans most likely use to describe an object (it's "entry-level" category).

Another important area that's closely related to this paper's goal – adaptive construction of image datasets – is the adaptive learning of network models. The rapidly-growing field of Neural Architecture Search [24, 36, 50] explores novel paradigms for building/discovering high-performing network architectures within the immensely broad space of potential network architectures. Also related is Knowledge Distillation [15, 46], which seeks to extract a larger and more powerful network's "expertise", embedding it in another, sometimes lighter-weight network. The ability to deploy small mobile-scale networks will be critical for promising fields of the future such as *Edge-AI* [32, 49].

3. Approach

As we explain our method for intelligent image collection, we first outline both the primary objective and a key constraint in our investigation. The **primary objective** is to maximize categorization accuracy. A **key constraint** is that the model can only add images to *half* of the classes, and it adds the *same* number of images to each of these classes. We do not directly consider alternate goals, such as maximizing image diversity, or other policies, such as adding an unequal amount of images to all categories.

In this section, we first describe our method step-by-step (3.1). We then explain how a class's ease of categorization is evaluated (3.2). We conclude by explaining how we simulate the construction of a dataset (3.3).

3.1. Steps

First, our model begins with some initial dataset. In all of our experiments, this starting dataset has the same number of images for each class; however, this need not be the case. In this paper, we refer to the number of images per class in the initial dataset as the ***base*** number.

Next, CNNs pre-trained on ImageNet [47] are trained and then evaluated on 5 folds of the dataset, each with a 60-40 train-validation split. We use this cross-fold validation approach instead of holding out a single fixed validation set because it means each image can be used for both training and testing (3 times for training, 2 for testing). This maximizes the amount of information gained from the images in the initial set. This helps the model more accurately gauge which classes should receive more images.

Last, the image-adding step is performed. To do this, the model averages performance across the folds based on a desired criterion (criteria are described in Section 3.2, below). It then sorts the classes according to performance, and identifies the half of the categories that performed the worst. These are the categories that images will be added to. In our experiments, the number of images added to each class is referred to as the ***jump*** size.

3.2. Criteria

To determine which categories are most difficult and thus need additional images, each model relies on a criterion based on one of the following metrics: per-class accuracy, precision, F1, loss, or KL-Divergence.

Criteria based on **Accuracy**, **Precision**, and **F1** scores work in a similar fashion. The model measures the average

for each class across all folds, and adds images to those with the *lowest* scores. **Loss** is the opposite – the model measures the cross entropy loss corresponding to each class, and adds images to the half of the dataset's categories that have the *highest* losses.

Our other criterion utilizes **KL-Divergence**. The n-dimensional vectors (where n is the number of classes) output by the network during the testing of each fold are passed through a softmax function. We represent each class as a distribution in this n-dimensional space, fitting a multivariate gaussian to the set of softmax output vectors for the images in the class. Taking the symmetric KL-Divergence between the gaussian distributions for pairs of classes yields a square $n \times n$ matrix. The average value for each row is used as a score for each class. Since the half of the classes with the lowest scores are the ones with the greatest visual similarity (the most easily confused), the model adds images to those classes.

3.3. Mock Datasets

We utilize a framework that allows us to act as though we were creating a new dataset. For a given domain, our model takes a subset of the training portion of an existing FGVC dataset as a "mock" initial dataset, and treats the rest of the training portion as a bank of images. Once the model decides which categories need more images, it adds images to the mock dataset from that bank of images, simulating the targeted collection of new images for a dataset. In our experiments, additional networks are trained on the resulting dataset and tested against the standard test portion of the FGVC dataset chosen. This measures the effectiveness of our method, allowing comparison between our method and a baseline (random selection of categories).

This is more easily understood with an example. For Cars 10-*base* 10-*jump* (*base* number is 10, *jump* size is 10), using the **Accuracy** criterion, the dataset starts with 10 images per class (1960 images total). The model then trains and evaluates on 5 folds of that dataset, with a fold ratio of 0.6, meaning that 5 different train-test dataset splits will be used, each with 6 images per class for training and 4 images per class for evaluation. The per-class evaluation accuracies are then averaged. The half of the categories (98 categories for Cars) with the lowest average accuracies are each given 10 additional images (980 total). The viability of the class-selection method is then tested by training on the resulting dataset (1960 + 980 = 2940 images) and evaluating on the standard, held-out test set (8041 images).

It is important to note that the test portion of the FGVC dataset (Cars in the example above) is not presented to our model while it is determining which half of the categories to add images to. This means that *we are accurately simulating the real-world application of our method* by selecting categories based on results from a limited initial dataset. While there are valid concerns about dataset construction beyond the per-class distribution of images, such as diversity of images in the initial dataset (see Section 2), *using a "mock" dataset allows us to prove that such concerns do not prevent our method from being demonstrably effective*.

4. Experiments

This section thoroughly evaluates the proposed method via the following set of targeted experiments:

- **Simulation** (4.1) proves the overall effectiveness of the proposed approach.
- **Architecture Swapping** (4.2) indicates our approach doesn't overfit to the chosen network.
- **Diversity Dilemma** (4.3) establishes robustness against lack of diversity in the initial dataset.
- **Adaptive Jumps** (4.4) shows the impact of determining the amount of images to add on a per-class basis.
- **Oracle Performance** (4.5) demonstrates viability for various domains, *base* numbers, and *jump* sizes.
- **Granular Analysis** (4.6) examines the effects of our method on the per-class accuracy distribution.

Dataset	Classes	# Images	Accuracy
Aircraft	102	10200	84.68
Birds	200	11788	81.12
Cars	196	16185	89.91
Dogs	120	28580	82.36
NABirds	555	48000	78.71

Table 2. Baseline performance (averaged over 10 trials) of our ResNet-50s on different FGVC datasets. This is not intended to be competitive with SOTA results, but instead gives benchmarks for our models. From this table onward, "Birds" refers to CUB.

Each experiment's setup is described in the first paragraph of its subsection. For all experiments, we use the same network architecture – a ResNet-50 [14] pre-trained on ImageNet (from PyTorch [31]). We use the Adam [20] optimizer with cross-entropy loss. Unless otherwise stated, our networks train for 50 epochs. A basic learning rate scheduler is used, starting at 10^{-4} and ending at 10^{-7}. Augmentation during training includes random cropping, horizontal flipping, and resizing to form batches of 16 images of size 224x224. We show the baseline performance of this architecture for different popular FGVC datasets in Table 2.

The experiments below are conducted on subsets drawn from the Birds, Cars, and Dogs datasets. We keep all of the categories in each dataset, but only use some of the images. In all of these experiments, we use *base* numbers and *jump* sizes as defined in Section 3.1.

4.1. Simulation

Here, the model proves it outperforms random category selection (**Random**), regardless of the criterion chosen. For each jump size/criterion pair (each entry in Table 3), we run 5 trials whose results we average, where each trial proceeds as follows. First, image adding is done as explained in Section 3.1, where the criterion is used as explained in Section 3.2. Then, 5 networks are trained (differing only in batch sampling and augmentation) on the resulting dataset (initial set + added images). Finally, those networks are evaluated on the corresponding FGVC test set.

Criteria	5-Jump	10-Jump	15-Jump	20-Jump
Simulation (Cars)				
Random	70.32	73.45	75.85	76.82
Accuracy	70.58	74.38	**76.89**	78.20
Precision	70.23	73.72	76.46	78.12
F1	70.17	73.75	76.04	77.53
Loss	**71.00**	**74.46**	**76.89**	**78.43**
KLDiv	70.88	74.34	76.71	77.91
Simulation (Birds)				
Random	70.12	72.16	72.77	73.53
Accuracy	70.71	72.69	74.04	74.73
Precision	**71.02**	**72.78**	**74.16**	**74.98**
F1	70.20	71.88	73.10	73.71
Loss	70.60	72.56	73.95	74.47
KLDiv	70.34	72.17	73.15	74.01
Architecture Swapping (Cars)				
Random	70.32	73.45	75.85	76.82
Accuracy	70.69	74.25	76.59	78.02
Precision	70.25	73.79	76.10	77.49
F1	70.02	74.03	75.98	77.42
Loss	70.94	74.61	76.87	**78.5**
KLDiv	**71.09**	**74.78**	**77.02**	78.49
Diversity Dilemma (Birds)				
Random	69.49	71.40	72.84	72.77
Accuracy	70.32	72.06	**73.75**	**74.30**
Precision	69.88	71.70	73.42	73.86
F1	69.78	71.57	73.04	73.78
Loss	**70.51**	**72.18**	73.64	74.19
KLDiv	69.98	71.96	73.24	73.84

Table 3. Presents 10-*base* accuracy results for **Simulation** (Section 4.1, top half of this table), **Architecture Swapping** (Section 4.2), and **Diversity Dilemma** (Section 4.3, bottom fourth of this table).

The **Simulation** results (top half of Table 3) prove the effectiveness of our method. The criteria based on **Accuracy** and **Loss** are in all tested cases demonstrably better than randomly selecting which half of classes the images should be added to. While the resulting dataset is not perfect (discussed in Section 4.5), due to potential issues like image diversity in the initial dataset (discussed in Section 4.3), *this experiment proves that our model is effective* for making recommendations based on an initial dataset.

4.2. Architecture Swapping

The clear success of our method invites questions about its weaknesses and limitations. One such question is: *does the model select categories that are good in general, or only categories that are good for the architecture making the selections?* To address this concern, we present the results of an experiment on the Cars dataset where DenseNet-169 networks [16] are used (we choose this DenseNet because its overall accuracy is similar to the ResNet-50) to obtain the metrics and select the categories which receive additional images, while a ResNet-50 is only used to train on the resulting set and evaluate its performance on the test set.

The middle (third quarter) of Table 3 demonstrates that this network-swapping approach not only matches the effectiveness of our initial approach (select and evaluate with ResNet-50), but in many cases exceeds it. The maximum accuracy achieved for each *jump* size is higher, including major improvements for **KL-Divergence**. This provides convincing evidence, across all of our criteria, that *the model does not overfit to a specific network architecture*.

4.3. Diversity Dilemma

Since our method begins with a small dataset, it is possible a "hard" category could be mis-identified as "easy" simply because the 10 images for that category do not accurately represent it. Since this kind of issue would be exposed when testing on a comprehensive set (as seen in sections 4.1 and 4.2), it is clear that limitations in diversity do not prevent our method from outperforming the baseline. Nevertheless, to fully investigate the effects of using this method with a minimally-diverse dataset, we take a pose-limited subset of Birds for an initial dataset. This set has 10 images per class, where the images gathered for each class were nearly identical in terms of pose (as shown in Figure 3). For each class, the 10 images with the smallest pairwise Procrustes distance between keypoint "constellations" (the CUB dataset provides 15 keypoint annotations per image) are selected. We run the experiments from 4.1 again for a new baseline (random selection for adding to the pose-limited set) and each criterion.

The results at the bottom of Table 3 confirm our hypothesis. **Random** now performs worse than it did for the **Simulation** experiments since the 10 initial images for each class are less diverse. In spite of the lack of representativeness, our method is still able to obtain the information necessary to outperform the baseline, doing so by *greater percentages* than before (in **Simulation**). These results suggest our method is not only robust to situations where diversity is limited, but that *it may be even better* in such scenarios.

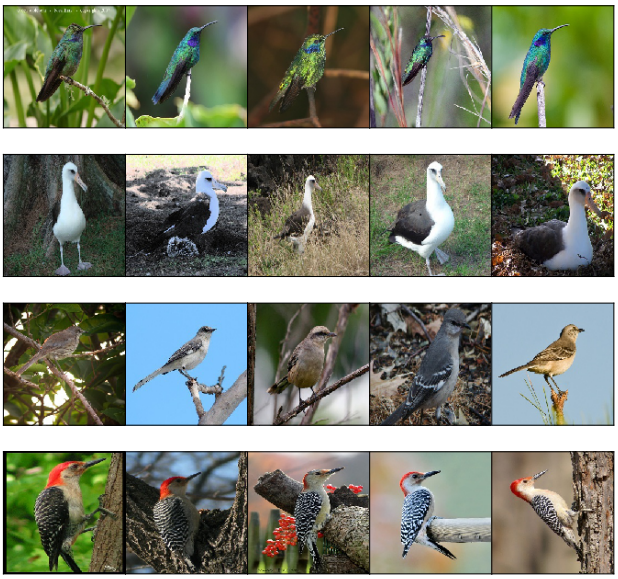

Figure 3. Shows example images from the diversity-lacking, pose-clustered initial dataset described in Section 4.3. From top: green violetear, laysan albatross, mockingbird, red-bellied woodpecker. Notice, for example, that only one of the woodpecker images shows its distinctive, faint red belly.

4.4. Adaptive Jumps

In our other experiments, the number of images added to each class is fixed, specified by the *jump* size. Here, we use the *jump* number only to determine the *total* number of images to add to the chosen categories – the *jump* size times the number of selected categories (98 for Cars). The exact number of images to add for each class is chosen adaptively, adding a number of images proportional to a class-assigned weight. For **Accuracy**, **Precision**, **F1**, and **KL-Divergence**, a class's weight is calculated by subtracting the respective metric for that class from the maximum observed across all classes. For **Loss**, the weight is the category's loss. This way, the classes with worse performance (based on the selected criterion) receive more images. Figure 4 shows the resulting image distributions for our 10-*base* experiments with weights determined using the **Loss** criterion.

	Cars		Birds	
Criteria	5-Jump	10-Jump	5-Jump	10-Jump
Random	70.32	73.45	70.12	72.16
Accuracy	70.36	74.53	**70.65**	72.56
Precision	70.73	74.46	70.30	72.37
F1	**70.85**	**74.67**	70.40	72.32
Loss	70.39	74.44	70.52	**72.79**
KLDiv	69.97	73.89	70.49	72.61

Table 4. Presents 10-*base* accuracy results for **Adaptive Jumps** (Section 4.4).

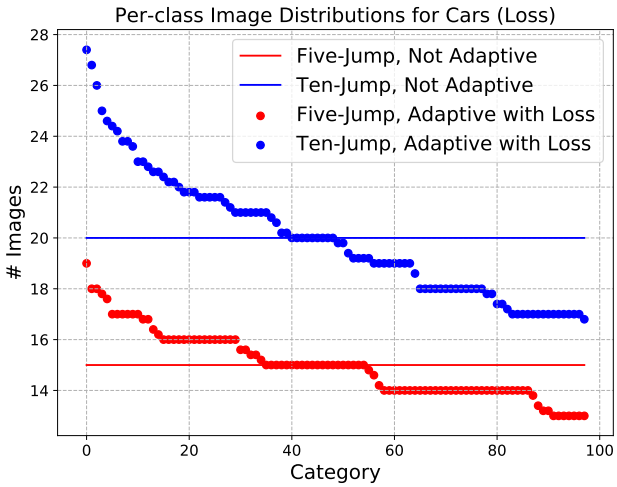

Figure 4. Distributions of per-class image counts (averaged over 5 trials) for the 98 Cars categories that receive images.

Comparing these **Adaptive Jumps** results (in Table 4) to those in Table 3 shows that this approach matches the effectiveness of adding a fixed number of images to each class. The **Accuracy**, **Loss**, and **F1** criteria seem to be particularly well-suited to adding images this way. These results also make clear that intelligent category selection has a much greater effect on accuracy than the modest difference that may result from adaptively varying the exact number of images added to each selected category.

4.5. Oracle Performance

These experiments, unlike those in previous subsections, are not intended to mimic the real-world application of our method. Previously, we limited the evaluation of the metrics/criteria to the test parts of each fold. In this experiment, the networks are trained on the complete initial dataset (a subset of the training portion of an FGVC dataset) and the metrics/criteria are evaluated on the *complete test portion* of the FGVC dataset, similar to an Oracle. By allowing the model to use this extra information to make its recommendations, we demonstrate the extent to which intelligently selecting categories boosts performance.

Two different methods are compared: **Random**, which serves as a baseline, and **Intelligent**. For **Random**, the half of categories that images are added to is chosen randomly. For **Intelligent**, images are added using the **Accuracy** criterion (from Section 3.2), with classes chosen for each domain based on the results of 10 networks that are trained on the initial dataset (which has the *base* number of images per class) and evaluated on the full FGVC test set. For a given *jump* number, after the appropriate number of images are added to the chosen half of the classes, 10 networks are trained (representing 10 trials) on the resulting dataset and evaluated on the full corresponding FGVC test set.

Imgs/Class		Birds			Cars			Dogs		
B+J	# Images	Random	Intelligent	Diff.	Random	Intelligent	Diff.	Random	Intelligent	Diff.
5+0	980	51.02	-	-	34.38	-	-	56.90	-	-
5+5	1470	58.84	59.79	*+0.95*	50.33	50.87	*+0.65*	59.86	60.95	*+1.09*
5+10	1960	61.56	63.55	*+1.99*	56.56	59.23	*+2.67*	61.07	62.58	*+1.51*
5+15	2450	62.93	65.27	*+2.34*	60.24	63.48	*+3.24*	60.75	62.46	*+1.71*
5+20	2940	63.77	66.69	*+2.92*	63.33	66.96	*+3.63*	60.85	62.36	*+1.51*
10+0	1960	67.52	-	-	63.89	-	-	65.91	-	-
10+5	2450	70.07	71.17	*+1.10*	69.70	70.79	*+1.09*	67.36	67.90	*+0.54*
10+10	2940	71.89	73.146	*+1.57*	73.41	75.01	*+1.60*	67.78	68.44	*+0.66*
10+15	3430	72.92	75.23	*+2.31*	75.75	77.99	*+2.24*	68.37	68.66	*+0.29*
10+20	3920	73.56	75.90	*+2.34*	77.27	79.89	*+2.62*	68.39	68.71	*+0.32*

Table 5. Shows 5-*base* and 10-*base* results for **Oracle Performance** (Section 4.5). B+J refers to the *base* number and *jump* size for the chosen classes. # Images gives the number of images in the resulting dataset. **Diff** gives the difference between **Intelligent** and **Random**.

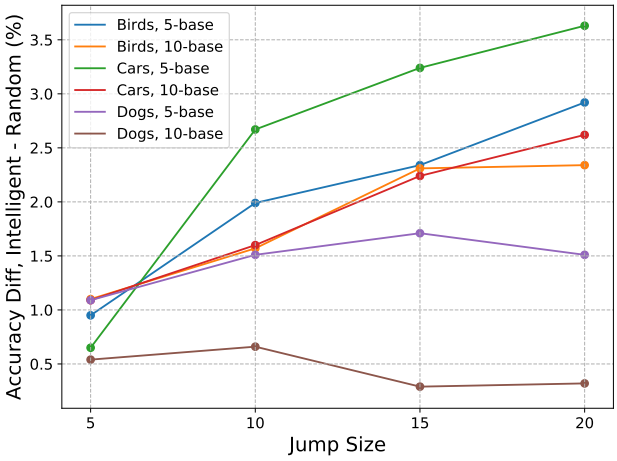

Figure 5. The difference in accuracy between **Intelligent** and **Random** (from Table 5) for each *base* number and each *jump* size. **Intelligent** is superior for all cases.

Table 5 demonstrates that major improvements are seen where the model applies the "intelligent" approach instead "random." A great example is Cars, 5-*base*, 20-*jump*, where the difference in categorization accuracy is huge: 3.63%. These improvements are much greater than what we observe in previous sections. This could mean that since the model uses the final test set to make recommendations, it is simply overfitting to the test set (*this was not possible in previous subsections where the experimental setup was different*). However, we believe it is more likely that the recommendations result in higher accuracy on the final test set because they were made based on more information – evaluating the chosen criterion, **Accuracy**, on 30+ images per class instead of different folds of the same 10 images. While an Oracle may be unrealistic in practice, it clearly demonstrates that the potential of our method is even higher than indicated by the earlier experiments.

Figure 5 highlights two key findings. One is that the differences are greater for 5-*base* than for 10-*base*. This is because, by virtue of having only half as many training images, the 5-*base* datasets have lower starting accuracies than 10-*base*, so there is more room for improvement. The other is that the advantage of **Intelligent** over **Random** is greater with Cars than with Birds, and greater with Birds than with Dogs. This could be due to the differences between intrinsic qualities of cars as opposed to birds and dogs (e.g. cars are rigid), dataset quality, or another factor.

4.6. Granular Analysis

Here, we analyze the effects of adding images to certain classes on the resulting per-class accuracy distributions. We do this in a granular fashion by targeting eighths (octiles) of the categories instead of halves. In these experiments, we identify the effects of adding images to the categories in each of the 8 octiles, not just the octile corresponding to the "hardest" categories. We train and test 10 networks after adding images to a given octile, averaging performance across them. In other regards, the experimental setup is the same as in Section 4.5.

Table 7 provides evidence that adding images to the "hardest" octile of categories (Octile 1) results in higher overall accuracy than adding images to any other octile. As in Section 4.5, differences are greater when the jump size is larger. Figure 6 shows that the improvements in accuracy are almost entirely driven by the categories that received more images. This means that dataset creators can target weak classes earlier on and gather more images to boost their performance. This will help them use their image budget more effectively, enabling them to include more categories in their datasets.

Also of note is that some per-class accuracies don't improve even when those classes receive the additional images. This suggests that some categories may be inherently very challenging to classify. Adding images to Octile 1 re-

Accuracy Diffs	Random	Octile 1	Octile 2	Octile 3	Octile 4	Octile 5	Octile 6	Octile 7	Octile 8
Images Added	20.22	34.75	23.44	22.07	16.46	13.07	11.98	10.50	9.80
Remained the Same	2.68	2.33	5.00	4.00	4.19	5.19	4.93	4.38	4.42

Table 6. Gives results for the Cars 10-*base* 20-*jump* **Granular Analysis** (Section 4.6). The "Images Added" row shows the average accuracy improvement resulting from adding more images for the classes that receive those images, while "Remained the Same" shows the accuracy difference for the classes that did not receive those images. For Octile 1, the 24 classes with the lowest categorization accuracy received 20 additional images. For Octile 2, the next worst 24 received the images, and so forth.

Octile	5-Jump	10-Jump	15-Jump	20-Jump
Random	65.54	66.35	67.37	68.26
1	**66.04**	**67.33**	**68.20**	**69.03**
2	65.67	66.73	67.98	68.04
3	65.44	67.07	67.53	67.95
4	65.71	66.47	67.53	67.90
5	65.86	66.75	67.99	68.93
6	65.51	66.91	67.15	67.66
7	65.32	65.87	66.49	67.48
8	65.11	65.71	66.67	66.93

Table 7. The average categorization accuracies for the Cars 10-*base* **Granular Analysis**. For Octile 1, images are added to the 24 categories corresponding to the lowest accuracy, for Octile 2 the next 24, etc. For **Random**, the 24 categories are chosen randomly.

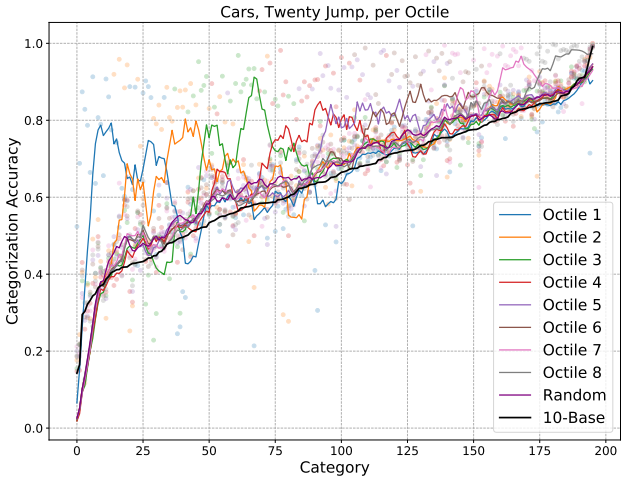

Figure 6. Moving averages for class-wise categorization accuracies for the 10-*base* 20-*jump* **Granular Analysis**. Each of the octile lines refer to the 24 classes to which images were added, where Octile 1 refers to the 24 corresponding to the lowest accuracy. Random refers to adding 20 images to 24 randomly selected classes. 10-*base* shows the accuracies before adding 480 images.

sults in less improvement for "Remained the Same" than adding to any other Octile, as shown in Table 6 – adding to any other octile in a concentrated manner results in significant accuracy improvements *for the classes that didn't receive additional images*, with Octile 5's improvement for those classes nearly doubling that of Random. This suggests our method may be successful because it focuses on *specific* segments of the accuracy distribution, not just because it focuses on *low-accuracy* segments.

Additional insight comes from comparing adding to octiles versus adding to halves. Consider that octiles 10-*base* 20-*jump* and halves 10-*base* 5-*jump* both start and end with the same number of images – 1960 and 2450. However, intelligently spreading images out through the whole worst half yielded an accuracy of 70.97%, while adding to Octile 1 was limited to 69.03%. In fact, at 69.70%, even adding 5 images to a *random* half of the categories is better for overall accuracy than adding to Octile 1. What this indicates is that there is a significant advantage in distributing the images more evenly, instead of adding only to a few classes.

These challenges and trade-offs could affect any approach. Adding images to a specific category, while helping the network accurately label images from that category, could negatively impact performance in a few different ways. First, adding those extra images could start to bias the network towards the class that received the images. Second, while performance on the chosen class may be enhanced, the potential of greater accuracy improvement for other classes (if the extra images had been added to them) is lost. Third, some classes are so intrinsically challenging that adding more images to them may not even significantly improve network performance.

5. Conclusion

As FGVC grows in prominence, researchers will be increasingly keen to apply it on the Edge (as opposed to in the Cloud), and it will be important to establish a smart approach for building datasets in the wild. In this paper, we proposed a model that maximizes accuracy for a dataset building task. We proved its ability to generalize across different domains, architectures, and dataset qualities. We conducted analysis which provides intuition for adapting our method for uses beyond those addressed here. This work helps enable FGVC applications to extend to domains where images have historically *not* been "easy" to acquire.

Acknowledgements We gratefully acknowledge the support of the National Science Foundation (NSF) under Grant No. IIS1651832 and the generous donation from NVIDIA Corporation of multiple GPUs used in this research.

References

[1] Amazon. Mechanical Turk. http://www.mturk.com/, 2019. 2

[2] T. Berg, J. Liu, S. W. Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur. Birdsnap: Large-scale Fine-grained Visual Categorization of Birds. In *CVPR*, 2014. 2

[3] T. L. Berg, A. Sorokin, G. Wang, D. A. Forsyth, D. Hoiem, I. Endres, and A. Farhadi. It's All About the Data. *Proceedings of the IEEE*, 98(8):1434–1452, 8 2010. 3

[4] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual Recognition with Humans in the Loop. In *ECCV*. 2010. 1, 2

[5] R. A. Butler. Countries with the most bird species. https://rainforests.mongabay.com/03birds.htm, 2019. 2

[6] Y. Chen, Y. Bai, W. Zhang, and T. Mei. Destruction and construction learning for fine-grained image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[7] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 3

[9] J. Deng, J. Krause, A. Berg, and L. Fei-Fei. Hedging Your Bets: Optimizing Accuracy-Specificity Trade-offs in Large Scale Visual Recognition. In *CVPR*, 2012. 3

[10] T. Devries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. 1

[11] A. Dubey, O. Gupta, R. Raskar, and N. Naik. Maximum-Entropy Fine Grained Classification. In *NeurIPS*, 2018. 1

[12] W. Ge, X. Lin, and Y. Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[13] R. G. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In *NIPS*, 2011. 3

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 4

[15] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 3

[16] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 5

[17] H. Inoue. Data augmentation by pairing samples for images classification. *CoRR*, abs/1801.02929, 2018. 1

[18] L. Karlinsky, J. Shtok, S. Harary, E. Schwartz, A. Aides, R. Feris, R. Giryes, and A. M. Bronstein. Repmet: Representative-based metric learning for classification and few-shot object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[19] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for Fine-Grained Image Categorization. In *CVPR Workshops (FGVC)*, 2011. 1

[20] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec 2014. 4

[21] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The Unreasonable Effectiveness of Noisy Data for Fine-Grained Recognition. In *ECCV*, 2016. 2

[22] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D Object Representations for Fine-Grained Categorization. In *ICCV Workshops (3DRR)*, 2013. 1

[23] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN Models for Fine-Grained Visual Recognition. In *ICCV*, 2015. 1

[24] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive Neural Architecture Search. In *ECCV*, 9 2018. 3

[25] C. Long and G. Hua. Multi-class Multi-annotator Active Learning with Robust Gaussian Process for Visual Recognition. In *ICCV*, 2015. 3

[26] S. Maji, E. Rahtu, J. Kannala, M. B. Blaschko, and A. Vedaldi. Fine-Grained Visual Classification of Aircraft. *arXiv.org*, 2013. 1

[27] M.-E. Nilsback and A. Zisserman. Automated Flower Classification over a Large Number of Classes. In *ICVGIP*, 2008. 1

[28] C. A. of Sciences and the National Geographic Society. iNaturalist.org Website. http://www.inaturalist.org/, 2019. 1

[29] V. Ordonez, J. Deng, Y. Choi, A. C. Berg, and T. Berg. From large scale image categorization to entry-level categories. In *ICCV*, 2013. 3

[30] V. Ordonez, W. Liu, J. Deng, Y. Choi, A. C. Berg, and T. L. Berg. Predicting Entry-Level Categories. *IJCV*, 2015. 3

[31] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 4

[32] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 2009. 3

[33] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. 1

[34] A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *CVPR Workshops*, 2008. 2

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[36] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *CVPR*, 2019. 3

[37] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie. Building a Bird Recognition App and Large Scale Dataset With Citizen Scientists: The Fine Print in Fine-Grained Dataset Collection. In *CVPR*, 2015. 2, 3

[38] G. Van Horn, S. Branson, S. Loarie, S. Belongie, and P. Perona. Lean Multiclass Crowdsourcing. In *CVPR*, 2018. 3

[39] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The INaturalist Species Classification and Detection Dataset. In *CVPR*, 2018. 3

[40] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc., 2016. 1

[41] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, California Institute of Technology, 2011. 1, 2, 3

[42] Y. Wang, V. I. Morariu, and L. S. Davis. Learning a Discriminative Filter Bank within a CNN for Fine-grained Recognition. In *CVPR*, 2018. 1

[43] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7278–7286, 2018. 1

[44] P. Welinder, S. Branson, P. Perona, and S. Belongie. The Multidimensional Wisdom of Crowds. In *NIPS*. 2010. 3

[45] Q. Xuan, H. Xiao, C. Fu, and Y. Liu. Evolving convolutional neural network and its application in fine-grained visual categorization. *IEEE Access*, 6:31110–31116, 2018. 2

[46] J. Yim, D. Joo, J. Bae, and J. Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141, 2017. 3

[47] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. 3

[48] H. Zheng, J. Fu, T. Mei, and J. Luo. Learning Multi-attention Convolutional Neural Network for Fine-Grained Image Recognition. In *ICCV*, 2017. 1

[49] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *arXiv preprint arXiv:1905.10083*, 2019. 3

[50] B. Zoph and Q. V. Le. Neural Architecture Search with Reinforcement Learning. In *ICLR*, 2017. 3