Synthetic Examples Improve Generalization for Rare Classes

Sara Beery*◇, Yang Liu*◇, Dan Morris^, Jim Piavis†,
Ashish Kapoor†, Markus Meister◇, Neel Joshi†, Pietro Perona◇

California Institute of Technology◇
1200 E California Blvd, Pasadena, CA 91125

Microsoft AI for Earth^, Microsoft Research†
14820 NE 36th Street, Redmond, WA, 98052

Abstract

The ability to detect and classify rare occurrences in images has important applications – for example, counting rare and endangered species when studying biodiversity, or detecting infrequent traffic scenarios that pose a danger to self-driving cars. Few-shot learning is an open problem: current computer vision systems struggle to categorize objects they have seen only rarely during training, and collecting a sufficient number of training examples of rare events is often challenging and expensive, and sometimes outright impossible. We explore in depth an approach to this problem: complementing the few available training images with ad-hoc simulated data.

Our testbed is animal species classification, which has a real-world long-tailed distribution. We present two natural world simulators, and analyze the effect of different axes of variation in simulation, such as pose, lighting, model, and simulation method, and we prescribe best practices for efficiently incorporating simulated data for real-world performance gain. Our experiments reveal that synthetic data can considerably reduce error rates for classes that are rare, that as the amount of simulated data is increased, accuracy on the target class improves, and that high variation of simulated data provides maximum performance gain.

1. Introduction

[1] In recent years computer vision researchers have made substantial progress towards automated visual recognition across a wide variety of visual domains [57, 20, 51, 68, 47, 67, 8]. However, applications are hampered by the fact that in the real world the distribution of visual classes is long-tailed, and state-of-the-art recognition algorithms struggle to learn classes with limited data [69]. In some cases (such as recognition of rare endangered species) classifying rare occurrences correctly is crucial. Simulated data, which is plentiful, and comes with annotation "for free", has been shown to be useful for various computer vision tasks [70, 50, 32, 53, 24, 61, 55, 49, 28, 26, 36]. However, an exploration of this approach in a long-tailed setting is still missing (see Section 2.4).

As a testbed, we focus on the effect of simulated data augmentation on the real-world application of recognizing animal species in camera trap images. Camera traps are heat- or motion-activated cameras placed in the wild to monitor animal populations and behavior. The processing of camera trap images is currently limited by human review capacity; consequently, automated detection and classification of animals is a necessity for scalable biodiversity assessment. A single sighting of a rare species is of immense importance. However, training data of rare species is, by definition, scarce. This makes this domain ideal for studying methods for training detection and classification algorithms with few training examples. We utilize a technique from [8] which tests performance at camera locations both seen (cis) and unseen (trans) during training in order to explicitly study generalization (see Section 3.1 for a more detailed explanation).

We introduce two novel natural world simulators based on popular 3D game development engines for generalizable, realistic and efficient synthetic data generation. We investigate the use of simulated data as augmentation during training, and how to best combine real data for common classes with simulated data for rare classes to achieve optimal performance across the class set at test time. We consider four different data simulation methods (see Fig.1) and compare the effects of each on classification performance. Finally, we analyze the effect of both increasing the number of simulated images and controlling for axes of variation to provide best practices for leveraging simulated data for real-world performance gain on rare classes.

2. Related work

2.1. Visual Categorization Datasets

Large and well-annotated public datasets allow scientists to train, analyze, and compare the performance of different methods, and have provided large performance im-

[1] * denotes equal contribution

(a) Real Camera Traps **(b)** TrapCam-Unity **(c)** TrapCam-AirSim **(d)** Sim on Empty **(e)** Real on Empty

Figure 1: **Day and night examples for each simulation method.** We compare four different simulation methods and compare the effects of each on classification performance.

provements over traditional vision approaches [64, 34, 31]. The most popular datasets used for this purpose are ImageNet, COCO, PascalVOC, and OpenImages, all of which are human-curated from images scraped from the web [17, 44, 21, 39]. These datasets cover a wide set of classes across both the manufactured and natural world, and are usually designed to provide "enough" data per class to avoid the low-data regime. More recently researchers have proposed datasets that focus specifically on long-tailed distributions [68, 8, 41]. The Caltech Camera Traps dataset [8] introduced the challenge of learning from limited locations, and generalizing to new locations.

2.2. Handling Imbalanced Datasets

Imbalanced datasets lead to bias in algorithm performance toward well-represented classes [12]. Algorithmic solutions often use a non-uniform cost per misclassification via weighted loss [19, 30, 29]. One example, focal loss, was recently proposed to deal with the large foreground/background imbalance in detection [43].

Data solutions employ data augmentation, either by 1) over-sampling the minority classes, 2) under-sampling the majority classes, or 3) generating new examples for the minority classes. When using mini-batch gradient descent, oversampling the minority classes is similar to weighted loss. Under-sampling the majority classes is non-ideal, as this reduces information about common classes. Our paper falls into the third category: generating new training data for rare classes. Data augmentation via pre-processing, using affine and photometric transformations, is a well-established tool for improving generalization [40, 33]. Data generation and simulation have begun to be explored as data augmentation methods, see Section 2.4.

Algorithmic and data solutions for imbalanced data are complementary, algorithmic advances can be used in conjunction with augmented training data.

2.3. Low-shot Learning

Low-shot learning attempts to learn categories from few examples [42]. Wang and Herbert [71] do low-shot classification by regressing from small-dataset classifiers to large-dataset classifiers. Hariharan and Girshick [27] look specifically at ImageNet, using classes that are unbalanced, some with large amounts of training data, and some with little training data. Metric learning learns a representation space where distance corresponds to similarity, and uses this as a basis for low-shot solutions [15]. We consider the low-shot regime with regard to *real* data for our rare target class, but investigate the use of added synthetic data based on a human-generated articulated model of the unseen class during training instead of additional class-specific attribute labels at training and test time. This takes us outside of the traditional low-shot framework into the realm of domain transfer from simulated to real data.

2.4. Data Augmentation via Style Transfer, Generation, and Simulation

Image generation via generative adversarial networks (GANs) and recurrent neural networks (RNNs), as well as style transfer and image-to-image translation have all been considered as sources for data augmentation [11, 25, 35, 52, 66, 45, 74]. These techniques require large amounts of data to generate realistic images making them non-ideal solutions for low-data regimes. Though conditional generation allows for class-specific output, the results can be difficult to interpret or control.

Graphics engines such an Unreal [7, 72] and Unity [6] leverage the expertise of human artists and complex physics

models to generate photorealistic simulated images, which can be used for data augmentation. Because ground truth is known at generation, simulated data has proved particularly useful for tasks requiring detailed and expensive annotation, such as keypoints, semantic segmentations, or depth information [70, 50, 32, 53, 24, 61, 55, 49, 28]. Varol *et al*. [70] use synthetically-generated humans placed on top of real image backgrounds as pretraining for human pose estimation, and suggest fine-tuning a synthetically-trained model on real data. [61] uses a combination of unlabeled real data and labeled simulated data of the same class to improve real-world performance on an eye-tracking task by using GANs [24]. This method requires a large number of unlabled examples from the target class. [50, 32, 53] find that simulated data improves detection performance, and the degree of realism and variability of simulation affects the amount of improvement. They consider only small sets of non-deformable man-made objects. Richter *et al*. [55] showed that a segmentation model for city scenes trained with a subset of their real dataset and a large synthetic set outperforms a model trained with the full real dataset. [49] proposes a dataset and benchmark for evaluating models for unsupervised domain transfer from synthetic to real data with all-simulated training data, as opposed to simulated data only for rare classes. While this literature is encouraging, a number of questions are left unexplored. The first is a careful analysis of when simulated data is useful and, in particular, if it is useful in generalizing to new scenarios. Second, whether simulated data can be useful in highly complex and relatively unpredictable scenes such as natural scenes, as opposed to indoors and urban scenes. Third, whether it is just the synthetic objects or also the synthetic environments that contribute to learning.

2.5. Simulated Datasets

Previous efforts on synthetic dataset generation focus on non-deformable man-made objects and indoor scenes [62, 58, 73, 32, 53, 38], human pose/actions [70, 16], or urban scenes [56, 22, 55, 18, 26, 36].

Bondi *et al*. [10] previously released the AirSim-w data simulator within the domain of wildlife conservation, focused on creating aerial infrared imagery. The resolution and quality of the assets is designed to replicate data from 100 meters in the air, but is not realistic close-up. We contribute the first image data generators specifically for the natural world with the ability to recreate natural environments and generate near-photorealistic images of animals within the scene, including real-world nuisance factors such as challenging pose, lighting, and occlusion.

3. Data and Simulation

3.1. Real Data

Our real-world training and test data comes from the Caltech Camera Traps (CCT) dataset [8]. CCT contains $243,187$ images from 140 camera trap locations covering 30 classes of animals, curated from data provided by the United States Geological Survey and the National Park Service. We follow the CCT-20 data split laid out in [8], which was explicitly designed for in-depth generalization analysis. The split uses a subset of $57,868$ images from 20 camera locations covering 15 classes in CCT to simultaneously investigate performance on locations seen during training and generalization performance to new locations. Bounding-box annotations are provided for all images in CCT-20, whereas the rest of CCT has only class labels. In the CCT-20 data split, **cis-locations** are defined as locations seen during training and **trans-locations** as locations not seen during training (see Fig.3). Nine locations are used for trans-test data, one location for trans-validation data, and data from the remaining 10 locations is split between odd and even days, with odd days as cis-test data and even days as training and cis-validation data (a 95% of data from even days for training, 5% for testing).

To study the effect of simulated data on rare species, we focus on deer, which are rare in CCT-20, with only 44 deer examples out of the $13,553$ images in the training set (see Fig.3). To focus on the performance of a single rare class, we remove the other two rare classes in CCT-20: badgers and foxes. We note that there are no deer images in the established CCT-20 trans sets. In reality, deer are far from uncommon: unlike a truly rare species, there exist sufficient images of deer in the CCT dataset outside of the CCT-20 locations to rigorously evaluate performance. To facilitate deeper investigation of generalization we have collected bounding-box annotations for an additional 16K images from CCT across 65 new locations, which we add to the trans-validation and trans-test sets to cover a wider variety of locations and classes (including deer). We call this augmented trans set *trans+* (see Fig.3) and will release the annotations at publication. To further analyze generalization, we also test on data containing deer from the iNaturalist 2017 dataset [68], which represents a domain shift to human-captured and human-selected photographs. We consider *Odocoileus hemionus* (mule deer) and *Odocoileus* virginianus (white-tailed deer) images from iNaturalist, the two species of deer seen in the CCT data. In Supplementary Material we show results on an additional class, wolf.

3.2. Synthetic Data

To assess generality we leverage multiple collections of woodland and animal models to create two simulation environments, which we call TrapCam-Unity and TrapCam-AirSim. Both simulation environments and source code to generate images will be provided publicly, along with the data generated for this paper. To synthesize daytime images we varied the orientation of the simulated sun in both azimuth and elevation. To create images taken at night we used a spotlight attached to the simulated camera to sim-

(a) Training images

(b) Cis test images

(c) Trans+ test images

(d) iNaturalist images

Figure 2: **Cis vs. Trans:** The cis-test data can be very similar to the training data: animals tend to behave similarly at a single location even across different days, so the images collected of each species are easy to memorize intra-location. The trans data has biases towards specific angles and lighting conditions that are different from those in the cis locations, and as such is very hard to learn from the training data. iNaturalist data represents a domain shift to human-curated images.

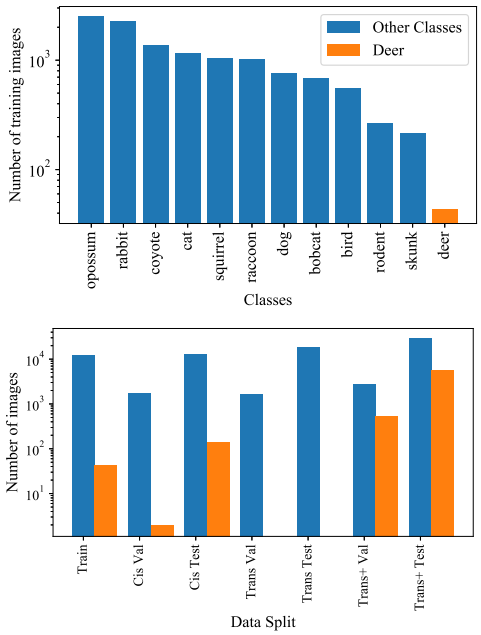


Figure 3: **(Top) Number of training examples for each class.** Deer are rare in the training locations from the CCT-20 data split. We focus on deer as a test species in order to investigate whether we can improve performance on a "rare" class. Since deer are not rare at other camera locations within the CCT dataset, we have enough test data to thoroughly evaluate the effect. **(Bottom) Number of examples for each data split, for deer and other classes.** In the CCT-20 data split there were no trans examples of deer. We added annotations to the trans val and test sets for an additional 16K images across 65 new locations from CCT, including 6K examples of deer. We call these augmented sets *trans+*.

ulate a white-light or IR flash and qualitatively match the low color saturation of the nighttime images. To simulate animals' eyeshine (a result of the reflection of camera flash from the back of the eye), we placed small reflective balls on top of the eyes of model animals.

TrapCam-AirSim. We create a modular natural environment within Microsoft AirSim [60] that can be randomly populated with flora and fauna. The distribution and types of trees, bushes, rocks, and logs can be varied and randomly seeded to create a diverse set of landscapes, from an open plain to a dense forest. We used various off-the-shelf components such as an animal pack from Epic Studios [1] (Animals Vol 01: Forest Animals by GiM [2]), background terrain also from Unreal Marketplace [7], vegetation from SpeedTree [4], and rocks/obstructions from Megascans [3]. The actual area of the environment is small, at 50 meters, but the modularity allows many possible scenes to be built.

TrapCam-Unity. Unity 3D game development engine is a popular game development tool that offers realistic graphics, real time performance and abundant 3D assets. We take advantage of the "Book of The Dead" environment [5], a near-photorealistic, open-source forest environment published by Unity to demonstrate its high definition rendering pipeline. This off-the-shelf environment is large and rich in details, it has a diversity of subregions with significantly different statistics. We change the lighting and move throughout this large, static environment to collect data with various background scenes. We make use of 17 animated deer models from five off-the-shelf model sets, purchased from Unity Asset Store and originally developed for game development, including the GiM models used in TrapCam-AirSim. A single gaming PC (Core i7 5820K, 16GB RAM, GTX 1080Ti) generates over 300,000 full-HD images with pixel-level instance annotation per day and the throughput linearly scales to additional machines.

Simulated animals on empty images. Similar to the data generated in [70], we generate synthetic images of deer by rendering deer on top of real camera trap images containing no animals, which we call *Sim on Empty* (see Fig.1). We first generate animal foreground images by randomizing the location, orientation in azimuth, pose and illumination of the deer, then paste the foreground images on top of the real empty images. A limitation is that the deer are not in realistic relationships or occlusion scenarios with the environment around them. We also note that the empty images used to construct this data come from both cis and trans locations, so Sim on Empty contains information about test-set backgrounds unavailable in the purely simulated sets. This choice is based on current camera trap literature, which first detects the presence of any animal, and then determines animal species [47, 8]. After the initial animal detection step, the empty images are known and can be utilized.

Segmented animals on empty images. We manually segment the 44 examples of deer from the training set and paste them at random on top of real empty camera trap images, which we call *Real on Empty* (see Fig.1). This allows us to analyze whether the generalization challenge is related to memorizing the training deer+background or memorizing the training deer regardless of background. Similar to the Sim on Empty set, these images do not have realistic foreground/background relationships and the empty images come from both cis and trans locations.

4. Experiments

Beery, *et al*. [8] showed that detecting and localizing the presence of an "animal" (where all animals are grouped into a single class) both generalizes well to new locations and improves classification performance. We focus on classification of cropped ground-truth bounding boxes as opposed to training multi-class detectors in order to disambiguate classification and detection errors. We specifically investigate how added synthetic training data for rare classes effects model performance on both rare and common classes.

We find that the Inception-Resnet-V2 architecture [63] works best for the cropped-box classification task, based on performance comparison across architectures (see Supplementary Material). Most classification systems are pretrained on Imagenet, which contains animal classes. To ensure that our "rare" class is truly something the model is unfamiliar with, as opposed to something seen in pretraining, we pretrain our classifiers on *no-animal ImageNet*, a dataset we define by removing the "animal" subtree (all classes under synset node n00015388) from ImageNet. We use an initial learning rate of 0.0045, RMSprop with a momentum of 0.9 [65], and a square input resolution of 299. We employ random cropping (containing at least 65% of the region), horizontal flipping, color distortion, and blur as data augmentation. Model selection is performed using early stopping based on trans+ validation set performance [9].

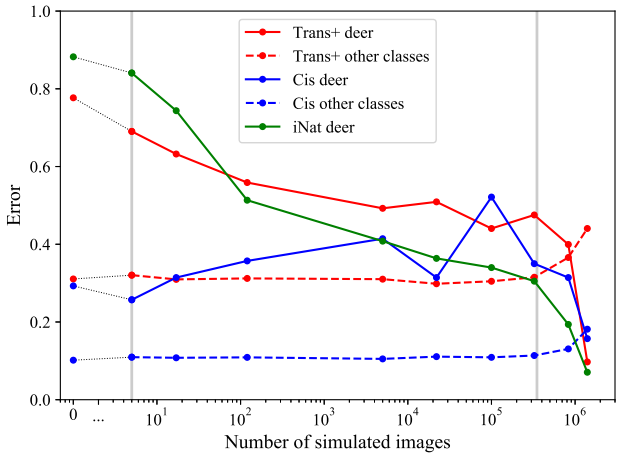
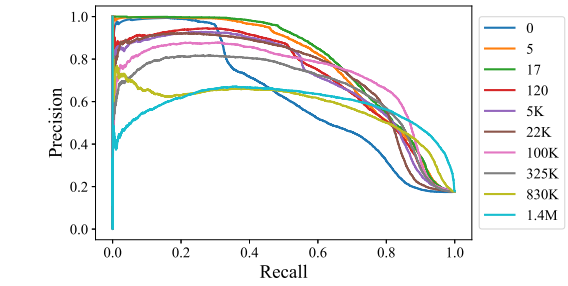


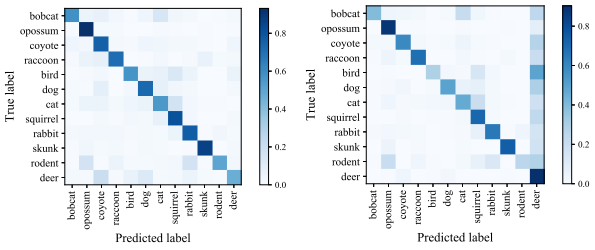
Figure 4: **Error as a function of number of simulated images seen during training. We divide this plot into three regions.** The leftmost region is the baseline performance with no simulated data, shown at x=0 (Note x-axis is in log scale). In the middle region, additional simulated training data increases performance on the rare class and does not harm the performance of the remaining classes (trend lines are visualized). The rightmost region, where many simulated images are added to the training set, results in a biased classifier, hurting the performance of the other classes (see Fig.5 (b-c) for details). We compare the class error for "deer" and "other classes" in both the "cis" and "trans+" testing regimes. Lines marked "deer" use only the deer test images for the error computation. Lines marked "other classes" use all the images in the other classes (excluding deer) for the error computation. Error is defined as the number of incorrectly identified images divided by the number of images.

4.1. Effect of increase in simulated data

We explore the trade-off in performance when increasing the number of simulated images, from 5 to 1.4 million, spanning 5 log units (see Fig.4). Very little simulated data is needed to see a trans+ performance boost: with as few as 5 simulated images we see a 10% decrease in per-class error on trans+ deer, with $< 0.5\%$ increase in average per-class error on the other trans+ classes. As we increase the number of simulated images, trans+ performance improves: with 100K simulated images we see a 39% decrease in trans+ deer error, with $< 0.5\%$ increase in error for the other trans classes. There exists some threshold (> 325K) where, if passed, an increase in simulated data noticeably biases the classifier towards the deer class (see Fig.5): with 1.4 million simulated images, our trans+ deer error decreases by 88%, but it comes at the cost of a 13% increase in average per-class error across the other classes. At this point there is an overwhelming class prior towards deer: the next-largest



(a) Trans+ deer precision-recall curves



(b) Confusion matrix: 100K (c) Confusion matrix: 1.4M

Figure 5: **(a) Trans+ PR curves for the deer class:** Note the development of a biased classifier as we add simulated training data. The baseline model (in blue) has high precision but suffers low recall. The model trained with 1.4M simulated images (in grey) has higher recall, but suffers a loss in precision. **(b-c) Evidence of a biased classifier:** Compare the deer column in the confusion matrices, the model trained with 1.4M simulated images predicts more test images as deer.

class at training time would be opossums with $2,514$ images, 3 orders of magnitude less.

Unsurprisingly, cis deer performance decreases with added simulated data. Although the images were taken on different days (train from even days, cis-test from odd days) the animals captured were to some extent creatures of habit. Thus, training and test images can be nearly identical from within the same locations (see Fig.2). Almost all cis test deer images have at least one visually similar training image. As simulated data is added at training time, the model is forced to learn a more complex, varied representation of deer. As a result, we see cis deer performance decrease. To quantify robustness, we ran the 100K experiment three times. We found that trans+ deer error had a standard deviation of 2% and cis deer error had a standard deviation of 4%, whereas the average error across other classes had a standard deviation of 0.2% for both cis and trans.

We also investigate performance on deer images from iNaturalist [68], which are individually collected by humans and are usually relatively centered and well-focused (and therefore easier to classify) but represent a domain shift (see Fig.2). Adding simulated data improves performance on the iNaturalist deer images (see Fig.4), demonstrating the

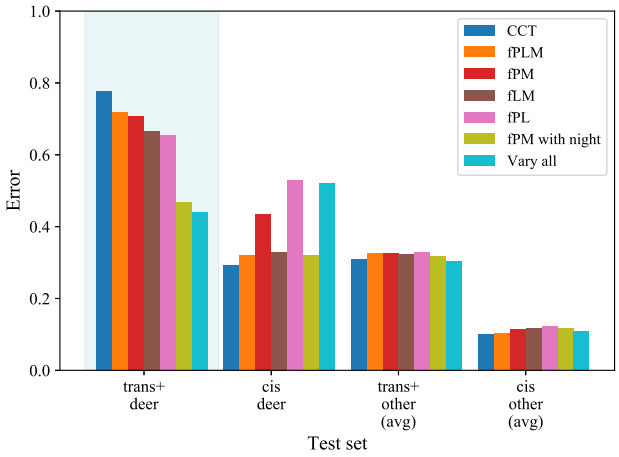


Figure 6: **Error as a function of variability of simulated images seen during training: 100K simulated deer images.** Error is calculated as in Fig.4, and all data is from TrapCam-Unity. Trans+ deer performance is highlighted. In the legend "CCT" means the model was trained only on the CCT-20 training set with no added simulated data. "P" means "pose," "L" means "lighting," and "M" means "model," while the prefix "f" for "fixed" denotes which of these variables were controlled for a particular experiment. For example "fPM" means the pose and the animal model were held fixed, while the lighting was allowed to vary. The variability of simulated data is extremely important, and that while all axes of variability matter, simulating nighttime images has the largest effect.

robustness and generality of the representation learned.

4.2. Effect of variation in simulation

In order to understand which aspects of the simulated data are most beneficial, we consider three dimensions of variation during simulation: pose, lighting, and animal model. Using the TrapCam-Unity simulator, we generate 100K daytime simulated images for each of these experiments. As a control, we create a set of data where the pose, lighting, and animal model are all fixed. We then create sets with varied pose, varied lighting, and varied animal model, each with the other variables held fixed. An additional set of data is generated varying all of the above. Unsurprisingly, widest variation results in the best trans+ deer performance. The individual axes of variation do have an effect of performance, and some are more "valuable" than others (see Fig.6). There are many more dimensions of variation that could be explored, such as simulated motion blur or variation in camera perspective. For CCT data, we find adding simulated nighttime images has the largest effect on performance. We have determined that for deer 49% of training images, 53% of cis test images, and 56% of trans+ test images were captured at night, using either IR or white flash.

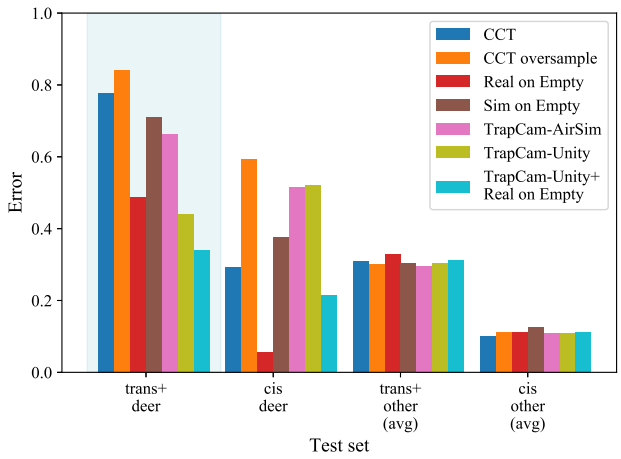


Figure 7: **Error as a function of simulated data generation method: 100K simulated deer images.** Per-class error is calculated as in Fig.4. Trans+ deer performance is highlighted. Oversampling decreases performance, and there is a large boost in performance from incorporating real segmented animals on different backgrounds (Real on Empty). TrapCam-Unity with everything allowed to vary (model, lighting, pose, including nighttime simulation) gives us slightly better trans+ performance, without requiring additional segmentation annotations. Combining Real on Empty with TrapCam-Unity (50K of each) gives us the best trans+ deer performance.

Simulating only daytime images injects a prior towards deer being seen during the day. By training on half day and half night images we match the day/night prior for deer in the data. Not all species occur equally during the day or night, some are strictly nocturnal. Our results suggest that a good strategy is to determine the appropriate ratio of day to night images using your training set and match that ratio when adding simulated data.

4.3. Comparing simulated data generation methods

We compare performance gain from 4 methods of data synthesis, using 100K added deer images for each (see Fig.7. The animal model is controlled (each simulated set uses the same GiM deer model for these experiments) for fair comparison of the efficacy of each generation method. As an additional control, we consider oversampling the rare class. This creates the same sampling prior towards deer without introducing any new information. Oversampling performs worse than just training on the unbalanced training set by causing the model to overfit the deer class to the training images. By manually segmenting out the deer in the 44 training images and randomly pasting them onto empty backgrounds we see a large improvement in performance. Cis error goes down to 6% with this method of data augmentation, which makes sense in the view of the strong similarities between the training and cis-test data (see Fig.2).

Real on Empty and Sim on Empty are able to approximate both "day" and "night" imagery, a deer pasted onto a nighttime empty image is actually a reasonable approximation of an animal illuminated by a flash at night (see Fig.1). They also have the additional benefit of using backgrounds from both cis and trans sets, giving them trans information not provided by the simulated datasets. TrapCam-Unity with all variability enabled is our best-performing model without requiring additional segmentation annotations. If segmentation information is available, Real on Empty combined with TrapCam-Unity (50K of each) improves both cis and trans deer performance: trans deer error decreases to 36% (a 54% decrease compared to CCT only), with $< 2\%$ increase in error on trans other classes.

4.4. Visualizing the representation of data

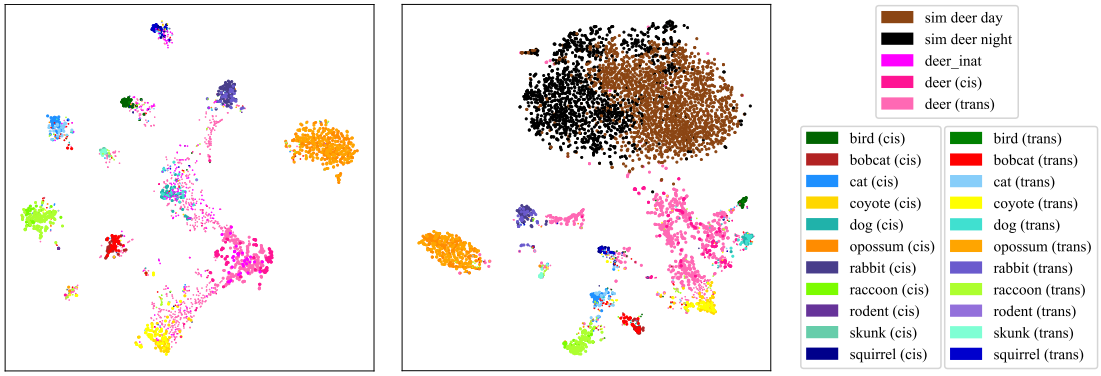
In order to visualize how the network represents simulated data vs. real data, we use PCA and tSNE [46] to cluster the activations of the final pre-logit layer of the network. These visualizations can be seen in Fig.8. Interestingly, the model learns "deer" bimodally: simulated deer are clustered almost entirely separately from real deer, with a few datapoints of each ending up in the opposite cluster. Even though those clusters overlap only slightly, the network is surprisingly able to classify more deer images correctly.

5. Conclusions and Future Work

We present two fast, realistic natural world data simulators based on popular 3D game development engines. Our simulators have 3 major advantages. **First**, they are generalizable. Thanks to the abundant 3D assets available online in the game development community, integrating a new species in a new environment from off the shelf assets is simple and fast. **Second**, not only are the graphics near-photorealistic, the pipeline also generates animals with realistic pose, animation, and interactions with the environment. **Third**, data generation is efficient. A single gaming PC generates over 300,000 full-HD images with pixel-level instance annotation per day and the throughput linearly scales to additional machines.

We explore using the simulated data to augment rare classes during training. Towards this goal, we compare multiple sources of natural-world data simulation, explicitly measure generalization via the cis-vs-trans paradigm, examine trade-offs in performance as the number of simulated images seen during training is increased, and analyze the effect of controlling for different axes of variation and data generation methods.

From our experiments we draw three main lessons. First: using synthetic data can considerably reduce error rates for classes that are rare, and with segmentation annotations we can reduce error rates even further by additionally randomly pasting segmented images of rare classes on empty back-



(a) No simulated deer (b) 1.4M simulated deer

Figure 8: **Visualization of network activations: more deer are classified correctly as we add synthetic data, despite the synthetic data being clustered separately.** The pink points are real deer, the brown are simulated day images and the grey are simulated night images. Large markers are points that are classified correctly, while small markers are points classified incorrectly. The plots were generated by running 200-dimensional PCA over the activations at the last pre-logit layer of the network when running inference on the test sets, and then running 2-dimensional tSNE over the resulting PCA embedding.

ground images. Second: as the amount of simulated data is increased, accuracy on the target class improves. However, with 1000x more simulated data than the common classes, we see negative effects on the performance of other classes due to the high class imbalance. Third: the variation of simulated data generated is very important, and maximum variation provides maximum performance gain.

While an increase in simulated data corresponds to an increase in target class performance, the representation of simulated data overlaps only rarely with real data (see Fig.8). It remains to be studied whether embedding techniques [59], domain adaptation techniques [23, 75], or style transfer [24, 61] could be used to encourage a higher overlap in representation between the synthetic and real data, and if that overlap would lead to an increase in categorization accuracy. Additionally, the bias induced by adding large amounts of simulated data could be addressed with algorithmic solutions such as those in [14, 19, 30, 29]. We have not discussed the drawbacks related to model training with large quantities of synthetic data (epoch time, data storage, etc.). In future, we will explore merging the simulator and classifier so that highly variable synthetic data could be requested "online" without storing raw frames.

Simulation is a fast, interpretable, and controllable method of data generation that is easy to use and easy to adapt to new classes. This allows for an integrated and evolving training pipeline with new classes of interest: simulated data can be generated iteratively based on needs or gaps in performance. Our analysis suggests a general methodology when using simulated data to improve rare-class performance: 1) generate small, variable sets of simulated data (even small sets can drive improvement), 2) add these sets to training and analyze performance to determine ideal ratios and dimensions of variation, 3) take advantage of ease and speed of generation to create an abundance of data based on this ideal distribution, and determine an operating point of number of added simulated images to optimize performance between rare target class and other classes based on the project goal.

Further, the performance gains we have demonstrated, along with the data generation tools we contribute to the community, will allow biodiversity researchers focused endangered species to improve classification performance on their target species. Adding each new species to the simulation tools currently requires the assistance of a graphics artist. However, automated 3D modeling techniques, such as those proposed in [37, 54, 13, 48], might eventually become an inexpensive and practical source of data to improve few-shot learning.

The improvement we have found in rare-class categorization is encouraging, and the release of our data generation tools and the data we have generated will provide a good starting point for other researchers studying imbalanced data, simulated data augmentation, or natural-world domains.

6. Acknowledgements

We would like to thank the USGS and NPS for providing data. This work was supported by NSFGRFP Grant No. 1745301, the views are those of the authors and do not necessarily reflect the views of the NSF. Compute provided by Microsoft AI for Earth and AWS.

References

[1] Epic studios. http://epicstudios.com/. Accessed: 2019-03-21. 4

[2] Forest animals by GiM. https://www.unrealengine.com/marketplace/en-US/animals-vol-01-forest-animals. Accessed: 2019-03-21. 4

[3] Quixel megascans library. https://quixel.com/megascans. Accessed: 2019-03-21. 4

[4] Speedtree. https://store.speedtree.com/. Accessed: 2019-03-21. 4

[5] Unity book of the dead. https://unity3d.com/book-of-the-dead. Accessed: 2019-03-21. 4

[6] Unity game engine. https://unity3d.com/. Accessed: 2019-02-05. 2

[7] Unreal game engine. https://www.unrealengine.com/en-US/what-is-unreal-engine-4. Accessed: 2019-02-05. 2, 4

[8] S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1, 2, 3, 5

[9] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012. 5

[10] E. Bondi, D. Dey, A. Kapoor, J. Piavis, S. Shah, F. Fang, B. Dilkina, R. Hannaford, A. Iyer, L. Joppa, et al. Airsim-w: A simulation environment for wildlife conservation with uavs. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, page 40. ACM, 2018. 3

[11] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017. 2

[12] M. Buda, A. Maki, and M. A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018. 2

[13] T. J. Cashman and A. W. Fitzgibbon. What shape are dolphins? building 3d morphable models from 2d images. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):232–244, 2013. 8

[14] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. Class-balanced loss based on effective number of samples. *arXiv preprint arXiv:1901.05555*, 2019. 8

[15] Y. Cui, F. Zhou, Y. Lin, and S. Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1153–1162, 2016. 2

[16] C. R. de Souza12, A. Gaidon, Y. Cabon, and A. M. López. Procedural generation of videos to train deep action recognition networks. 2017. 3

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2

[18] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 3

[19] C. Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001. 2, 8

[20] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017. 1

[21] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2

[22] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016. 3

[23] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015. 8

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1, 3, 8

[25] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015. 2

[26] S. Han, A. Fafard, J. Kerekes, M. Gartley, E. Ientilucci, A. Savakis, C. Law, J. Parhan, M. Turek, K. Fieldhouse, et al. Efficient generation of image chips for training deep learning algorithms. In *Automatic Target Recognition XXVII*, volume 10202, page 1020203. International Society for Optics and Photonics, 2017. 1, 3

[27] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proc. of IEEE Int. Conf. on Computer Vision (ICCV), Venice, Italy*, 2017. 2

[28] H. Hattori, V. N. Boddeti, K. Kitani, and T. Kanade. Learning scene-specific pedestrian detectors without real data. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3819–3827. IEEE, 2015. 1, 3

[29] H. He, Y. Bai, E. A. Garcia, and S. Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE, 2008. 2, 8

[30] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008. 2, 8

[31] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[32] S. Hinterstoisser, O. Pauly, H. Heibel, M. Marek, and M. Bokeloh. An annotation saved is an annotation earned:

Using fully synthetic training for object instance detection, 2019. 1, 3

[33] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013. 2

[34] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*, 2017. 2

[35] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*, 2016. 2

[36] S. Ji, Y. Shen, M. Lu, and Y. Zhang. Building instance change detection from large-scale aerial images using convolutional neural networks and simulated samples. *Remote Sensing*, 11(11):1343, 2019. 1, 3

[37] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018. 8

[38] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, abs/1712.05474, 2017. 3

[39] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy. Open-images: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2017. 2

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[41] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. Lopez, and J. V. B. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *The 12th European Conference on Computer Vision (ECCV)*, October 2012. 2

[42] F.-F. Li, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006. 2

[43] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2

[44] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[45] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4990–4998, 2017. 2

[46] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 7

[47] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, C. Packer, and J. Clune. Automatically identifying wild animals in camera trap images with deep learning. *arXiv preprint arXiv:1703.05830*, 2017. 1, 5

[48] F. Pahde, M. Puscas, J. Wolff, T. Klein, N. Sebe, and M. Nabi. Low-shot learning from imaginary 3d model. *arXiv preprint arXiv:1901.01868*, 2019. 8

[49] X. Peng, B. Usman, K. Saito, N. Kaushik, J. Hoffman, and K. Saenko. Syn2real: A new benchmark forsynthetic-to-real visual domain adaptation. *arXiv preprint arXiv:1806.09755*, 2018. 1, 3

[50] B. Pepik, R. Benenson, T. Ritschel, and B. Schiele. What is holding back convnets for detection? In *German Conference on Pattern Recognition*, pages 517–528. Springer, 2015. 1, 3

[51] R. Poplin, A. V. Varadarajan, K. Blumer, Y. Liu, M. V. McConnell, G. S. Corrado, L. Peng, and D. R. Webster. Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, page 1, 2018. 1

[52] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2

[53] P. S. Rajpura, H. Bojinov, and R. S. Hegde. Object detection using deep cnns trained on synthetic images, 2017. 1, 3

[54] B. Reinert, T. Ritschel, and H.-P. Seidel. Animated 3d creatures from single-view video by skeletal sketching. In *Graphics Interface*, pages 133–141, 2016. 8

[55] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. *Lecture Notes in Computer Science*, page 102118, 2016. 1, 3

[56] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016. 3

[57] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1

[58] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017. 3

[59] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 8

[60] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018. 4

[61] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 3, 8

[62] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3

[63] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 5

[64] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016. 2

[65] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. 5

[66] T. Tran, T. Pham, G. Carneiro, L. Palmer, and I. Reid. A bayesian data augmentation approach for learning deep models. In *Advances in Neural Information Processing Systems*, pages 2797–2806, 2017. 2

[67] G. van Horn, J. Barry, S. Belongie, and P. Perona. The Merlin Bird ID smartphone app (`http://merlin.allaboutbirds.org/download/`). 1

[68] G. Van Horn, O. Mac Aodha, Y. Song, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist challenge 2017 dataset. *arXiv preprint arXiv:1707.06642*, 2017. 1, 2, 3, 6

[69] G. Van Horn and P. Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017. 1

[70] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 3, 5

[71] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*, pages 616–634. Springer, 2016. 2

[72] Y. Z. S. Q. Z. X. T. S. K. Y. W. A. Y. Weichao Qiu, Fangwei Zhong. Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*, 2017. 2

[73] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018. 3

[74] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017. 2

[75] Y. Zou, Z. Yu, B. Vijaya Kumar, and J. Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 289–305, 2018. 8