Harder or Different?
A Closer Look at Distribution Shift in Dataset Reproduction

Shangyun Lu [*1] **Bradley Nott** [*1] **Aaron Olson** [*1]
Alberto Todeschini [1] **Hossein Vahabi** [1] **Yair Carmon** [2] **Ludwig Schmidt** [1]

Abstract

We introduce CIFAR-10.2, a new variant of the CIFAR-10 image classification dataset. CIFAR-10.2 is derived from the same source as CIFAR-10 and assembled via a similar process. In contrast to the CIFAR-10.1 reproduction, our new dataset is six times larger and contains both a training split (size 10,000) and a test split (size 2,000). This allows us to probe the distribution shift from CIFAR-10 by comparing the performance of classifiers *trained* on the original and new datasets. We decompose the accuracy drop due to shift into "harder" and "different" components based on the performance gain from training on the shifted data. Our experiments show that the "different" component is nearly constant across the 33 models in our testbed. We complement our experiments with a theoretical model that predicts similar behavior.

1. Introduction

Over the past decade, machine learning has made tremendous progress on benchmarks such as CIFAR-10, ImageNet, and SQuAD (Krizhevsky, 2009; Deng et al., 2009; Russakovsky et al., 2015; Rajpurkar et al., 2016). At the same time, small deviations away from existing test sets still lead to substantial performance degradation, which poses obstacles for safety-critical applications (Quionero-Candela et al., 2009; Torralba & Efros, 2011). For instance, small changes in the dataset creation process of CIFAR-10 and ImageNet reduce the accuracy of popular ResNet models by 9% and 12%, respectively (Recht et al., 2019). Since these new test sets are different from the distributions the models are trained on, the following fundamental question arises: *What would happen if we trained the models instead on data from the new test distribution?*

To explore the space of possible answers, we consider two extreme scenarios. In the first scenario, training a model on the new distribution does not improve the test performance at all. We then say that the new distribution is purely *harder* than the original distribution for this model. This can happen if, for example, the new distribution contains a larger portion of blurry objects or if the model architecture cannot capture aspects of the new distribution.

In the second scenario, training on the new distribution recovers the entire performance drop. We then say that the new distribution is merely *different* for this model, for instance because the new distribution contains images from different perspectives or with different backgrounds. Performance drops due to "harder" distributions shift may be inevitable for current techniques, while drops due to "different" distribution shifts can at least be addressed by changing the training data. It is therefore important to quantify where distribution shifts stand in the spectrum between "harder" and "different."

We experimentally instantiate our different vs. harder framework in the context of a distribution shift stemming from a CIFAR-10 reproduction. Similarly to the CIFAR-10.1 test set introduced in (Recht et al., 2019), we build a reproduction of CIFAR-10 by following the original dataset creation process. However, our dataset, which we call CIFAR-10.2, is six times larger than CIFAR-10.1. This allows us to form a train-test split of size 10,000 and 2,000, respectively. As in CIFAR-10.1, there is an accuracy drop when testing CIFAR-10 models on the CIFAR-10.2 test set. Using our CIFAR-10.2 training set, we find evidence that CIFAR-10.2 is *both harder and different* than CIFAR-10.

For example, a ResNet-32 trained on 10,000 images from CIFAR-10 achieves an accuracy of 68.4% on CIFAR-10.2, compared to an accuracy of 83.0% on CIFAR-10. Training the model on 10,000 images from CIFAR-10.2 increases the accuracy to 73.8%. This 5.4% improvement indicates that CIFAR-10.2 is different, but the remaining 9% show that CIFAR-10.2 is also harder. Interestingly, we find that the roughly 5% "different" component of the accuracy drop is a consistent trend across the 33 models in our testbed spanning a wide range of accuracies. Models with a smaller

[*]Equal contribution [1]University of California, Berkeley [2]Stanford University. Correspondence to: Ludwig Schmidt <ludwig@berkeley.edu>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

accuracy drop on CIFAR-10.2 make progress mainly by reducing the "harder" component. We exhibit a similar trend in simulations with a simple Gaussian data model.

We hope that our new dataset CIFAR-10.2 and the distinction between harder and different distributions will be useful for future research. CIFAR-10.2 is available at https://github.com/modestyachts/cifar-10.2.

2. Formal setup

We now formally describe our classification protocol and the distribution shift metrics of interest. We fix a learning rule $\mathcal{A}$ (a combination of model architecture and training algorithm) that takes in a training set sampled i.i.d. from a distribution D and outputs a classifier. To measure the accuracy of this classifier, we sample a test set from a (possibly different) distribution D' and evaluate the classifier on it. We denote the resulting accuracy by $\mathrm{acc}^{\mathcal{A}}_{D \to D'}$.[1]

To measure the performance change when training on distribution D and testing on distribution D', it natural to consider the accuracy gap $\mathrm{acc}^{\mathcal{A}}_{D \to D} - \mathrm{acc}^{\mathcal{A}}_{D \to D'}$, i.e., the performance loss a classifier suffers when trained on distribution D and tested on D'. In this work, we seek a more fine-grained understanding of this accuracy gap. To this end, we decompose the quantity into two terms:

$$\mathrm{acc}^{\mathcal{A}}_{D \to D} - \mathrm{acc}^{\mathcal{A}}_{D \to D'} = \underbrace{\mathrm{acc}^{\mathcal{A}}_{D \to D} - \mathrm{acc}^{\mathcal{A}}_{D' \to D'}}_{\text{Hardness gap}} + \underbrace{\mathrm{acc}^{\mathcal{A}}_{D' \to D'} - \mathrm{acc}^{\mathcal{A}}_{D \to D'}}_{\text{Difference gap}}.$$

The *hardness gap* quantifies how much accuracy drop we should expect even if we had had access to the second distribution at training time. Note that the hardness gap depends on the learning algorithm $\mathcal{A}$: it is possible that there is a different classifier that achieves perfect accuracy on both D and D', but the hardness gap w.r.t. $\mathcal{A}$ is still nonzero due to limitations of the hypothesis class or training algorithm. In particular, "harder" distributions could arise from presence of less informative features or more label noise (and thus higher Bayes risk), but they could also exist when the Bayes risks for D and D' are the same.

The *difference gap* quantifies the advantage of applying $\mathcal{A}$ on the matching training distribution D' as opposed to the original training distribution D. A large difference gap indicates that $\mathcal{A}$ could in principle perform well on the second distribution D', but fails to generalize when trained on the original distribution D.

Recht et al. (2019) evaluate the accuracy gap for a distribution shift between CIFAR-10 and a reproduction (CIFAR-10.1) for multiple learning algorithms. They reveal a notable pattern: the accuracy gap $\mathrm{acc}^{\mathcal{A}}_{D \to D} - \mathrm{acc}^{\mathcal{A}}_{D \to D'}$ is roughly linear in $\mathrm{acc}^{\mathcal{A}}_{D \to D}$. The goal of our paper is to provide additional insight into this pattern by evaluating the hardness and difference gaps for a similar distribution shift and learning algorithms.

[1] Note that $\mathrm{acc}^{\mathcal{A}}_{D \to D'}$ is a random variable. As is commonly done in machine learning, we estimate it from a finite number of samples. In the full version of the paper, we will quantify the resulting random variation in more detail.

3. Dataset construction

The overall goal for our new dataset CIFAR-10.2 was to create a similar (but not identical) distribution as the original CIFAR-10 dataset. Hence we closely followed the original dataset creation process (Krizhevsky, 2009). In particular, all images in CIFAR-10.2 also come from the Tiny Images dataset (Torralba et al., 2008). Tiny Images contains about 80 million color images of size 32×32 pixels and is organized into about 75,000 different keywords. Each class in CIFAR-10 was assembled by human annotators reviewing candidate images from Tiny Images for about 10 – 25 corresponding keywords per class (e.g., "airbus" is one of the keywords for the class "airplane"). We followed a similar process with the authors of this paper taking the role of the human annotators.

The CIFAR-10.2 dataset size of 12,000 images (10,000 train and 2,000 test) balances two goals. On the one hand, a larger dataset has multiple advantages: larger test sets lead to smaller error bars and a larger training set makes the training setup more similar to the original CIFAR-10 dataset, which has 50,000 training images. On the other hand, using the same keywords as CIFAR-10 is desirable to minimize distribution shift. However, some of these keywords have only few images remaining. In particular, we found only 791 suitable cat images in Tiny Images among the keywords used for the cat class in CIFAR-10, while other classes have at least 1,200 usable images remaining within their corresponding keywords.

To balance these competing desiderata, we obtained additional images for the cat class from other keywords in Tiny Images but kept the keywords for the other classes unchanged. Concretely, we utilized the following additional keywords: *feline, tiger_cat, Egyptian_cat, Persian_cat, Siamese_cat*, and *Siamese*. We identified about 1,000 additional suitable cat images among these keywords, which allowed us to manually assess roughly 32,000 candidate images per class and assemble a dataset of size 12,000.

We took two steps to ensure data quality. First, we removed all exact matches and near-duplicates between our candidate images and all images contained in CIFAR-10 (train and test) as well as CIFAR-10.1. In particular, we filtered out images with structural similarity index (SSIM) above a threshold of 0.6 (Wang et al., 2004); see Appendix A

for additional details. After removing near-duplicates, we performed a final data cleaning step to remove mislabeled images. We reviewed all images that were misclassfied by most of the classifiers in our testbed and removed the images that were incorrectly labeled.

4. Experiments

Our experiments have two main goals. The first goal is to check whether our new CIFAR-10.2 dataset is qualitatively similar to the recent CIFAR-10.1 reproduction from Recht et al. (2019). In particular, do models again exhibit a substantial accuracy drop? The second goal is to decompose this accuracy drop into a *harder* and *different* component as outlined in Section 2.

Before we proceed to our results, we briefly outline our experimental setup. We assembled a testbed of 33 image classification models, largely following Recht et al. (2019). Among other models, the testbed includes common architectures such as ResNets, VGG, Shake-Shake models as well as the original AlexNet and random feature models (He et al., 2016a; Simonyan & Zisserman, 2014; Xavier, 2016; Krizhevsky et al., 2012; Coates et al., 2011). We adapted the models from published code (see Appendix B for the a list of all models). For each model, we verified that the code reproduces the original published accuracies when trained and tested on CIFAR-10.

Comparing CIFAR-10.1 and CIFAR-10.2. Figure 1 plots CIFAR-10.1 and 10.2 test accuracies as a function of CIFAR-10 accuracy for models trained on CIFAR-10. Similar to CIFAR-10.1, the models exhibit a substantial accuracy drop with a linear trend between CIFAR-10 and CIFAR-10.2 accuracies. The accuracy drop for CIFAR-10.2 is larger than for CIFAR-10.1, e.g., the accuracy of a ResNet-32 model now drops by 12%, as oppposed to 9% on CIFAR-10.1. The slope of the linear fit on CIFAR-10.2 is 1.34, which is lower than the slope of 1.7 for CIFAR-10.1. This indicates that higher performing models do not close the accuracy drop as much as on CIFAR-10.1.

Decomposing the accuracy drops. Next we decompose the accuracy drop described in the previous paragraph into different and harder components. These experiments involved training models on both the CIFAR-10 and 10.2 distributions. To match the size of the CIFAR-10.2 training set, we sampled a random class-balanced subset of the original CIFAR-10 training set with size 10,000 for training on the CIFAR-10 distribution.

For training on the smaller datasets, we did not change the model or training hyper-parameters (e.g., we kept the learning rate schedules as for training on the full CIFAR-10 training set). Optimizing hyperparameters for the smaller training sets may improve accuracy, and exploring whether there are better hyperparameter settings for CIFAR-10.2 is an interesting direction for future work. However, for this paper we are interested in broad trends that apply to a range of training algorithms and hence we do not pursue this avenue further.

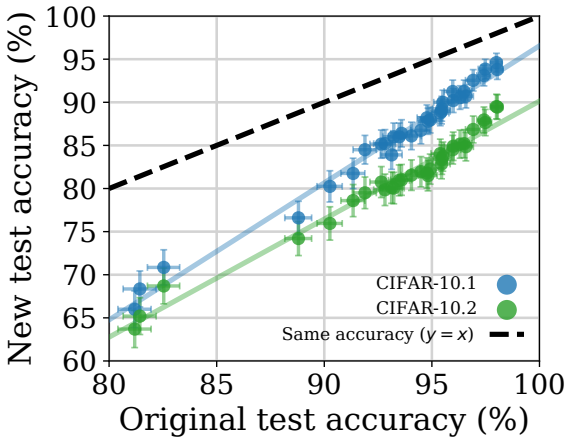

Figure 1. Comparison of CIFAR-10.1 and CIFAR-10.2. Similar to CIFAR-10.1, the accuracies of several models follow a linear trend on CIFAR-10.2. The accuracy drop on CIFAR-10.2 is larger and the slope of the linear fit is smaller than on CIFAR-10.1. Overall CIFAR-10.2 is still qualitatively similar to CIFAR-10.1.

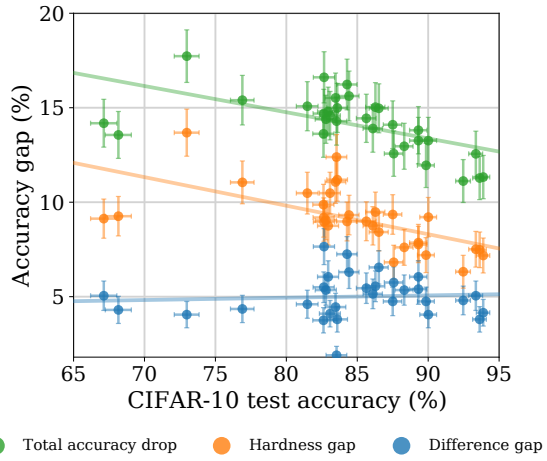

Figure 2. Decomposition of the CIFAR-10.2 accuracy drop into "harder" and "different" components. The "different" component is approximately constant across the models in our testbed.

Table 1 contains the accuracies for a few key models and the full experimental results are detailed in Appendix B. Figure 2 depicts the overall accuracy drop, harder component, and different component as a function of model accuracy when trained on the 10,000 image subset of CIFAR-10. Interestingly, the "different" component is mostly flat while the slope of the overall accuracy gap is largely accounted for by the "harder" component.

Model	Trained on CIFAR-10.0 $n_{\text{train}} = 50{,}000$		Trained on CIFAR-10.0 $n_{\text{train}} = 10{,}000$		Trained on CIFAR-10.2 $n_{\text{train}} = 10{,}000$	
	10.0 Acc.	10.2 Acc.	10.0 Acc.	10.2 Acc.	10.0 Acc.	10.2 Acc.
autoaug_shake_shake_112	98.0	89.5	93.6	82.4	90.1	86.2
wide_resnet_28_10	96.0	84.5	87.6	75.0	84.0	80.8
resnet_basic_32	92.8	79.9	83.0	68.4	79.1	73.8
alexnet	82.5	68.7	67.1	53.0	59.5	58.0
randomfeatures16k	79.9	61.9	72.8	54.3	62.1	58.1

Table 1. Test set accuracies for five representative models. The first two columns show the accuracies for models trained on the full CIFAR-10 training set with $n_{\text{train}} = 50{,}000$. We use 10.0 as a shorthand for the original CIFAR-10 dataset and for consistency with the 10.1 and 10.2 reproductions. The other four columns show the accuracies for models with the same architecture and training algorithm, but trained on training sets with size $n_{\text{train}} = 10{,}000$ for CIFAR-10.0 and 10.2, respectively. All numbers are percentage points. Tables 4 and 5 in the appendix contain the accuracy numbers for all 33 models in our testbed.

5. Theoretical model

To further our understanding of the different vs. harder distinction, we analyze distribution shift in a simple Gaussian linear discriminant setting. Our setting aims to capture the salient: a family of learning rules with growing capacity and a "natural" (non-adversarial) distribution shift. The setting provides explicit control over the "hard" and "different" part of the distribution shift, allowing us to explore how they affect drops in accuracy.

Formally, we consider a C-way classification task with input $x \in \mathbb{R}^d$ and target label $y \in \{1, \dots, C\}$. For every dimension $q \leq d$, we consider linear classifiers operating only on the first q coordinates of x. That is, we consider models of the form $\boldsymbol{\theta} \in \mathbb{R}^{q \times C}$ that for input x predict output $\hat{y} = \arg\max_{i \leq C} [\boldsymbol{\theta}^\top x_{:q}]_i$, where $x_{:q}$ denotes the first q coordinates of x. Clearly, the capacity of these linear classifiers grows with q, but how does their sensitivity to distribution shift behave?

The answer to this question depends closely on how the distribution shift affects the different coordinates of x. To capture non-adversarial "average case" shifts, we consider two transformations that are agnostic to the choice of coordinate system: (1) making the distribution "harder" by adding standard Gaussian noise to x and (2) making the distribution "different" by applying a random rotation to x.

For concreteness and simplicity, we consider the distribution D corresponding to y uniform on $\{1, \dots, C\}$ and $x|y \sim \mathcal{N}(\mu_y, \sigma^2 I)$, where $\mu_1, \dots, \mu_C \in \mathbb{R}^d$ are *orthogonal unit vectors* (chosen randomly). For q-dimensional linear classifiers, we consider the optimal learning rule (realizable with infinite training data) that takes θ_i to be the first q coordinates of μ_i.

We parameterize the shifted distribution D' by coefficients $\alpha_{\text{hard}}, \alpha_{\text{diff}} \in [0, 1]$. The closer the coefficients are to zero, the larger the shift is. Specifically, under D' we set $x|y \sim \mathcal{N}(\mu'_y, \sigma^2 I)$ for

$$\mu'_i = \alpha_{\text{hard}} \cdot \left(\alpha_{\text{diff}} \cdot \mu_i + \sqrt{1 - \alpha_{\text{diff}}^2} \cdot \tilde{\mu}_i\right),$$

where $\tilde{\mu}_1, \dots, \tilde{\mu}_C$ are orthonormal vectors chosen uniformly from the subspace orthogonal to $\{\mu_i\}$.

Figure 3 shows the different accuracy drops as a function of accuracy on the original distribution, with every point corresponding to a different model dimension q. The figure shows a single realization of D and D' as described above, but due to concentration in high d, the results depend only on α_{hard} and α_{diff} and not the draw of μ and $\tilde{\mu}$. Comparing Figure 3 to Figure 2, we see similar trends: accuracy drops are decreasing as original accuracy increases, but the "different" part of the accuracy drop is more stable.

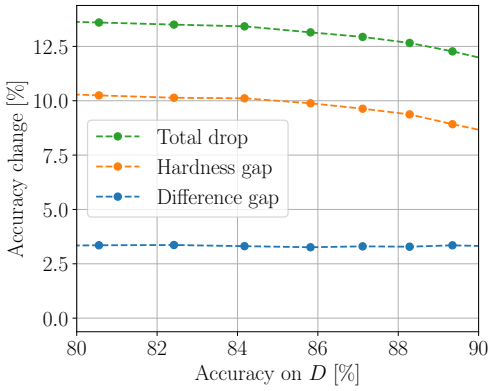

Figure 3. Distribution shift effect in the setting of Section 5, with $d = 10^4$, $C = 10$ and distribution shift parameters $\alpha_{\text{hard}} = 0.85$ and $\alpha_{\text{diff}} = 0.95$. Similar to real data (see Figure 2), the difference gap is approximately constant.

References

Chen, Y., li, J., Xiao, H., Yan, S., and Feng, J. Dual path networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2017. https://papers.nips.cc/paper/7033-dual-path-networks.pdf.

Coates, A., Ng, A., , and Lee, H. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. https://ai.stanford.edu/~ang/papers/nipsdlufl10-AnalysisSingleLayerUnsupervisedFeatureLearning.pdf.

Cubuk, E., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. Autoaugment: Learning augmentation policies from data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. https://arxiv.org/abs/1805.09501.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. http://www.image-net.org/papers/imagenet_cvpr09.pdf.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with Cutout, 2017. https://arxiv.org/abs/1708.04552.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *In Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016a. https://arxiv.org/abs/1512.03385.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, 2016b. https://arxiv.org/abs/1603.05027.

Howard, A., Zhu, M., Chen, B., Kalencichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. https://arxiv.org/abs/1704.04861.

Huang, G., Liu, Z., Maaten, L. V. D., and Weinberger, K. Densely connected convultional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. https://arxiv.org/abs/1608.06993.

Krizhevsky, A. Learning Multiple Layers of Features From Tiny Images, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Krizhevsky, A., Sutskever, I., and Hinton, G. ImageNet Classification With Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L., Fei-Fei, L., Yuille, A. L., Huang, J., and Murphy, K. Progressive neural architecture search. In *European Conference on Computer Vision (ECCV)*, 2018. http://arxiv.org/abs/1712.00559.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ Questions For Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. https://arxiv.org/abs/1606.05250.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do Imagenet Classifiers Generalize to Imagenet? In *International Conference on Machine Learning (ICML)*, 2019. https://arxiv.org/abs/1902.10811.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Li, F.-F. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015. https://arxiv.org/abs/1409.0575.

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014. https://arxiv.org/abs/1409.1556.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2019. http://arxiv.org/abs/1905.11946.

Torralba, A. and Efros, A. A. Unbiased Look at Dataset Bias. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. http://people.csail

.mit.edu/torralba/publications/datasets_cvpr11.pdf.

Torralba, A., Fergus, R., and Freeman, W. T. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008. https://ieeexplore.ieee.org/document/4531741.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 2004. http://www.cns.nyu.edu/pub/lcv/wang03-preprint.pdf.

Xavier, G. Shake-Shake Regularization, 2016. https://arxiv.org/abs/1705.07485.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. https://arxiv.org/abs/1611.05431.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016. https://arxiv.org/abs/1605.07146.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. http://arxiv.org/abs/1707.01083.

A. Dataset details

To create CIFAR-10.2, we first selected the TinyImages data with keywords appearing in CIFAR-10, and manually reviewed roughly 32k images per class, keeping only images whose content matches the class label. We then filtered out any image whose SSIM metric to any of the images in CIFAR-10 and CIFAR-10.1 was above 0.6; we marked the remaining images as "suitable." We then selected 1,200 suitable images for each class to create CIFAR-10.2. Table (2) provides a per-class breakdown of the numbers of candidate, reviewed, suitable, and selected images. As we mention in Section 3, for the "cat" label we had to include additional keywords not appearing in CIFAR-10; see Table 3 for a per-keyword breakdown.

We determined the SSIM threshold for near-duplicate removal by manually reviewing the nearest neighbors for a large number of images and picking a threshold for which very few duplicates appear. Figure 4 illustrates that 0.6 is a valid threshold by showing, for 10 random "suitable" images, the nearest neighbors (in CIFAR-10 and CIFAR-10.1) in terms of SSIM. The near-duplicate removal stage decreased the number of suitable images by roughly 15%.

	Airplane	Automobile	Bird	Cat†	Deer	Dog	Frog	Horse	Ship	Truck
Candidates	37,458	40,052	55,046	32,511	45,340	40,051	40,454	45,666	87,121	56,017
Reviewed	≈32k	≈32k	≈32k	≈32k	≈32k	≈32k	≈32k	≈32k	≈32k	≈32k
Suitable	1,656	2,031	1,957	1,889	1,237	1,755	1,207	2,137	2,226	2,031
Selected	1,200	1,200	1,200	1,200	1,200	1,200	1,200	1,200	1,200	1,200

Table 2. Class breakdown of CIFAR-10.2 creation process. †For the "cat" class there were only 21,394 candidate (and 791 suitable) images with keywords appearing in CIFAR-10. We therefore collected images from additional "cat" keywords shown in Table 3 below.

	egyptian_cat	feline	persian_cat	siamese	siamese_cat	tiger_cat
Candidates	1,303	1,997	2,167	1,957	1,972	1,721
Suitable	140	208	214	114	226	196

Table 3. Breakdown of images from additional "cat" keywords.

B. Experiment details

Table 4 lists the CIFAR-10 and CIFAR-10.2 test accuracies for three different training set, from which we compute the hardness and difference gaps shown in Figure 2; we also list them in Table 5 for ease of reference. Finally, in Table 6 we provide a url and reference for the learning algorithms in our testbed.

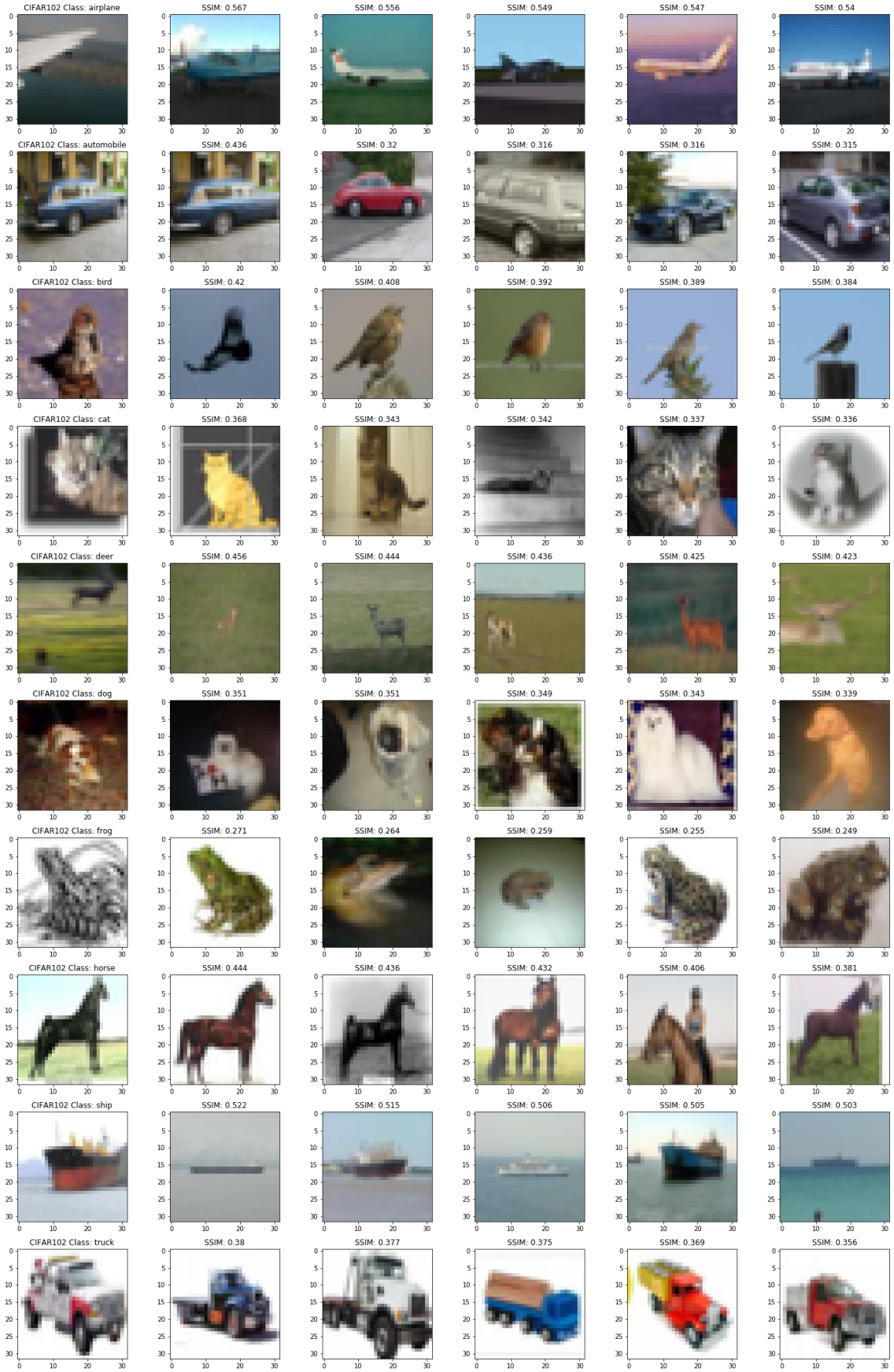

Figure 4. Illustration of near-duplicate filtering. The left column shows one random CIFAR-10.2 candidate image from each class, and the remaining columns contain their nearest neighbors in CIFAR10 and CIFAR10.1 (in terms of SSIM).

Model	Trained on CIFAR10.0 $n_{\text{train}} = 50{,}000$		Trained on CIFAR-10.0 $n_{\text{train}} = 10{,}000$		Trained on CIFAR-10.2 $n_{\text{train}} = 10{,}000$	
	10.0 Acc.	10.2 Acc.	10.0 Acc.	10.2 Acc.	10.0 Acc.	10.2 Acc.
autoaug_shake_shake_112	98.0	89.5	93.6	82.4	90.1	86.2
autoaug_shake_shake_96	98.0	89.5	93.9	82.6	90.3	86.7
autoaug_shake_shake_32	97.5	87.7	93.4	80.8	89.4	85.9
autoaug_wide_resnet	97.4	88.0	92.5	81.4	88.7	86.2
wide_resnet_28_10_cutout	96.9	86.9	86.0	82.7	89.9	77.9
shake_shake_26_2x32d_ssi	96.6	84.9	85.5	80.8	90.0	76.8
resnext_29_8x64d	96.5	85.2	85.1	80.7	88.3	75.4
wide_resnet_16_8_cutout	96.3	85.2	89.3	75.5	84.7	81.2
resnext_29_4x64d	96.0	84.9	85.4	81.5	89.3	76.1
wide_resnet_28_10	96.0	84.5	84.0	80.8	87.6	75.0
wide_resnet_16_8	95.5	83.7	87.5	73.4	83.4	78.2
googlenet	95.5	83.4	81.7	78.1	86.5	71.6
densenet121	95.5	82.9	80.6	76.7	85.6	71.2
resnext29	95.4	84.1	81.0	76.8	86.3	71.3
dpn92	94.9	82.3	79.2	75.1	84.4	68.8
preactres	94.8	81.5	81.8	77.4	86.1	72.2
mobilenetv2	94.8	81.8	78.5	75.3	84.3	68.1
resnet_preact_bottleneck_164	94.5	82.0	78.1	73.7	82.7	66.1
resnet_basic_110	94.1	81.6	75.1	71.0	81.5	66.4
resnet_preact_basic_110	93.6	81.0	77.4	72.4	83.5	68.0
resnet_basic_56	93.5	81.0	77.7	71.2	83.5	69.3
resnet_basic_44	93.2	80.1	77.8	72.6	83.2	68.5
vgg_15_bn_64	93.1	80.1	79.4	72.4	83.6	68.6
resnet_basic_32	92.8	79.9	79.1	73.8	83.0	68.4
resnet_basic_38	92.7	80.8	82.6	68	78.4	73.5
resnet_basic_26	91.9	79.5	82.6	69	77.6	72.8
vgg19	91.3	78.6	78.1	74.2	83.0	68.2
shufflenetv2	90.3	76.0	72.5	70.0	80.5	65.1
efficientnet	88.8	74.2	70.1	65.9	76.9	61.5
alexnet	82.5	68.7	59.5	58.0	67.1	53.0
pnasneta	81.4	65.2	61.3	58.9	68.2	54.6
randomfeatures32k	81.2	63.7	63.1	59.3	73.0	55.3
randomfeatures16k	79.9	61.9	62.1	58.1	72.8	54.3
lenet	74.9	59.1	58.4	55.5	65.2	50.8

Table 4. Full experiment results. The first two columns show the accuracies when the models are trained on the full CIFAR-10 training set of $n_{\text{train}} = 50{,}000$ images. The other four columns show training on $n_{\text{train}} = 10{,}000$ images from CIFAR10.0 or 10.2.

Model	$\text{acc}^{\mathcal{A}}_{10.0(50K)\to 10.0}$	$\text{acc}^{\mathcal{A}}_{10.0(10K)\to 10.0}$	$\text{acc}^{\mathcal{A}}_{10.0(10K)\to 10.2}$	Accuracy gap	Hardness gap	Difference gap
autoaug_shake_shake_112	98.0	93.6	82.4	11.3	7.5	3.8
autoaug_shake_shake_96	98.0	93.9	82.6	11.3	7.2	4.1
autoaug_shake_shake_32	97.5	93.4	80.8	12.6	7.5	5.1
autoaug_wrn	97.4	92.5	81.4	11.1	6.3	4.8
wide_resnet_28_10_cutout	96.9	89.9	77.9	11.9	7.2	4.7
shake_shake_26_2x32d_SSI	96.6	90.0	76.8	13.3	9.2	4.0
resnext_29_8x64d	96.5	88.3	75.4	13.0	7.6	5.4
wide_resnet_16_8_cutout	96.3	89.3	75.5	13.8	7.8	6.1
resnext_29_4x64d	96.0	89.3	76.1	13.3	7.9	5.4
wrn_28_10	96.0	87.6	75.0	12.6	6.8	5.7
wrn_16_8	95.5	87.5	73.4	14.1	9.4	4.7
googlenet	95.5	86.5	71.6	15.0	8.4	6.5
densenet121	95.5	85.6	71.2	14.4	9.0	5.4
resnext29	95.4	86.3	71.3	15.0	9.5	5.5
dpn92	94.9	84.4	68.8	15.6	9.3	6.3
preactres	94.8	86.1	72.2	13.9	8.8	5.1
mobilenetv2	94.8	84.3	68.1	16.2	9.0	7.3
resnet_preact_bottleneck_164	94.5	82.7	66.1	16.6	9.0	7.7
resnet_basic_110	94.1	81.5	66.4	15.1	10.5	4.6
resnet_preact_basic_110	93.6	83.5	68.0	15.5	11.1	4.5
resnet_basic_56	93.5	83.5	69.3	14.3	12.4	1.9
resnet_basic_44	93.2	83.1	68.5	14.6	10.5	4.1
vgg_15_BN_64	93.1	83.6	68.6	15.0	11.2	3.8
resnet_basic_32	92.8	82.8	68.4	14.4	9.0	5.3
resnet_basic_38	92.7	82.6	68.0	14.7	9.2	5.5
resnet_basic_26	91.9	82.6	69.0	13.6	9.9	3.8
vgg19	91.3	83.0	68.2	14.8	8.8	6.1
efficientnet	88.8	76.9	61.5	15.4	11.1	4.3
alexnet	82.5	67.1	53.0	14.2	9.1	5.0
pnasneta	81.4	68.2	54.6	13.6	9.3	4.3
randomfeatures32k	81.2	73.0	55.3	17.7	13.7	4.0
randomfeatures16k	79.9	72.8	54.3	18.5	14.8	3.8
lenet	74.9	65.2	50.8	14.4	9.7	4.7

Table 5. Tabulation of Figure 2; column titles of the form $\text{acc}^{\mathcal{A}}_{x(n)\to y}$ denote the accuracy when training on n images from CIFAR-x and testing on the test set of CIFAR-y.

Model	Code Repository	Reference
autoaug_shake_shake_96	https://github.com/tensorflow/models/tree/master/research/autoaugment	(Cubuk et al., 2018)
autoaug_shake_shake_112	https://github.com/tensorflow/models/tree/master/research/autoaugment	(Cubuk et al., 2018)
autoaug_shake_shake_32	https://github.com/tensorflow/models/tree/master/research/autoaugment	(Cubuk et al., 2018)
autoaug_wrn	https://github.com/tensorflow/models/tree/master/research/autoaugment	(Cubuk et al., 2018)
wide_resnet_28_10_cutout	https://github.com/hysts/pytorch_image_classification/	(Zagoruyko & Komodakis, 2016) (DeVries & Taylor, 2017)
shake_shake_26_2x32d_SSI	https://github.com/hysts/pytorch_image_classification/	(Xavier, 2016)
resnext_29_8x64d	https://github.com/hysts/pytorch_image_classification/	(He et al., 2016a)
wide_resnet_16_8_cutout	https://github.com/hysts/pytorch_image_classification/	(Zagoruyko & Komodakis, 2016) (DeVries & Taylor, 2017)
resnext_29_4x64d	https://github.com/hysts/pytorch_image_classification/	(Xie et al., 2017)
wrn_28_10	https://github.com/hysts/pytorch_image_classification/	(Zagoruyko & Komodakis, 2016)
wrn_16_8	https://github.com/hysts/pytorch_image_classification/	(Zagoruyko & Komodakis, 2016)
googlenet	https://github.com/kuangliu/pytorch-cifar	(Szegedy et al., 2015)
densenet121	https://github.com/kuangliu/pytorch-cifar	(Huang et al., 2016)
resnext29	https://github.com/hysts/pytorch_image_classification/	(Xie et al., 2017)
dpn92	https://github.com/kuangliu/pytorch-cifar	(Chen et al., 2017)
preactres	https://github.com/kuangliu/pytorch-cifar	(He et al., 2016b)
mobilenetv2	https://github.com/kuangliu/pytorch-cifar	(Howard et al., 2017)
resnet_preact_bottleneck_164	https://github.com/hysts/pytorch_image_classification/	(He et al., 2016b)
resnet_basic_110	https://github.com/hysts/pytorch_image_classification/	(He et al., 2016a)
resnet_preact_basic_110	https://github.com/hysts/pytorch_image_classification/	(He et al., 2016b)
resnet_basic_56	https://github.com/hysts/pytorch_image_classification/	(He et al., 2016a)
resnet_basic_44	https://github.com/hysts/pytorch_image_classification/	(He et al., 2016a)
vgg_15_BN_64	https://github.com/hysts/pytorch_image_classification/	(Simonyan & Zisserman, 2014)
resnet_basic_32	https://github.com/hysts/pytorch_image_classification/	(He et al., 2016a)
resnet_basic_38	https://github.com/hysts/pytorch_image_classification/	(He et al., 2016a)
resnet_basic_26	https://github.com/hysts/pytorch_image_classification/	(He et al., 2016a)
vgg19	https://github.com/hysts/pytorch_image_classification/	(Simonyan & Zisserman, 2014)
shufflenetv2	https://github.com/kuangliu/pytorch-cifar	(Zhang et al., 2018)
efficientnet	https://github.com/kuangliu/pytorch-cifar	(Tan & Le, 2019)
alexnet	https://github.com/dansuh17/alexnet-pytorch	(Krizhevsky et al., 2012)
pnasneta	https://github.com/kuangliu/pytorch-cifar	(Liu et al., 2018)
randomfeatures32k	https://github.com/modestyachts/nondeep	(Coates et al., 2011)
randomfeatures16k	https://github.com/modestyachts/nondeep	(Coates et al., 2011)
lenet	https://github.com/kuangliu/pytorch-cifar	(LeCun et al., 1998)

Table 6. Repositories for the models in our testbed.