A. Related Work

We now briefly discuss related threads in machine learning. To the best of our knowledge, there are no reproducibility experiments directly comparable to ours in the literature.

Dataset Biases. The computer vision community has a rich history of creating new datasets and discussing their relative merits, e.g., (Deng et al., 2009; Everingham et al., 2010; Fei-Fei et al., 2007; Lin et al., 2014; Ponce et al., 2006; Russakovsky et al., 2015; Torralba and Efros, 2011; Yao et al., 2007). The paper closest to ours is (Torralba and Efros, 2011), which studies dataset biases by measuring how models trained on one dataset generalize to other datasets. The main difference to our work is that the authors test generalization across *different* datasets, where larger changes in the distribution (and hence larger drops in accuracy) are expected. In contrast, our experiments explicitly attempt to reproduce the original data distribution and demonstrate that even small variations arising in this process can lead to significant accuracy drops. Moreover, (Torralba and Efros, 2011) do not test on previously unseen data, so their experiments cannot rule out adaptive overfitting.

Transfer Learning From ImageNet. Kornblith et al. (2018) study how well accuracy on ImageNet transfers to other image classification datasets. An important difference from both our work and (Torralba and Efros, 2011) is that the the ImageNet models are re-trained on the target datasets. The authors find that better ImageNet models usually perform better on the target dataset as well. Similar to (Torralba and Efros, 2011), these experiments cannot rule out adaptive overfitting since the authors do not use new data. Moreover, the experiments do not measure accuracy drops due to small variations in the data generating process since the models are evaluated on a different task with an explicit adaptation step. Interestingly, the authors also find an approximately linear relationship between ImageNet and transfer accuracy.

Adversarial Examples. While adversarial examples (Biggio and Roli, 2018; Szegedy et al., 2013) also show that existing models are brittle, the perturbations have to be finely tuned since models are much more robust to random perturbations. In contrast, our results demonstrate that even small, benign variations in the data sampling process can already lead to a significant accuracy drop without an adversary.

A natural question is whether adversarially robust models are also more robust to the distribution shifts observed in our work. As a first data point, we tested the common ℓ_∞-robustness baseline from (Madry et al., 2018) for CIFAR-10. Interestingly, the accuracy numbers of this model fall almost exactly on the linear fit given by the other models in our testbed. Hence ℓ_∞-robustness does not seem to offer benefits for the distribution shift arising from our reproducibility experiment. However, we note that more forms of adversarial robustness such as spatial transformations or color space changes have been studied (Engstrom et al., 2017; Fawzi and Frossard, 2015; Hosseini and Poovendran, 2018; Kanbak et al., 2018; Xiao et al., 2018). Testing these variants is an interesting direction for future work.

Non-Adversarial Image Perturbations. Recent work also explores less adversarial changes to the input, e.g., (Geirhos et al., 2018; Hendrycks and Dietterich, 2019). In these papers, the authors modify the ImageNet validation set via well-specified perturbations such as Gaussian noise, a fixed rotation, or adding a synthetic snow-like pattern. Standard ImageNet models then achieve significantly lower accuracy on the perturbed examples than on the unmodified validation set. While this is an interesting test of robustness, the mechanism underlying the accuracy drops is significantly different from our work. The aforementioned papers rely on an intentional, clearly-visible, and well-defined perturbation of existing validation images. Moreover, some of the interventions are quite different from the ImageNet validation set (e.g., ImageNet contains few images of falling snow). In contrast, our experiments use new images and match the distribution of the existing validation set as closely as possible. Hence it is unclear what properties of our new images cause the accuracy drops.

B. A Model for the Linear Fit

Finally, we briefly comment on the striking linear relationship between original and new test accuracies that we observe in all our experiments (for instance, see Figure 1 in the introduction or Figures 12 and 13 in the appendix). To illustrate how this phenomenon could arise, we present a simple data model where a small modification of the data distribution can lead to significant changes in accuracy, yet the relative order of models is preserved as a linear relationship. We emphasize that this model should not be seen as the true explanation. Instead, we hope it can inform future experiments that explore natural variations in test distributions.

First, as we describe in Appendix D.2, we find that we achieve better fits to our data under a *probit scaling* of the accuracies. Over a wide range from 21% to 83% (all models in our ImageNet testbed), the accuracies on the new test set, α_{new}, are related to the accuracies on the original test set, α_{orig}, by the relationship

$$\Phi^{-1}(\alpha_{\text{new}}) \;=\; u \cdot \Phi^{-1}(\alpha_{\text{orig}}) + v$$

where Φ is the Gaussian CDF, and u and v are scalars. The probit scale is in a sense more natural than a linear scale as the accuracy numbers are probabilities. When we plot accuracies on a probit scale in Figures 6 and 13, we effectively visualize $\Phi^{-1}(\alpha)$ instead of α.

We now provide a simple plausible model where the original and new accuracies are related linearly on a probit scale. Assume that every example i has a scalar "difficulty" $\tau_i \in \mathbb{R}$ that quantifies how easy it is to classify. Further assume the probability of a model j correctly classifying an image with difficulty τ is given by an increasing function $\zeta_j(\tau)$. We show that for restricted classes of difficulty functions ζ_j, we find a linear relationship between average accuracies after distribution shifts.

To be specific, we focus on the following parameterization. Assume the difficulty distribution of images in a test set follows a normal distribution with mean μ and variance σ^2. Further assume that

$$\zeta_j(\tau) \;=\; \Phi(s_j - \tau) \; ,$$

where $\Phi : \mathbb{R} \to (0, 1)$ is the CDF of a standard normal distribution, and s_j is the "skill" of model j. Models with higher skill have higher classification accuracy, and images with higher difficulty lead to smaller classification accuracy. Again, the choice of Φ here is somewhat arbitrary: any sigmoidal function that maps $(-\infty, +\infty)$ to $(0, 1)$ is plausible. But using the Gaussian CDF yields a simple calculation illustrating the linear phenomenon.

Using the above notation, the accuracy $\alpha_{j,\mu,\sigma}$ of a model j on a test set with difficulty mean μ and variance σ is then given by

$$\alpha_{j,\mu,\sigma} \;=\; \mathop{\mathbb{E}}_{\tau \sim \mathcal{N}(\mu,\sigma)} \left[\Phi(s_j - \tau)\right] \; .$$

We can expand the CDF into an expectation and combine the two expectations by utilizing the fact that a linear combination of two Gaussians is again Gaussian. This yields:

$$\alpha_{j,\mu,\sigma} \;=\; \Phi\left(\frac{s_j - \mu}{\sqrt{\sigma^2 + 1}}\right) \; .$$

On a probit scale, the quantities we plot are given by

$$\tilde{\alpha}_{j,\mu,\sigma} \;=\; \Phi^{-1}(\alpha_{j,\mu,\sigma}) \;=\; \frac{s_j - \mu}{\sqrt{\sigma^2 + 1}} \; .$$

Next, we consider the case where we have multiple models and two test sets with difficulty parameters μ_k and σ_k respectively for $k \in \{1, 2\}$. Then $\tilde{\alpha}_{j,2}$, the probit-scaled accuracy on the second test set, is a linear function of the accuracy on the first test set, $\tilde{\alpha}_{j,1}$:

$$\tilde{\alpha}_{j,2} \;=\; u \cdot \tilde{\alpha}_{j,1} + v \; ,$$

with

$$u \;=\; \frac{\sqrt{\sigma_1^2 + 1}}{\sqrt{\sigma_2^2 + 1}} \quad \text{and} \quad v \;=\; \frac{\mu_1 - \mu_2}{\sqrt{\sigma_2^2 + 1}} \; .$$

Hence, we see that the Gaussian difficulty model above yields a linear relationship between original and new test accuracy in the probit domain. While the Gaussian assumptions here made the calculations simple, a variety of different simple classes of ζ_j will give rise to the same linear relationship between the accuracies on two different test sets.

C. Details of the CIFAR-10 Experiments

We first present our reproducibility experiment for the CIFAR-10 image classification dataset (Krizhevsky, 2009). There are multiple reasons why CIFAR-10 is an important example for measuring how well current models generalize to unseen data.

- CIFAR-10 is one of the most widely used datasets in machine learning and serves as a test ground for many image classification methods. A concrete measure of popularity is the fact that CIFAR-10 was the second most common dataset in NIPS 2017 (after MNIST) (Hamner, 2017).
- The dataset creation process for CIFAR-10 is transparent and well documented (Krizhevsky, 2009). Importantly, CIFAR-10 draws from the larger Tiny Images repository that has more fine-grained labels than the ten CIFAR-10 classes (Torralba et al., 2008). This enables us to minimize various forms of distribution shift between the original and new test set.
- CIFAR-10 poses a difficult enough problem so that the dataset is still the subject of active research (e.g., see (Cubuk et al., 2018; DeVries and Taylor, 2017; Gastaldi, 2017; Real et al., 2018; Yamada et al., 2018; Zoph et al., 2018)). Moreover, there is a wide range of classification models that achieve significantly different accuracy scores. Since code for these models has been published in various open source repositories, they can be treated as independent of our new test set.

Compared to ImageNet, CIFAR-10 is significantly smaller both in the number of images and in the size of each image. This makes it easier to conduct various follow-up experiments that require training new classification models. Moreover, the smaller size of CIFAR-10 also means that the dataset has been accessible to more researchers for a longer time. Hence it is plausible that CIFAR-10 experienced more test set adaptivity than ImageNet, where it is much more costly to tune hyperparameters.

Before we describe how we created our new test set, we briefly review relevant background on CIFAR-10 and Tiny Images.

Tiny Images. The dataset contains 80 million RGB color images with resolution 32×32 pixels and was released in 2007 (Torralba et al., 2008). The images are organized by roughly 75,000 *keywords* that correspond to the non-abstract nouns from the WordNet database (Miller, 1995) Each keyword was entered into multiple Internet search engines to collect roughly 1,000 to 2,500 images per keyword. It is important to note that Tiny Images is a fairly noisy dataset. Many of the images filed under a certain keyword do not clearly (or not at all) correspond to the respective keyword.

CIFAR-10. The CIFAR-10 dataset was created as a cleanly labeled subset of Tiny Images for experiments with multi-layer networks. To this end, the researchers assembled a dataset consisting of ten classes with 6,000 images per class, which was published in 2009 (Krizhevsky, 2009). These classes are `airplane`, `automobile`, `bird`, `cat`, `deer`, `dog`, `frog`, `horse`, `ship`, and `truck`. The standard train / test split is class-balanced and contains 50,000 training images and 10,000 test images.

The CIFAR-10 creation process is well-documented (Krizhevsky, 2009). First, the researchers assembled a set of relevant keywords for each class by using the hyponym relations in WordNet (Miller, 1995) (for instance, "Chihuahua" is a hyponym of "dog"). Since directly using the corresponding images from Tiny Images would not give a high quality dataset, the researchers paid student annotators to label the images from Tiny Images. The labeler instructions can be found in Appendix C of (Krizhevsky, 2009) and include a set of specific guidelines (e.g., an image should not contain two object of the corresponding class). The researchers then verified the labels of the images selected by the annotators and removed near-duplicates from the dataset via an ℓ_2 nearest neighbor search.

C.1. Dataset Creation Methodology

Our overall goal was to create a new test set that is as close as possible to being drawn from the same distribution as the original CIFAR-10 dataset. One crucial aspect here is that the CIFAR-10 dataset did not exhaust any of the Tiny Image keywords it is drawn from. So by collecting new images from the same keywords as CIFAR-10, our new test set can match the sub-class distribution of the original dataset.

Understanding the Sub-Class Distribution. As the first step, we determined the Tiny Image keyword for every image in the CIFAR-10 dataset. A simple nearest-neighbor search sufficed since every image in CIFAR-10 had an exact duplicate (ℓ_2-distance 0) in Tiny Images. Based on this information, we then assembled a list of the 25 most common keywords for each class. We decided on 25 keywords per class since the 250 total keywords make up more than 95% of CIFAR-10. Moreover, we wanted to avoid accidentally creating a harder dataset with infrequent keywords that the classifiers had little incentive to learn based on the original CIFAR-10 dataset.

The keyword distribution can be found in Appendix C.3.1. Inspecting this list reveals the importance of matching the sub-class distribution. For instance, the most common keyword in the `airplane` class is `stealth_bomber` and not a more common civilian type of airplane. In addition, the third most common keyword for the `airplane` class is `stealth_fighter`. Both types of planes are highly distinctive. There are more examples where certain sub-classes are considerably different. For instance, trucks from the keyword `fire_truck` are mostly red, which is quite different from pictures for `dump_truck` or other keywords.

Collecting New Images. After determining the keywords, we collected corresponding images. To simulate the student / researcher split in the original CIFAR-10 collection procedure, we introduced a similar split among two authors of this paper. Author A took the role of the original student annotators and selected new suitable images for the 250 keywords. In order to ensure a close match between the original and new images for each keyword, we built a user interface that allowed Author A to first look through existing CIFAR-10 images for a given keyword and then select new candidates from the remaining pictures in Tiny Images. Author A followed the labeling guidelines in the original instruction sheet (Krizhevsky, 2009). The number of images Author A selected per keyword was so that our final dataset would contain between 2,000 and 4,000 images. We decided on 2,000 images as a target number for two reasons:

- While the original CIFAR-10 test set contains 10,000 images, a test set of size 2,000 is already sufficient for a fairly small confidence interval. In particular, a conservative confidence interval (Clopper-Pearson at confidence level 95%) for accuracy 90% has size about $\pm 1\%$ with $n = 2{,}000$ (to be precise, [88.6%, 91.3%]). Since we considered a potential discrepancy between original and new test accuracy only interesting if it is significantly larger than 1%, we decided that a new test set of size 2,000 was large enough for our study.
- As with very infrequent keywords, our goal was to avoid accidentally creating a harder test set. Since some of the Tiny Image keywords have only a limited supply of remaining adequate images, we decided that a smaller target size for the new dataset would reduce bias to include images of more questionable difficulty.

After Author A had selected a set of about 9,000 candidate images, Author B adopted the role of the researchers in the original CIFAR-10 dataset creation process. In particular, Author B reviewed all candidate images and removed images that were unclear to Author B or did not conform to the labeling instructions in their opinion (some of the criteria are subjective). In the process, a small number of keywords did not have enough images remaining to reach the $n = 2{,}000$ threshold. Author B then notified Author A about the respective keywords and Author A selected a further set of images for these keywords. In this process, there was only one keyword where Author A had to carefully examine all available images in Tiny Images. This keyword was `alley_cat` and comprises less than 0.3% of the overall CIFAR-10 dataset.

Final Assembly. After collecting a sufficient number of high-quality images for each keyword, we sampled a random subset from our pruned candidate set. The sampling procedure was such that the keyword-level distribution of our new dataset matches the keyword-level distribution of CIFAR-10 (see Appendix C.3.1). In the final stage, we again proceeded similar to the original CIFAR-10 dataset creation process and used ℓ_2-nearest neighbors to filter out near duplicates. In particular, we removed near-duplicates within our new dataset and also images that had a near duplicate in the original CIFAR-10 dataset (train or test). The latter aspect is particularly important since our reproducibility study is only interesting if we evaluate on truly unseen data. Hence we manually reviewed the top-10 nearest neighbors for each image in our new test set. After removing near-duplicates in our dataset, we re-sampled the respective keywords until this process converged to our final dataset.

Figure 3b shows a random subset of images from the original and our new test set.

We remark that we did not run any classifiers on our new dataset during the data collection phase of our study. In order to ensure that the new data does not depend on the existing classifiers, it is important to strictly separate the data collection phase from the following evaluation phase.

C.2. Follow-up Hypotheses

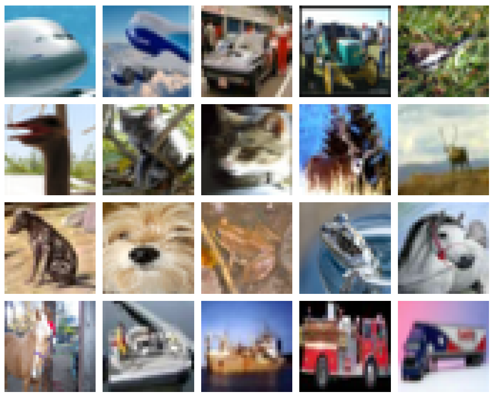
Since the gap between original and new accuracy is concerningly large, we investigated multiple hypotheses for explaining this gap.

C.2.1. Statistical Error

A first natural guess is that the gap is simply due to statistical fluctuations. But as noted before, the sample size of our new test set is large enough so that a 95% confidence interval has size about $\pm 1.2\%$. Since a 95% confidence interval for the original CIFAR-10 test accuracy is even smaller (roughly $\pm 0.6\%$ for 90% classification accuracy and $\pm 0.3\%$ for 97% classification accuracy), we can rule out statistical error as the main explanation.

C.2.2. Differences in Near-Duplicate Removal

As mentioned in Section C.1, the final step of both the original CIFAR-10 and our dataset creation procedure is to remove near-duplicates. While removing near-duplicates between our new test set and the original CIFAR-10 dataset, we noticed that the original test set contained images that we would have ruled out as near-duplicates. A large number of near-duplicates between CIFAR-10 train and test, combined with our more stringent near-duplicate removal, could explain some of the accuracy drop. Indeed, we found about 800 images in the original CIFAR-10 test set that we would classify as near-duplicates (8% of the entire test set). More-


(a) Test set A

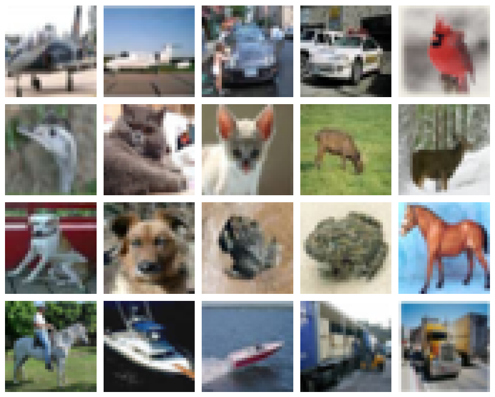

(b) Test set B

Figure 3. Randomly selected images from the original and new CIFAR-10 test sets. Each grid contains two images for each of the ten classes. The following footnote reveals which of the two grids corresponds to the new test set.[8]

over, most classifiers have accuracy between 99% and 100% on these near-duplicates (recall that most models achieve 100% training error). However, the following calculation shows that the near-duplicates can explain at most 1% of the observed difference.

For concreteness, we consider a model with 93% original test set accuracy such as a common VGG or ResNet architecture. Let acc_{true} be the "true" accuracy of the model on test images that are not near-duplicates, and let acc_{nd} be the accuracy on near-duplicates. Then for 8% near-duplicates, the overall accuracy is given by

$$\text{acc} \;=\; 0.92 \cdot \text{acc}_{\text{true}} + 0.08 \cdot \text{acc}_{\text{nd}} \;.$$

Using $\text{acc} = 0.93$, $\text{acc}_{\text{nd}} = 1.0$, and solving for acc_{true} then yields $\text{acc}_{\text{true}} \approx 0.924$. So the accuracy on original test images that are not near-duplicates is indeed lower, but only by a small amount (0.6%). This is in contrast to the 8% - 9% accuracy drop that VGG and ResNet models with 93% original accuracy see in our experiments.

For completeness, we describe our process for finding near duplicates in detail. For every test image, we visually inspected the top-10 nearest neighbors in both ℓ_2-distance and the SSIM (structural similarity) metric. We compared the original test set to the CIFAR-10 training set, and our new test set to both the original training and test sets. We consider an image pair as near-duplicates if both images have the same object in the same pose. We include images that have different zoom, color scale, stretch in the horizontal or vertical direction, or small shifts in vertical or horizontal position. If the object was rotated or in a different pose, we did not include it as a near-duplicate.

C.2.3. Hyperparameter Tuning

Another conjecture is that we can recover some of the missing accuracy by re-tuning hyperparameters of a model. To this end, we performed a grid search over multiple parameters of a VGG model. We selected three standard hyperparameters known to strongly influence test set performance: initial learning rate, dropout, and weight decay. The `vgg16_keras` architecture uses different amounts of dropout across different layers of the network, so we chose to tune a multiplicative scaling factor for the amount of dropout. This keeps the ratio of dropout across different layers constant.

We initialized a hyperparameter configuration from values tuned to the original test set (learning rate 0.1, dropout ratio 1, weight decay 5×10^{-4}), and performed a grid search across the following values:

- Learning rate in $\{0.0125, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8\}$.
- Dropout ratio in $\{0.5, 0.75, 1, 1.25, 1.75\}$.
- Weight decay in $\{5 \times 10^{-5},\ 1 \times 10^{-4},\ 5 \times 10^{-4},\ 1 \times 10^{-3}, 5 \times 10^{-3}\}$.

We ensured that the best performance was never at an extreme point of any range we tested for an individual hyperparameter. Overall, we did not find a hyperparameter setting with a significantly better accuracy on the new test set (the biggest improvement was from 85.3% to 85.8%).

C.2.4. Visually Inspecting Hard Images

It is also possible that we accidentally created a more difficult test set by including a set of "harder" images. To explore this question, we visually inspected the set of images that most models incorrectly classified. Figure 4 in Appendix C.3.5 shows examples of the hard images in our new test set that no model correctly classified. We find that all the new images are valid images that are recognizable to humans.

[8]Test Set A is the new test set and Test Set B is the original test set.

C.2.5. Human Accuracy Comparison

The visual inspection of hard images in the previous section is one way to compare the original and new test sets. However, our conclusion may be biased since we have created the new test set ourselves. To compare the relative hardness of the two test sets more objectively, we also conducted a small experiment to measure human accuray on the two test sets.[9] The goal of the experiment was to measure if human accuracy is significantly different on the original and new test sets.

Since we conjectured that our new test set included particularly hard images, we focused our experiment on the approximately 5% hardest images in both test sets. Here, "hardness" is defined by how many models correctly classified an image. After rounding to include all images that were classified by the same number of models, we obtained 500 images from the original test set and 115 images from our new test set.

We recruited nine graduate students from three different research groups in the Electrical Engineering & Computer Sciences Department at UC Berkeley. We wrote a simple user interface that allowed the participants to label images with one of the ten CIFAR-10 classes. To ensure that the participants did not know which dataset an image came from, we presented the images in random order.

Table 3 shows the results of our experiment. We find that four participants performed better on the original test set and five participants were better on our new test set. The average difference is -0.8%, i.e., the participants do not see a drop in average accuracy on this subset of original and new test images. This suggests that our new test set is not significantly harder for humans. However, we remark that our results here should only be seen as a preliminary study. Understanding human accuracy on CIFAR-10 in more detail will require further experiments.

C.2.6. Training on Part of Our New Test Set

If our new test set distribution is significantly different from the original CIFAR-10 distribution, retraining on part of our new test set (plus the original training data) may improve the accuracy on the held-out fraction of our new test set.

We conducted this experiment by randomly drawing a class-balanced split containing about 1,000 images from the new test set. We then added these images to the full CIFAR-10 training set and retrained the `vgg16_keras` model. After training, we tested the model on the remaining half of the new test set. We repeated this experiment twice with different randomly selected splits from our test set, obtaining accuracies of 85.1% and 85.4% (compared to 84.9% without the extra training data[10]). This provides evidence that there is no large distribution shift between our new test set and the original CIFAR-10 dataset, or that the model is unable to learn the modified distribution.

C.2.7. Cross-validation

Cross-validation can be a more reliable way of measuring a model's generalization ability than using only a single train / test split. Hence we tested if cross-validation on the original CIFAR-10 dataset could predict a model's error on our new test set. We created cross-validation data by randomly dividing the training set into 5 class-balanced splits. We then randomly shuffled together 4 out of the 5 training splits with the original test set. The leftover held-out split from the training set then became the new test set.

We retrained the models `vgg_15_BN_64`, `wide_resnet_28_10`, and `shake_shake_64d_cutout` on each of the 5 new datasets we created. The accuracies are reported in Table 4. The accuracies on the cross-validation splits did not differ much from the accuracy on the original test set. The variation among the cross-validation splits is significantly smaller than the drop on our new test set.

C.2.8. Training a Discriminator for Original vs. New Test Set

Our main hypothesis for the accuracy drop is that small variations in the test set creation process suffice to significantly reduce a model's accuracy. To test whether these variations could be detected by a convolutional network, we investigated whether a discriminator model could distinguish between the two test sets.

We first created a training set consisting of $3,200$ images (1,600 from the original test set and 1,600 from our new test set) and a test set of 800 images (consisting of 400 images from original and new test set each). Each image had a binary label indicating whether it came from the original or new test set. Additionally, we ensured that that both datasets were class balanced.

We then trained `resnet_32` and `resnet_110` models for 160 epochs using a standard SGD optimizer to learn a binary classifier between the two datasets. We conducted two variants of this experiment: in one variant, we traind the model from scratch. In the other variant, we started with a model pre-trained on the regular CIFAR-10 classification task.

Our results are summarized in Table 5. Overall we found

[9]Use of this data was permitted by the Berkelely Committee for Protection of Human Subjects (CPHS).

[10]This number is slightly lower than the accuracy of `vgg16_keras` on our new test set in Table 11, but still within the 95% confidence interval $[83.6, 86.8]$. Hence we conjecture that the difference is due to the random fluctuation arising from randomly initializing the model.

	Human Accuracy (%)		
	Original Test Set	New Test Set	Gap
Participant 1	85 [81.6, 88.0]	83 [74.2, 89.8]	2
Participant 2	83 [79.4, 86.2]	81 [71.9, 88.2]	2
Participant 3	82 [78.3, 85.3]	78 [68.6, 85.7]	4
Participant 4	79 [75.2, 82.5]	84 [75.3, 90.6]	-5
Participant 5	76 [72.0, 79.7]	77 [67.5, 84.8]	-1
Participant 6	75 [71.0, 78.7]	73 [63.2, 81.4]	2
Participant 7	74 [69.9, 77.8]	79 [69.7, 86.5]	-5
Participant 8	74 [69.9, 77.8]	76 [66.4, 84.0]	-2
Participant 9	67 [62.7, 71.1]	71 [61.1, 79.6]	-4

Table 3. Human accuracy on the "hardest" images in the original and our new CIFAR-10 test set. We ordered the images by number of incorrect classifications from models in our testbed and then selected the top 5% images from the original and new test set (500 images from the original test set, 115 images from our new test set). The results show that on average humans do not see a drop in accuracy on this subset of images.

	Model Accuracy (%)		
Dataset	`vgg_15_BN_64`	`wide_resnet_28_10`	`shake_shake_64d_cutout`
Original Test Set	93.6 [93.1, 94.1]	95.7 [95.3, 96.1]	97.1 [96.8, 97.4]
Split 1	93.9 [93.4, 94.3]	96.2 [95.8, 96.6]	97.2 [96.9, 97.5]
Split 2	93.8 [93.3, 94.3]	96.0 [95.6, 96.4]	97.3 [97.0, 97.6]
Split 3	94.0 [93.5, 94.5]	96.4 [96.0, 96.8]	97.4 [97.1, 97.7]
Split 4	94.0 [93.5, 94.5]	96.2 [95.8, 96.6]	97.4 [97.1, 97.7]
Split 5	93.5 [93.0, 94.0]	96.5 [96.1, 96.9]	97.4 [97.1, 97.7]
New Test Set	84.9 [83.2, 86.4]	89.7 [88.3, 91.0]	93.0 [91.8, 94.1]

Table 4. Model accuracies on cross-validation splits for the original CIFAR-10 data. The difference in cross-validation accuracies is significantly smaller than the drop to the new test set.

that the resulting models could not discriminate well between the original and our new test set: the best accuracy we obtained is 53.1%.

C.2.9. An Exactly Class-balanced Test Set

The top 25 keywords of each class in CIFAR-10 capture approximately 95% of the dataset. However, the remaining 5% of the dataset are skewed towards the class `ship`. As a result, our new dataset was not exactly class-balanced and contained only 8% images of class `ship` (as opposed to 10% in the original test set).

To measure whether this imbalance affected the acccuracy scores, we created an exactly class-balanced version of our new test set with 2,000 images (200 per class). In this version, we selected the top 50 keywords in each class and computed a fractional number of images for each keyword. We then rounded these numbers so that images for keywords with the largest fractional part were added first. The resulting model accuracies can be found in Table 12 (Appendix C.3.4). Models with lower original accuracies achieve a small accuracy improvement on the exactly class-balanced test set (around 0.3%), but the accuracy drop of the best-performing model remains unchanged.

C.3. Additional Figures, Tables, and Lists

In this appendix we provide large figures etc. that did not fit into the preceding sections about our CIFAR-10 experiments.

C.3.1. Keyword Distribution in CIFAR-10

The sub-tables in Table 6 show the keyword distribution for each of the ten classes in the original CIFAR-10 test set and our new test set.

Model	Discriminator Accuracy (%) random initialization	Discriminator Accuracy (%) pre-trained
resnet_32	50.1 [46.6, 53.6]	52.9 [49.4, 56.4]
resnet_110	50.3 [46.7, 53.8]	53.1 [49.6, 56.6]

Table 5. Accuracies for discriminator models trained to distinugish between the original and new CIFAR-10 test sets. The models were initialized either randomly or using a model pre-trained on the original CIFAR-10 dataset. Although the models performed slightly better than random chance, the confidence intervals (95% Clopper Pearson) still overlap with 50% accuracy.

Table 6. Distribution of the top 25 keywords in each class for the new and original test set.

Frog		
	New	Original
bufo_bufo	0.64%	0.63%
leopard_frog	0.64%	0.64%
bufo_viridis	0.59%	0.57%
rana_temporaria	0.54%	0.53%
bufo	0.49%	0.47%
bufo_americanus	0.49%	0.46%
toad	0.49%	0.46%
green_frog	0.45%	0.44%
rana_catesbeiana	0.45%	0.43%
bufo_marinus	0.45%	0.43%
bullfrog	0.45%	0.42%
american_toad	0.45%	0.43%
frog	0.35%	0.35%
rana_pipiens	0.35%	0.32%
toad_frog	0.30%	0.30%
spadefoot	0.30%	0.27%
western_toad	0.30%	0.26%
grass_frog	0.30%	0.27%
pickerel_frog	0.25%	0.24%
spring_frog	0.25%	0.22%
rana_clamitans	0.20%	0.20%
natterjack	0.20%	0.17%
crapaud	0.20%	0.18%
bufo_calamita	0.20%	0.18%
alytes_obstetricans	0.20%	0.16%

Cat		
	New	Original
tabby_cat	1.78%	1.78%
tabby	1.53%	1.52%
domestic_cat	1.34%	1.33%
cat	1.24%	1.25%
house_cat	0.79%	0.79%
felis_catus	0.69%	0.69%
mouser	0.64%	0.63%
felis_domesticus	0.54%	0.50%
true_cat	0.49%	0.47%
tomcat	0.49%	0.49%
alley_cat	0.30%	0.30%
felis_bengalensis	0.15%	0.11%
nougat	0.10%	0.05%
gray	0.05%	0.03%
manx_cat	0.05%	0.04%
fissiped	0.05%	0.03%
persian_cat	0.05%	0.03%
puss	0.05%	0.05%
catnap	0.05%	0.03%
tiger_cat	0.05%	0.03%
black_cat	0.05%	0.04%
bedspread	0.00%	0.02%
siamese_cat	0.00%	0.02%
tortoiseshell	0.00%	0.02%
kitty-cat	0.00%	0.02%

Dog		
	New	Original
pekingese	1.24%	1.22%
maltese	0.94%	0.93%
puppy	0.89%	0.87%
chihuahua	0.84%	0.81%
dog	0.69%	0.67%
pekinese	0.69%	0.66%
toy_spaniel	0.59%	0.60%
mutt	0.49%	0.47%
mongrel	0.49%	0.49%
maltese_dog	0.45%	0.43%
toy_dog	0.40%	0.36%
japanese_spaniel	0.40%	0.38%
blenheim_spaniel	0.35%	0.35%
english_toy_spaniel	0.35%	0.31%
domestic_dog	0.35%	0.32%
peke	0.30%	0.28%
canis_familiaris	0.30%	0.27%
lapdog	0.30%	0.30%
king_charles_spaniel	0.20%	0.17%
toy	0.15%	0.13%
feist	0.10%	0.06%
pet	0.10%	0.07%
cavalier	0.10%	0.05%
canine	0.05%	0.04%
cur	0.05%	0.04%

Deer		
	New	Original
elk	0.79%	0.77%
capreolus_capreolus	0.74%	0.71%
cervus_elaphus	0.64%	0.61%
fallow_deer	0.64%	0.63%
roe_deer	0.59%	0.60%
deer	0.59%	0.60%
muntjac	0.54%	0.51%
mule_deer	0.54%	0.51%
odocoileus_hemionus	0.49%	0.50%
fawn	0.49%	0.49%
alces_alces	0.40%	0.36%
wapiti	0.40%	0.36%
american_elk	0.40%	0.35%
red_deer	0.35%	0.33%
moose	0.35%	0.35%
rangifer_caribou	0.25%	0.24%
rangifer_tarandus	0.25%	0.24%
caribou	0.25%	0.23%
sika	0.25%	0.22%
woodland_caribou	0.25%	0.21%
dama_dama	0.20%	0.19%
cervus_sika	0.20%	0.16%
barking_deer	0.20%	0.18%
sambar	0.15%	0.15%
stag	0.15%	0.13%

Bird		
	New	Original
cassowary	0.89%	0.85%
bird	0.84%	0.84%
wagtail	0.74%	0.74%
ostrich	0.69%	0.68%
struthio_camelus	0.54%	0.51%
sparrow	0.54%	0.52%
emu	0.54%	0.51%
pipit	0.49%	0.47%
passerine	0.49%	0.50%
accentor	0.49%	0.49%
honey_eater	0.40%	0.37%
dunnock	0.40%	0.37%
alauda_arvensis	0.30%	0.26%
nandu	0.30%	0.27%
prunella_modularis	0.30%	0.30%
anthus_pratensis	0.30%	0.28%
finch	0.25%	0.24%
lark	0.25%	0.20%
meadow_pipit	0.25%	0.20%
rhea_americana	0.25%	0.21%
flightless_bird	0.15%	0.10%
emu_novaehollandiae	0.15%	0.12%
dromaius_novaehollandiae	0.15%	0.14%
apteryx	0.15%	0.10%
flying_bird	0.15%	0.13%

Ship		
	New	Original
passenger_ship	0.79%	0.78%
boat	0.64%	0.64%
cargo_ship	0.40%	0.37%
cargo_vessel	0.40%	0.39%
pontoon	0.35%	0.31%
container_ship	0.35%	0.31%
speedboat	0.35%	0.32%
freighter	0.35%	0.32%
pilot_boat	0.35%	0.31%
ship	0.35%	0.31%
cabin_cruiser	0.30%	0.29%
police_boat	0.30%	0.25%
sea_boat	0.30%	0.29%
oil_tanker	0.30%	0.29%
pleasure_boat	0.25%	0.21%
lightship	0.25%	0.22%
powerboat	0.25%	0.25%
guard_boat	0.25%	0.20%
dredger	0.25%	0.20%
hospital_ship	0.25%	0.21%
banana_boat	0.20%	0.19%
merchant_ship	0.20%	0.17%
liberty_ship	0.20%	0.15%
container_vessel	0.20%	0.19%
tanker	0.20%	0.18%

Truck		
	New	Original
dump_truck	0.89%	0.89%
trucking_rig	0.79%	0.76%
delivery_truck	0.64%	0.61%
truck	0.64%	0.65%
tipper_truck	0.64%	0.60%
camion	0.59%	0.58%
fire_truck	0.59%	0.55%
lorry	0.54%	0.53%
garbage_truck	0.54%	0.53%
moving_van	0.35%	0.32%
tractor_trailer	0.35%	0.34%
tipper	0.35%	0.30%
aerial_ladder_truck	0.35%	0.34%
ladder_truck	0.30%	0.26%
fire_engine	0.30%	0.27%
dumper	0.30%	0.28%
trailer_truck	0.30%	0.28%
wrecker	0.30%	0.27%
articulated_lorry	0.25%	0.24%
tipper_lorry	0.25%	0.25%
semi	0.20%	0.18%
sound_truck	0.15%	0.12%
tow_truck	0.15%	0.12%
delivery_van	0.15%	0.11%
bookmobile	0.10%	0.10%

Horse		
	New	Original
arabian	1.14%	1.12%
lipizzan	1.04%	1.02%
broodmare	0.99%	0.97%
gelding	0.74%	0.73%
quarter_horse	0.74%	0.72%
stud_mare	0.69%	0.69%
lippizaner	0.54%	0.52%
appaloosa	0.49%	0.45%
lippizan	0.49%	0.46%
dawn_horse	0.45%	0.42%
stallion	0.45%	0.43%
tennessee_walker	0.45%	0.45%
tennessee_walking_horse	0.40%	0.38%
walking_horse	0.30%	0.28%
riding_horse	0.20%	0.20%
saddle_horse	0.20%	0.18%
female_horse	0.15%	0.11%
cow_pony	0.15%	0.11%
male_horse	0.15%	0.14%
buckskin	0.15%	0.13%
horse	0.10%	0.08%
equine	0.10%	0.08%
quarter	0.10%	0.07%
cavalry_horse	0.10%	0.09%
thoroughbred	0.10%	0.06%

Airplane		
	New	Original
stealth_bomber	0.94%	0.92%
airbus	0.89%	0.89%
stealth_fighter	0.79%	0.80%
fighter_aircraft	0.79%	0.76%
biplane	0.74%	0.74%
attack_aircraft	0.69%	0.67%
airliner	0.64%	0.61%
jetliner	0.59%	0.56%
monoplane	0.54%	0.55%
twinjet	0.54%	0.52%
dive_bomber	0.54%	0.52%
jumbo_jet	0.49%	0.47%
jumbojet	0.35%	0.35%
propeller_plane	0.30%	0.28%
fighter	0.20%	0.20%
plane	0.20%	0.15%
amphibious_aircraft	0.20%	0.20%
multiengine_airplane	0.15%	0.14%
seaplane	0.15%	0.14%
floatplane	0.10%	0.05%
multiengine_plane	0.10%	0.06%
reconnaissance_plane	0.10%	0.09%
airplane	0.10%	0.08%
tail	0.10%	0.05%
joint	0.05%	0.04%

Automobile		
	New	Original
coupe	1.29%	1.26%
convertible	1.19%	1.18%
station_wagon	0.99%	0.98%
automobile	0.89%	0.90%
car	0.84%	0.81%
auto	0.84%	0.83%
compact_car	0.79%	0.76%
shooting_brake	0.64%	0.63%
estate_car	0.59%	0.59%
wagon	0.54%	0.51%
police_cruiser	0.45%	0.45%
motorcar	0.40%	0.40%
taxi	0.20%	0.17%
cruiser	0.15%	0.13%
compact	0.15%	0.11%
beach_wagon	0.15%	0.13%
funny_wagon	0.10%	0.05%
gallery	0.10%	0.07%
cab	0.10%	0.07%
ambulance	0.10%	0.07%
door	0.00%	0.03%
ford	0.00%	0.03%
opel	0.00%	0.03%
sport_car	0.00%	0.03%
sports_car	0.00%	0.03%

C.3.2. Full List of Models Evaluated on CIFAR-10

The following list contains all models we evaluated on CIFAR-10 with references and links to the corresponding source code.

1. `autoaug_pyramid_net` (Cubuk et al., 2018; Han et al., 2017) https://github.com/tensorflow/models/tree/master/research/autoaugment
2. `autoaug_shake_shake_112` (Cubuk et al., 2018; Gastaldi, 2017) https://github.com/tensorflow/models/tree/master/research/autoaugment
3. `autoaug_shake_shake_32` (Cubuk et al., 2018; Gastaldi, 2017) https://github.com/tensorflow/models/tree/master/research/autoaugment
4. `autoaug_shake_shake_96` (Cubuk et al., 2018; Gastaldi, 2017) https://github.com/tensorflow/models/tree/master/research/autoaugment
5. `autoaug_wrn` (Cubuk et al., 2018; Zagoruyko and Komodakis, 2016) https://github.com/tensorflow/models/tree/master/research/autoaugment
6. `cudaconvnet` (Krizhevsky et al., 2012) https://github.com/akrizhevsky/cuda-convnet2
7. `darc` (Kawaguchi et al., 2017) http://lis.csail.mit.edu/code/gdl.html
8. `densenet_BC_100_12` (Huang et al., 2017) https://github.com/hysts/pytorch_image_classification/
9. `nas` (Zoph et al., 2018) https://github.com/tensorflow/models/blob/master/research/slim/nets/nasnet/nasnet.py#L32
10. `pyramidnet_basic_110_270` (Han et al., 2017) https://github.com/hysts/pytorch_image_classification/
11. `pyramidnet_basic_110_84` (Han et al., 2017) https://github.com/hysts/pytorch_image_classification/
12. `random_features_256k_aug` (Coates et al., 2011) https://github.com/modestyachts/nondeep Random 1 layer convolutional network with 256k filters sampled from image patches, patch size = 6, pool size 15, pool stride 6, and horizontal flip data augmentation.
13. `random_features_256k` (Coates et al., 2011) ht

tps://github.com/modestyachts/nondeep Random 1 layer convolutional network with 256k filters sampled from image patches, patch size = 6, pool size 15, pool stride 6.

14. random_features_32k_aug (Coates et al., 2011) https://github.com/modestyachts/nondeep Random 1 layer convolutional network with 32k filters sampled from image patches, patch size = 6, pool size 15, pool stride 6, and horizontal flip data augmentation.
15. random_features_32k (Coates et al., 2011) Random 1 layer convolutional network with 32k filters sampled from image patches, patch size = 6, pool size 15, pool stride 16.
16. resnet_basic_32 (He et al., 2016a) https://github.com/hysts/pytorch_image_classification/
17. resnet_basic_44 (He et al., 2016a) https://github.com/hysts/pytorch_image_classification/
18. resnet_basic_56 (He et al., 2016a) https://github.com/hysts/pytorch_image_classification/
19. resnet_basic_110 (He et al., 2016a) https://github.com/hysts/pytorch_image_classification/
20. resnet_preact_basic_110 (He et al., 2016b) https://github.com/hysts/pytorch_image_classification/
21. resnet_preact_bottleneck_164 (He et al., 2016b) https://github.com/hysts/pytorch_image_classification/
22. resnet_preact_tf (He et al., 2016b) https://github.com/tensorflow/models/tree/b871670b5ae29aaa6cad1b2d4e004882f716c466/resnet
23. resnext_29_4x64d (Xie et al., 2017) https://github.com/hysts/pytorch_image_classification/
24. resnext_29_8x64d (Xie et al., 2017) https://github.com/hysts/pytorch_image_classification/
25. shake_drop (Yamada et al., 2018) https://github.com/imenurok/ShakeDrop
26. shake_shake_32d (Gastaldi, 2017) https://github.com/hysts/pytorch_image_classification/
27. shake_shake_64d (Gastaldi, 2017) https://github.com/hysts/pytorch_image_classification/
28. shake_shake_96d (Gastaldi, 2017) https://github.com/hysts/pytorch_image_classification/
29. shake_shake_64d_cutout (DeVries and Taylor, 2017; Gastaldi, 2017) https://github.com/hysts/pytorch_image_classification/
30. vgg16_keras (Liu and Deng, 2015; Simonyan and Zisserman, 2014) https://github.com/geifmany/cifar-vgg
31. vgg_15_BN_64 (Liu and Deng, 2015; Simonyan and Zisserman, 2014) https://github.com/hysts/pytorch_image_classification/
32. wide_resnet_tf (Zagoruyko and Komodakis, 2016) https://github.com/tensorflow/models/tree/b871670b5ae29aaa6cad1b2d4e004882f716c466/resnet
33. wide_resnet_28_10 (Zagoruyko and Komodakis, 2016) https://github.com/hysts/pytorch_image_classification/
34. wide_resnet_28_10_cutout (DeVries and Taylor, 2017; Zagoruyko and Komodakis, 2016) https://github.com/hysts/pytorch_image_classification/

C.3.3. Full Results Table

Table 11 contains the detailed accuracy scores for the original CIFAR-10 test set and our new test set.

C.3.4. Full Results Table for the Exactly Class-Balanced Test Set

Table 12 contains the detailed accuracy scores for the original CIFAR-10 test set and the exactly class-balanced variant of our new test set.

C.3.5. Hard Images

Figure 4 shows the images in our new CIFAR-10 test set that were misclassified by all models in our testbed. As can be seen in the figure, the class labels for these images are correct.

D. Details of the ImageNet Experiments

Our results on CIFAR-10 show that current models fail to reliably generalize in the presence of small variations in the data distribution. One hypothesis is that the accuracy drop stems from the limited nature of the CIFAR-10 dataset. Compared to other datasets, CIFAR-10 is relatively small, both in terms of image resolution and the number of images in the dataset. Since the CIFAR-10 models are only exposed to a constrained visual environment, they may be unable to

Table 11. Model accuracy on the original CIFAR-10 test set and our new test set. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, $\Delta\text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. Due to space constraints, references for the models can be found in Appendix C.3.2.

CIFAR-10

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	autoaug_pyramid_net_tf	98.4 [98.1, 98.6]	95.5 [94.5, 96.4]	2.9	1	0
2	autoaug_shake_shake_112_	98.1 [97.8, 98.4]	93.9 [92.7, 94.9]	4.3	2	0
3	autoaug_shake_shake_96_t	98.0 [97.7, 98.3]	93.7 [92.6, 94.7]	4.3	3	0
4	autoaug_wrn_tf	97.5 [97.1, 97.8]	93.0 [91.8, 94.1]	4.4	4	0
5	autoaug_shake_shake_32_t	97.3 [97.0, 97.6]	92.9 [91.7, 94.0]	4.4	6	-1
6	shake_shake_64d_cutout	97.1 [96.8, 97.4]	93.0 [91.8, 94.1]	4.1	5	1
7	shake_shake_26_2x96d_SS	97.1 [96.7, 97.4]	91.9 [90.7, 93.1]	5.1	9	-2
8	shake_shake_64d	97.0 [96.6, 97.3]	91.4 [90.1, 92.6]	5.6	10	-2
9	wrn_28_10_cutout16	97.0 [96.6, 97.3]	92.0 [90.7, 93.1]	5.0	8	1
10	shake_drop	96.9 [96.5, 97.2]	92.3 [91.0, 93.4]	4.6	7	3
11	shake_shake_32d	96.6 [96.2, 96.9]	89.8 [88.4, 91.1]	6.8	13	-2
12	darc	96.6 [96.2, 96.9]	89.5 [88.1, 90.8]	7.1	16	-4
13	resnext_29_4x64d	96.4 [96.0, 96.7]	89.6 [88.2, 90.9]	6.8	15	-2
14	pyramidnet_basic_110_27	96.3 [96.0, 96.7]	90.5 [89.1, 91.7]	5.9	11	3
15	resnext_29_8x64d	96.2 [95.8, 96.6]	90.0 [88.6, 91.2]	6.3	12	3
16	wrn_28_10	95.9 [95.5, 96.3]	89.7 [88.3, 91.0]	6.2	14	2
17	pyramidnet_basic_110_84	95.7 [95.3, 96.1]	89.3 [87.8, 90.6]	6.5	17	0
18	densenet_BC_100_12	95.5 [95.1, 95.9]	87.6 [86.1, 89.0]	8.0	20	-2
19	nas	95.4 [95.0, 95.8]	88.8 [87.4, 90.2]	6.6	18	1
20	wide_resnet_tf_28_10	95.0 [94.6, 95.4]	88.5 [87.0, 89.9]	6.5	19	1
21	resnet_v2_bottleneck_16	94.2 [93.7, 94.6]	85.9 [84.3, 87.4]	8.3	22	-1
22	vgg16_keras	93.6 [93.1, 94.1]	85.3 [83.6, 86.8]	8.3	23	-1
23	resnet_basic_110	93.5 [93.0, 93.9]	85.2 [83.5, 86.7]	8.3	24	-1
24	resnet_v2_basic_110	93.4 [92.9, 93.9]	86.5 [84.9, 88.0]	6.9	21	3
25	resnet_basic_56	93.3 [92.8, 93.8]	85.0 [83.3, 86.5]	8.3	25	0
26	resnet_basic_44	93.0 [92.5, 93.5]	84.2 [82.6, 85.8]	8.8	29	-3
27	vgg_15_BN_64	93.0 [92.5, 93.5]	84.9 [83.2, 86.4]	8.1	27	0
28	resnetv2_tf_32	92.7 [92.2, 93.2]	84.4 [82.7, 85.9]	8.3	28	0
29	resnet_basic_32	92.5 [92.0, 93.0]	84.9 [83.2, 86.4]	7.7	26	3
30	cudaconvnet	88.5 [87.9, 89.2]	77.5 [75.7, 79.3]	11.0	30	0
31	random_features_256k_au	85.6 [84.9, 86.3]	73.1 [71.1, 75.1]	12.5	31	0
32	random_features_32k_aug	85.0 [84.3, 85.7]	71.9 [69.9, 73.9]	13.0	32	0
33	random_features_256k	84.2 [83.5, 84.9]	69.9 [67.8, 71.9]	14.3	33	0
34	random_features_32k	83.3 [82.6, 84.0]	67.9 [65.9, 70.0]	15.4	34	0

learn a more reliable representation.

To investigate whether ImageNet models generalize more reliably, we assemble a new test set for ImageNet. ImageNet captures a much broader variety of natural images: it contains about 24× more training images than CIFAR-10 with roughly 100× more pixels per image. As a result, ImageNet poses a significantly harder problem and is among the most prestigious machine learning benchmarks. The steadily improving accuracy numbers have also been cited as an important breakthrough in machine learning (Malik, 2017). If popular ImageNet models are indeed more robust to natural variations in the data (and there is again no adaptive overfitting), the accuracies on our new test set should roughly match the existing accuracies.

Before we proceed to our experiments, we briefly describe the relevant background concerning the ImageNet dataset. For more details, we refer the reader to the original ImageNet publications (Deng et al., 2009; Russakovsky et al.,

Table 12. Model accuracy on the original CIFAR-10 test set and the exactly class-balanced variant of our new test set. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, $\Delta\text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. Due to space constraints, references for the models can be found in Appendix C.3.2.

CIFAR-10

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	autoaug_pyramid_net_tf	98.4 [98.1, 98.6]	95.5 [94.5, 96.4]	2.9	1	0
2	autoaug_shake_shake_112_	98.1 [97.8, 98.4]	94.0 [92.9, 95.0]	4.1	2	0
3	autoaug_shake_shake_96_t	98.0 [97.7, 98.3]	93.9 [92.8, 94.9]	4.1	3	0
4	autoaug_wrn_tf	97.5 [97.1, 97.8]	93.0 [91.8, 94.1]	4.5	6	-2
5	autoaug_shake_shake_32_t	97.3 [97.0, 97.6]	93.2 [92.0, 94.2]	4.2	4	1
6	shake_shake_64d_cutout	97.1 [96.8, 97.4]	93.1 [91.9, 94.2]	4.0	5	1
7	shake_shake_26_2x96d_SSI	97.1 [96.7, 97.4]	92.0 [90.7, 93.1]	5.1	9	-2
8	shake_shake_64d	97.0 [96.6, 97.3]	91.9 [90.6, 93.1]	5.1	10	-2
9	wrn_28_10_cutout16	97.0 [96.6, 97.3]	92.1 [90.8, 93.2]	4.9	8	1
10	shake_drop	96.9 [96.5, 97.2]	92.3 [91.1, 93.4]	4.6	7	3
11	shake_shake_32d	96.6 [96.2, 96.9]	90.0 [88.6, 91.3]	6.6	15	-4
12	darc	96.6 [96.2, 96.9]	89.9 [88.5, 91.2]	6.7	16	-4
13	resnext_29_4x64d	96.4 [96.0, 96.7]	90.1 [88.8, 91.4]	6.2	12	1
14	pyramidnet_basic_110_27(	96.3 [96.0, 96.7]	90.5 [89.1, 91.7]	5.8	11	3
15	resnext_29_8x64d	96.2 [95.8, 96.6]	90.1 [88.7, 91.4]	6.1	14	1
16	wrn_28_10	95.9 [95.5, 96.3]	90.1 [88.8, 91.4]	5.8	13	3
17	pyramidnet_basic_110_84	95.7 [95.3, 96.1]	89.6 [88.2, 90.9]	6.1	17	0
18	densenet_BC_100_12	95.5 [95.1, 95.9]	87.9 [86.4, 89.3]	7.6	20	-2
19	nas	95.4 [95.0, 95.8]	89.2 [87.8, 90.5]	6.2	18	1
20	wide_resnet_tf_28_10	95.0 [94.6, 95.4]	88.8 [87.4, 90.2]	6.2	19	1
21	resnet_v2_bottleneck_164	94.2 [93.7, 94.6]	86.1 [84.5, 87.6]	8.1	22	-1
22	vgg16_keras	93.6 [93.1, 94.1]	85.6 [84.0, 87.1]	8.0	23	-1
23	resnet_basic_110	93.5 [93.0, 93.9]	85.4 [83.8, 86.9]	8.1	24	-1
24	resnet_v2_basic_110	93.4 [92.9, 93.9]	86.9 [85.4, 88.3]	6.5	21	3
25	resnet_basic_56	93.3 [92.8, 93.8]	84.9 [83.2, 86.4]	8.5	28	-3
26	resnet_basic_44	93.0 [92.5, 93.5]	84.8 [83.2, 86.3]	8.2	29	-3
27	vgg_15_BN_64	93.0 [92.5, 93.5]	85.0 [83.4, 86.6]	7.9	27	0
28	resnetv2_tf_32	92.7 [92.2, 93.2]	85.1 [83.5, 86.6]	7.6	26	2
29	resnet_basic_32	92.5 [92.0, 93.0]	85.2 [83.6, 86.7]	7.3	25	4
30	cudaconvnet	88.5 [87.9, 89.2]	78.2 [76.3, 80.0]	10.3	30	0
31	random_features_256k_auc	85.6 [84.9, 86.3]	73.6 [71.6, 75.5]	12.0	31	0
32	random_features_32k_aug	85.0 [84.3, 85.7]	72.2 [70.2, 74.1]	12.8	32	0
33	random_features_256k	84.2 [83.5, 84.9]	70.5 [68.4, 72.4]	13.8	33	0
34	random_features_32k	83.3 [82.6, 84.0]	68.7 [66.6, 70.7]	14.6	34	0

2015).

ImageNet. ImageNet (Deng et al., 2009; Russakovsky et al., 2015) is a large image database consisting of more than 14 million human-annotated images depicting almost 22,000 classes. The images do not have a uniform size, but most of them are stored as RGB color images with a resolution around 500×400 pixels. The classes are derived from the WordNet hierarchy (Miller, 1995), which represents each class by a set of synonyms (“synset”) and is organized into semantically meaningful relations. Each class has an associated definition (“gloss”) and a unique WordNet ID (“wnid”).

The ImageNet team populated the classes with images downloaded from various image search engines, using the WordNet synonyms as queries. The researchers then annotated the images via Amazon Mechanical Turk (MTurk). A class-specific threshold decided how many agreements among the MTurk workers were necessary for an image to be consid-

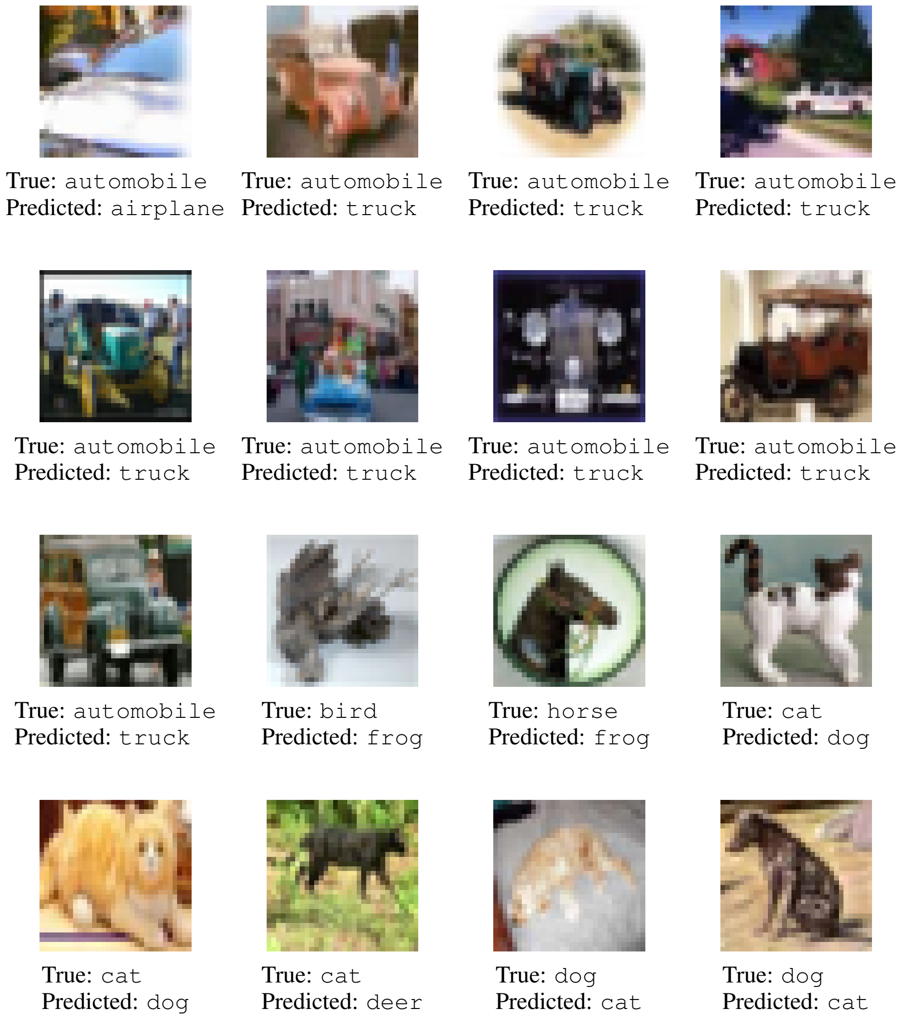


Figure 4. Hard images from our new test set that no model correctly. The caption of each image states the correct class label ("True") and the label predicted by most models ("Predicted").

ered valid. Overall, the researchers employed over 49,000 workers from 167 countries (Li and Deng, 2017).

Since 2010, the ImageNet team has run the yearly ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which consists of separate tracks for object classification, localization, and detection. All three tracks are based on subsets of the ImageNet data. The classification track has received the most attention and is also the focus of our paper.

The ILSVRC2012 competition data has become the de facto benchmark version of the dataset and comprises 1.2 million training images, 50,000 validation images, and 100,000 test images depicting 1,000 categories. We generally refer to this data as the ImageNet training, validation, and test set. The labels for the ImageNet test set were never publicly released in order to minimize adaptive overfitting. Instead, teams could submit a limited number of requests to an evaluation server in order to obtain accuracy scores. There were no similar limitations in place for the validation set. Most publications report accuracy numbers on the validation set.

The training, validation, and test sets were not drawn strictly i.i.d. from the same distribution (i.e., there was not a single data collection run with the result split randomly into training, validation, and test). Instead, the data collection was an ongoing process and both the validation and test sets were refreshed in various years of the ILSVRC. One notable difference is that the ImageNet training and validation sets do not have the same data source: while the ImageNet training set consists of images from several search engines (e.g., Google, MSN, Yahoo, and Flickr), the validation set consists almost entirely of images from Flickr (Berg, 2018).

D.1. Dataset Creation Methodology

Since the existing training, validation, and test sets are not strictly i.i.d. (see above), the first question was which dataset part to replicate. For our experiment, we decided to match the distribution of the *validation set*. There are multiple reasons for this choice:

- In contrast to the training set, the validation set comes from only one data source (Flickr). Moreover, the Flickr API allows fine-grained searches, which makes it easier to control the data source and match the original distribution.
- In contrast to the original test set, the validation set comes with label information. This makes it easier to inspect the existing image distribution for each class, which is important to ensure that we match various intricacies of the dataset (e.g., see Appendix D.4.8 for examples of ambiguous classes).
- Most papers report accuracy numbers on the validation set. Hence comparing new vs. existing accuracies is most relevant for the validation set.
- The validation set is commonly used to develop new architectures and tune hyperparameters, which leads to the possibility of adaptive overfitting. If we again observe no diminishing returns in accuracy on our new test set, this indicates that even the validation set is resilient to adaptive overfitting.

Therefore, our goal was to replicate the distribution of the original validation set as closely as possible. We aimed for a new test set of size 10,000 since this would already result in accuracy scores with small confidence intervals (see Section 2). While a larger dataset would result in even smaller confidence intervals, we were also concerned that searching for more images might lead to a larger distribution shift. In particular, we decided to use a time range for our Flickr queries *after* the original ImageNet collection period (see below for the corresponding considerations). Since a given time period only has a limited supply of high quality images, a larger test set would have required a longer time range. This in turn may create a larger temporal distribution shift. To balance these two concerns, we decided on a size of 10,000 images for the new test set.

Figure 5 presents a visual overview of our dataset creation pipeline. It consists of two parts: creating a pool of candidate images and sampling a clean dataset from this candidate pool. We now describe each part in detail to give the reader insights into the design choices potentially affecting the final distribution.

D.1.1. Creating a Candidate Pool

Similar to the creation procedure for the original ImageNet validation set, we collected candidate images from the Flickr image hosting service and then annotated them with Amazon Mechanical Turk (MTurk).

Downloading images from Flickr. The Flickr API has a range of parameters for image searches such as the query terms, an allowed time range, a maximum number of returned images, and a sorting order. We summarize the main points here:

- **Query terms:** For each class, we used each of the WordNet synonyms as a search term in separate queries.
- **Date range:** There were two main options for the date range associated with our queries to Flickr: either the same date range as the original ImageNet data collection, or a date range directly after ImageNet. The advantage of using the ImageNet date range is that it avoids a distribution shift due to the time the images were taken. However, this option also comes with two important caveats: First, the pool of high quality images in the original ImageNet date range could have been largely exhausted by ImageNet. Second, the new dataset could end up with near-duplicates of images in the original validation or training set that are hard to detect. Especially the first issue is difficult to quantify, so we decided on a time range directly after the ImageNet collection period.

  In particular, we initially searched for images taken and uploaded to Flickr between July 11, 2012 and July 11, 2013 because the final ILSVRC2012 public data release was on July 10, 2012. Since we used a period of only one year (significantly shorter than the ImageNet collection period), we believe that the temporal component of the distribution shift is small.
- **Result size:** We initially downloaded up to 100 images for each class. If a class has k synonyms associated with it, we requested $100/k$ images for each synonym. We decided on 100 images per class since we aimed for 10,000 images overall and estimated that 10% of the candidate images would be of sufficiently high quality (similar to ImageNet (Deng et al., 2009)).
- **Result order:** Flickr offers the sorting options "relevance", "interestingness", and various temporal orderings. Note that the "relevance" and "interestingness" orderings may rely on machine learning models trained on ImageNet. Since these orderings may introduce a significant bias (e.g., by mainly showing images that current ImageNet models recognize for the respective search term), we chose to order the images by their upload date. This helps to ensure that our new test set is independent of current classifiers.

After our first data collection, we found it necessary to expand the initial candidate pool for particular classes in order to reach a sufficient number of valid images. This is similar to the original ImageNet creation process, where the authors expanded the set of queries using two methods

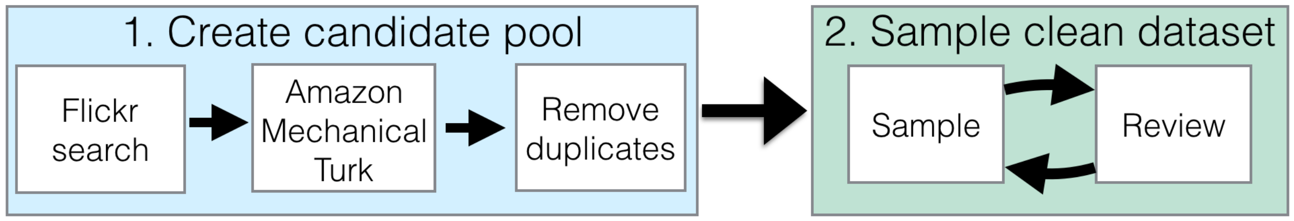


Figure 5. The pipeline for the new ImageNet test set. It consists of two parts: creating the candidate pool and sampling the final dataset from this candidate pool.

(Deng et al., 2009; Russakovsky et al., 2015). The first method appended a word from the parent class to the queries if this word also appeared in the gloss of the target class. The second method included translations of the queries into other languages such as Chinese, Spanish, Dutch, and Italian.

We took the following steps to expand our search queries, only proceeding to the next step for a given class when in need of more images.

1. Append a word from the parent class if the word appears in the gloss of the target class.
2. Expand the maximum number of images to 200 for this class.
3. Expand the search range to include photos taken or uploaded before July 11, 2014 (i.e., a time span of two years instead of one).
4. Concatenate compound queries, i.e., search for "dialphone" instead of "dial phone".
5. Manually pick alternative query words, including translations of the queries.

In total, we obtained 208,145 candidate images from Flickr.

Amazon Mechanical Turk. While the candidate images from Flickr are correlated with their corresponding class, a large number of images are still unsuitable for an image classification dataset. For instance, the images may be of low quality (blurry, unclear object presence, etc.), violate dataset rules (e.g., no paintings), or be simply unrelated to the target class. So similar to ImageNet, we utilized MTurk to filter our pool of candidate images.

We designed our MTurk tasks and UI to be close to those used in ImageNet. As in ImageNet, we showed each MTurk worker a grid of 48 candidate images for a given target class. The task description was derived from the original ImageNet instructions and included the definition of the target class with a link to a corresponding Wikipedia page. We asked the MTurk workers to select images belonging to the target class regardless of "occlusions, other objects, and clutter or text in the scene" and to avoid drawings or paintings (both as in ImageNet). Appendix D.4.1 shows a screenshot of our UI and a screenshot of the original UI for comparison.

For quality control, we embedded at least six randomly selected images from the original validation set in each MTurk task (three from the same class, three from a class that is nearby in the WordNet hierarchy). These images appeared in random locations of the image grid for each task. We obfuscated all image URLs and resized our images to match the most common size of the existing validation images so that the original validation images were not easy to spot.

The main outcome of the MTurk tasks is a *selection frequency* for each image, i.e., what fraction of MTurk workers selected the image in a task for its target class. We recruited at least ten MTurk workers for each task (and hence for each image), which is similar to ImageNet. Since each task contained original validation images, we could also estimate how often images from the original dataset were selected by our MTurk workers.

Removing near-duplicate images. The final step in creating the candidate pool was to remove near-duplicates, both within our new test set and between our new test set and the original ImageNet dataset. Both types of near-duplicates could harm the quality of our dataset.

Since we obtained results from Flickr in a temporal ordering, certain events (e.g., the 2012 Olympics) led to a large number of similar images depicting the same scene (e.g., in the class for the "horizontal bar" gymnastics instrument). Inspecting the ImageNet validation set revealed only very few sets of images from a single event. Moreover, the ImageNet paper also remarks that they removed near-duplicates (Deng et al., 2009). Hence we decided to remove near-duplicates within our new test set.

Near-duplicates between our dataset and the original test set are also problematic. Since the models typically achieve high accuracy on the training set, testing on a near-duplicate of a training image checks for memorization more than generalization. A near-duplicate between the existing validation set and our new test set also defeats the purpose of measuring generalization to previously unseen data (as opposed to data that may already have been the victim of adaptive overfitting).

To find near-duplicates, we computed the 30 nearest neighbors for each candidate image in three different metrics: ℓ_2-distance on raw pixels, ℓ_2-distance on features extracted from a pre-trained VGG (Simonyan and Zisserman, 2014) model (fc7), and SSIM (structural similarity) (Wang et al., 2004), which is a popular image similarity metric. For metrics that were cheap to evaluate (ℓ_2-distance on pixels and ℓ_2-distance on fc7), we computed nearest neighbor distances to all candidate images and all of the original ImageNet data. For the more compute-intensive SSIM metric, we restricted the set of reference images to include all candidate images and the five closest ImageNet classes based on the tree distance in the WordNet hierarchy. We then manually reviewed nearest neighbor pairs below certain thresholds for each metric and removed any duplicates we discovered.

To the best of our knowledge, ImageNet used only nearest neighbors in the ℓ_2-distance to find near-duplicates (Berg, 2018). While this difference may lead to a small change in distribution, we still decided to use multiple metrics since including images that have near-duplicates in ImageNet would be contrary to the main goal of our experiment. Moreover, a manual inspection of the original validation set revealed only a very small number of near-duplicates within the existing dataset.

D.1.2. Sampling a Clean Dataset

The result of collecting a candidate pool was a set of images with annotations from MTurk, most importantly the selection frequency of each image. In the next step, we used this candidate pool to sample a new test set that closely resembles the distribution of the existing validation set. There were two main difficulties in this process.

First, the ImageNet publications do not provide the agreement thresholds for each class that were used to determine which images were valid. One possibility could be to run the algorithm the ImageNet authors designed to compute the agreement thresholds. However, this algorithm would need to be exactly specified, which is unfortunately not the case to the best of our knowledge.[11]

[11]To be precise: Jia Deng's PhD thesis (Deng, 2012) provides a clear high-level description of their algorithm for computing agreement thresholds. However – as is commonly the case in synopses of algorithms – the description still omits some details such as the binning procedure or the number of images used to compute the thresholds. Since it is usually hard to exactly reconstruct a non-trivial algorithm from an informal summary, we instead decided to implement three different sampling strategies and compare their outcomes. Potential deviations from the ImageNet sampling procedure are also alleviated by the fact that our MTurk tasks always included at least a few images from the original validation set, which allowed us to calibrate our sampling strategies to match the existing ImageNet data.

Second, and more fundamentally, it is impossible to exactly replicate the MTurk worker population from 2010 – 2012 with a reproducibility experiment in 2018. Even if we had access to the original agreement thresholds, it is unclear if they would be meaningful for our MTurk data collection (e.g., because the judgments of our annotations could be different). Similarly, re-running the algorithm for computing agreement thresholds could give different results with our MTurk worker population.

So instead of attempting to directly replicate the original agreement thresholds, we instead explored three different sampling strategies. An important asset in this part of our experiment was that we had inserted original validation images into the MTurk tasks (see the previous subsection). So at least for *our* MTurk worker population, we could estimate how frequently the MTurk workers select the original validation images.

In this subsection, we describe our sampling strategy that closely matches the selection frequency distribution of the original validation set. The follow-up experiments in Section 4 then explore the impact of this design choice in more detail. As we will see, the sampling strategy has significant influence on the model accuracies.

Matching the Per-class Selection Frequency. A simple approach to matching the selection frequency of the existing validation set would be to sample new images so that the mean selection frequency is the same as for the original dataset. However, a closer inspection of the selection frequencies reveals significant differences between the various classes. For instance, well-defined and well-known classes such as "African elephant" tend to have high selection frequencies ranging from 0.8 to 1.0. At the other end of the spectrum are classes with an unclear definition or easily confused alternative classes. For instance, the MTurk workers in our experiment often confused the class "nail" (the fastener) with fingernails, which led to significantly lower selection frequencies for the original validation images belonging to this class. In order to match these class-level details, we designed a sampling process that approximately matches the selection frequency distribution for each class.

As a first step, we built an estimate of the per-class distribution of selection frequencies. For each class, we divided the annotated validation images into five histogram bins based on their selection frequency. These frequency bins were $[0.0, 0.2)$, $[0.2, 0.4)$, $[0.4, 0.6)$, $[0.6, 0.8)$, and $[0.8, 1.0]$. Intuitively, these bins correspond to a notion of image quality assessed by the MTurk workers, with the $[0.0, 0.2)$ bin containing the worst images and the $[0.8, 1.0]$ bin containing the best images. Normalizing the resulting histograms then yielded a distribution over these selection frequency bins for each class.

Next, we sampled ten images for each class from our candidate pool, following the distribution given by the class-specific selection frequency histograms. More precisely, we first computed the target number of images for each histogram bin, and then sampled from the candidates images falling into this histogram bin uniformly at random. Since we only had a limited number of images for each class, this process ran out of images for a small number of classes. In these cases, we then sampled candidate images from the next higher bin until we found a histogram bin that still had images remaining. While this slightly changes the distribution, we remark that it makes our new test set easier and only affected 0.8% of the images in the new test set.

At the end of this sampling process, we had a test set with $10,000$ images and an average sampling frequency of 0.73. This is close to the average sampling frequency of the annotated validation images (0.71).

Final Reviewing. While the methodology outlined so far closely matches the original ImageNet distribution, it is still hard to ensure that no unintended biases crept into the dataset (e.g., our MTurk workers could interpret some of the class definitions differently and select different images). So for quality control, we added a final reviewing step to our dataset creation pipeline. Its purpose was to rule out obvious biases and ensure that the dataset satisfies our quality expectations *before* we ran any models on the new dataset. This minimizes dependencies between the new test set and the existing models.

In the final reviewing step, the authors of this paper manually reviewed every image in the dataset. Appendix D.4.2 includes a screenshot and brief description of the user interface. When we found an incorrect image or a near-duplicate, we removed it from the dataset. After a pass through the dataset, we then re-sampled new images from our candidate pool. In some cases, this also required new targeted Flickr searches for certain classes. We repeated this process until the dataset converged after 33 iterations. We remark that the majority of iterations only changed a small number of images.

One potential downside of the final reviewing step is that it may lead to a distribution shift. However, we accepted this possibility since we view dataset correctness to be more important than minimizing distribution shift. In the end, a test set is only interesting if it has correct labels. Note also that removing incorrect images from the dataset makes it easier, which goes *against* the main trend we observe (a drop in accuracy). Finally, we kept track of all intermediate iterations of our dataset so that we could measure the impact of this final reviewing step (see Section D.3.2). This analysis shows that the main trends (a significant accuracy drop and an approximately linear relationship between original and new accuracy) also hold for the first iteration of the dataset without any additional reviewing.

D.2. Model Performance Results

After assembling our new test sets, we evaluated a broad range of models on both the original validation set and our new test sets. Section D.4.3 contains a list of all models we evaluated with corresponding references and links to source code repositories. Tables 14 and 15 show the top-1 and top-5 accuracies for our main test set MatchedFrequency. Figure 12 visualizes the top-1 and top-5 accuracies on all three test sets.

In the main text of the paper and Figure 12, we have chosen to exclude the Fisher Vector models and show accuracies only for the convolutional neural networks (convnets). Since the Fisher Vector models achieve significantly lower accuracy, a plot involving both model families would have sacrificed resolution among the convnets. We decided to focus on convnets in the main text because they have become the most widely used model family on ImageNet.

Moreover, a linear model of accuracies (as shown in previous plots) is often not a good fit when the accuracies span a wide range. Instead, a non-linear model such as a logistic or probit model can sometimes describe the data better. Indeed, this is also the case for our data on ImageNet. Figure 6 shows the accuracies both on a linear scale as in the previous plots, and on a *probit* scale, i.e., after applying the inverse of the Gaussian CDF to all accuracy scores. As can be seen by comparing the two plots in Figure 6, the probit model is a better fit for our data. It accurately summarizes the relationship between original and new test set accuracy for all models from both model families in our testbed.

Similar to Figure 12, we also show the top-1 and top-5 accuracies for all three datasets in the probit domain in Figure 13. Section B describes a possible generative model that leads to a linear fit in the probit domain as exhibited by the plots in Figures 6 and 13.

D.3. Follow-up Hypotheses

As for CIFAR-10, the gap between original and new accuracy is concerningly large. Hence we investigated multiple hypotheses for explaining this gap.

D.3.1. Cross Validation

A natural question is whether cross-validation with the existing ImageNet data could have pointed towards a significant drop in accuracy. If adaptive overfitting to the images in the validation set is a cause for the accuracy drop, testing on different images from another cross-validation fold could produce lower accuracies.[12] Moreover, recall that the ImageNet validation set is not a strictly i.i.d. sample from the

[12]Note however that the training images may also be affected by adaptive overfitting since the model hyperparameters are often tuned for fast training speed and high training accuracy.

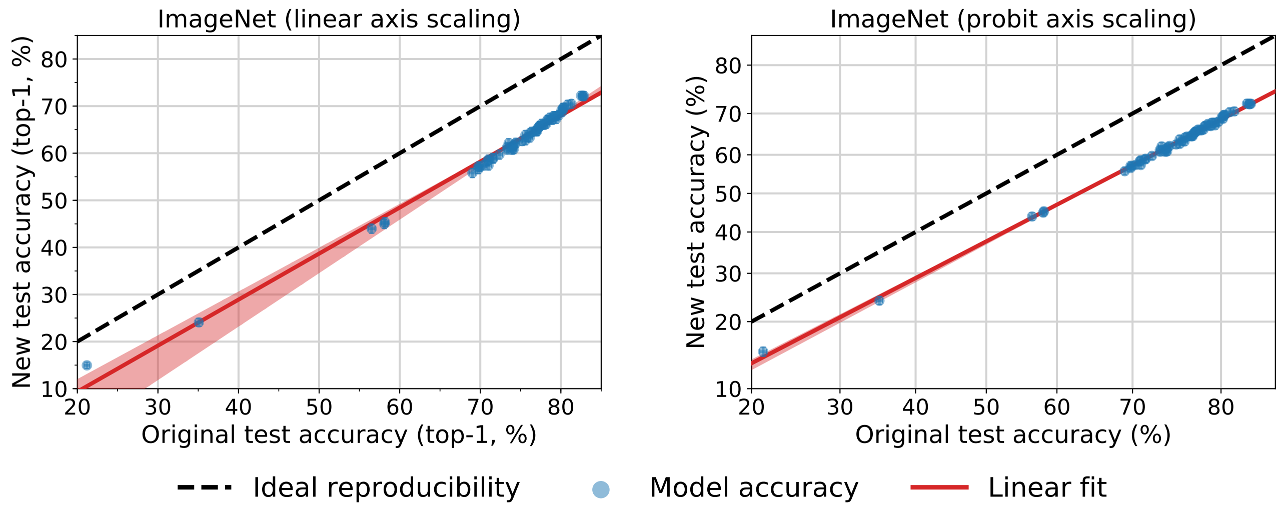


Figure 6. Model accuracy on the original ImageNet validation set vs. our new test set MatchedFrequency. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals), and we now also include the Fisher Vector models. The left plot shows the model accuracies with a linear scale on the axes. The right plot instead uses a *probit* scale, i.e., accuracy α appears at $\Phi^{-1}(\alpha)$, where Φ is the Gaussian CDF. Comparing the two plot provides evidence that the probit model is a better fit for the accuracy scores. Over a range of 60 percentage points, the linear fit in the probit domain accurately describes the relationship between original and new test set accuracy. The shaded region around the linear fit is a 95% confidence region from 100,000 bootstrap samples. The confidence region is present in both plots but is significantly smaller in the right plot due to the better fit in the probit domain.

same distribution as the training set (see the beginning of Section 4). This also raises the question of how well a model would perform on a cross-validation fold from the training data.

To investigate these two effects, we conducted a cross-validation experiment with the ImageNet training and validation sets. In order to ensure that the new cross-validation folds contain only training images, we treated the existing validation set as one fold and created five additional folds with 50,000 images each. To this end, we drew a class-balanced sample of 250,000 images from the training set and then randomly partitioned this sample into five cross-validation folds (again class-balanced). For each of these five folds, we added the validation set (and the other training folds) to the training data so that the size of the training set was unchanged. We then trained one `resnet50` model[13] (He et al., 2016a) for each of the five training sets and evaluated them on the corresponding held-out data. Table 13 shows the resulting accuracies for each split.

Overall, we do not see a large difference in accuracy on the new cross validation splits: all differences fall within the 95% confidence intervals around the accuracy scores. This is in contrast to the significantly larger accuracy drops on our new test sets.

[13] To save computational resources, we used the optimized training code from `https://github.com/fastai/imagenet-fast`. Hence the top-5 accuracy on the original validation set is 0.4% lower than in Table 15.

D.3.2. Impact of Dataset Revisions

As mentioned in Section D.1.2, our final reviewing pass may have led to a distribution shift compared to the original ImageNet validation set. In general, our reviewing criterion was to blacklist images that were

- not representative of the target class,
- cartoons, paintings, drawings, or renderings,
- significantly different in distribution from the original ImageNet validation set,
- unclear, blurry, severely occluded, overly edited, or including only a small target object.

For each class, our reviewing UI (screenshot in Appendix D.4.2) displayed a random sample of ten original validation images directly next to the ten new candidate images currently chosen. At least to some extent, this allowed us to detect and correct distribution shifts between the original validation set and our candidate pool. As a concrete example, we noted in one revision of our dataset that approximately half of the images for “great white shark” were not live sharks in the water but models in museums or statues outside. In contrast, the ImageNet validation set had fewer examples of such artificial sharks. Hence we decided to remove some non-live sharks from our candidate pool and sampled new shark images as a replacement in the dataset.

Unfortunately, some of these reviewing choices are subjective. However, such choices are often an inherent part of creating a dataset and it is unclear whether a more “hands-

Dataset	resnet50 Top-5 Accuracy (%)
Original validation set	92.5 [92.3, 92.8]
Split 1	92.60 [92.4, 92.8]
Split 2	92.59 [92.4, 92.8]
Split 3	92.61 [92.4, 92.8]
Split 4	92.55 [92.3, 92.8]
Split 5	92.62 [92.4, 92.9]
New test set (MatchedFrequency)	84.7 [83.9, 85.4]

Table 13. `resnet50` accuracy on cross-validation splits created from the original ImageNet train and validation sets. The accuracy increase is likely caused by a small shift in distribution between the ImageNet training and validation sets.

off" approach would lead to more meaningful conclusions. For instance, if the drop in accuracy was mainly caused by a distribution shift that is easy to identify and correct (e.g., an increase in black-and-white images), the resulting drop may not be an interesting phenomenon (beyond counting black-and-white images). Hence we decided to *both* remove distribution shifts that we found easy to identify visually, and also to measure the effect of these interventions.

Our reviewing process was iterative, i.e., we made a full pass over every incomplete class in a given dataset revision before sampling new images to fill the resulting gaps. Each time we re-sampled our dataset, we saved the current list of images in our version control system. This allowed us to track the datasets over time and later measure the model accuracy for each dataset revision. We remark that we only computed model accuracies on intermediate revisions after we had arrived at the final revision of the corresponding dataset.

Figure 7 plots the top-1 accuracy of a `resnet50` model versus the dataset revision for our new MatchedFrequency test set. Overall, reviewing improved model accuracy by about 4% for this dataset. This is evidence that our manual reviewing did not cause the drop in accuracy between the original and new dataset.

In addition, we also investigated whether the linear relationship between original and new test accuracy was affected by our final reviewing passes. To this end, we evaluated our model testbed on the first revision of our MatchedFrequency test set. As can be seen in Figure 8, the resulting accuracies still show a good linear fit that is of similar quality as in Figure 12. This shows that the linear relationship between the test accuracies is not a result of our reviewing intervention.

D.4. Additional Figures, Tables, and Lists

In this appendix we provide large figures etc. that did not fit into the preceding sections about our ImageNet experiments.

D.4.1. MTurk User Interfaces

For comparison, we include the original ImageNet MTurk user interface (UI) in Figure 9 and the MTurk UI we used in our experiments in Figure 10. Each UI corresponds to one task for the MTurk workers, which consists of 48 images in both cases. In contrast to the original ImageNet UI, our UI takes up more than one screen. This requires the MTurk workers to scroll but also provides more details in the images. While the task descriptions are not exactly the same, they are very similar and contain the same directions (e.g., both descriptions ask the MTurk workers to exclude drawings or paintings).

D.4.2. User Interface for our Final Reviewing Process

Figure 11 shows a screenshot of the reviewing UI that the paper authors used to manually review the new ImageNet datasets. At the top, the UI displays the wnid ("n01667114"), the synset (**mud turtle**), and the gloss. Next, a grid of 20 images is shown in 4 rows.

The top two rows correspond to the candidate images currently sampled for the new dataset. Below each image, our UI shows a unique identifier for the image and the date the image was taken. There is also a check box to blacklist any incorrect images. In addition, there is a check box for each image to add it to the blacklist of incorrect images. If an image is added to the blacklist, it will be removed in the next revision of the dataset and replaced with a new image from the candidate pools. The candidate images are sorted by the date they were taken, which makes it easier to spot and remove near-duplicates. Images are marked as near-duplicates by adding their identifier to the "Near-duplicate set" text field.

The bottom two rows correspond to a random sample of images from the validation set that belong to the target class. We display these images to make it easier to detect and correct for distribution shifts between our new test sets and the original ImageNet validation dataset.

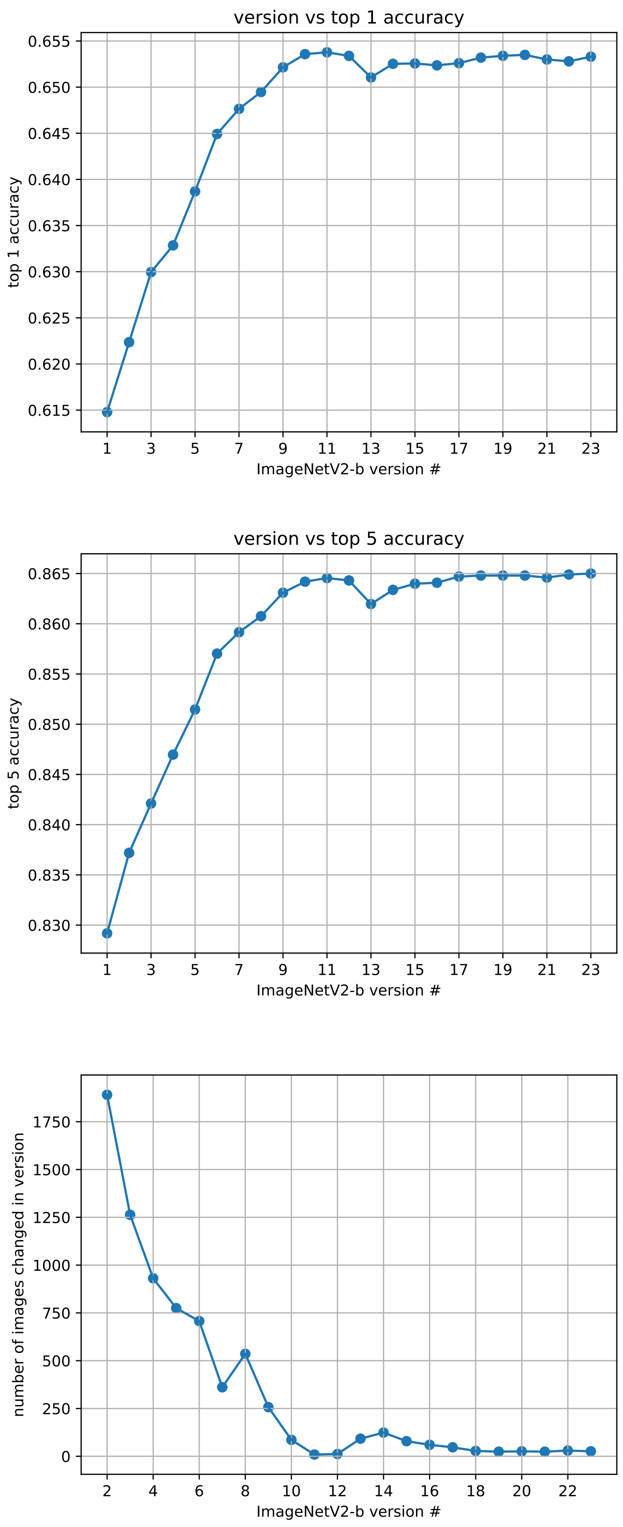


Figure 7. Impact of the reviewing passes on the accuracy of a `resnet152` on our new MatchedFrequency test set. The revision numbers correspond to the chronological ordering in which we created the dataset revisions

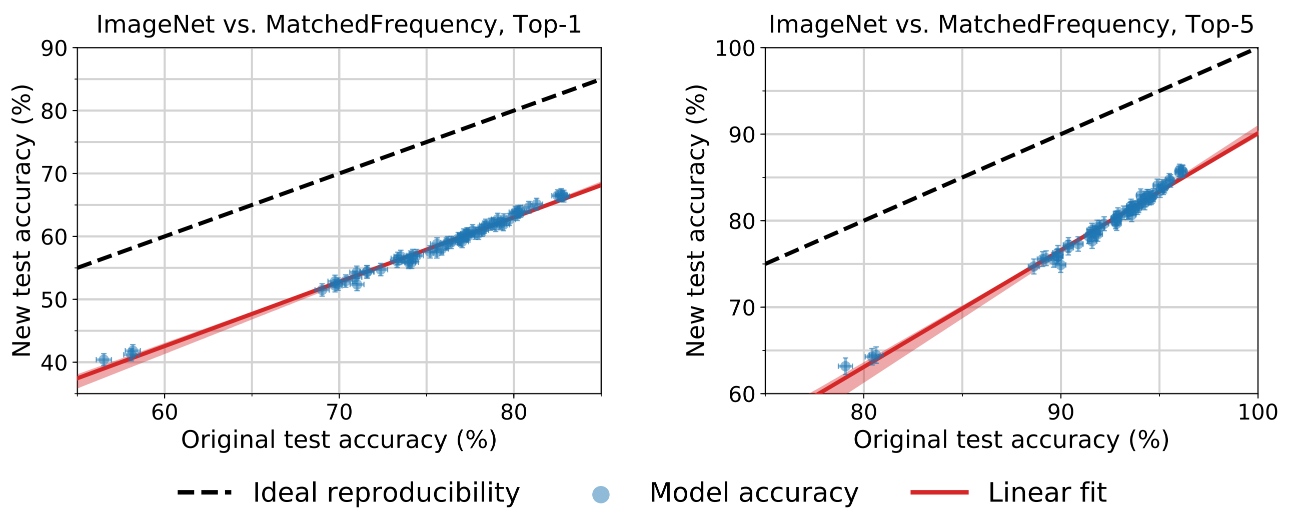
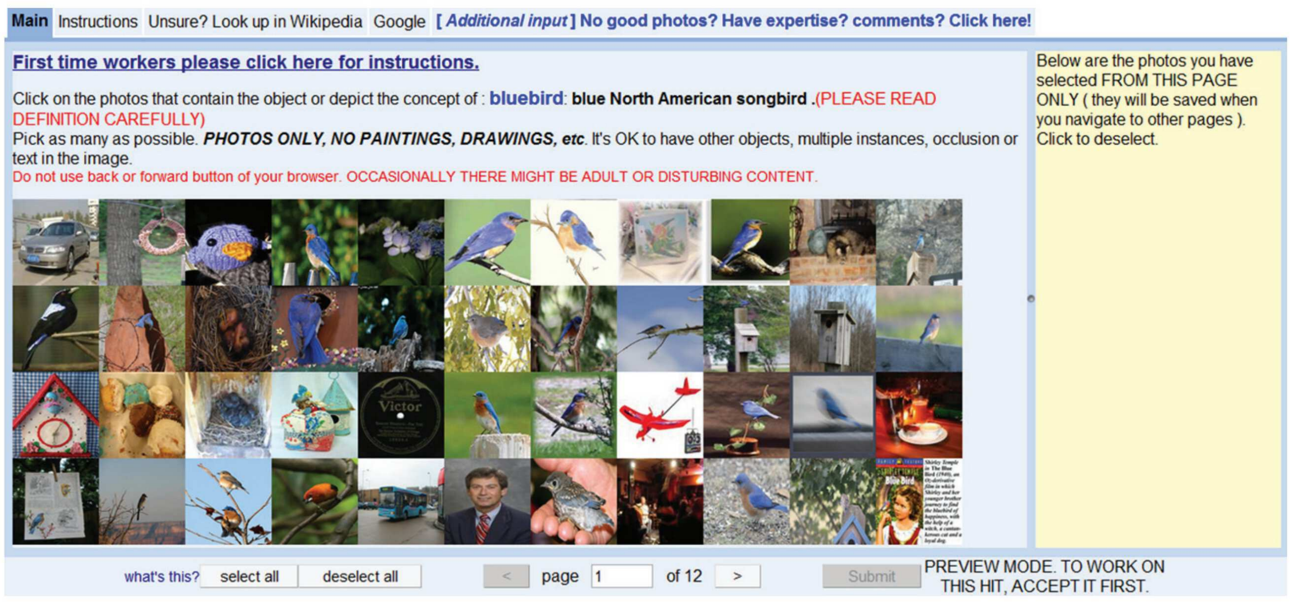


Figure 8. Model accuracy on the original ImageNet validation set vs. accuracy on *the first revision* of our MatchedFrequency test set. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). The red shaded region is a 95% confidence region for the linear fit from 100,000 bootstrap samples. The plots show that the linear relationship between original and new test accuracy also occurs without our final dataset reviewing step. The accuracy plots for the final revision of MatchedFrequency can be found in Figure 12.


Main
Instructions
Unsure? Look up in Wikipedia
Google
[Additional input] No good photos? Have expertise? comments? Click here!
what's this?
select all
deselect all
page
of 12
Submit

First time workers please click here for instructions.

Click on the photos that contain the object or depict the concept of : bluebird: blue North American songbird .(PLEASE READ DEFINITION CAREFULLY)
Pick as many as possible. ***PHOTOS ONLY, NO PAINTINGS, DRAWINGS, etc.*** It's OK to have other objects, multiple instances, occlusion or text in the image.
Do not use back or forward button of your browser. OCCASIONALLY THERE MIGHT BE ADULT OR DISTURBING CONTENT.

Below are the photos you have selected FROM THIS PAGE ONLY (they will be saved when you navigate to other pages). Click to deselect.

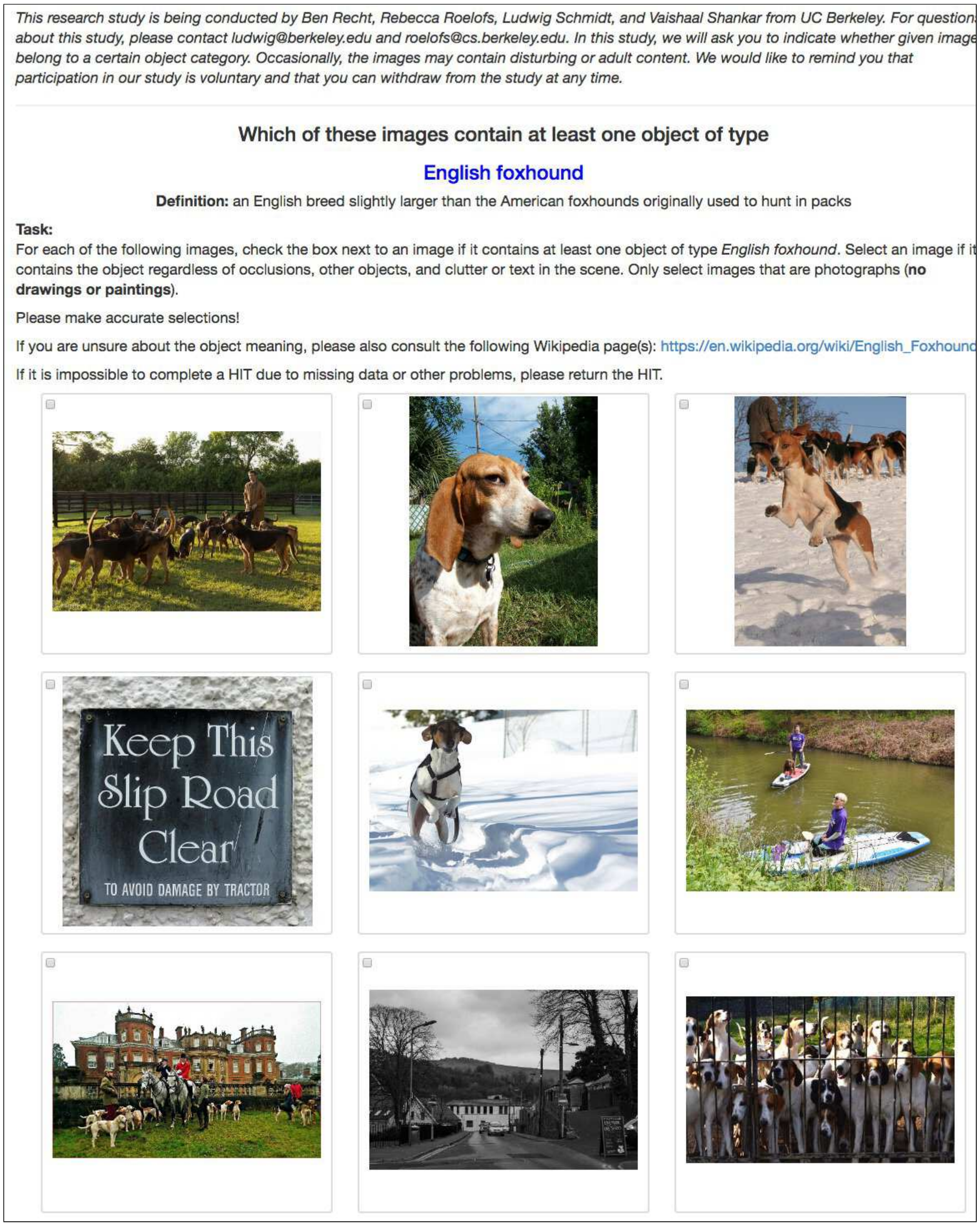
PREVIEW MODE. TO WORK ON THIS HIT, ACCEPT IT FIRST.

Figure 9. The user interface employed in the original ImageNet collection process for the labeling tasks on Amazon Mechanical Turk.

D.4.3. Full List of Models Evaluated on ImageNet

The following list contains all models we evaluated on ImageNet with references and links to the corresponding source code.

1. `alexnet` (Krizhevsky et al., 2012) `https://github.com/Cadene/pretrained-models.pytorch`
2. `bninception` (Ioffe and Szegedy, 2015) `https://github.com/Cadene/pretrained-models.pytorch`
3. `cafferesnet101` (He et al., 2016a) `https://github.com/Cadene/pretrained-models.pytorch`
4. `densenet121` (Huang et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`



This research study is being conducted by Ben Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar from UC Berkeley. For question about this study, please contact ludwig@berkeley.edu and roelofs@cs.berkeley.edu. In this study, we will ask you to indicate whether given image belong to a certain object category. Occasionally, the images may contain disturbing or adult content. We would like to remind you that participation in our study is voluntary and that you can withdraw from the study at any time.

Which of these images contain at least one object of type

English foxhound

Definition: an English breed slightly larger than the American foxhounds originally used to hunt in packs

Task:
For each of the following images, check the box next to an image if it contains at least one object of type *English foxhound*. Select an image if it contains the object regardless of occlusions, other objects, and clutter or text in the scene. Only select images that are photographs (**no drawings or paintings**).

Please make accurate selections!

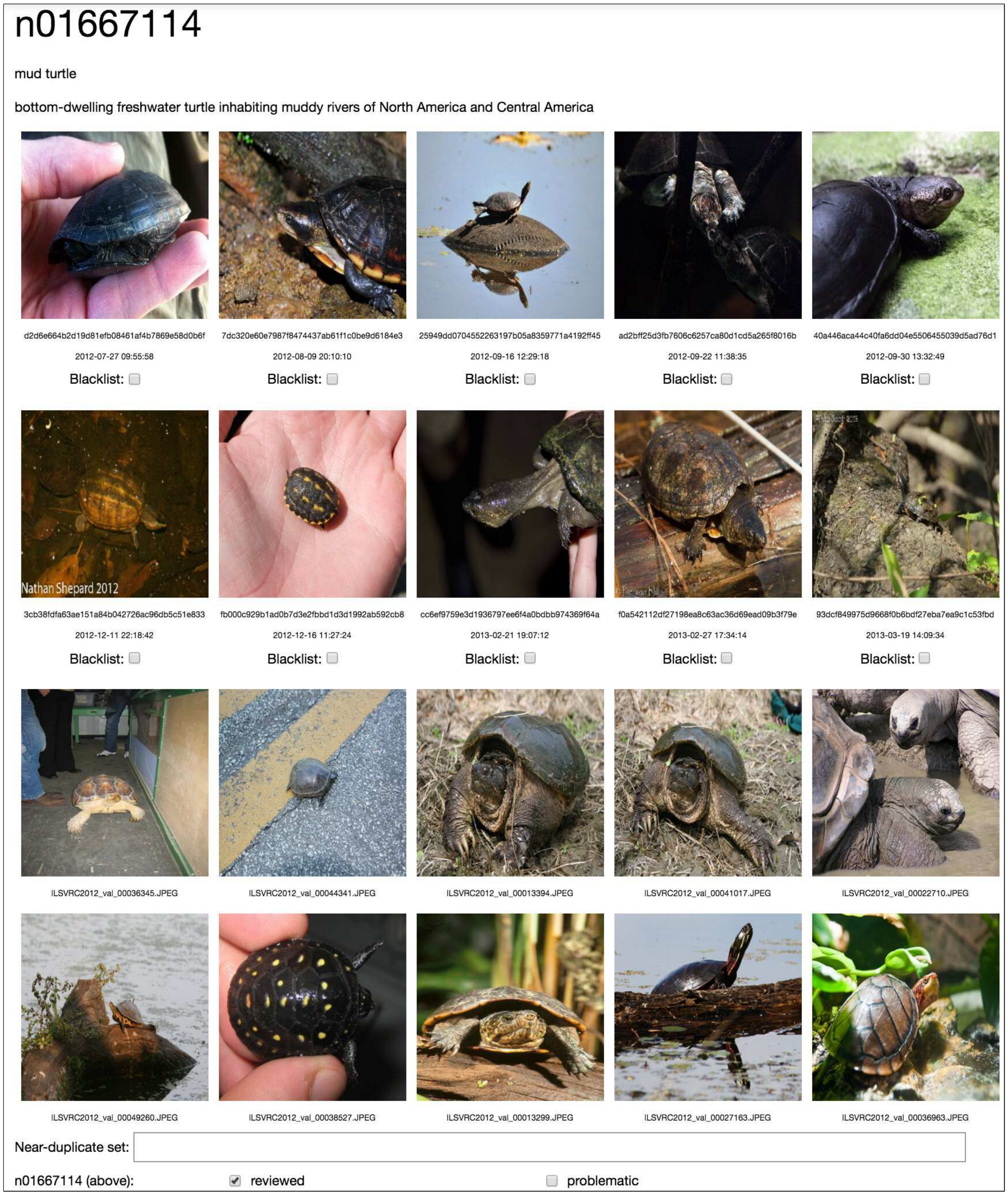
If you are unsure about the object meaning, please also consult the following Wikipedia page(s): https://en.wikipedia.org/wiki/English_Foxhound

If it is impossible to complete a HIT due to missing data or other problems, please return the HIT.

Figure 10. Our user interface for labeling tasks on Amazon Mechanical Turk. The screenshot here omits the scroll bar and shows only a subset of the entire MTurk task. As in the ImageNet UI, the full task involves a grid of 48 images.

5. `densenet161` (Huang et al., 2017)`https://github.com/Cadene/pretrained-models.pytorch`

6. `densenet169` (Huang et al., 2017) `https://github.com/Cadene/pretrained-models.pytorch`



n01667114

mud turtle

bottom-dwelling freshwater turtle inhabiting muddy rivers of North America and Central America

d2d6e664b2d19d81efb08461af4b7869e58d0b6f	7dc320e60e7987f8474437ab61f1c0be9d6184e3	25949dd0704552263197b05a8359771a4192ff45	ad2bff25d3fb7606c6257ca80d1cd5a265f8016b	40a446aca44c40fa6dd04e5506455039d5ad76d1
2012-07-27 09:55:58	2012-08-09 20:10:10	2012-09-16 12:29:18	2012-09-22 11:38:35	2012-09-30 13:32:49
Blacklist: ☐	Blacklist: ☐	Blacklist: ☐	Blacklist: ☐	Blacklist: ☐
3cb38fdfa63ae151a84b042726ac96db5c51e833	fb000c929b1ad0b7d3e2fbbd1d3d1992ab592cb8	cc6ef9759e3d1936797ee6f4a0bdbb974369f64a	f0a542112df27198ea8c63ac36d69ead09b3f79e	93dcf849975d9668f0b6bdf27eba7ea9c1c53fbd
2012-12-11 22:18:42	2012-12-16 11:27:24	2013-02-21 19:07:12	2013-02-27 17:34:14	2013-03-19 14:09:34
Blacklist: ☐	Blacklist: ☐	Blacklist: ☐	Blacklist: ☐	Blacklist: ☐
ILSVRC2012_val_00036345.JPEG	ILSVRC2012_val_00044341.JPEG	ILSVRC2012_val_00013394.JPEG	ILSVRC2012_val_00041017.JPEG	ILSVRC2012_val_00022710.JPEG
ILSVRC2012_val_00049260.JPEG	ILSVRC2012_val_00038527.JPEG	ILSVRC2012_val_00013299.JPEG	ILSVRC2012_val_00027163.JPEG	ILSVRC2012_val_00036963.JPEG

Near-duplicate set:

n01667114 (above): ✓ reviewed ☐ problematic

Figure 11. The user interface we built to review dataset revisions and remove incorrect or near duplicate images. This user interface was not used for MTurk but only in the final dataset review step conducted by the authors of this paper.

7. densenet201 (Huang et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
8. dpn107 (Chen et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
9. dpn131 (Chen et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
10. dpn68b (Chen et al., 2017)https://github.com/Cadene/pretrained-models.pytorch
11. dpn68 (Chen et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
12. dpn92 (Chen et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
13. dpn98 (Chen et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
14. fbresnet152 (He et al., 2016a) https://github.com/tensorflow/models/tree/master/research/slim/
15. fv_4k (Clinchant et al., 2007; Perronnin et al., 2010) https://github.com/modestyachts/nondeep FisherVector model using SIFT, local color statistic features, and 16 GMM centers.
16. fv_16k (Clinchant et al., 2007; Perronnin et al., 2010) https://github.com/modestyachts/nondeep FisherVector model using SIFT, local color statistic features, and 64 GMM centers.
17. fv_64k (Clinchant et al., 2007; Perronnin et al., 2010) https://github.com/modestyachts/nondeep FisherVector model using SIFT, local color statistic features, and 256 GMM centers.
18. inception_resnet_v2_tf (Szegedy et al., 2017) https://github.com/tensorflow/models/tree/master/research/slim/
19. inception_v1_tf (Szegedy et al., 2015) https://github.com/tensorflow/models/tree/master/research/slim/
20. inception_v2_tf (Ioffe and Szegedy, 2015) https://github.com/tensorflow/models/tree/master/research/slim/
21. inception_v3_tf (Szegedy et al., 2016) https://github.com/tensorflow/models/tree/master/research/slim/
22. inception_v3 (Szegedy et al., 2016) https://github.com/Cadene/pretrained-models.pytorch
23. inception_v4_tf (Szegedy et al., 2017) https://github.com/tensorflow/models/tree/master/research/slim/
24. inceptionresnetv2 (Ioffe and Szegedy, 2015) https://github.com/Cadene/pretrained-models.pytorch
25. inceptionv3 (Szegedy et al., 2016) https://github.com/Cadene/pretrained-models.pytorch
26. inceptionv4 (Szegedy et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
27. mobilenet_v1_tf (Howard et al., 2017) https://github.com/tensorflow/models/tree/master/research/slim/
28. nasnet_large_tf (Zoph et al., 2018) https://github.com/tensorflow/models/tree/master/research/slim/
29. nasnet_mobile_tf (Zoph et al., 2018) https://github.com/tensorflow/models/tree/master/research/slim/
30. nasnetalarge (Zoph et al., 2018) https://github.com/Cadene/pretrained-models.pytorch
31. nasnetamobile (Zoph et al., 2018) https://github.com/Cadene/pretrained-models.pytorch
32. pnasnet5large (Liu et al., 2018) https://github.com/Cadene/pretrained-models.pytorch
33. pnasnet_large_tf (Liu et al., 2018) https://github.com/tensorflow/models/tree/master/research/slim/
34. pnasnet_mobile_tf (Liu et al., 2018) https://github.com/tensorflow/models/tree/master/research/slim/
35. polynet (Zhang et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
36. resnet101 (He et al., 2016a) https://github.com/Cadene/pretrained-models.pytorch
37. resnet152 (He et al., 2016a) https://github.com/Cadene/pretrained-models.pytorch
38. resnet18 (He et al., 2016a) https://github.com/Cadene/pretrained-models.pytorch
39. resnet34 (He et al., 2016a) https://github.com/Cadene/pretrained-models.pytorch
40. resnet50 (He et al., 2016a) https://github.com/Cadene/pretrained-models.pytorch

41. resnet_v1_101_tf (He et al., 2016a) https://github.com/tensorflow/models/tree/master/research/slim/
42. resnet_v1_152_tf (He et al., 2016a) https://github.com/tensorflow/models/tree/master/research/slim/
43. resnet_v1_50_tf (He et al., 2016a) https://github.com/tensorflow/models/tree/master/research/slim/
44. resnet_v2_101_tf (He et al., 2016b) https://github.com/tensorflow/models/tree/master/research/slim/
45. resnet_v2_152_tf (He et al., 2016b) https://github.com/tensorflow/models/tree/master/research/slim/
46. resnet_v2_50_tf (He et al., 2016b) https://github.com/tensorflow/models/tree/master/research/slim/
47. resnext101_32x4d (Xie et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
48. resnext101_64x4d (Xie et al., 2017) https://github.com/Cadene/pretrained-models.pytorch
49. se_resnet101 (Hu et al., 2018) https://github.com/Cadene/pretrained-models.pytorch
50. se_resnet152 (Hu et al., 2018) https://github.com/Cadene/pretrained-models.pytorch
51. se_resnet50 (Hu et al., 2018) https://github.com/Cadene/pretrained-models.pytorch
52. se_resnext101_32x4d (Hu et al., 2018) https://github.com/Cadene/pretrained-models.pytorch
53. se_resnext50_32x4d (Hu et al., 2018) https://github.com/Cadene/pretrained-models.pytorch
54. senet154 (Hu et al., 2018) https://github.com/Cadene/pretrained-models.pytorch
55. squeezenet1_0 (Iandola et al., 2016) https://github.com/Cadene/pretrained-models.pytorch
56. squeezenet1_1 (Iandola et al., 2016) https://github.com/Cadene/pretrained-models.pytorch
57. vgg11_bn (Ioffe and Szegedy, 2015) https://github.com/Cadene/pretrained-models.pytorch
58. vgg11 (Simonyan and Zisserman, 2014) https://github.com/Cadene/pretrained-models.pytorch
59. vgg13_bn (Ioffe and Szegedy, 2015) https://github.com/Cadene/pretrained-models.pytorch
60. vgg13 (Simonyan and Zisserman, 2014) https://github.com/Cadene/pretrained-models.pytorch
61. vgg16_bn (Ioffe and Szegedy, 2015) https://github.com/Cadene/pretrained-models.pytorch
62. vgg16 (Simonyan and Zisserman, 2014) https://github.com/Cadene/pretrained-models.pytorch
63. vgg19_bn (Ioffe and Szegedy, 2015) https://github.com/Cadene/pretrained-models.pytorch
64. vgg19 (Simonyan and Zisserman, 2014) https://github.com/Cadene/pretrained-models.pytorch
65. vgg_16_tf (Simonyan and Zisserman, 2014) https://github.com/tensorflow/models/tree/master/research/slim/
66. vgg_19_tf (Simonyan and Zisserman, 2014) https://github.com/tensorflow/models/tree/master/research/slim/
67. xception (Chollet, 2017) https://github.com/Cadene/pretrained-models.pytorch

D.4.4. Full Results Tables

Tables 14 and 15 contain the detailed accuracy scores (top-1 and top-5, respectively) for the original ImageNet validation set and our main new test set MatchedFrequency. Tables 16 – 19 contain the accuracy scores for our Threshold0.7 and TopImages test sets.

Table 14. Top-1 model accuracy on the original ImageNet validation set and our new test set MatchedFrequency. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, $\Delta\text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. Due to space constraints, references for the models can be found in Appendix D.4.3. The second part of the table can be found on the following page.

ImageNet Top-1 MatchedFrequency

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	82.9 [82.5, 83.2]	72.2 [71.3, 73.1]	10.7	3	-2
2	pnasnet5large	82.7 [82.4, 83.1]	72.1 [71.2, 73.0]	10.7	4	-2
3	nasnet_large_tf	82.7 [82.4, 83.0]	72.2 [71.3, 73.1]	10.5	2	1
4	nasnetalarge	82.5 [82.2, 82.8]	72.2 [71.3, 73.1]	10.3	1	3
5	senet154	81.3 [81.0, 81.6]	70.5 [69.6, 71.4]	10.8	5	0
6	polynet	80.9 [80.5, 81.2]	70.3 [69.4, 71.2]	10.5	6	0
7	inception_resnet_v2_tf	80.4 [80.0, 80.7]	69.7 [68.7, 70.6]	10.7	7	0
8	inceptionresnetv2	80.3 [79.9, 80.6]	69.6 [68.7, 70.5]	10.6	8	0
9	se_resnext101_32x4d	80.2 [79.9, 80.6]	69.3 [68.4, 70.2]	10.9	9	0
10	inception_v4_tf	80.2 [79.8, 80.5]	68.8 [67.9, 69.7]	11.4	11	-1
11	inceptionv4	80.1 [79.7, 80.4]	69.1 [68.2, 70.0]	10.9	10	1
12	dpn107	79.7 [79.4, 80.1]	68.1 [67.2, 69.0]	11.7	12	0
13	dpn131	79.4 [79.1, 79.8]	67.9 [67.0, 68.8]	11.5	13	0
14	dpn92	79.4 [79.0, 79.8]	67.3 [66.3, 68.2]	12.1	17	-3
15	dpn98	79.2 [78.9, 79.6]	67.8 [66.9, 68.8]	11.4	15	0
16	se_resnext50_32x4d	79.1 [78.7, 79.4]	67.9 [66.9, 68.8]	11.2	14	2
17	resnext101_64x4d	79.0 [78.6, 79.3]	67.1 [66.2, 68.0]	11.9	20	-3
18	xception	78.8 [78.5, 79.2]	67.2 [66.2, 68.1]	11.7	18	0
19	se_resnet152	78.7 [78.3, 79.0]	67.5 [66.6, 68.5]	11.1	16	3
20	se_resnet101	78.4 [78.0, 78.8]	67.2 [66.2, 68.1]	11.2	19	1
21	resnet152	78.3 [77.9, 78.7]	67.0 [66.1, 67.9]	11.3	21	0
22	resnext101_32x4d	78.2 [77.8, 78.5]	66.2 [65.3, 67.2]	11.9	22	0
23	inception_v3_tf	78.0 [77.6, 78.3]	66.1 [65.1, 67.0]	11.9	24	-1
24	resnet_v2_152_tf	77.8 [77.4, 78.1]	66.1 [65.1, 67.0]	11.7	25	-1
25	se_resnet50	77.6 [77.3, 78.0]	66.2 [65.3, 67.2]	11.4	23	2
26	fbresnet152	77.4 [77.0, 77.8]	65.8 [64.9, 66.7]	11.6	26	0
27	resnet101	77.4 [77.0, 77.7]	65.7 [64.7, 66.6]	11.7	28	-1
28	inceptionv3	77.3 [77.0, 77.7]	65.7 [64.8, 66.7]	11.6	27	1
29	inception_v3	77.2 [76.8, 77.6]	65.4 [64.5, 66.4]	11.8	29	0
30	densenet161	77.1 [76.8, 77.5]	65.3 [64.4, 66.2]	11.8	30	0
31	dpn68b	77.0 [76.7, 77.4]	64.7 [63.7, 65.6]	12.4	32	-1
32	resnet_v2_101_tf	77.0 [76.6, 77.3]	64.6 [63.7, 65.6]	12.3	34	-2
33	densenet201	76.9 [76.5, 77.3]	64.7 [63.7, 65.6]	12.2	31	2

ImageNet Top-1 MatchedFrequency						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	76.8 [76.4, 77.2]	64.6 [63.7, 65.6]	12.2	33	1
35	resnet_v1_101_tf	76.4 [76.0, 76.8]	64.5 [63.6, 65.5]	11.9	35	0
36	cafferesnet101	76.2 [75.8, 76.6]	64.3 [63.4, 65.2]	11.9	36	0
37	resnet50	76.1 [75.8, 76.5]	63.3 [62.4, 64.3]	12.8	39	-2
38	dpn68	75.9 [75.5, 76.2]	63.4 [62.5, 64.4]	12.4	38	0
39	densenet169	75.6 [75.2, 76.0]	63.9 [62.9, 64.8]	11.7	37	2
40	resnet_v2_50_tf	75.6 [75.2, 76.0]	62.7 [61.8, 63.7]	12.9	40	0
41	resnet_v1_50_tf	75.2 [74.8, 75.6]	62.6 [61.6, 63.5]	12.6	41	0
42	densenet121	74.4 [74.0, 74.8]	62.2 [61.3, 63.2]	12.2	42	0
43	vgg19_bn	74.2 [73.8, 74.6]	61.9 [60.9, 62.8]	12.3	44	-1
44	pnasnet_mobile_tf	74.1 [73.8, 74.5]	60.9 [59.9, 61.8]	13.3	48	-4
45	nasnetamobile	74.1 [73.7, 74.5]	61.6 [60.6, 62.5]	12.5	45	0
46	inception_v2_tf	74.0 [73.6, 74.4]	61.2 [60.2, 62.2]	12.8	46	0
47	nasnet_mobile_tf	74.0 [73.6, 74.4]	60.8 [59.8, 61.7]	13.2	50	-3
48	bninception	73.5 [73.1, 73.9]	62.1 [61.2, 63.1]	11.4	43	5
49	vgg16_bn	73.4 [73.0, 73.7]	60.8 [59.8, 61.7]	12.6	49	0
50	resnet34	73.3 [72.9, 73.7]	61.2 [60.2, 62.2]	12.1	47	3
51	vgg19	72.4 [72.0, 72.8]	59.7 [58.7, 60.7]	12.7	51	0
52	vgg16	71.6 [71.2, 72.0]	58.8 [57.9, 59.8]	12.8	53	-1
53	vgg13_bn	71.6 [71.2, 72.0]	59.0 [58.0, 59.9]	12.6	52	1
54	mobilenet_v1_tf	71.0 [70.6, 71.4]	57.4 [56.4, 58.4]	13.6	56	-2
55	vgg_19_tf	71.0 [70.6, 71.4]	58.6 [57.7, 59.6]	12.4	54	1
56	vgg_16_tf	70.9 [70.5, 71.3]	58.4 [57.4, 59.3]	12.5	55	1
57	vgg11_bn	70.4 [70.0, 70.8]	57.4 [56.4, 58.4]	13.0	57	0
58	vgg13	69.9 [69.5, 70.3]	57.1 [56.2, 58.1]	12.8	59	-1
59	inception_v1_tf	69.8 [69.4, 70.2]	56.6 [55.7, 57.6]	13.1	60	-1
60	resnet18	69.8 [69.4, 70.2]	57.2 [56.2, 58.2]	12.6	58	2
61	vgg11	69.0 [68.6, 69.4]	55.8 [54.8, 56.8]	13.2	61	0
62	squeezenet1_1	58.2 [57.7, 58.6]	45.3 [44.4, 46.3]	12.8	62	0
63	squeezenet1_0	58.1 [57.7, 58.5]	45.0 [44.0, 46.0]	13.1	63	0
64	alexnet	56.5 [56.1, 57.0]	44.0 [43.0, 45.0]	12.5	64	0
65	fv_64k	35.1 [34.7, 35.5]	24.1 [23.2, 24.9]	11.0	65	0
66	fv_16k	28.3 [27.9, 28.7]	19.2 [18.5, 20.0]	9.1	66	0
67	fv_4k	21.2 [20.8, 21.5]	15.0 [14.3, 15.7]	6.2	67	0

Table 15. Top-5 model accuracy on the original ImageNet validation set and our new test set MatchedFrequency. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, $\Delta\text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. Due to space constraints, references for the models can be found in Appendix D.4.3. The second part of the table can be found on the following page.

ImageNet Top-5 MatchedFrequency

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	`pnasnet_large_tf`	96.2 [96.0, 96.3]	90.1 [89.5, 90.7]	6.1	3	-2
2	`nasnet_large_tf`	96.2 [96.0, 96.3]	90.1 [89.5, 90.6]	6.1	4	-2
3	`nasnetalarge`	96.0 [95.8, 96.2]	90.4 [89.8, 91.0]	5.6	1	2
4	`pnasnet5large`	96.0 [95.8, 96.2]	90.2 [89.6, 90.8]	5.8	2	2
5	`polynet`	95.6 [95.4, 95.7]	89.1 [88.5, 89.7]	6.4	5	0
6	`senet154`	95.5 [95.3, 95.7]	89.0 [88.4, 89.6]	6.5	6	0
7	`inception_resnet_v2_tf`	95.2 [95.1, 95.4]	88.4 [87.7, 89.0]	6.9	9	-2
8	`inception_v4_tf`	95.2 [95.0, 95.4]	88.3 [87.6, 88.9]	6.9	10	-2
9	`inceptionresnetv2`	95.1 [94.9, 95.3]	88.5 [87.8, 89.1]	6.7	8	1
10	`se_resnext101_32x4d`	95.0 [94.8, 95.2]	88.0 [87.4, 88.7]	7.0	11	-1
11	`inceptionv4`	94.9 [94.7, 95.1]	88.7 [88.1, 89.3]	6.2	7	4
12	`dpn107`	94.7 [94.5, 94.9]	87.6 [86.9, 88.2]	7.1	13	-1
13	`dpn92`	94.6 [94.4, 94.8]	87.2 [86.5, 87.8]	7.5	17	-4
14	`dpn131`	94.6 [94.4, 94.8]	87.0 [86.3, 87.7]	7.6	19	-5
15	`dpn98`	94.5 [94.3, 94.7]	87.2 [86.5, 87.8]	7.3	16	-1
16	`se_resnext50_32x4d`	94.4 [94.2, 94.6]	87.6 [87.0, 88.3]	6.8	12	4
17	`se_resnet152`	94.4 [94.2, 94.6]	87.4 [86.7, 88.0]	7.0	15	2
18	`xception`	94.3 [94.1, 94.5]	87.0 [86.3, 87.7]	7.3	20	-2
19	`se_resnet101`	94.3 [94.1, 94.5]	87.1 [86.4, 87.7]	7.2	18	1
20	`resnext101_64x4d`	94.3 [94.0, 94.5]	86.9 [86.2, 87.5]	7.4	22	-2
21	`resnet_v2_152_tf`	94.1 [93.9, 94.3]	86.9 [86.2, 87.5]	7.2	21	0
22	`resnet152`	94.0 [93.8, 94.3]	87.6 [86.9, 88.2]	6.5	14	8
23	`inception_v3_tf`	93.9 [93.7, 94.1]	86.4 [85.7, 87.0]	7.6	23	0
24	`resnext101_32x4d`	93.9 [93.7, 94.1]	86.2 [85.5, 86.8]	7.7	25	-1
25	`se_resnet50`	93.8 [93.5, 94.0]	86.3 [85.6, 87.0]	7.4	24	1
26	`resnet_v2_101_tf`	93.7 [93.5, 93.9]	86.1 [85.4, 86.8]	7.6	27	-1
27	`fbresnet152`	93.6 [93.4, 93.8]	86.1 [85.4, 86.7]	7.5	28	-1
28	`dpn68b`	93.6 [93.4, 93.8]	85.3 [84.6, 86.0]	8.3	33	-5
29	`densenet161`	93.6 [93.3, 93.8]	86.1 [85.4, 86.8]	7.4	26	3
30	`resnet101`	93.5 [93.3, 93.8]	86.0 [85.3, 86.7]	7.6	30	0
31	`inception_v3`	93.5 [93.3, 93.7]	85.9 [85.2, 86.6]	7.6	31	0
32	`inceptionv3`	93.4 [93.2, 93.6]	86.1 [85.4, 86.7]	7.4	29	3
33	`densenet201`	93.4 [93.1, 93.6]	85.3 [84.6, 86.0]	8.1	34	-1

ImageNet Top-5 MatchedFrequency						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	93.2 [92.9, 93.4]	85.4 [84.6, 86.0]	7.8	32	2
35	resnet_v1_101_tf	92.9 [92.7, 93.1]	85.2 [84.5, 85.9]	7.7	35	0
36	resnet50	92.9 [92.6, 93.1]	84.7 [83.9, 85.4]	8.2	38	-2
37	resnet_v2_50_tf	92.8 [92.6, 93.1]	84.4 [83.6, 85.1]	8.5	40	-3
38	densenet169	92.8 [92.6, 93.0]	84.7 [84.0, 85.4]	8.1	37	1
39	dpn68	92.8 [92.5, 93.0]	84.6 [83.9, 85.3]	8.2	39	0
40	cafferesnet101	92.8 [92.5, 93.0]	84.9 [84.1, 85.6]	7.9	36	4
41	resnet_v1_50_tf	92.2 [92.0, 92.4]	84.1 [83.4, 84.8]	8.1	41	0
42	densenet121	92.0 [91.7, 92.2]	83.8 [83.1, 84.5]	8.2	42	0
43	pnasnet_mobile_tf	91.9 [91.6, 92.1]	83.1 [82.4, 83.8]	8.8	46	-3
44	vgg19_bn	91.8 [91.6, 92.1]	83.5 [82.7, 84.2]	8.4	43	1
45	inception_v2_tf	91.8 [91.5, 92.0]	83.1 [82.3, 83.8]	8.7	47	-2
46	nasnetamobile	91.7 [91.5, 92.0]	83.4 [82.6, 84.1]	8.4	45	1
47	nasnet_mobile_tf	91.6 [91.3, 91.8]	82.2 [81.4, 82.9]	9.4	50	-3
48	bninception	91.6 [91.3, 91.8]	83.4 [82.7, 84.2]	8.1	44	4
49	vgg16_bn	91.5 [91.3, 91.8]	83.0 [82.2, 83.7]	8.6	48	1
50	resnet34	91.4 [91.2, 91.7]	82.7 [82.0, 83.5]	8.7	49	1
51	vgg19	90.9 [90.6, 91.1]	81.5 [80.7, 82.2]	9.4	52	-1
52	vgg16	90.4 [90.1, 90.6]	81.7 [80.9, 82.4]	8.7	51	1
53	vgg13_bn	90.4 [90.1, 90.6]	81.1 [80.3, 81.9]	9.3	53	0
54	mobilenet_v1_tf	90.0 [89.7, 90.2]	79.4 [78.6, 80.1]	10.6	60	-6
56	vgg_19_tf	89.8 [89.6, 90.1]	80.7 [79.9, 81.4]	9.2	54	2
55	vgg_16_tf	89.8 [89.6, 90.1]	80.5 [79.7, 81.3]	9.3	55	0
57	vgg11_bn	89.8 [89.5, 90.1]	80.0 [79.2, 80.8]	9.8	58	-1
58	inception_v1_tf	89.6 [89.4, 89.9]	80.1 [79.3, 80.9]	9.5	57	1
59	vgg13	89.2 [89.0, 89.5]	79.5 [78.7, 80.3]	9.7	59	0
60	resnet18	89.1 [88.8, 89.3]	80.2 [79.4, 81.0]	8.9	56	4
61	vgg11	88.6 [88.3, 88.9]	78.8 [78.0, 79.6]	9.8	61	0
62	squeezenet1_1	80.6 [80.3, 81.0]	69.0 [68.1, 69.9]	11.6	62	0
63	squeezenet1_0	80.4 [80.1, 80.8]	68.5 [67.6, 69.4]	11.9	63	0
64	alexnet	79.1 [78.7, 79.4]	67.4 [66.5, 68.3]	11.7	64	0
65	fv_64k	55.7 [55.3, 56.2]	42.6 [41.6, 43.6]	13.2	65	0
66	fv_16k	49.9 [49.5, 50.4]	37.5 [36.6, 38.5]	12.4	66	0
67	fv_4k	41.3 [40.8, 41.7]	31.0 [30.1, 31.9]	10.3	67	0

Table 16. Top-1 model accuracy on the original ImageNet validation set and our new test set Threshold0.7. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, $\Delta \text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. Due to space constraints, references for the models can be found in Appendix D.4.3. The second part of the table can be found on the following page.

ImageNet Top-1 Threshold0.7

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	82.9 [82.5, 83.2]	80.2 [79.4, 80.9]	2.7	2	-1
2	pnasnet5large	82.7 [82.4, 83.1]	80.3 [79.5, 81.1]	2.4	1	1
3	nasnet_large_tf	82.7 [82.4, 83.0]	80.1 [79.3, 80.9]	2.6	3	0
4	nasnetalarge	82.5 [82.2, 82.8]	80.0 [79.2, 80.8]	2.5	4	0
5	senet154	81.3 [81.0, 81.6]	78.7 [77.8, 79.5]	2.6	5	0
6	polynet	80.9 [80.5, 81.2]	78.5 [77.7, 79.3]	2.3	6	0
7	inception_resnet_v2_tf	80.4 [80.0, 80.7]	77.9 [77.1, 78.7]	2.5	8	-1
8	inceptionresnetv2	80.3 [79.9, 80.6]	78.0 [77.2, 78.8]	2.3	7	1
9	se_resnext101_32x4d	80.2 [79.9, 80.6]	77.6 [76.8, 78.5]	2.6	11	-2
10	inception_v4_tf	80.2 [79.8, 80.5]	77.8 [77.0, 78.6]	2.4	10	0
11	inceptionv4	80.1 [79.7, 80.4]	77.9 [77.0, 78.7]	2.2	9	2
12	dpn107	79.7 [79.4, 80.1]	76.6 [75.8, 77.5]	3.1	12	0
13	dpn131	79.4 [79.1, 79.8]	76.6 [75.7, 77.4]	2.9	13	0
14	dpn92	79.4 [79.0, 79.8]	76.3 [75.5, 77.1]	3.1	17	-3
15	dpn98	79.2 [78.9, 79.6]	76.3 [75.5, 77.2]	2.9	16	-1
16	se_resnext50_32x4d	79.1 [78.7, 79.4]	76.5 [75.7, 77.3]	2.6	14	2
17	resnext101_64x4d	79.0 [78.6, 79.3]	75.6 [74.7, 76.4]	3.4	20	-3
18	xception	78.8 [78.5, 79.2]	76.4 [75.5, 77.2]	2.5	15	3
19	se_resnet152	78.7 [78.3, 79.0]	76.1 [75.3, 76.9]	2.5	18	1
20	se_resnet101	78.4 [78.0, 78.8]	75.8 [75.0, 76.7]	2.6	19	1
21	resnet152	78.3 [77.9, 78.7]	75.3 [74.5, 76.2]	3.0	22	-1
22	resnext101_32x4d	78.2 [77.8, 78.5]	75.4 [74.5, 76.2]	2.8	21	1
23	inception_v3_tf	78.0 [77.6, 78.3]	75.0 [74.2, 75.9]	2.9	24	-1
24	resnet_v2_152_tf	77.8 [77.4, 78.1]	75.2 [74.4, 76.1]	2.6	23	1
25	se_resnet50	77.6 [77.3, 78.0]	74.2 [73.3, 75.1]	3.4	30	-5
26	fbresnet152	77.4 [77.0, 77.8]	74.8 [74.0, 75.7]	2.6	25	1
27	resnet101	77.4 [77.0, 77.7]	74.5 [73.6, 75.3]	2.9	29	-2
28	inceptionv3	77.3 [77.0, 77.7]	74.5 [73.6, 75.4]	2.8	28	0
29	inception_v3	77.2 [76.8, 77.6]	74.7 [73.8, 75.6]	2.5	26	3
30	densenet161	77.1 [76.8, 77.5]	74.6 [73.7, 75.4]	2.6	27	3
31	dpn68b	77.0 [76.7, 77.4]	73.8 [72.9, 74.7]	3.2	33	-2
32	resnet_v2_101_tf	77.0 [76.6, 77.3]	74.0 [73.1, 74.8]	3.0	31	1
33	densenet201	76.9 [76.5, 77.3]	73.9 [73.1, 74.8]	3.0	32	1

ImageNet Top-1 Threshold0.7						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	76.8 [76.4, 77.2]	73.7 [72.9, 74.6]	3.1	34	0
35	resnet_v1_101_tf	76.4 [76.0, 76.8]	73.4 [72.5, 74.2]	3.0	35	0
36	cafferesnet101	76.2 [75.8, 76.6]	72.9 [72.0, 73.7]	3.3	37	-1
37	resnet50	76.1 [75.8, 76.5]	72.7 [71.8, 73.6]	3.4	38	-1
38	dpn68	75.9 [75.5, 76.2]	73.0 [72.1, 73.8]	2.9	36	2
39	densenet169	75.6 [75.2, 76.0]	72.3 [71.4, 73.1]	3.3	40	-1
40	resnet_v2_50_tf	75.6 [75.2, 76.0]	72.3 [71.4, 73.2]	3.3	39	1
41	resnet_v1_50_tf	75.2 [74.8, 75.6]	71.9 [71.0, 72.8]	3.3	41	0
42	densenet121	74.4 [74.0, 74.8]	70.5 [69.6, 71.4]	3.9	47	-5
43	vgg19_bn	74.2 [73.8, 74.6]	71.4 [70.5, 72.3]	2.8	42	1
44	pnasnet_mobile_tf	74.1 [73.8, 74.5]	70.6 [69.7, 71.5]	3.6	46	-2
45	nasnetamobile	74.1 [73.7, 74.5]	70.9 [70.0, 71.8]	3.2	45	0
46	inception_v2_tf	74.0 [73.6, 74.4]	71.1 [70.2, 72.0]	2.9	44	2
47	nasnet_mobile_tf	74.0 [73.6, 74.4]	70.0 [69.0, 70.8]	4.0	50	-3
48	bninception	73.5 [73.1, 73.9]	71.3 [70.4, 72.2]	2.2	43	5
49	vgg16_bn	73.4 [73.0, 73.7]	70.2 [69.3, 71.1]	3.1	48	1
50	resnet34	73.3 [72.9, 73.7]	70.2 [69.2, 71.0]	3.2	49	1
51	vgg19	72.4 [72.0, 72.8]	68.7 [67.8, 69.6]	3.7	51	0
52	vgg16	71.6 [71.2, 72.0]	68.0 [67.0, 68.9]	3.6	52	0
53	vgg13_bn	71.6 [71.2, 72.0]	67.3 [66.4, 68.2]	4.3	55	-2
54	mobilenet_v1_tf	71.0 [70.6, 71.4]	66.1 [65.2, 67.0]	4.9	59	-5
55	vgg_19_tf	71.0 [70.6, 71.4]	67.4 [66.5, 68.3]	3.6	54	1
56	vgg_16_tf	70.9 [70.5, 71.3]	67.6 [66.7, 68.5]	3.3	53	3
57	vgg11_bn	70.4 [70.0, 70.8]	66.4 [65.5, 67.3]	4.0	58	-1
58	vgg13	69.9 [69.5, 70.3]	66.0 [65.0, 66.9]	4.0	60	-2
59	inception_v1_tf	69.8 [69.4, 70.2]	66.4 [65.5, 67.4]	3.3	57	2
60	resnet18	69.8 [69.4, 70.2]	66.6 [65.7, 67.5]	3.2	56	4
61	vgg11	69.0 [68.6, 69.4]	64.6 [63.7, 65.6]	4.4	61	0
62	squeezenet1_1	58.2 [57.7, 58.6]	54.4 [53.4, 55.4]	3.8	62	0
63	squeezenet1_0	58.1 [57.7, 58.5]	53.4 [52.4, 54.4]	4.7	63	0
64	alexnet	56.5 [56.1, 57.0]	51.3 [50.3, 52.3]	5.2	64	0
65	fv_64k	35.1 [34.7, 35.5]	29.1 [28.2, 30.0]	6.0	65	0
66	fv_16k	28.3 [27.9, 28.7]	23.4 [22.5, 24.2]	5.0	66	0
67	fv_4k	21.2 [20.8, 21.5]	17.8 [17.0, 18.5]	3.4	67	0

Table 17. Top-5 model accuracy on the original ImageNet validation set and our new test set Threshold0.7. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, $\Delta\text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. Due to space constraints, references for the models can be found in Appendix D.4.3. The second part of the table can be found on the following page.

ImageNet Top-5 Threshold0.7

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	96.2 [96.0, 96.3]	95.6 [95.2, 96.0]	0.6	2	-1
2	nasnet_large_tf	96.2 [96.0, 96.3]	95.7 [95.2, 96.0]	0.5	1	1
3	nasnetalarge	96.0 [95.8, 96.2]	95.3 [94.9, 95.8]	0.7	4	-1
4	pnasnet5large	96.0 [95.8, 96.2]	95.5 [95.0, 95.9]	0.5	3	1
5	polynet	95.6 [95.4, 95.7]	94.9 [94.4, 95.3]	0.7	5	0
6	senet154	95.5 [95.3, 95.7]	94.8 [94.3, 95.2]	0.7	6	0
7	inception_resnet_v2_tf	95.2 [95.1, 95.4]	94.7 [94.2, 95.1]	0.6	7	0
8	inception_v4_tf	95.2 [95.0, 95.4]	94.4 [94.0, 94.9]	0.8	9	-1
9	inceptionresnetv2	95.1 [94.9, 95.3]	94.5 [94.1, 95.0]	0.6	8	1
10	se_resnext101_32x4d	95.0 [94.8, 95.2]	94.3 [93.8, 94.7]	0.7	11	-1
11	inceptionv4	94.9 [94.7, 95.1]	94.3 [93.8, 94.7]	0.6	10	1
12	dpn107	94.7 [94.5, 94.9]	93.7 [93.2, 94.2]	1.0	12	0
13	dpn92	94.6 [94.4, 94.8]	93.7 [93.2, 94.2]	0.9	14	-1
14	dpn131	94.6 [94.4, 94.8]	93.5 [92.9, 93.9]	1.1	20	-6
15	dpn98	94.5 [94.3, 94.7]	93.6 [93.1, 94.1]	0.9	17	-2
16	se_resnext50_32x4d	94.4 [94.2, 94.6]	93.6 [93.1, 94.1]	0.8	16	0
17	se_resnet152	94.4 [94.2, 94.6]	93.7 [93.2, 94.2]	0.7	13	4
18	xception	94.3 [94.1, 94.5]	93.6 [93.1, 94.1]	0.7	18	0
19	se_resnet101	94.3 [94.1, 94.5]	93.6 [93.1, 94.0]	0.7	19	0
20	resnext101_64x4d	94.3 [94.0, 94.5]	93.3 [92.8, 93.8]	0.9	22	-2
21	resnet_v2_152_tf	94.1 [93.9, 94.3]	93.4 [92.9, 93.9]	0.7	21	0
22	resnet152	94.0 [93.8, 94.3]	93.7 [93.2, 94.2]	0.4	15	7
23	inception_v3_tf	93.9 [93.7, 94.1]	92.8 [92.3, 93.3]	1.1	25	-2
24	resnext101_32x4d	93.9 [93.7, 94.1]	92.7 [92.2, 93.2]	1.2	28	-4
25	se_resnet50	93.8 [93.5, 94.0]	93.0 [92.4, 93.5]	0.8	24	1
26	resnet_v2_101_tf	93.7 [93.5, 93.9]	93.2 [92.7, 93.7]	0.5	23	3
27	fbresnet152	93.6 [93.4, 93.8]	92.7 [92.1, 93.2]	0.9	29	-2
28	dpn68b	93.6 [93.4, 93.8]	92.7 [92.1, 93.2]	0.9	31	-3
29	densenet161	93.6 [93.3, 93.8]	92.8 [92.3, 93.3]	0.8	26	3
30	resnet101	93.5 [93.3, 93.8]	92.8 [92.3, 93.3]	0.8	27	3
31	inception_v3	93.5 [93.3, 93.7]	92.7 [92.1, 93.2]	0.9	30	1
32	inceptionv3	93.4 [93.2, 93.6]	92.6 [92.1, 93.1]	0.8	32	0
33	densenet201	93.4 [93.1, 93.6]	92.4 [91.9, 92.9]	1.0	33	0

ImageNet Top-5 Threshold0.7						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	`resnet_v1_152_tf`	93.2 [92.9, 93.4]	92.2 [91.7, 92.7]	1.0	34	0
35	`resnet_v1_101_tf`	92.9 [92.7, 93.1]	92.0 [91.5, 92.5]	0.9	36	-1
36	`resnet50`	92.9 [92.6, 93.1]	92.0 [91.5, 92.5]	0.9	37	-1
37	`resnet_v2_50_tf`	92.8 [92.6, 93.1]	91.9 [91.4, 92.5]	0.9	38	-1
38	`densenet169`	92.8 [92.6, 93.0]	91.9 [91.4, 92.4]	0.9	39	-1
39	`dpn68`	92.8 [92.5, 93.0]	92.1 [91.5, 92.6]	0.7	35	4
40	`cafferesnet101`	92.8 [92.5, 93.0]	91.6 [91.1, 92.2]	1.1	40	0
41	`resnet_v1_50_tf`	92.2 [92.0, 92.4]	91.1 [90.6, 91.7]	1.0	41	0
42	`densenet121`	92.0 [91.7, 92.2]	91.1 [90.5, 91.6]	0.9	42	0
43	`pnasnet_mobile_tf`	91.9 [91.6, 92.1]	90.7 [90.1, 91.3]	1.1	47	-4
44	`vgg19_bn`	91.8 [91.6, 92.1]	91.0 [90.4, 91.5]	0.9	44	0
45	`inception_v2_tf`	91.8 [91.5, 92.0]	91.0 [90.5, 91.6]	0.7	43	2
46	`nasnetamobile`	91.7 [91.5, 92.0]	90.9 [90.3, 91.4]	0.9	46	0
47	`nasnet_mobile_tf`	91.6 [91.3, 91.8]	90.1 [89.5, 90.7]	1.4	50	-3
48	`bninception`	91.6 [91.3, 91.8]	90.9 [90.3, 91.5]	0.7	45	3
49	`vgg16_bn`	91.5 [91.3, 91.8]	90.4 [89.8, 90.9]	1.1	49	0
50	`resnet34`	91.4 [91.2, 91.7]	90.5 [89.9, 91.0]	1.0	48	2
51	`vgg19`	90.9 [90.6, 91.1]	89.7 [89.1, 90.3]	1.2	51	0
52	`vgg16`	90.4 [90.1, 90.6]	88.8 [88.1, 89.4]	1.6	53	-1
53	`vgg13_bn`	90.4 [90.1, 90.6]	89.0 [88.3, 89.6]	1.4	52	1
54	`mobilenet_v1_tf`	90.0 [89.7, 90.2]	87.6 [86.9, 88.2]	2.4	60	-6
56	`vgg_19_tf`	89.8 [89.6, 90.1]	88.5 [87.8, 89.1]	1.4	55	1
55	`vgg_16_tf`	89.8 [89.6, 90.1]	88.6 [87.9, 89.2]	1.3	54	1
57	`vgg11_bn`	89.8 [89.5, 90.1]	88.3 [87.6, 88.9]	1.5	56	1
58	`inception_v1_tf`	89.6 [89.4, 89.9]	88.1 [87.4, 88.7]	1.5	57	1
59	`vgg13`	89.2 [89.0, 89.5]	87.6 [86.9, 88.2]	1.6	59	0
60	`resnet18`	89.1 [88.8, 89.3]	88.1 [87.4, 88.7]	1.0	58	2
61	`vgg11`	88.6 [88.3, 88.9]	86.9 [86.2, 87.5]	1.7	61	0
62	`squeezenet1_1`	80.6 [80.3, 81.0]	78.0 [77.2, 78.8]	2.6	62	0
63	`squeezenet1_0`	80.4 [80.1, 80.8]	77.7 [76.9, 78.5]	2.7	63	0
64	`alexnet`	79.1 [78.7, 79.4]	75.9 [75.0, 76.7]	3.2	64	0
65	`fv_64k`	55.7 [55.3, 56.2]	49.8 [48.8, 50.7]	6.0	65	0
66	`fv_16k`	49.9 [49.5, 50.4]	44.2 [43.2, 45.2]	5.7	66	0
67	`fv_4k`	41.3 [40.8, 41.7]	36.5 [35.6, 37.5]	4.8	67	0

Table 18. Top-1 model accuracy on the original ImageNet validation set and our new test set TopImages. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, $\Delta\text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. Due to space constraints, references for the models can be found in Appendix D.4.3. The second part of the table can be found on the following page.

ImageNet Top-1 TopImages

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	`pnasnet_large_tf`	82.9 [82.5, 83.2]	83.9 [83.2, 84.6]	-1.0	3	-2
2	`pnasnet5large`	82.7 [82.4, 83.1]	83.9 [83.1, 84.6]	-1.1	4	-2
3	`nasnet_large_tf`	82.7 [82.4, 83.0]	84.0 [83.3, 84.7]	-1.3	2	1
4	`nasnetalarge`	82.5 [82.2, 82.8]	84.2 [83.4, 84.9]	-1.7	1	3
5	`senet154`	81.3 [81.0, 81.6]	82.8 [82.1, 83.6]	-1.5	6	-1
6	`polynet`	80.9 [80.5, 81.2]	83.0 [82.2, 83.7]	-2.1	5	1
7	`inception_resnet_v2_tf`	80.4 [80.0, 80.7]	82.5 [81.7, 83.2]	-2.1	8	-1
8	`inceptionresnetv2`	80.3 [79.9, 80.6]	82.8 [82.0, 83.5]	-2.5	7	1
9	`se_resnext101_32x4d`	80.2 [79.9, 80.6]	82.2 [81.5, 83.0]	-2.0	11	-2
10	`inception_v4_tf`	80.2 [79.8, 80.5]	82.3 [81.5, 83.0]	-2.1	9	1
11	`inceptionv4`	80.1 [79.7, 80.4]	82.3 [81.5, 83.0]	-2.2	10	1
12	`dpn107`	79.7 [79.4, 80.1]	81.4 [80.6, 82.1]	-1.6	13	-1
13	`dpn131`	79.4 [79.1, 79.8]	81.3 [80.5, 82.1]	-1.9	15	-2
14	`dpn92`	79.4 [79.0, 79.8]	81.2 [80.5, 82.0]	-1.8	16	-2
15	`dpn98`	79.2 [78.9, 79.6]	81.5 [80.7, 82.3]	-2.3	12	3
16	`se_resnext50_32x4d`	79.1 [78.7, 79.4]	81.4 [80.6, 82.1]	-2.3	14	2
17	`resnext101_64x4d`	79.0 [78.6, 79.3]	80.3 [79.5, 81.0]	-1.3	22	-5
18	`xception`	78.8 [78.5, 79.2]	81.0 [80.2, 81.8]	-2.2	18	0
19	`se_resnet152`	78.7 [78.3, 79.0]	81.0 [80.3, 81.8]	-2.4	17	2
20	`se_resnet101`	78.4 [78.0, 78.8]	80.5 [79.7, 81.3]	-2.1	19	1
21	`resnet152`	78.3 [77.9, 78.7]	80.3 [79.5, 81.1]	-2.0	21	0
22	`resnext101_32x4d`	78.2 [77.8, 78.5]	79.9 [79.1, 80.6]	-1.7	26	-4
23	`inception_v3_tf`	78.0 [77.6, 78.3]	80.1 [79.3, 80.9]	-2.1	23	0
24	`resnet_v2_152_tf`	77.8 [77.4, 78.1]	80.3 [79.5, 81.1]	-2.6	20	4
25	`se_resnet50`	77.6 [77.3, 78.0]	79.4 [78.6, 80.2]	-1.8	31	-6
26	`fbresnet152`	77.4 [77.0, 77.8]	80.1 [79.3, 80.9]	-2.7	24	2
27	`resnet101`	77.4 [77.0, 77.7]	79.0 [78.2, 79.8]	-1.7	32	-5
28	`inceptionv3`	77.3 [77.0, 77.7]	79.6 [78.8, 80.4]	-2.3	27	1
29	`inception_v3`	77.2 [76.8, 77.6]	79.6 [78.8, 80.4]	-2.4	28	1
30	`densenet161`	77.1 [76.8, 77.5]	79.5 [78.7, 80.3]	-2.4	29	1
31	`dpn68b`	77.0 [76.7, 77.4]	79.4 [78.6, 80.2]	-2.4	30	1
32	`resnet_v2_101_tf`	77.0 [76.6, 77.3]	80.1 [79.3, 80.8]	-3.1	25	7
33	`densenet201`	76.9 [76.5, 77.3]	79.0 [78.1, 79.7]	-2.1	34	-1

ImageNet Top-1 TopImages						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	76.8 [76.4, 77.2]	79.0 [78.2, 79.8]	-2.2	33	1
35	resnet_v1_101_tf	76.4 [76.0, 76.8]	78.6 [77.8, 79.4]	-2.2	35	0
36	cafferesnet101	76.2 [75.8, 76.6]	78.3 [77.4, 79.1]	-2.1	37	-1
37	resnet50	76.1 [75.8, 76.5]	78.1 [77.3, 78.9]	-2.0	38	-1
38	dpn68	75.9 [75.5, 76.2]	78.4 [77.6, 79.2]	-2.6	36	2
39	densenet169	75.6 [75.2, 76.0]	78.0 [77.2, 78.8]	-2.4	39	0
40	resnet_v2_50_tf	75.6 [75.2, 76.0]	78.0 [77.2, 78.8]	-2.4	40	0
41	resnet_v1_50_tf	75.2 [74.8, 75.6]	77.0 [76.2, 77.9]	-1.8	41	0
42	densenet121	74.4 [74.0, 74.8]	76.8 [75.9, 77.6]	-2.3	45	-3
43	vgg19_bn	74.2 [73.8, 74.6]	76.6 [75.7, 77.4]	-2.3	46	-3
44	pnasnet_mobile_tf	74.1 [73.8, 74.5]	76.8 [76.0, 77.6]	-2.7	44	0
45	nasnetamobile	74.1 [73.7, 74.5]	76.4 [75.5, 77.2]	-2.3	47	-2
46	inception_v2_tf	74.0 [73.6, 74.4]	77.0 [76.1, 77.8]	-3.0	43	3
47	nasnet_mobile_tf	74.0 [73.6, 74.4]	76.0 [75.1, 76.8]	-2.0	49	-2
48	bninception	73.5 [73.1, 73.9]	77.0 [76.1, 77.8]	-3.4	42	6
49	vgg16_bn	73.4 [73.0, 73.7]	75.9 [75.1, 76.8]	-2.6	50	-1
50	resnet34	73.3 [72.9, 73.7]	76.3 [75.4, 77.1]	-3.0	48	2
51	vgg19	72.4 [72.0, 72.8]	74.2 [73.3, 75.0]	-1.8	51	0
52	vgg16	71.6 [71.2, 72.0]	73.9 [73.0, 74.7]	-2.3	52	0
53	vgg13_bn	71.6 [71.2, 72.0]	73.5 [72.7, 74.4]	-1.9	55	-2
54	mobilenet_v1_tf	71.0 [70.6, 71.4]	72.4 [71.5, 73.3]	-1.4	59	-5
55	vgg_19_tf	71.0 [70.6, 71.4]	73.6 [72.7, 74.5]	-2.6	53	2
56	vgg_16_tf	70.9 [70.5, 71.3]	73.5 [72.7, 74.4]	-2.6	54	2
57	vgg11_bn	70.4 [70.0, 70.8]	73.0 [72.1, 73.8]	-2.6	58	-1
58	vgg13	69.9 [69.5, 70.3]	72.0 [71.1, 72.9]	-2.1	60	-2
59	inception_v1_tf	69.8 [69.4, 70.2]	73.1 [72.2, 73.9]	-3.3	56	3
60	resnet18	69.8 [69.4, 70.2]	73.0 [72.2, 73.9]	-3.3	57	3
61	vgg11	69.0 [68.6, 69.4]	70.8 [69.9, 71.7]	-1.8	61	0
62	squeezenet1_1	58.2 [57.7, 58.6]	61.7 [60.7, 62.6]	-3.5	62	0
63	squeezenet1_0	58.1 [57.7, 58.5]	60.7 [59.7, 61.7]	-2.6	63	0
64	alexnet	56.5 [56.1, 57.0]	58.2 [57.2, 59.1]	-1.7	64	0
65	fv_64k	35.1 [34.7, 35.5]	34.2 [33.3, 35.2]	0.8	65	0
66	fv_16k	28.3 [27.9, 28.7]	27.4 [26.6, 28.3]	0.9	66	0
67	fv_4k	21.2 [20.8, 21.5]	21.1 [20.3, 21.9]	0.1	67	0

Table 19. Top-5 model accuracy on the original ImageNet validation set and our new test set TopImages. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, $\Delta\text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. Due to space constraints, references for the models can be found in Appendix D.4.3. The second part of the table can be found on the following page.

ImageNet Top-5 TopImages

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	`pnasnet_large_tf`	96.2 [96.0, 96.3]	97.2 [96.9, 97.5]	-1.0	2	-1
2	`nasnet_large_tf`	96.2 [96.0, 96.3]	97.2 [96.9, 97.5]	-1.0	1	1
3	`nasnetalarge`	96.0 [95.8, 96.2]	97.1 [96.7, 97.4]	-1.1	3	0
4	`pnasnet5large`	96.0 [95.8, 96.2]	96.9 [96.6, 97.2]	-0.9	4	0
5	`polynet`	95.6 [95.4, 95.7]	96.8 [96.4, 97.1]	-1.2	5	0
6	`senet154`	95.5 [95.3, 95.7]	96.6 [96.2, 97.0]	-1.1	8	-2
7	`inception_resnet_v2_tf`	95.2 [95.1, 95.4]	96.8 [96.4, 97.1]	-1.5	6	1
8	`inception_v4_tf`	95.2 [95.0, 95.4]	96.5 [96.1, 96.9]	-1.3	9	-1
9	`inceptionresnetv2`	95.1 [94.9, 95.3]	96.7 [96.3, 97.0]	-1.5	7	2
10	`se_resnext101_32x4d`	95.0 [94.8, 95.2]	96.2 [95.8, 96.6]	-1.2	11	-1
11	`inceptionv4`	94.9 [94.7, 95.1]	96.4 [96.0, 96.7]	-1.5	10	1
12	`dpn107`	94.7 [94.5, 94.9]	96.0 [95.6, 96.4]	-1.4	13	-1
13	`dpn92`	94.6 [94.4, 94.8]	95.9 [95.5, 96.3]	-1.3	17	-4
14	`dpn131`	94.6 [94.4, 94.8]	96.0 [95.6, 96.4]	-1.5	14	0
15	`dpn98`	94.5 [94.3, 94.7]	96.0 [95.6, 96.4]	-1.5	15	0
16	`se_resnext50_32x4d`	94.4 [94.2, 94.6]	95.9 [95.5, 96.3]	-1.5	18	-2
17	`se_resnet152`	94.4 [94.2, 94.6]	95.9 [95.5, 96.3]	-1.5	19	-2
18	`xception`	94.3 [94.1, 94.5]	95.9 [95.5, 96.3]	-1.6	20	-2
19	`se_resnet101`	94.3 [94.1, 94.5]	95.9 [95.5, 96.3]	-1.6	21	-2
20	`resnext101_64x4d`	94.3 [94.0, 94.5]	95.7 [95.3, 96.1]	-1.5	23	-3
21	`resnet_v2_152_tf`	94.1 [93.9, 94.3]	96.0 [95.6, 96.3]	-1.9	16	5
22	`resnet152`	94.0 [93.8, 94.3]	96.2 [95.8, 96.5]	-2.1	12	10
23	`inception_v3_tf`	93.9 [93.7, 94.1]	95.5 [95.1, 95.9]	-1.5	25	-2
24	`resnext101_32x4d`	93.9 [93.7, 94.1]	95.2 [94.8, 95.6]	-1.3	31	-7
25	`se_resnet50`	93.8 [93.5, 94.0]	95.5 [95.1, 95.9]	-1.8	24	1
26	`resnet_v2_101_tf`	93.7 [93.5, 93.9]	95.8 [95.4, 96.2]	-2.1	22	4
27	`fbresnet152`	93.6 [93.4, 93.8]	95.2 [94.8, 95.7]	-1.7	28	-1
28	`dpn68b`	93.6 [93.4, 93.8]	95.2 [94.8, 95.6]	-1.6	32	-4
29	`densenet161`	93.6 [93.3, 93.8]	95.2 [94.8, 95.6]	-1.7	29	0
30	`resnet101`	93.5 [93.3, 93.8]	95.4 [95.0, 95.8]	-1.9	26	4
31	`inception_v3`	93.5 [93.3, 93.7]	95.1 [94.7, 95.5]	-1.6	34	-3
32	`inceptionv3`	93.4 [93.2, 93.6]	95.2 [94.8, 95.6]	-1.8	30	2
33	`densenet201`	93.4 [93.1, 93.6]	95.2 [94.8, 95.7]	-1.9	27	6

ImageNet Top-5 TopImages						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	93.2 [92.9, 93.4]	95.2 [94.7, 95.6]	-2.0	33	1
35	resnet_v1_101_tf	92.9 [92.7, 93.1]	94.9 [94.4, 95.3]	-2.0	35	0
36	resnet50	92.9 [92.6, 93.1]	94.7 [94.2, 95.1]	-1.8	39	-3
37	resnet_v2_50_tf	92.8 [92.6, 93.1]	94.8 [94.3, 95.2]	-1.9	37	0
38	densenet169	92.8 [92.6, 93.0]	94.7 [94.2, 95.1]	-1.9	38	0
39	dpn68	92.8 [92.5, 93.0]	94.8 [94.3, 95.2]	-2.0	36	3
40	cafferesnet101	92.8 [92.5, 93.0]	94.6 [94.1, 95.0]	-1.8	40	0
41	resnet_v1_50_tf	92.2 [92.0, 92.4]	94.2 [93.8, 94.7]	-2.1	41	0
42	densenet121	92.0 [91.7, 92.2]	94.0 [93.5, 94.5]	-2.0	46	-4
43	pnasnet_mobile_tf	91.9 [91.6, 92.1]	94.1 [93.6, 94.5]	-2.2	44	-1
44	vgg19_bn	91.8 [91.6, 92.1]	94.0 [93.5, 94.4]	-2.1	47	-3
45	inception_v2_tf	91.8 [91.5, 92.0]	94.2 [93.7, 94.7]	-2.5	42	3
46	nasnetamobile	91.7 [91.5, 92.0]	94.1 [93.6, 94.5]	-2.3	43	3
47	nasnet_mobile_tf	91.6 [91.3, 91.8]	93.8 [93.4, 94.3]	-2.3	49	-2
48	bninception	91.6 [91.3, 91.8]	94.0 [93.6, 94.5]	-2.5	45	3
49	vgg16_bn	91.5 [91.3, 91.8]	93.7 [93.2, 94.1]	-2.1	50	-1
50	resnet34	91.4 [91.2, 91.7]	93.9 [93.4, 94.3]	-2.5	48	2
51	vgg19	90.9 [90.6, 91.1]	92.8 [92.2, 93.3]	-1.9	51	0
52	vgg16	90.4 [90.1, 90.6]	92.5 [92.0, 93.0]	-2.1	53	-1
53	vgg13_bn	90.4 [90.1, 90.6]	92.6 [92.1, 93.1]	-2.2	52	1
54	mobilenet_v1_tf	90.0 [89.7, 90.2]	91.4 [90.8, 91.9]	-1.4	59	-5
56	vgg_19_tf	89.8 [89.6, 90.1]	92.1 [91.5, 92.6]	-2.2	56	0
55	vgg_16_tf	89.8 [89.6, 90.1]	92.2 [91.6, 92.7]	-2.3	54	1
57	vgg11_bn	89.8 [89.5, 90.1]	91.9 [91.4, 92.5]	-2.1	58	-1
58	inception_v1_tf	89.6 [89.4, 89.9]	92.1 [91.6, 92.6]	-2.5	55	3
59	vgg13	89.2 [89.0, 89.5]	91.4 [90.8, 91.9]	-2.2	60	-1
60	resnet18	89.1 [88.8, 89.3]	92.0 [91.4, 92.5]	-2.9	57	3
61	vgg11	88.6 [88.3, 88.9]	91.0 [90.4, 91.5]	-2.4	61	0
62	squeezenet1_1	80.6 [80.3, 81.0]	83.9 [83.1, 84.6]	-3.2	62	0
63	squeezenet1_0	80.4 [80.1, 80.8]	83.5 [82.8, 84.3]	-3.1	63	0
64	alexnet	79.1 [78.7, 79.4]	81.8 [81.0, 82.6]	-2.7	64	0
65	fv_64k	55.7 [55.3, 56.2]	55.9 [54.9, 56.8]	-0.1	65	0
66	fv_16k	49.9 [49.5, 50.4]	49.8 [48.8, 50.8]	0.1	66	0
67	fv_4k	41.3 [40.8, 41.7]	41.9 [40.9, 42.8]	-0.6	67	0

D.4.5. Accuracy Plots for All ImageNet Test Sets

Figure 12 shows the top-1 and top-5 accuracies for our three test sets and all convolutional networks in our model testbed. Figure 13 shows the accuracies for all models (including Fisher Vector models) with a probit scale on the axes.

D.4.6. Example Images

Figure 14 shows randomly selected images for three randomly selected classes for both the original ImageNet validation set and our three new test sets.

D.4.7. Effect of Selection Frequency on Model Accuracy

To better understand how the selection frequency of an image impacts the model accuracies, Figures 15, 16, and 17 show model accuracies stratified into five selection frequency bins.

D.4.8. Ambiguous Class Examples

Figure 18 shows randomly selected images from the original ImageNet validation set for three pairs of classes with ambiguous class boundaries. We remark that several more classes in ImageNet have ill-defined boundaries. The three pairs of classes here were chosen only as illustrative examples.

The following list shows names and definitions for the three class pairs:

- Pair 1
    a. `projectile, missile`: “a weapon that is forcibly thrown or projected at a targets but is not self-propelled”
    b. `missile`: “a rocket carrying a warhead of conventional or nuclear explosives; may be ballistic or directed by remote control”
- Pair 2
    c. `tusker`: “any mammal with prominent tusks (especially an elephant or wild boar)”
    d. `Indian elephant, Elephas maximus`: “Asian elephant having smaller ears and tusks primarily in the male”
- Pair 3
    e. `screen, CRT screen`: “the display that is electronically created on the surface of the large end of a cathode-ray tube”
    f. `monitor`: “electronic equipment that is used to check the quality or content of electronic transmissions”

[15]Test Set A is the original validation set, Test Set B is the MatchedFrequencydataset, Test Set C is the Threshold0.7, Test set D is TopImages.

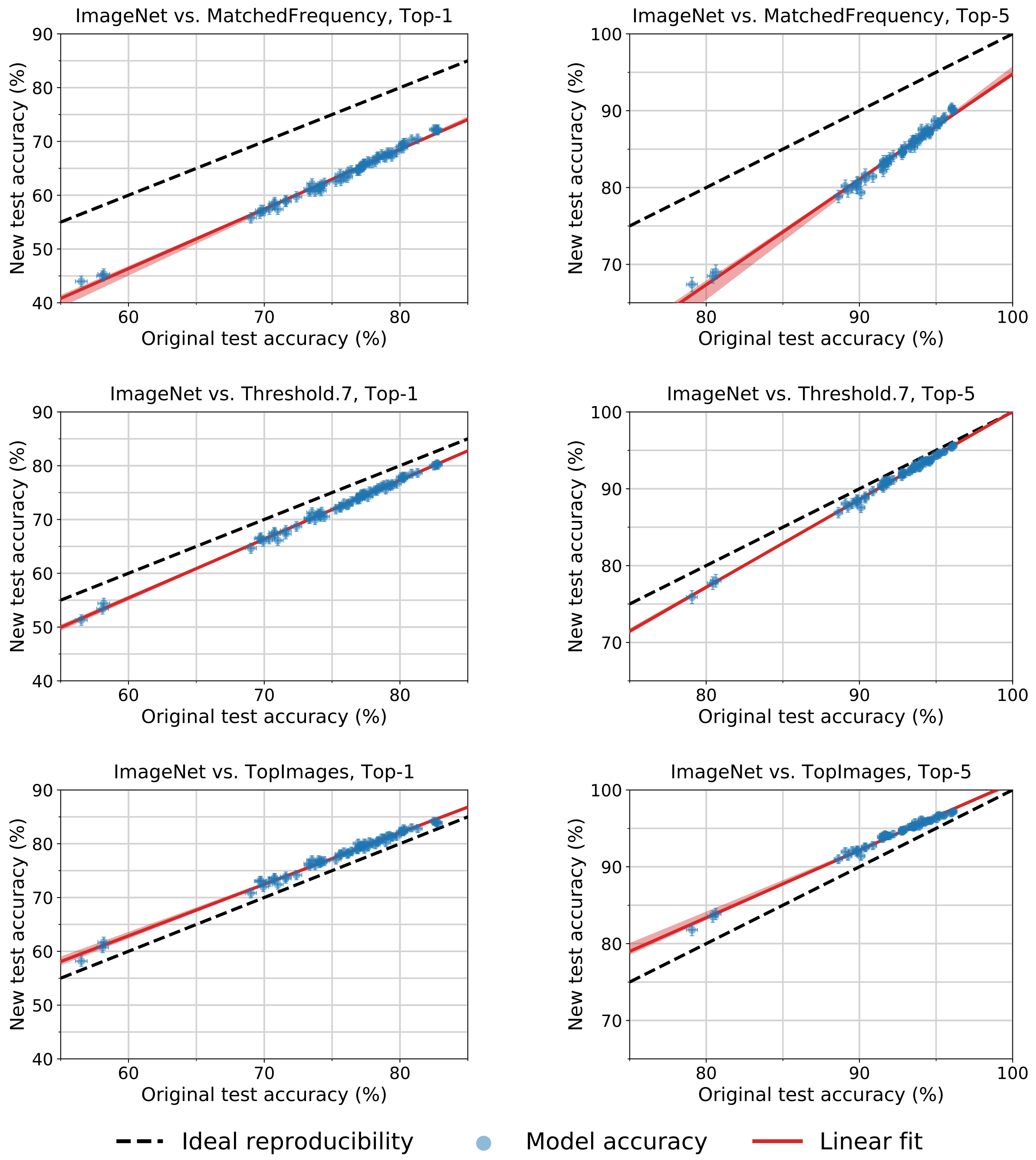


Figure 12. Model accuracy on the original ImageNet validation set vs. our new test sets. See Section 4 for a description of these test sets. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). The red shaded region is a 95% confidence region for the linear fit from 100,000 bootstrap samples. For MatchedFrequency, the accuracies on the new test set are significantly below the original accuracies. The accuracies for Threshold0.7 are still below the original counterpart, but for TopImages they improve over the original test accuracies. This shows that small variations in the data generation process can have significant impact on the accuracy scores. As for CIFAR-10, all plots reveal an approximatly linear relationship between original and new test accuracy. Only the slope for the top-5 accuracies on TopImages is significantly smaller than 1 (0.88, 95% confidence interval from 100,000 bootstrap samples: $[0.81, 0.91]$). It is unclear if this is a sign of adaptive overfitting or due to the models approaching the 100% accuracy regime. Investigating this further is an interesting question for future work.

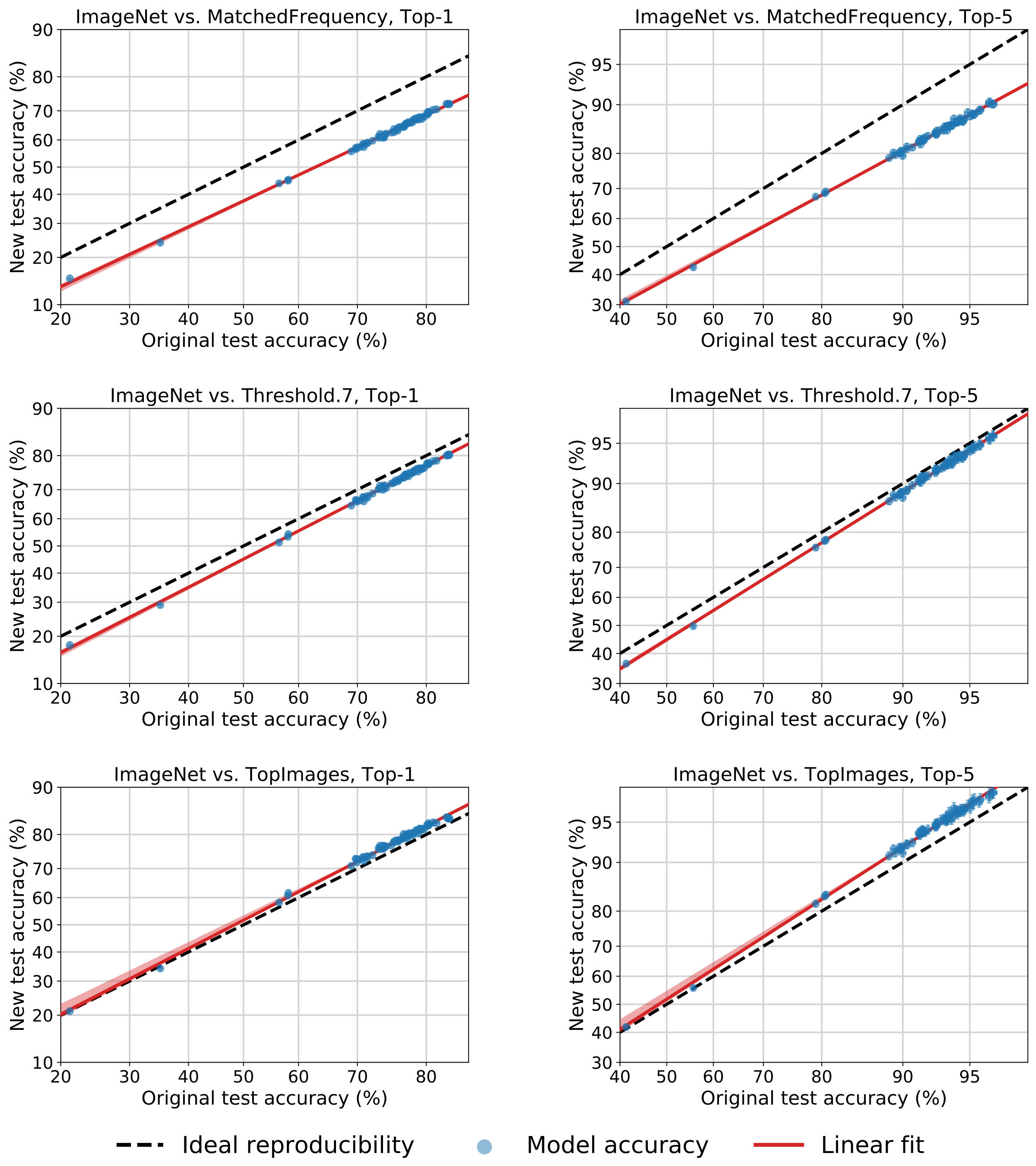
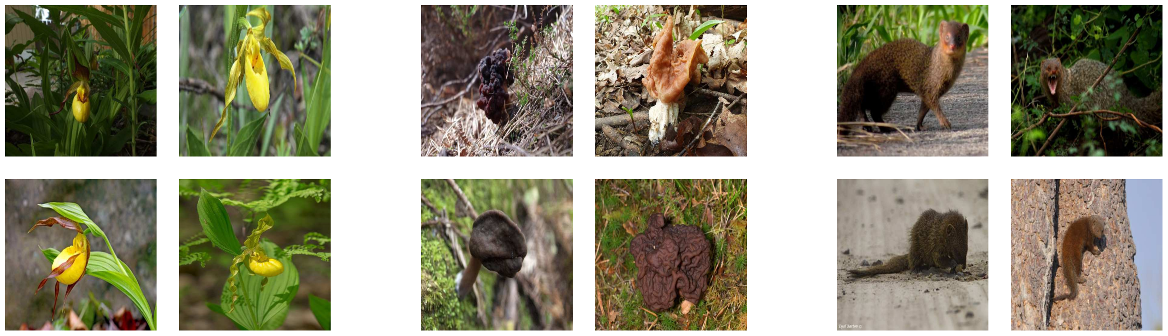
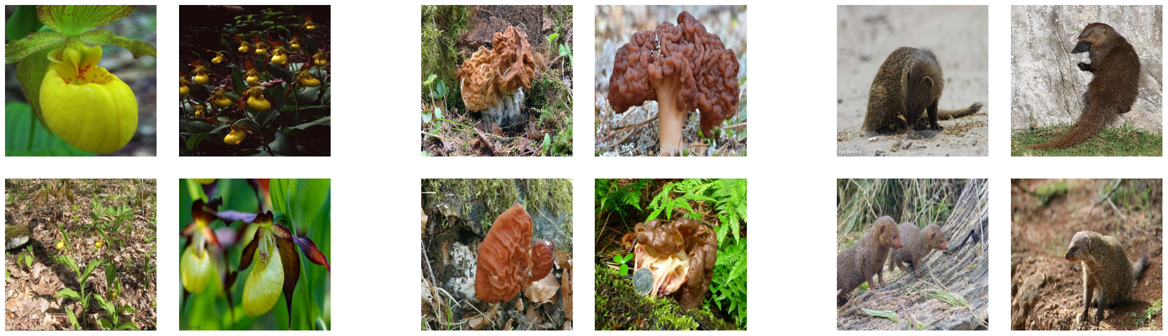


Figure 13. Model accuracy on the original ImageNet validation set vs. our new test sets. The structure of the plots is similar to Figure 12 and we refer the reader to the description there. In contrast to Figure 12, the plots here contain also the Fisher Vector models. Moreover, the axes are scaled according to the probit transformation, i.e., accuracy α appears at $\Phi^{-1}(\alpha)$, where Φ is the Gaussian CDF. For all three datasets and both top-1 and top-5 accuracy, the plots reveal a good linear fit in the probit domain spanning around 60 percentage points of accuracy. All plots include a 95% confidence region for the linear fit as in Figure 12, but the red shaded region is hard to see in some of the plots due to its small size.

(a) Test Set A


(b) Test Set B


(c) Test Set C

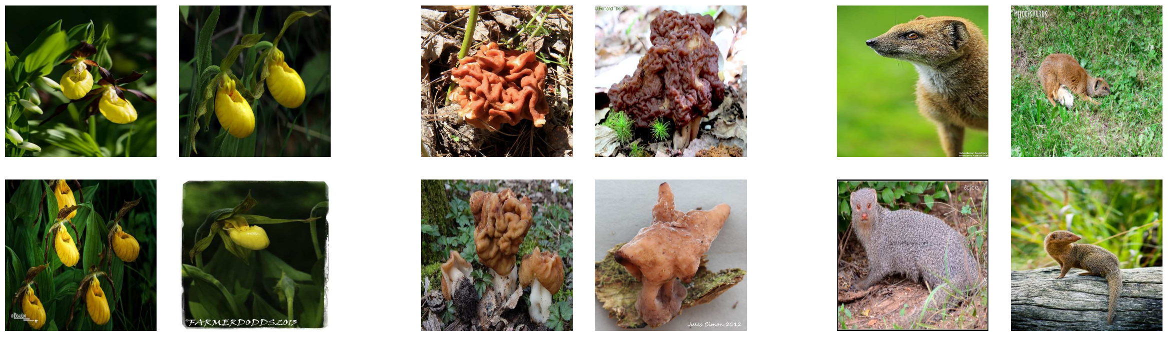

(d) Test Set D

Figure 14. Randomly selected images from the original ImageNet validation set and our new ImageNet test sets. We display four images from three randomly selected classes for each of the four datasets (the original validation set and our three test sets described in Section 4). The displayed classes are “Cypripedium calceolus”, “gyromitra”, and “mongoose”. The following footnote reveals which datasets correspond to original and new ImageNet test sets. [15]

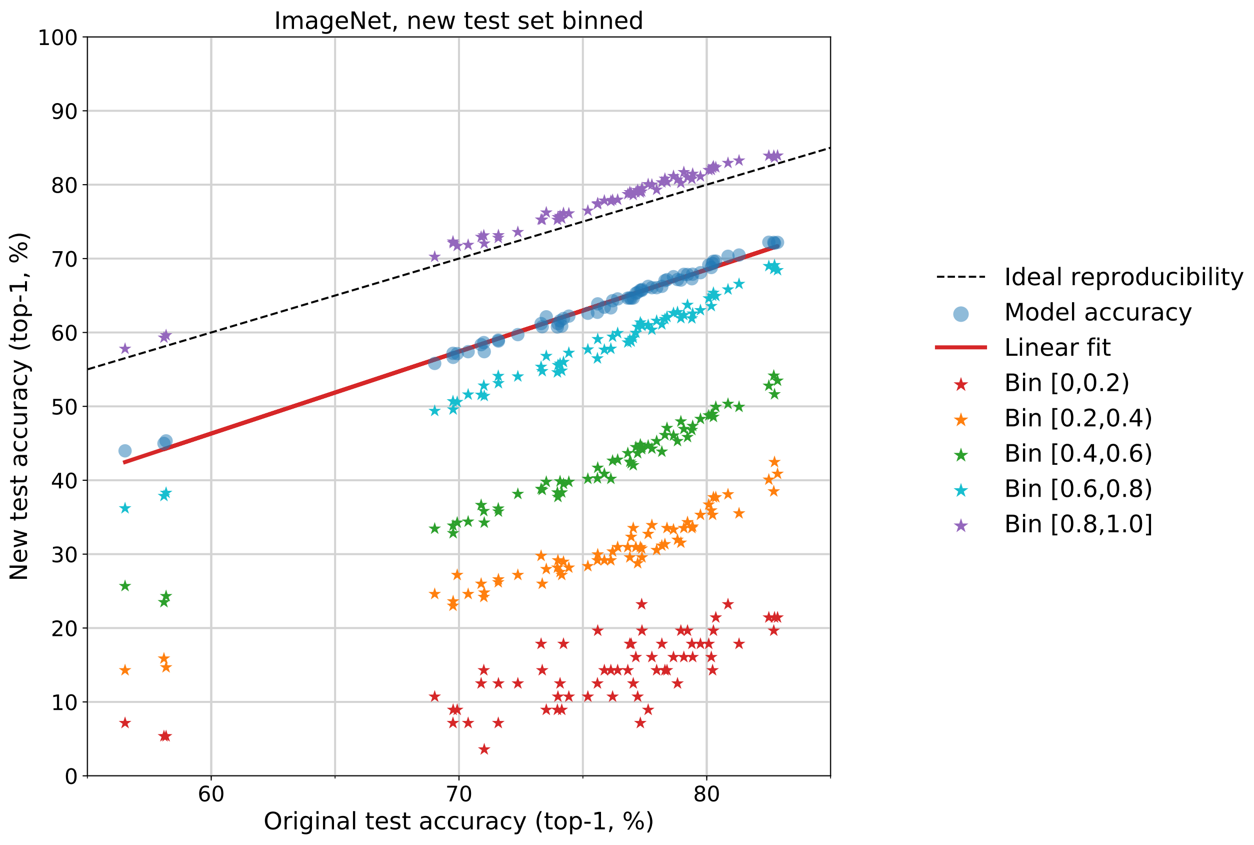


Figure 15. Model accuracy on the original ImageNet validation set vs. accuracy on our new test set MatchedFrequency, stratified into five selection frequency bins. Every bin contains the images with MTurk selection frequency falling into the corresponding range. Each data point corresponds to one model and one of the five frequency bins (indicated by the different colors). The x-value of each data point is given by the model's accuracy on the entire original validation set. The y-value is given by the model's accuracy on our new test images falling into the respective selection frequency bin. The plot shows that the selection frequency has strong influence on the model accuracy. For instance, images with selection frequencies in the $[0.4, 0.6)$ bin lead to an average model accuracy about 20% lower than for the entire test set MatchedFrequency, and 30% lower than the original validation set. We remark that we manually reviewed all images in MatchedFrequency to ensure that (almost) all images have the correct class label, regardless of selection frequency bin.

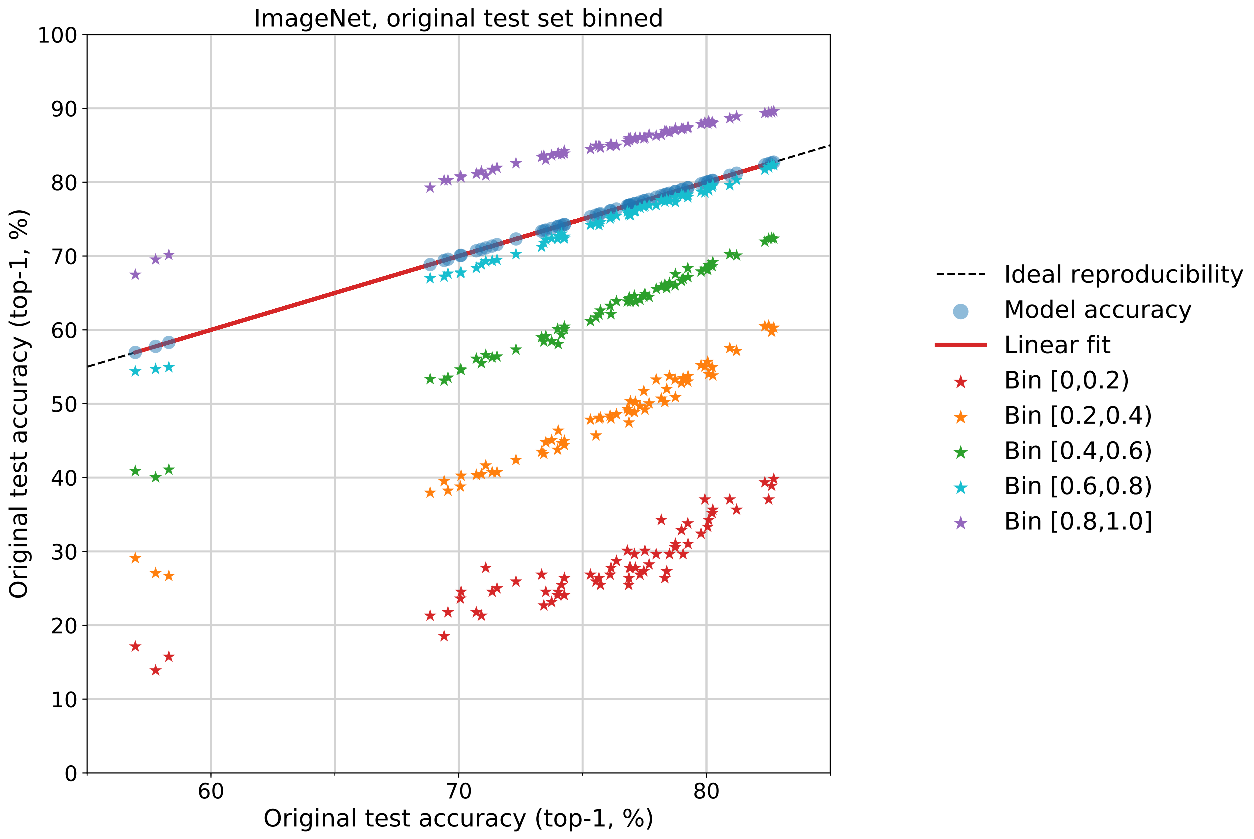


Figure 16. Model accuracy on the original ImageNet validation set stratified into five selection frequency bins. This plot has a similar structure as Figure 15 above, but contains the original validation set accuracy on both axes (as before, the images are binned on the y-axis and not binned on the x-axis, i.e., the x-value is the accuracy on the entire validation set). The plot shows that the selection frequency has strong influence on the model accuracy on the original ImageNet validation set as well. For instance, images with selection frequencies in the $[0.4, 0.6)$ bin lead to an average model accuracy about 10 – 15% lower than for the entire validation set.

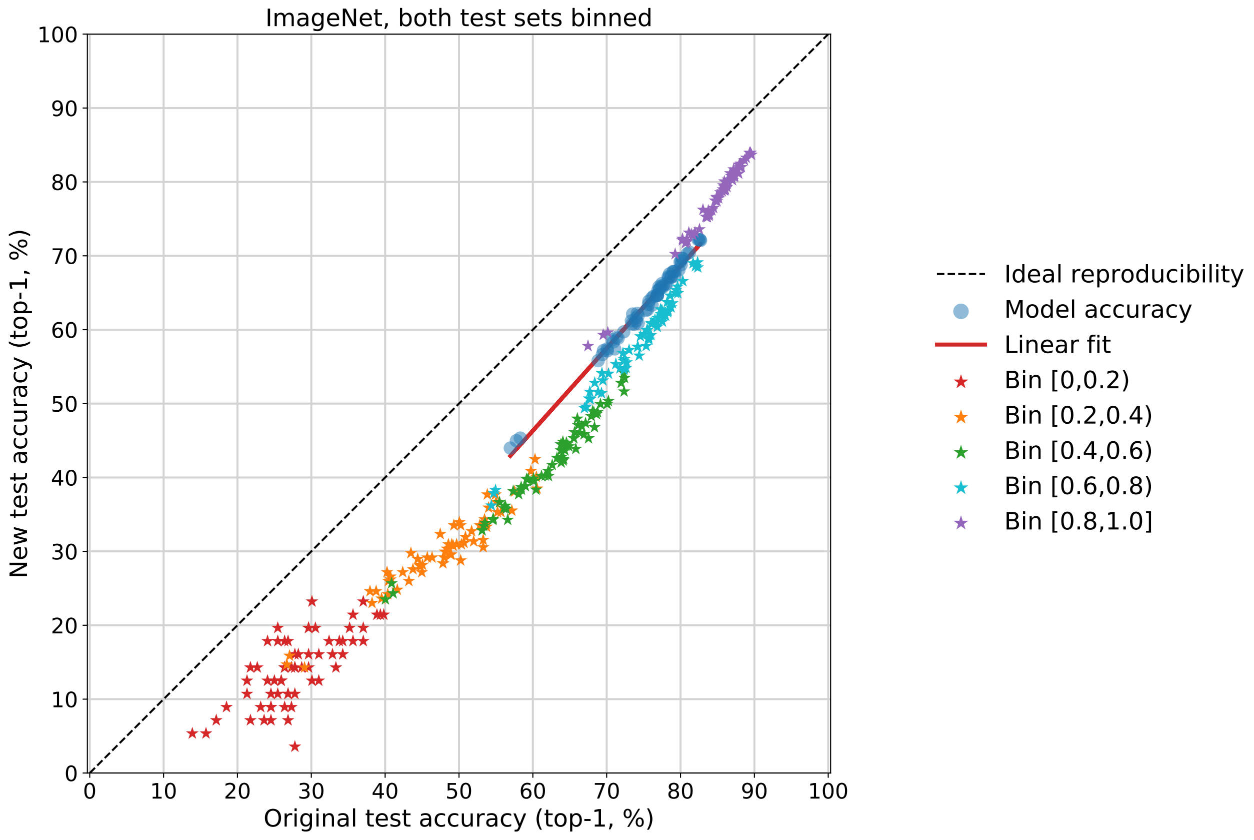
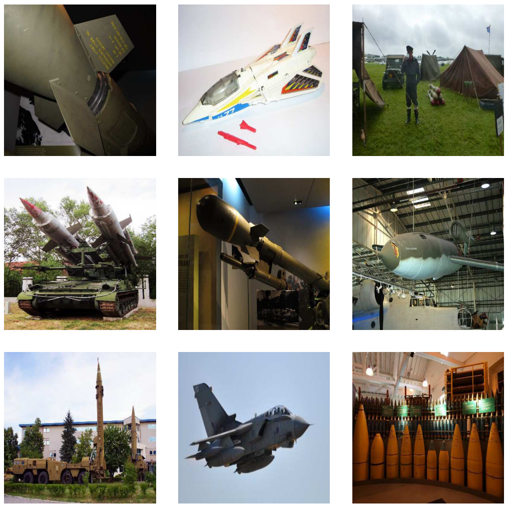
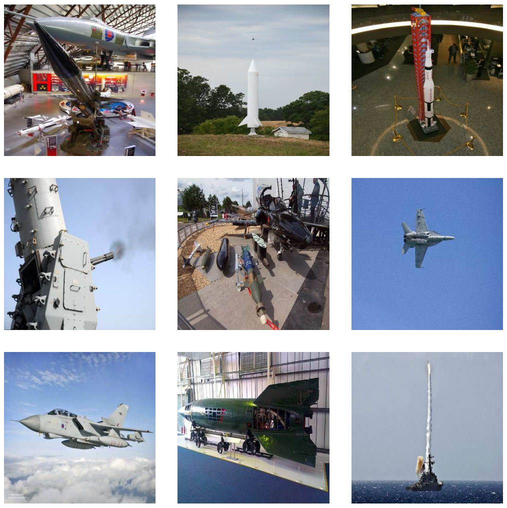
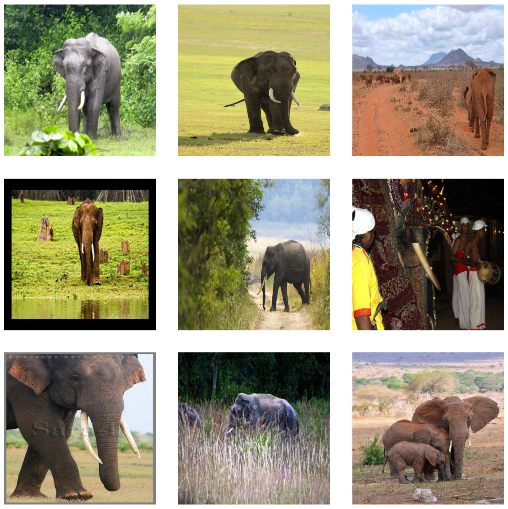
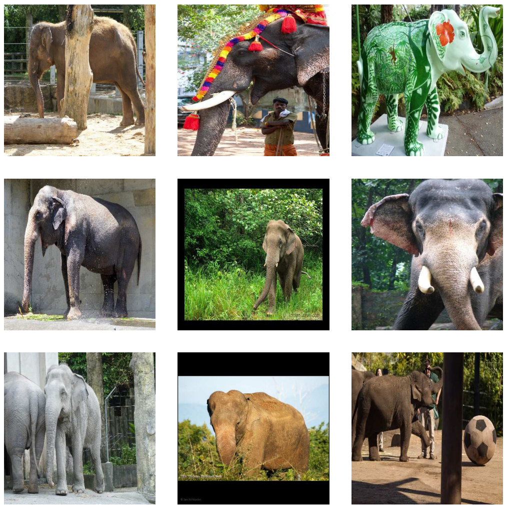


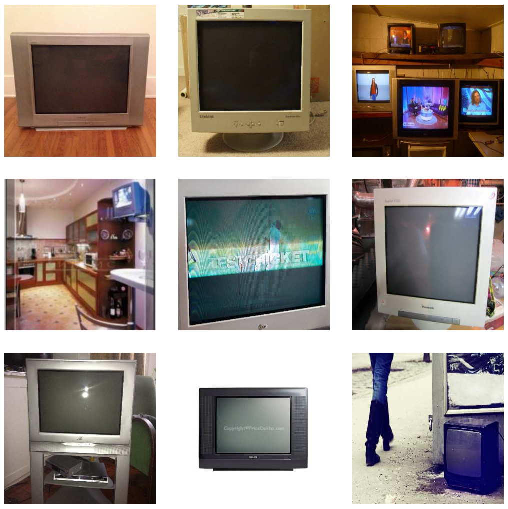
Figure 17. Model accuracy on the original ImageNet validation set vs. accuracy on our new test set MatchedFrequency. In contrast to the preceding Figures 15 and 16, both original and new test accuracy is now stratified into five selection frequency bins. Each data point corresponds to the accuracy achieved by one model on the images from one of the five frequency bins (indicated by the different colors). The plot shows that the model accuracies in the various bins are strongly correlated, but the accuracy on images in our new test is consistently lower. The gap is largest for images in the middle frequency bins (about 20% accuracy difference) and smallest for images in the lowest and highest frequency bins (5 – 10 % difference).


(a) `projectile, missile`

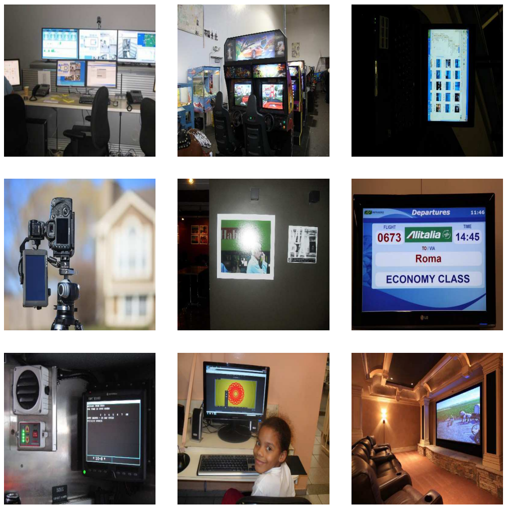

(b) `missile`


(c) `tusker`


(d) `Indian elephant, Elephas maximus`



(e) `screen, CRT screen`



(f) `monitor`

Figure 18. Random images from the original ImageNet validation set for three pairs of classes with ambiguous class boundaries.