From Distribution Shift to Kernel Methods: A study of empirical phenomena in machine learning

Vaishaal Shankar

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-141
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-141.html

August 11, 2020

Copyright © 2020, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

From Distribution Shift to Kernel Methods: A study of empirical phenomena in machine learning

by

Vaishaal Shankar

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Benjamin Recht, Chair
Professor Ion Stoica
Professor Solomon Hsiang

Summer 2020

From Distribution Shift to Kernel Methods: A study of empirical phenomena in machine learning

Copyright 2020
by
Vaishaal Shankar

Abstract

From Distribution Shift to Kernel Methods: A study of empirical phenomena in machine learning

by

Vaishaal Shankar

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Benjamin Recht, Chair

As machine learning becomes a progressively empirical field, the need for rigorous empirical evaluation of existing methodology grows. In this dissertation I present a line of work that studies the effect of subtle distribution shifts to classifier accuracy. I then present a construction and evaluation of high performance classifiers using tools from classical kernel literature.

To study the effect of distribution shfts in machine learning, we build new test sets for the CIFAR-10 and ImageNet datasets. By closely following the original dataset creation processes, we test to what extent current classification models generalize to new data. We evaluate a broad range of models and find accuracy drops of 3% – 15% on CIFAR-10 and 11% – 14% on ImageNet. However, accuracy gains on the original test sets translate to larger gains on the new test sets. Our results suggest that the accuracy drops are not caused by adaptivity, but by the models' inability to generalize to slightly "harder" images than those found in the original test sets.

We then perform an in-depth evaluation of human accuracy on the ImageNet dataset. First, three expert labelers re-annotated 30,000 images from the original ImageNet validation set and the ImageNetV2 replication experiment with multi-label annotations to enable a semantically coherent accuracy measurement. Then we evaluated five trained humans on both datasets. The median of the five labelers outperforms the best publicly released ImageNet model by 1.5% on the original validation set and by 6.2% on ImageNetV2. Moreover, the human labelers see a substantially smaller drop in accuracy between the

two datasets compared to the best available model (less than 1% vs 5.4%). Our results put claims of superhuman performance on ImageNet in context and show that robustly classifying ImageNet at human-level performance is still an open problem.

To study the effect of another form of distribution shift, we study the robustness of image classifiers to temporal perturbations derived from videos. As part of this study, we construct two new datasets, `ImageNet-Vid-Robust` and `YTBB-Robust`, containing a total of 57,897 images grouped into 3,139 sets of perceptually similar images. Our datasets were derived from ImageNet-Vid and Youtube-BB respectively and thoroughly re-annotated by human experts for image similarity. We evaluate a diverse array of classifiers pre-trained on ImageNet and show a median classification accuracy drop of 16 and 10 percent, respectively, on our two datasets. Additionally, we evaluate three detection models and show that natural perturbations induce both classification as well as localization errors, leading to a median drop in detection mAP of 14 points. Our analysis demonstrates that perturbations occurring naturally in videos pose a substantial and realistic challenge to deploying convolutional neural networks in environments that require both reliable and low-latency predictions.

Finally, we investigate the connections between neural networks and simple building blocks in kernel space. In particular, using well established feature space tools such as direct sum, averaging, and moment lifting, we present an algebra for creating “compositional" kernels from bags of features. We show that these operations correspond to many of the building blocks of “neural tangent kernels" (NTK). Experimentally, we show a correlation in test error between neural network architectures and the associated kernels. We construct a simple neural network architecture using only 3×3 convolutions, 2×2 average pooling, ReLU, and optimized with SGD and MSE loss that achieves 96% accuracy on CIFAR10, and whose corresponding compositional kernel achieves 90% accuracy. We also use our constructions to investigate the relative performance of neural networks, NTKs, and compositional kernels in the small dataset regime. In particular, we find that compositional kernels outperform NTKs and neural networks outperform both kernel methods.

To Merra

Contents

Acknowledgments

While I must put this thesis together with one functioning hand due to an unfortunate bike accident at the end of my PhD, the work in this document could not have been possible single handedly (pun intented). I was fortunate to have an excellent advisor, brilliant colleagues, wonderful friends, a supportive girlfriend and a loving family. My strong support system made graduate school truly the best five years of my life so far (not to say it didn't get extremely difficult at times).

To this day I am unsure why Ben decided to take me as an advisee, as coming out of undergrad at Berkeley I knew far more about filesystems and compilers than vector spaces or statistics. I did not have the profile of a typical machine learning graduate student. Nevertheless, Ben's unique and contrarian view of machine learning helped me both grow as a researcher and appreciate the nuances of field without being drawn into the headline grabbing "trend-of-the-week" types of work. While not pumping out 44 ICLR papers may have harmed my H-index, the careful, methodological approach to research Ben encouraged in our group has made me a better scientist.

While Ben gave me the hard problems to work on during my PhD, our group gave me the tools to solve them. I probably sent Shivaram upwards of a $10,000$ slack messages trying to understand how to use a computer properly. From ssh-ing into our local compute nodes, running 500 machine jobs on AWS to tweaking `SPARK_MEM_FRACTION`. All I'm saying is I thought I knew how to use a computer when I got a B.S in Computer Science, but in reality I didn't know anything until I met Shiv.

If Shiv taught me how computers work, Eric taught me how science works. For the two years of my PhD I was convinced Eric knew everything. Operationally, he taught me how to pick good problems to work on, the importance of a good experimental setup, and how to effectively communicate your work. Beyond practical skills, Eric also taught me how to **love** my work, and how science can be simultaneously the coolest and most frustrating thing in the world.

Towards the end of my PhD virtually every paper had Ludwig as a co-author, simply because adding Ludwig makes any project stronger For every project Ludwig will occupy a almost paradoxical dual role of both an advisor dictating high level motivations of the project and a boots-on-the-ground experimentalist writing frontend javascript. While Ludwig and I have had more than a few heated arguments because he wants to do things the "right" way and I want to do things the "pragmatic" way, I've greatly enjoyed working (labeling dogs, writing database code, writing javascript, implementing block coordinate descent, reading the JPEG spec, and writing papers) with him. Hopefully we can continue collaborating while he is Professor-ing at University of Washington.

I cannot emphasize how fortunate I was to have such a strong and diverse group in graduate school. Each member of Modest Yachts shaped my experience in a unique way. I know Horia has probably taught me a lot but in my opinion his greatest contribution to my life was introducing me to the Reuben at Saul's. Esther taught me both about the difficulty and importance of collaborations outside our little machine learning bubble. Max was an excellent source of South Park memes. Sarah taught me to think about the consequences and ethical impacts of seemingly innocous algorithms. Becca helped me understand the value of not getting distracted and finishing things on time (ImageNetV2 would have been impossible if it wasn't for Becca's unrelenting schedule). Ross was a great desk-mate and got me to google for many obscure 90s pop-culture and sports trivia. Stephen actually taught me a lot of random technical tid-bits via his blog, he also taught me its ok to make big pivots during your PhD. Alex and Rohan were fantastic undergraduates to work with very excited to see what they do in the future! Beyond each individual person in our group, the Modest Yachts slack was almost a character of its own with interesting spontaneous technical, political and sometimes philosphical conversations with the smartest people I know.

Outside of our group, I am also thankful to all my senior collaborators, Joanathan Ragan Kelley, Solomon Hsiang and Ion Stoica that helped me work on diverse projects and also served on my thesis commitee.

I of course must mention the never-ending group chat with Achal and Steve, `encrypted tensors` (formerly `remedial tensor discussion`) that served as group therapy, a way to stay connected with my closest friends from undergrad, and most importantly a place to argue about machine learning.

While graduate school was mostly work, there was indeed life outside of the research that served as a crucial pressure release valve. These spanned from much-needed midday coffee-breaks to complain about work with Alyssa to hiking through hot springs in pamukkale with Achal, Radhika and Merra.

As an avid runner, I was delighted to find that tunning was a consistent social activity across my five years in Berkeley. I was fortunate to be able to go on runs with **many** people in my social circle: Jeff, Robert, Ludwig, Becca, Steve, Alyssa, Esther, Horia. Due to Alyssa's encouragement I tried to get into biking but after this accident, perhaps I'll stick to running for now.

Speaking of my accident, Nilesh has always been an excellent room mate and put up with my annoying habbits (loud phone calls, late night tv, etc..), he has literally been a lifesaver last few weeks while I was recovering.

Of course I cannot forget the four most important people in my life: Mom, Dad, Vishan and Merra. Even though they are thousands of miles away, their support really got me through the lowest points of graduate school. My family though they didn't always understand my decision process (spend 5 more years in school rather than get a "real" six figure job?) they were always supportive with whatever I chose to do. Merra has been with me from day 1 of graduate school, and though for most of it we were physically apart, it feels like she was with me every step of the way. Graduate school would not have been the same without our late-night phone calls, elaborate vacation plans and food delivery surprises. I am extremely fortunate to have such a support system.

Chapter 1

Introduction

Over the past decade machine learning has become a heavily empirical science. In this thesis we will hone in on two avenues in which rigorous experimental work led to interesting findings, distribution shifts and high performance kernel machines. We first present some background on the two topics of interest.

1.1 Distribution Shift

The past decade has seen substantial progress on a wide range of machine learning benchmarks. A key challenge for the field now is translating the emerging technologies into safe, dependable, and secure systems that can be deployed in the real world and in interactions with humans. However, it is not clear whether machine learning currently has the evaluation methodology to underwrite dependable performance. In short, what can we expect from a trained model with a good benchmark score?

Machine learning benchmarks evaluated with standard train / test splits only provide a narrow guarantee for future performance: as long as the data comes from the same distribution as the test set, we can expect a model to behave similarly well. However, i.i.d. data is a highly idealized scenario and small deviations from the data distribution can lead to substantially worse performance [46, 71, 114, 119, 127, 135, 144]. Providing broader performance guarantees is difficult because tasks in machine learning are often incompletely defined (what set of pixel values counts as a cat?). Instead of precisely characterizing the actual task of interest, we approximate it through train and test sets. The hope is that a model with good performance on this train / test split can imitate human behavior on the task of interest.

Unfortunately, we have little understanding of the extent to which current benchmark protocols can measure the relative generalization capabilities of trained models and humans on a given task. While widely used datasets such as ImageNet [32, 131] and SQuAD [123] have human baselines, it is unclear what conclusions we can draw from a direct comparison in the i.i.d. scenario favored by machine learning models. Trained models that "surpass" the human baseline on a benchmark still often fail in a variety of ways, while humans are usually reliable in a wide range of scenarios. Moreover, the benchmarks often attempt to render immeasurable ambiguous, cultural, and subjective aspects of their tasks measurable in a way that does not capture important dimensions of human experience, thus making the benchmarks problematic from a human perspective.

To address the above concerns we perform a systematic analysis to study how a wide array of models behave under various *natural* distribution shifts. We then study how susceptible humans are to one shifts to gain better context for model performance.

1.1.1 Potential Causes of Accuracy Drops

We adopt the standard classification setup and posit the existence of a "true" underlying data distribution $\mathcal{D}$ over labeled examples (x, y). The overall goal in classification is to find a model $\hat{f}$ that minimizes the population loss

$$L_{\mathcal{D}}(\hat{f}) \;=\; \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\Big[\mathbb{I}[\hat{f}(x) \neq y]\Big] \,. \tag{1.1.1}$$

Since we usually do not know the distribution $\mathcal{D}$, we instead measure the performance of a trained classifier via a *test set* S drawn from the distribution $\mathcal{D}$:

$$L_S(\hat{f}) \;=\; \frac{1}{|S|} \sum_{(x,y)\in S} \mathbb{I}[\hat{f}(x) \neq y] \,. \tag{1.1.2}$$

We then use this test error $L_S(\hat{f})$ as a proxy for the population loss $L_{\mathcal{D}}(\hat{f})$. If a model $\hat{f}$ achieves a low test error, we assume that it will perform similarly well on future examples from the distribution $\mathcal{D}$. This assumption underlies essentially all empirical evaluations in machine learning since it allows us to argue that the model $\hat{f}$ generalizes.

In our experiments, we test this assumption by collecting a new test set S' from a data distribution $\mathcal{D}'$ that we carefully control to resemble the original distribution $\mathcal{D}$. Ideally, the original test accuracy $L_S(\hat{f})$ and new test accuracy $L_{S'}(\hat{f})$ would then match up to the random sampling error. In contrast to this idealized view, our results in Figure 3.2 show a large drop in accuracy from the original test set S set to our new test set S'. To understand this accuracy drop in more detail, we decompose the difference between $L_S(\hat{f})$ and $L_{S'}(\hat{f})$ into three parts (dropping the dependence on $\hat{f}$ to simplify notation):

$$L_S - L_{S'} = \underbrace{(L_S - L_{\mathcal{D}})}_{\text{Adaptivity gap}} + \underbrace{(L_{\mathcal{D}} - L_{\mathcal{D}'})}_{\text{Distribution Gap}} + \underbrace{(L_{\mathcal{D}'} - L_{S'})}_{\text{Generalization gap}} .$$

We now discuss to what extent each of the three terms can lead to accuracy drops.

Generalization Gap. By construction, our new test set S' is independent of the existing classifier $\hat{f}$. Hence the third term $L_{\mathcal{D}'} - L_{S'}$ is the standard *generalization gap* commonly studied in machine learning. It is determined solely by the random sampling error.

A first guess is that this inherent sampling error suffices to explain the accuracy drops in Figure 3.2 (e.g., the new test set S' could have sampled certain "harder" modes of the distribution $\mathcal{D}$ more often). However, random fluctuations of this magnitude are unlikely for the size of our test sets. With 10,000 data points (as in our new ImageNet test set), a Clopper-Pearson 95% confidence interval for the test accuracy has size of at most $\pm 1\%$. Increasing the confidence level to 99.99% yields a confidence interval of size at most $\pm 2\%$. Moreover, these confidence intervals become smaller for higher accuracies, which is the relevant regime for the best-performing models. Hence random chance alone cannot explain the accuracy drops observed in our experiments.[1]

Adaptivity Gap. We call the term $L_S - L_{\mathcal{D}}$ the *adaptivity gap*. It measures how much adapting the model $\hat{f}$ to the test set S causes the test error L_S to underestimate the population loss $L_{\mathcal{D}}$. If we assumed that our model $\hat{f}$ is independent of the test set S, this terms would follow the same concentration laws as the generalization gap $L_{\mathcal{D}'} - L_{S'}$ above. But this assumption is undermined by the common practice of tuning model hyperparameters directly on the test set, which introduces dependencies between the model $\hat{f}$ and the test set S. In the extreme case, this can be seen as training directly on the test set. But milder forms of adaptivity may also artificially inflate accuracy scores by increasing the gap between L_S and $L_{\mathcal{D}}$ beyond the purely random error.

Distribution Gap. We call the term $L_{\mathcal{D}} - L_{\mathcal{D}'}$ the *distribution gap*. It quantifies how much the change from the original distribution $\mathcal{D}$ to our new distribution $\mathcal{D}'$ affects the

[1] We remark that the sampling process for the new test set S' could indeed *systematically* sample harder modes more often than under the original data distribution $\mathcal{D}$. Such a systematic change in the sampling process would not be an effect of random chance but captured by the distribution gap described below.

model $\hat{f}$. Note that this term is not influenced by random effects but quantifies the systematic difference between sampling the original and new test sets. While we went to great lengths to minimize such systematic differences, in practice it is hard to argue whether two high-dimensional distributions are exactly the same. We typically lack a precise definition of either distribution, and collecting a real dataset involves a plethora of design choices.

1.1.2 Distinguishing Between the Two Mechanisms

For a single model $\hat{f}$, it is unclear how to disentangle the adaptivity and distribution gaps. To gain a more nuanced understanding, we measure accuracies for *multiple* models $\hat{f}_1, \ldots, \hat{f}_k$. This provides additional insights because it allows us to determine how the two gaps have evolved over time.

For both CIFAR-10 and ImageNet, the classification models come from a long line of papers that incrementally improved accuracy scores over the past decade. A natural assumption is that later models have experienced more adaptive overfitting since they are the result of more successive hyperparameter tuning on the same test set. Their higher accuracy scores would then come from an increasing adaptivity gap and reflect progress only on the specific examples in the test set S but not on the actual distribution $\mathcal{D}$. In an extreme case, the population accuracies $L_{\mathcal{D}}(\hat{f}_i)$ would plateau (or even decrease) while the test accuracies $L_S(\hat{f}_i)$ would continue to grow for successive models $\hat{f}_i$.

However, this idealized scenario is in stark contrast to our results in Figure 3.2. Later models do not see diminishing returns but an *increased* advantage over earlier models. Hence we view our results as evidence that the accuracy drops mainly stem from a large distribution gap. After presenting our results in more detail in the next section, we will further discuss this point in Section 2.4.

1.2 Kernel Methods

Recent research has drawn exciting connections between neural networks and kernel methods, providing new insights into training dynamics, generalization, and expressibility [6, 27, 35, 36, 58, 80, 97]. This line of work relates "infinitely wide" neural networks to particular kernel spaces, showing that infinite limits of random initializations of neural networks lead to particular kernels on the same input data. Since these initial investigations, some have proposed to use these kernels in prediction problems, finding promising results on many benchmark problems [8, 100]. However, these kernels do not match the performance of neural networks on most tasks of interest, and the kernel constructions themselves are not only hard to compute, but their mathematical formulae are difficult to even write down [7].

We explore kernels constructions that are both *simple* and performant. Our kernel constructions perform well on both standard benchmark tasks and real world scientific tasks such as transcription factor binding site prediction 6 and solar flare prediction [82]

1.2.1 Nonparametric Prediction With Kernels

We proceed with kernel classification as follows. Let C be the total number of classes. Let $\{x_1...x_N\}$ be N training examples in d dimensions. Let $\{y_1...y_N\}$ be N one-hot encoded training labels. We use v_{im} to denote the m^{th} entry of the vector v_i. For a choice of kernel function $k(x, y)$, loss function $\mathcal{L}$, and regularization value λ, we solve the following optimization problem:

$$\underset{\alpha}{\text{minimize}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\Big(\sum_{j=1}^{N} \boldsymbol{K}_{j \cdot} \alpha, y_i \Big) + \lambda \operatorname{Tr}(\alpha^T K \alpha) \tag{1.2.1}$$

where $\boldsymbol{K}$ is the matrix of kernel evaluations on the data: $K_{ij} = k(x_i, x_j)$. The prediction for an example x_{test} is:

$$\underset{c}{\operatorname{argmax}} \sum_{i=1}^{N} \alpha_{ic} k(x_{test}, x_i) \tag{1.2.2}$$

If not otherwise specificed, we use the squared error loss, $\mathcal{L}(\hat{y}, y) = \|\hat{y} - y\|^2$, for our experiments. In this case, α in (1.2.1) is given by

$$\alpha = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{Y}$$

where $\boldsymbol{Y}$ is the $N \times C$ matrix of all one-hot-encodings of the labels. We allow for the value of $\lambda = 0$ in our experiments, and oftentimes this value produces the lowest test error.

Part I

Distribution Shift

Chapter 2

Do ImageNet Classifiers Generalize to ImageNet

2.1 Introduction

The overarching goal of machine learning is to produce models that *generalize*. We usually quantify generalization by measuring the performance of a model on a held-out test set. What does good performance on the test set then imply? At the very least, one would hope that the model also performs well on a new test set assembled from the same data source by following the same data cleaning protocol.

In this chapter, we realize this thought experiment by replicating the dataset creation process for two prominent benchmarks, CIFAR-10 and ImageNet [32, 91]. In contrast to the ideal outcome, we find that a wide range of classification models fail to reach their original accuracy scores. The accuracy drops range from 3% to 15% on CIFAR-10 and 11% to 14% on ImageNet. On ImageNet, the accuracy loss amounts to approximately five years of progress in a highly active period of machine learning research.

Conventional wisdom suggests that such drops arise because the models have been adapted to the specific images in the original test sets, e.g., via extensive hyperparameter tuning. However, our experiments show that the relative order of models is almost exactly preserved on our new test sets: the models with highest accuracy on the original test sets are still the models with highest accuracy on the new test sets. Moreover, there are no diminishing returns in accuracy. In fact, every percentage point of accuracy improvement on the original test set translates to a *larger* improvement on our new test sets. So although later models could have been adapted more to the test set, they see smaller drops in accuracy. These results provide evidence that exhaustive test set evaluations are an effective way to improve image classification models. Adaptivity is therefore an unlikely explanation for the accuracy drops.

Instead, we propose an alternative explanation based on the relative difficulty of the original and new test sets. We demonstrate that it is possible to recover the original ImageNet accuracies almost exactly if we only include the easiest images from our candidate pool. This suggests that the accuracy scores of even the best image classifiers are still highly sensitive to minutiae of the data cleaning process. This brittleness puts claims about human-level performance into context [67, 86, 132]. It also shows that current classifiers still do not generalize reliably even in the benign environment of a carefully controlled reproducibility experiment.

Figure 3.2 shows the main result of our experiment. Before we describe our methodology in Section 2.2, the next section provides relevant background. To enable future research, we release both our new test sets and the corresponding code.[1]

2.2 Summary of Our Experiments

We now give an overview of the main steps in our reproducibility experiment. Appendices 2.7 and 2.8 describe our methodology in more detail. We begin with the first decision, which was to choose informative datasets.

2.2.1 Choice of Datasets

We focus on image classification since it has become the most prominent task in machine learning and underlies a broad range of applications. The cumulative progress on ImageNet is often cited as one of the main breakthroughs in computer vision and machine learning [109]. State-of-the-art models now surpass human-level accuracy by some measure [67, 132]. This makes it particularly important to check if common image classification models can reliably generalize to new data from the same source.

We decided on CIFAR-10 and ImageNet, two of the most widely-used image classification benchmarks [63]. Both datasets have been the focus of intense research for almost ten years now. Due to the competitive nature of these benchmarks, they are an excellent example for testing whether adaptivity has led to overfitting. In addition to their popularity, their carefully documented dataset creation process makes them well suited for a reproducibility experiment [32, 91, 132].

[1] `https://github.com/modestyachts/CIFAR-10.1` and `https://github.com/modestyachts/ImageNetV2`

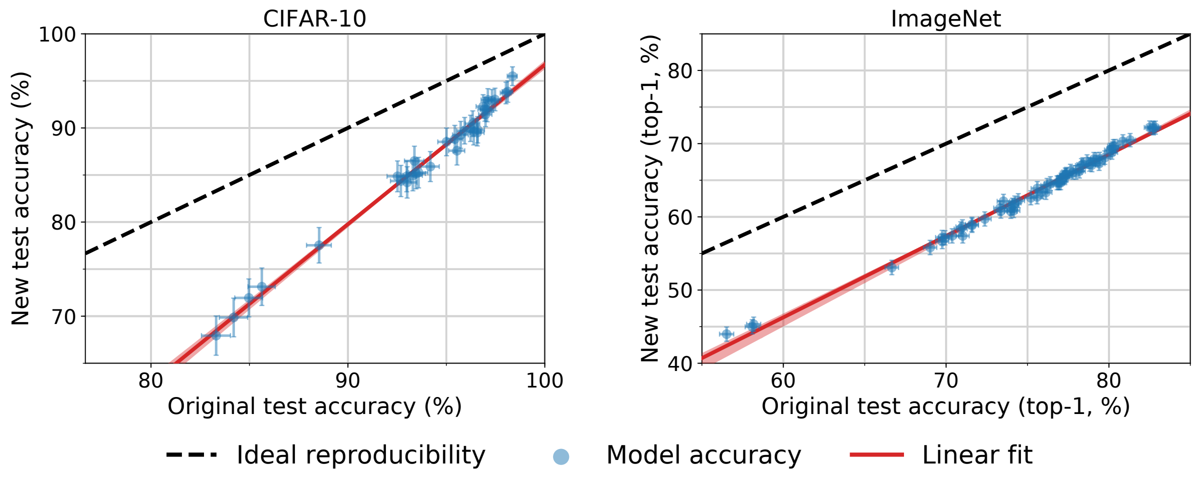


Figure 2.1: Model accuracy on the original test sets vs. our new test sets. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). The plots reveal two main phenomena: (i) There is a significant drop in accuracy from the original to the new test sets. (ii) The model accuracies closely follow a linear function with slope *greater* than 1 (1.7 for CIFAR-10 and 1.1 for ImageNet). This means that every percentage point of progress on the original test set translates into more than one percentage point on the new test set. The two plots are drawn so that their aspect ratio is the same, i.e., the slopes of the lines are visually comparable. The red shaded region is a 95% confidence region for the linear fit from 100,000 bootstrap samples.

Each of the two datasets has specific features that make it especially interesting for our replication study. CIFAR-10 is small enough so that many researchers developed and tested new models for this dataset. In contrast, ImageNet requires significantly more computational resources, and experimenting with new architectures has long been out of reach for many research groups. As a result, CIFAR-10 has likely experienced more hyperparameter tuning, which may also have led to more adaptive overfitting.

On the other hand, the limited size of CIFAR-10 could also make the models more susceptible to small changes in the distribution. Since the CIFAR-10 models are only exposed to a constrained visual environment, they may be unable to learn a robust representation. In contrast, ImageNet captures a much broader variety of images: it contains about 24× more training images than CIFAR-10 and roughly 100× more pixels per image. So conventional wisdom (such as the claims of human-level performance) would suggest that ImageNet models also generalize more reliably .

As we will see, neither of these conjectures is supported by our data: CIFAR-10 models do not suffer from more adaptive overfitting, and ImageNet models do not appear to be significantly more robust.

2.2.2 Dataset Creation Methodology

One way to test generalization would be to evaluate existing models on new i.i.d. data from the original test distribution. For example, this would be possible if the original dataset authors had collected a larger initial dataset and randomly split it into two test sets, keeping one of the test sets hidden for several years. Unfortunately, we are not aware of such a setup for CIFAR-10 or ImageNet.

In this paper, we instead mimic the original distribution as closely as possible by repeating the dataset curation process that selected the original test set[2] from a larger data source. While this introduces the difficulty of disentangling the adaptivity gap from the distribution gap, it also enables us to check whether independent replication affects current accuracy scores. In spite of our efforts, we found that it is astonishingly hard to replicate the test set distributions of CIFAR-10 and ImageNet. At a high level, creating a new test set consists of two parts:

Gathering Data. To obtain images for a new test set, a simple approach would be to use a different dataset, e.g., Open Images [90]. However, each dataset comes with specific biases [144]. For instance, CIFAR-10 and ImageNet were assembled in the late 2000s, and some classes such as `car` or `cell_phone` have changed significantly over the past decade. We avoided such biases by drawing new images from the same source as CIFAR-10 and ImageNet. For CIFAR-10, this was the larger Tiny Image dataset [145]. For ImageNet, we followed the original process of utilizing the Flickr image hosting service and only considered images uploaded in a similar time frame as for ImageNet. In addition to the data source and the class distribution, both datasets also have rich structure *within* each class. For instance, each class in CIFAR-10 consists of images from multiple specific keywords in Tiny Images. Similarly, each class in ImageNet was assembled from the results of multiple queries to the Flickr API. We relied on the documentation of the two datasets to closely match the sub-class distribution as well.

[2]For ImageNet, we repeat the creation process of the *validation set* because most papers developed and tested models on the validation set. We discuss this point in more detail in Appendix 2.8.1. In the context to this paper, we use the terms “validation set” and “test set” interchangeably for ImageNet.

Cleaning Data. Many images in Tiny Images and the Flickr results are only weakly related to the query (or not at all). To obtain a high-quality dataset with correct labels, it is therefore necessary to manually select valid images from the candidate pool. While this step may seem trivial, our results in Section 2.3 will show that it has major impact on the model accuracies.

The authors of CIFAR-10 relied on paid student labelers to annotate their dataset. The researchers in the ImageNet project utilized Amazon Mechanical Turk (MTurk) to handle the large size of their dataset. We again replicated both annotation processes. Two graduate students authors of this paper impersonated the CIFAR-10 labelers, and we employed MTurk workers for our new ImageNet test set. For both datasets, we also followed the original labeling instructions, MTurk task format, etc.

After collecting a set of correctly labeled images, we sampled our final test sets from the filtered candidate pool. We decided on a test set size of 2,000 for CIFAR-10 and 10,000 for ImageNet. While these are smaller than the original test sets, the sample sizes are still large enough to obtain 95% confidence intervals of about $\pm 1\%$. Moreover, our aim was to avoid bias due to CIFAR-10 and ImageNet possibly leaving only "harder" images in the respective data sources. This effect is minimized by building test sets that are small compared to the original datasets (about 3% of the overall CIFAR-10 dataset and less than 1% of the overall ImageNet dataset).

2.2.3 Results on the New Test Sets

After assembling our new test sets, we evaluated a broad range of image classification models spanning a decade of machine learning research. The models include the seminal AlexNet [92], widely used convolutional networks [68, 76, 137, 142], and the state-of-the-art [23, 103]. For all deep architectures, we used code previously published online. We relied on pre-trained models whenever possible and otherwise ran the training commands from the respective repositories. In addition, we also evaluated the best-performing approaches preceding convolutional networks on each dataset. These are random features for CIFAR-10 [21, 122] and Fisher vectors for ImageNet [117].[3] We wrote our own implementations for

[3]We remark that our implementation of Fisher vectors yields top-5 accuracy numbers that are 17% lower than the published numbers in ILSVRC 2012 [132]. Unfortunately, there is no publicly available reference implementation of Fisher vector models achieving this accuracy score. Hence our implementation should not be seen as an exact reproduction of the state-of-the-art Fisher vector model, but as a baseline inspired by this approach. The main goal of including Fisher vector models in our experiment is to investigate if they follow the same overall trends as convolutional neural networks.

CIFAR-10						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	autoaug_pyramid_net_	98.4 [98.1, 98.6]	95.5 [94.5, 96.4]	2.9	1	0
6	shake_shake_64d_cutc	97.1 [96.8, 97.4]	93.0 [91.8, 94.1]	4.1	5	1
16	wide_resnet_28_10	95.9 [95.5, 96.3]	89.7 [88.3, 91.0]	6.2	14	2
23	resnet_basic_110	93.5 [93.0, 93.9]	85.2 [83.5, 86.7]	8.3	24	-1
27	vgg_15_BN_64	93.0 [92.5, 93.5]	84.9 [83.2, 86.4]	8.1	27	0
30	cudaconvnet	88.5 [87.9, 89.2]	77.5 [75.7, 79.3]	11.0	30	0
31	random_features_256k	85.6 [84.9, 86.3]	73.1 [71.1, 75.1]	12.5	31	0

ImageNet Top-1						
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
4	nasnetalarge	82.5 [82.2, 82.8]	72.2 [71.3, 73.1]	10.3	1	3
21	resnet152	78.3 [77.9, 78.7]	67.0 [66.1, 67.9]	11.3	21	0
23	inception_v3_tf	78.0 [77.6, 78.3]	66.1 [65.1, 67.0]	11.9	24	-1
30	densenet161	77.1 [76.8, 77.5]	65.3 [64.4, 66.2]	11.8	30	0
43	vgg19_bn	74.2 [73.8, 74.6]	61.9 [60.9, 62.8]	12.3	44	-1
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

Table 2.1: Model accuracies on the original CIFAR-10 test set, the original ImageNet validation set, and our new test sets. Δ Rank is the relative difference in the ranking from the original test set to the new test set in the full ordering of all models (see Appendices 2.7.3.3 and 2.8.4.4). 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 Appendices 2.7.3.2 and 2.8.4.3.

these models, which we also release publicly.[4]

Overall, the top-1 accuracies range from 83% to 98% on the original CIFAR-10 test set and 21% to 83% on the original ImageNet validation set. We refer the reader to Appendices 2.8.4.3 and 2.7.3.2 for a full list of models and source repositories.

Figure 3.2 in the introduction plots original vs. new accuracies, and Table 2.1 in this section summarizes the numbers of key models. The remaining accuracy scores can be

[4] https://github.com/modestyachts/nondeep

found in Appendices 2.7.3.3 and 2.8.4.4. We now briefly describe the two main trends and discuss the results further in Section 2.4.

A Significant Drop in Accuracy. All models see a large drop in accuracy from the original test sets to our new test sets. For widely used architectures such as VGG [137] and ResNet [68], the drop is 8% on CIFAR-10 and 11% on ImageNet. On CIFAR-10, the state of the art [23] is more robust and only drops by 3% from 98.4% to 95.5%. In contrast, the best model on ImageNet [103] sees an 11% drop from 83% to 72% in top-1 accuracy and a 6% drop from 96% to 90% in top-5 accuracy. So the top-1 drop on ImageNet is larger than what we observed on CIFAR-10.

To put these accuracy numbers into perspective, we note that the best model in the ILSVRC[5] 2013 competition achieved 89% top-5 accuracy, and the best model from ILSVRC 2014 achieved 93% top-5 accuracy. So the 6% drop in top-5 accuracy from the 2018 state-of-the-art corresponds to approximately five years of progress in a very active period of machine learning research.

Few Changes in the Relative Order. When sorting the models in order of their original and new accuracy, there are few changes in the respective rankings. Models with comparable original accuracy tend to see a similar decrease in performance. In fact, Figure 3.2 shows that the original accuracy is highly predictive of the new accuracy and that the relationship can be summarized well with a linear function. On CIFAR-10, the new accuracy of a model is approximately given by the following formula:

$$\text{acc}_{\text{new}} \;=\; 1.69 \cdot \text{acc}_{\text{orig}} - 72.7\% \;.$$

On ImageNet, the top-1 accuracy of a model is given by

$$\text{acc}_{\text{new}} \;=\; 1.11 \cdot \text{acc}_{\text{orig}} - 20.2\% \;.$$

Computing a 95% confidence interval from 100,000 bootstrap samples gives $[1.63, 1.76]$ for the slope and $[-78.6, -67.5]$ for the offset on CIFAR-10, and $[1.07, 1.19]$ and $[-26.0, -17.8]$ respectively for ImageNet.

On both datasets, the slope of the linear fit is *greater* than 1. So models with higher original accuracy see a smaller drop on the new test sets. In other words, model robustness *improves* with increasing accuracy. This effect is less pronounced on ImageNet (slope 1.1) than on CIFAR-10 (slope 1.7). In contrast to a scenario with strong adaptive overfitting, neither dataset sees diminishing returns in accuracy scores when going from the original to the new test sets.

[5]ILSVRC is the ImageNet Large Scale Visual Recognition Challenge [132].

2.2.4 Experiments to Test Follow-Up Hypotheses

Since the drop from original to new accuracies is concerningly large, we investigated multiple hypotheses for explaining this drop. Appendices 2.7.2 and 2.8.3 list a range of follow-up experiments we conducted, e.g., re-tuning hyperparameters, training on part of our new test set, or performing cross-validation. However, none of these effects can explain the size of the drop. We conjecture that the accuracy drops stem from small variations in the human annotation process. As we will see in the next section, the resulting changes in the test sets can significantly affect model accuracies.

2.3 Understanding the Impact of Data Cleaning on ImageNet

A crucial aspect of ImageNet is the use of MTurk. There is a broad range of design choices for the MTurk tasks and how the resulting annotations determine the final dataset. To better understand the impact of these design choices, we assembled three different test sets for ImageNet. All of these test sets consist of images from the same Flickr candidate pool, are correctly labeled, and selected by more than 70% of the MTurk workers on average. Nevertheless, the resulting model accuracies vary by 14%. To put these numbers in context, we first describe our MTurk annotation pipeline.

MTurk Tasks. We designed our MTurk tasks and user interface to closely resemble those originally used for ImageNet. As in ImageNet, each MTurk task contained 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 2.8.4.1 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. In total, we collected sufficient MTurk annotations so that we have at least 20 annotated validation images for each class.

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.

Sampling Strategies. In order to understand how the MTurk selection frequency affects the model accuracies, we explored three sampling strategies.

- **MatchedFrequency:** First, we estimated the selection frequency distribution for each class from the annotated original validation images. We then sampled ten images from our candidate pool for each class according to these class-specific distributions (see Appendix 2.8.1.2 for details).
- **Threshold0.7:** For each class, we sampled ten images with selection frequency at least 0.7.
- **TopImages:** For each class, we chose the ten images with highest selection frequency.

In order to minimize labeling errors, we manually reviewed each dataset and removed incorrect images. The average selection frequencies of the three final datasets range from 0.93 for TopImages over 0.85 for Threshold0.7 to 0.73 for MatchedFrequency. For comparison, the original validation set has an average selection frequency of 0.71 in our experiments. Hence all three of our new test sets have higher selection frequencies than the original ImageNet validation set. In the preceding sections, we presented results on MatchedFrequency for ImageNet since it is closest to the validation set in terms of selection frequencies.

Results. Table 2.2 shows that the MTurk selection frequency has significant impact on both top-1 and top-5 accuracy. In particular, TopImages has the highest average MTurk selection frequency and sees a small *increase* of about 2% in both average top-1 and top-5 accuracy compared to the original validation set. This is in stark contrast to MatchedFrequency, which has the lowest average selection frequency and exhibits a significant drop of 12% and 8%, respectively. The Threshold0.7 dataset is in the middle and sees a small decrease of 3% in top-1 and 1% in top-5 accuracy.

In total, going from TopImages to MatchedFrequency decreases the accuracies by about 14% (top-1) and 10% (top-5). For comparison, note that after excluding AlexNet (and the SqueezeNet models tuned to match AlexNet [77]), the range of accuracies spanned by all remaining convolutional networks is roughly 14% (top-1) and 8% (top-5). So the

Sampling Strategy	Average MTurk Selection Freq.	Average Top-1 Accuracy Change	Average Top-5 Accuracy Change
MatchedFrequency	0.73	-11.8%	-8.2%
Threshold0.7	0.85	-3.2%	-1.2%
TopImages	0.93	+2.1%	+1.8%

Table 2.2: Impact of the three sampling strategies for our ImageNet test sets. The table shows the average MTurk selection frequency in the resulting datasets and the average changes in model accuracy compared to the original validation set. We refer the reader to Section 2.3 for a description of the three sampling strategies. All three test sets have an average selection frequency of more than 0.7, yet the model accuracies still vary widely. For comparison, the original ImageNet validation set has an average selection frequency of 0.71 in our MTurk experiments. The changes in average accuracy span 14% and 10% in top-1 and top-5, respectively. This shows that details of the sampling strategy have large influence on the resulting accuracies.

variation in accuracy caused by the three sampling strategies is larger than the variation in accuracy among all post-AlexNet models we tested.

Figure 2.2 plots the new vs. original top-1 accuracies on Threshold0.7 and TopImages, similar to Figure 3.2 for MatchedFrequency before. For easy comparison of top-1 and top-5 accuracy plots on all three datasets, we refer the reader to Figure 3.2 in Appendix 2.8.4.4. All three plots show a good linear fit.

2.4 Discussion

We now return to the main question from Section 1.1.1: *What causes the accuracy drops?* As before, we distinguish between two possible mechanisms.

2.4.1 Adaptivity Gap

In its prototypical form, *adaptive* overfitting would manifest itself in diminishing returns observed on the new test set (see Section 1.1.2). However, we do not observe this pattern on either CIFAR-10 or ImageNet. On both datasets, the slope of the linear fit is *greater* than 1, i.e., each point of accuracy improvement on the original test set translates to more than 1% on the new test set. This is the opposite of the standard overfitting scenario. So

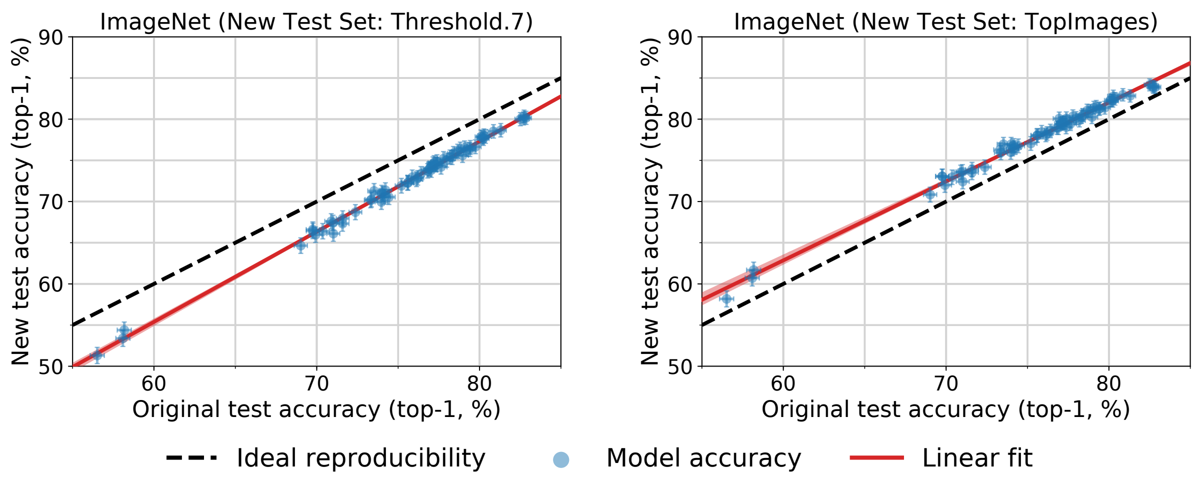


Figure 2.2: Model accuracy on the original ImageNet validation set vs. accuracy on two variants of our new test set. We refer the reader to Section 2.3 for a description of these test sets. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). On Threshold0.7, the model accuracies are 3% lower than on the original test set. On TopImages, which contains the images most frequently selected by MTurk workers, the models perform 2% *better* than on the original test set. The accuracies on both datasets closely follow a linear function, similar to MatchedFrequency in Figure 3.2. The red shaded region is a 95% confidence region for the linear fit from 100,000 bootstrap samples.

at least on CIFAR-10 and ImageNet, multiple years of competitive test set adaptivity did not lead to diminishing accuracy numbers.

While our experiments rule out the most dangerous form of adaptive overfitting, we remark that they do not exclude all variants. For instance, it could be that any test set adaptivity leads to a roughly constant drop in accuracy. Then all models are affected equally and we would see no diminishing returns since later models could still be better. Testing for this form of adaptive overfitting likely requires a new test set that is truly i.i.d. and not the result of a separate data collection effort. Finding a suitable dataset for such an experiment is an interesting direction for future research.

The lack of adaptive overfitting contradicts conventional wisdom in machine learning. We now describe two mechanisms that could have prevented adaptive overfitting:

The Ladder Mechanism. Blum and Hardt introduced the Ladder algorithm to protect machine learning competitions against adaptive overfitting [15]. The core idea is that constrained interaction with the test set can allow a large number of model evaluations to succeed, even if the models are chosen adaptively. Due to the natural form of their algorithm, the authors point out that it can also be seen as a mechanism that the machine learning community *implicitly* follows.

Limited Model Class. Adaptivity is only a problem if we can choose among models for which the test set accuracy differs significantly from the population accuracy. Importantly, this argument does not rely on the number of *all* possible models (e.g., all parameter settings of a neural network), but only on those models that could actually be evaluated on the test set. For instance, the standard deep learning workflow only produces models trained with SGD-style algorithms on a fixed training set, and requires that the models achieve high training accuracy (otherwise we would not consider the corresponding hyperparameters). Hence the number of different models arising from the current methodology may be small enough so that uniform convergence holds.

Our experiments offer little evidence for favoring one explanation over the other. One observation is that the convolutional networks shared many errors on CIFAR-10, which could be an indicator that the models are rather similar. But to gain a deeper understanding into adaptive overfitting, it is likely necessary to gather further data from more machine learning benchmarks, especially in scenarios where adaptive overfitting *does* occur naturally.

2.4.2 Distribution Gap

The lack of diminishing returns in our experiments points towards the distribution gap as the primary reason for the accuracy drops. Moreover, our results on ImageNet show that changes in the sampling strategy can indeed affect model accuracies by a large amount, even if the data source and other parts of the dataset creation process stay the same.

So in spite of our efforts to match the original dataset creation process, the distribution gap is still our leading hypothesis for the accuracy drops. This demonstrates that it is surprisingly hard to accurately replicate the distribution of current image classification datasets. The main difficulty likely is the subjective nature of the human annotation step. There are many parameters that can affect the quality of human labels such as the annotator population (MTurk vs. students, qualifications, location & time, etc.), the exact task format, and compensation. Moreover, there are no exact definitions for many classes in ImageNet (e.g., see Appendix 2.8.4.8). Understanding these aspects in more detail is an important direction for designing future datasets that contain challenging images while still being labeled correctly.

The difficulty of clearly defining the data distribution, combined with the brittle behavior of the tested models, calls into question whether the black-box and i.i.d. framework of learning can produce reliable classifiers. Our analysis of selection frequencies in Figure 2.15 (Appendix 2.8.4.7) shows that we could create a new test set with even lower model accuracies. The images in this hypothetical dataset would still be correct, from Flickr, and selected by more than half of the MTurk labelers on average. So in spite of the impressive accuracy scores on the original validation set, current ImageNet models still have difficulty generalizing from “easy” to “hard” images.

2.4.3 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 3.2 in the introduction or Figures 2.12 and 2.13 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 2.8.2, 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 2.6 and 2.13, 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

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

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.

2.5 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., [32, 43, 47, 102, 118, 132, 144, 159]. The paper closest to ours is [144], 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, [144] do not test on previously unseen data, so their experiments cannot rule out adaptive overfitting.

Transfer Learning From ImageNet. Kornblith et al. [89] study how well accuracy on ImageNet transfers to other image classification datasets. An important difference from both our work and [144] 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 [144], 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 [13, 140] 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 [106] 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 [40, 44, 73, 84, 153]. 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., [55, 70]. 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.

2.6 Conclusion & Future Work

The expansive growth of machine learning rests on the aspiration to deploy trained systems in a variety of challenging environments. Common examples include autonomous vehicles, content moderation, and medicine. In order to use machine learning in these areas responsibly, it is important that we can both train models with sufficient generalization abilities, and also reliably measure their performance. As our results show, these goals still pose significant hurdles even in a benign environment.

Our experiments are only one step in addressing this reliability challenge. There are multiple promising avenues for future work:

Understanding Adaptive Overfitting. In contrast to conventional wisdom, our experiments show that there are no diminishing returns associated with test set re-use on CIFAR-10 and ImageNet. A more nuanced understanding of this phenomenon will require studying whether other machine learning problems are also resilient to adaptive overfitting. For instance, one direction would be to conduct similar reproducibility experiments on tasks in natural language processing, or to analyze data from competition platforms such as Kaggle and CodaLab.[6]

Characterizing the Distribution Gap. Why do the classification models in our testbed perform worse on the new test sets? The selection frequency experiments in Section 2.3 suggest that images selected less frequently by the MTurk workers are harder for the models. However, the selection frequency analysis does not explain what aspects of the images make them harder. Candidate hypotheses are object size, special filters applied to the images (black & white, sepia, etc.), or unusual object poses. Exploring whether there is a succinct description of the difference between the original and new test distributions is an interesting direction for future work.

Learning More Robust Models. An overarching goal is to make classification models more robust to small variations in the data. If the change from the original to our new test sets can be characterized accurately, techniques such as data augmentation or robust optimization may be able to close some of the accuracy gap. Otherwise, one possible approach could be to gather particularly hard examples from Flickr or other data sources and expand the training set this way. However, it may also be necessary to develop entirely novel approaches to image classification.

Measuring Human Accuracy. One interesting question is whether our new test sets are also harder for humans. As a first step in this direction, our human accuracy experiment on CIFAR-10 (see Appendix 2.7.2.5) shows that average human performance is not affected significantly by the distribution shift between the original and new images that are most difficult for the models. This suggests that the images are only harder for the trained models and not for humans. But a more comprehensive understanding of the human baseline will require additional human accuracy experiments on both CIFAR-10 and ImageNet.

Building Further Test Sets. The dominant paradigm in machine learning is to evaluate the performance of a classification model on a single test set per benchmark. Our results suggest that this is not comprehensive enough to characterize the reliability of current models. To understand their generalization abilities more accurately, new test data from various sources may be needed. One intriguing question here is whether accuracy on other test sets will also follow a linear function of the original test accuracy.

[6]`https://www.kaggle.com/competitions` and `https://competitions.codalab.org/`.

Suggestions For Future Datasets. We found that it is surprisingly difficult to create a new test set that matches the distribution of an existing dataset. Based on our experience with this process, we provide some suggestions for improving machine learning datasets in the future:

- **Code release.** It is hard to fully document the dataset creation process in a paper because it involves a long list of design choices. Hence it would be beneficial for reproducibility efforts if future dataset papers released not only the data but also all code used to create the datasets.
- **Annotator quality.** Our results show that changes in the human annotation process can have significant impact on the difficulty of the resulting datasets. To better understand the quality of human annotations, it would be valuable if authors conducted a standardized test with their annotators (e.g., classifying a common set of images) and included the results in the description of the dataset. Moreover, building variants of the test set with different annotation processes could also shed light on the variability arising from this data cleaning step.
- **"Super hold-out".** Having access to data from the original CIFAR-10 and ImageNet data collection could have clarified the cause of the accuracy drops in our experiments. By keeping an additional test set hidden for multiple years, future benchmarks could explicitly test for adaptive overfitting after a certain time period.
- **Simpler tasks for humans.** The large number of classes and fine distinctions between them make ImageNet a particularly hard problem for humans without special training. While classifying a large variety of objects with fine-grained distinctions is an important research goal, there are also trade-offs. Often it becomes necessary to rely on images with high annotator agreement to ensure correct labels, which in turn leads to bias by excluding harder images. Moreover, the large number of classes causes difficulties when characterizing human performance. So an alternative approach for a dataset could be to choose a task that is simpler for humans in terms of class structure (fewer classes, clear class boundaries), but contains a larger variety of object poses, lighting conditions, occlusions, image corruptions, etc.
- **Test sets with expert annotations.** Compared to building a full training set, a test set requires a smaller number of human annotations. This makes it possible to employ a separate labeling process for the test set that relies on more costly expert annotations. While this violates the assumption that train and test splits are i.i.d.

from the same distribution, the expert labels can also increase quality both in terms of correctness and example diversity.

Finally, we emphasize that our recommendations here should *not* be seen as flaws in CIFAR-10 or ImageNet. Both datasets were assembled in the late 2000s for an accuracy regime that is very different from the state-of-the-art now. Over the past decade, especially ImageNet has successfully guided the field to increasingly better models, thereby clearly demonstrating the immense value of this dataset. But as models have increased in accuracy and our reliability expectations have grown accordingly, it is now time to revisit how we create and utilize datasets in machine learning.

2.7 Appendix: Details of the CIFAR-10 Experiments

We first present our reproducibility experiment for the CIFAR-10 image classification dataset [91]. 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) [63].
- The dataset creation process for CIFAR-10 is transparent and well documented [91]. Importantly, CIFAR-10 draws from the larger Tiny Images repository that has more fine-grained labels than the ten CIFAR-10 classes [145]. 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 [23, 33, 51, 126, 158, 166]). 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 $\times$ 32 pixels and was released in 2007 [145]. The images are organized by roughly 75,000 *keywords* that correspond to the non-abstract nouns from the WordNet database [112] 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 [91]. 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 [91]. First, the researchers assembled a set of relevant keywords for each class by using the hyponym relations in WordNet [112] (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 [91] 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.

2.7.1 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.

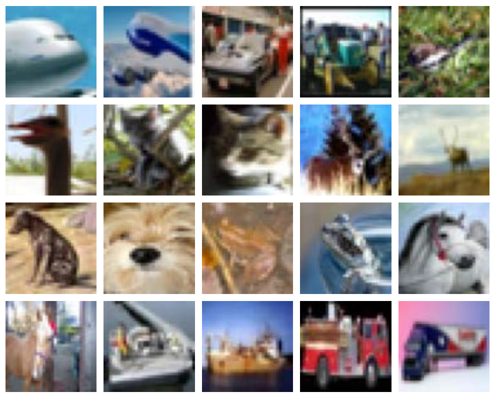
The keyword distribution can be found in Appendix 2.7.3.1. Inspecting this list reveals the importance of matching the sub-class distribution. For instance, the most common keyword in the `airplane` class is `stealth_bomber` and not a more common civilian type of airplane. In addition, the third most 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 [91]. 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 ±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


(a) Test set A

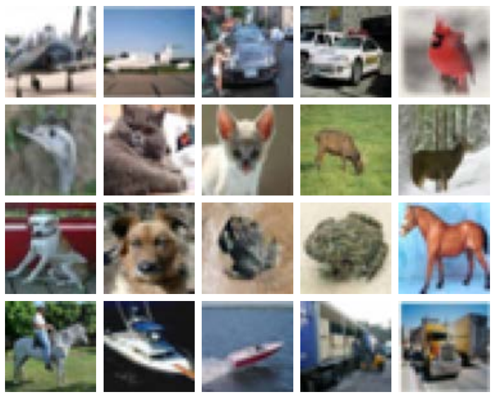

(b) Test set B

Figure 2.3: 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.[7]

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 2.7.3.1). 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 2.3b 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.

2.7.2 Follow-up Hypotheses

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

2.7.2.1 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.

2.7.2.2 Differences in Near-Duplicate Removal

As mentioned in Section 2.7.1, 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). Moreover, 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.

2.7.2.3 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 $\{5\times10^{-5}, 1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 5\times10^{-3}\}$.

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%).

2.7.2.4 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 2.4 in Appendix 2.7.3.5 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.

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

2.7.2.5 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.[8] 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 2.3 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.

2.7.2.6 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

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

	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 2.3: 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.

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[9]). 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.

2.7.2.7 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` and `wide_resnet_28_10` on each of the 5 new datasets we created. The accuracies are reported in Table 2.4. The accuracies on the cross-validation splits did not differ much from the accuracy on the original test set.

[9]This number is slightly lower than the accuracy of `vgg16_keras` on our new test set in Table 2.11, 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.

The variation among the cross-validation splits is significantly smaller than the drop on our new test set.

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

Table 2.4: 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.

2.7.2.8 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 2.5. Overall we found that the resulting models could not discriminate well between the original and our new test set: the best accuracy we obtained is 53.1%.

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 2.5: 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.

2.7.2.9 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 2.12 (Appendix 2.7.3.4). 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.

2.7.3 Additional Figures, Tables, and Lists

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

2.7.3.1 Keyword Distribution in CIFAR-10

The sub-tables in Table 2.6 show the keyword distribution for each of the ten classes in the original CIFAR-10 test set and our new test set.

Table 2.6: 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%

2.7.3.2 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` [23, 64] `https://github.com/tensorflow/models/tree/master/research/autoaugment`
2. `autoaug_shake_shake_112` [23, 51] `https://github.com/tensorflow/models/tree/master/research/autoaugment`
3. `autoaug_shake_shake_32` [23, 51] `https://github.com/tensorflow/models/tree/master/research/autoaugment`
4. `autoaug_shake_shake_96` [23, 51] `https://github.com/tensorflow/models/tree/master/research/autoaugment`
5. `autoaug_wrn` [23, 160] `https://github.com/tensorflow/models/tree/master/research/autoaugment`
6. `cudaconvnet` [92] `https://github.com/akrizhevsky/cuda-convnet2`
7. `darc` [87] `http://lis.csail.mit.edu/code/gdl.html`
8. `densenet_BC_100_12` [76] `https://github.com/hysts/pytorch_image_classification/`
9. `nas` [166] `https://github.com/tensorflow/models/blob/master/research/slim/nets/nasnet/nasnet.py#L32`
10. `pyramidnet_basic_110_270` [64] `https://github.com/hysts/pytorch_image_classification/`
11. `pyramidnet_basic_110_84` [64] `https://github.com/hysts/pytorch_image_classification/`
12. `random_features_256k_aug` [21] `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 [21] 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.

14. random_features_32k_aug [21] 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 [21] 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 [68] https://github.com/hysts/pytorch_image_classification/

17. resnet_basic_44 [68] https://github.com/hysts/pytorch_image_classification/

18. resnet_basic_56 [68] https://github.com/hysts/pytorch_image_classification/

19. resnet_basic_110 [68] https://github.com/hysts/pytorch_image_classification/

20. resnet_preact_basic_110 [69] https://github.com/hysts/pytorch_image_classification/

21. resnet_preact_bottleneck_164 [69] https://github.com/hysts/pytorch_image_classification/

22. resnet_preact_tf [69] https://github.com/tensorflow/models/tree/b871670b5ae29aaa6cad1b2d4e004882f716c466/resnet

23. resnext_29_4x64d [157] https://github.com/hysts/pytorch_image_classification/

24. resnext_29_8x64d [157] https://github.com/hysts/pytorch_image_classification/

25. shake_drop [158] https://github.com/imenurok/ShakeDrop

26. shake_shake_32d [51] https://github.com/hysts/pytorch_image_classification/
27. shake_shake_64d [51] https://github.com/hysts/pytorch_image_classification/
28. shake_shake_96d [51] https://github.com/hysts/pytorch_image_classification/
29. shake_shake_64d_cutout [33, 51] https://github.com/hysts/pytorch_image_classification/
30. vgg16_keras [104, 137] https://github.com/geifmany/cifar-vgg
31. vgg_15_BN_64 [104, 137] https://github.com/hysts/pytorch_image_classification/
32. wide_resnet_tf [160] https://github.com/tensorflow/models/tree/b871670b5ae29aaa6cad1b2d4e004882f716c466/resnet
33. wide_resnet_28_10 [160] https://github.com/hysts/pytorch_image_classification/
34. wide_resnet_28_10_cutout [33, 160] https://github.com/hysts/pytorch_image_classification/

2.7.3.3 Full Results Table

Table 2.11 contains the detailed accuracy scores for the original CIFAR-10 test set and our new test set.

2.7.3.4 Full Results Table for the Exactly Class-Balanced Test Set

Table 2.12 contains the detailed accuracy scores for the original CIFAR-10 test set and the exactly class-balanced variant of our new test set.

2.7.3.5 Hard Images

Figure 2.4 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.

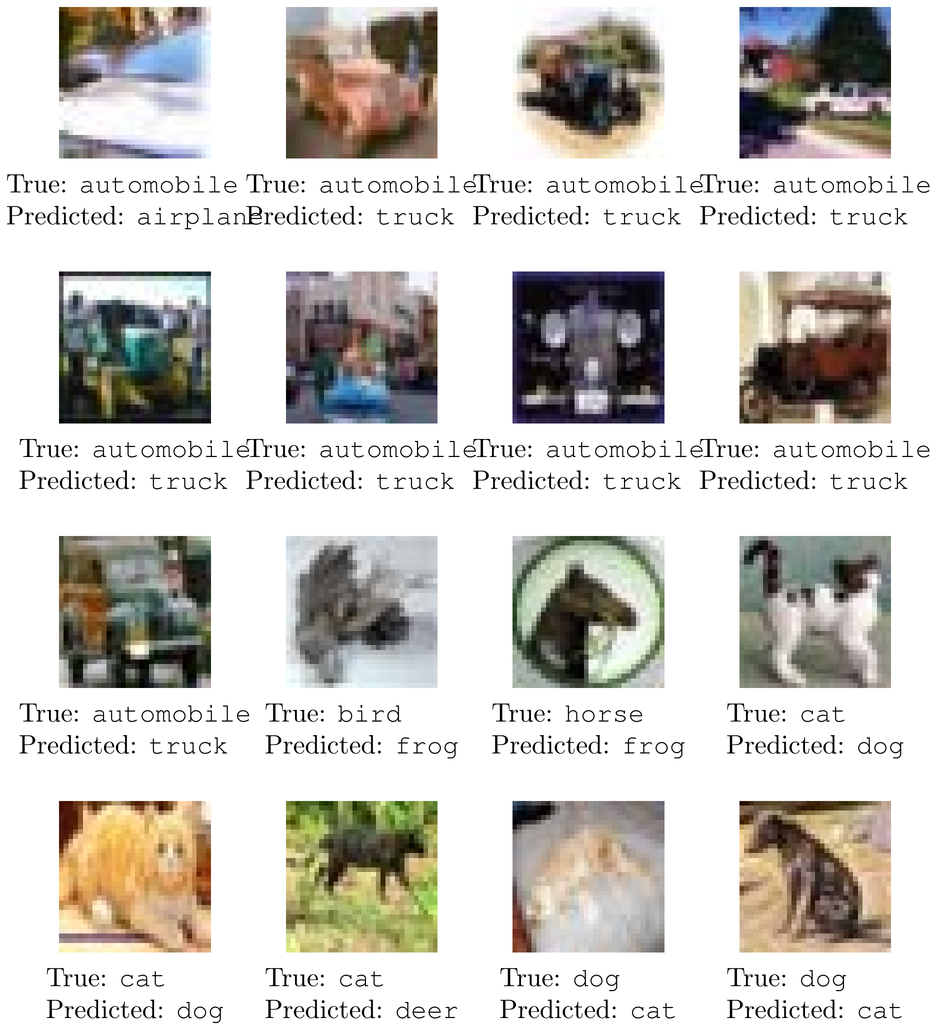


Figure 2.4: Hard images from our new test set that no model correctly. The caption of each image states the correct class label ("True") and the label predicted by most models ("Predicted").

Table 2.11: 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, ΔRank = −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 2.7.3.2.

CIFAR-10

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	autoaug_pyramid_net_	98.4 [98.1, 98.6]	95.5 [94.5, 96.4]	2.9	1	0
2	autoaug_shake_shake_	98.1 [97.8, 98.4]	93.9 [92.7, 94.9]	4.3	2	0
3	autoaug_shake_shake_	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_	97.3 [97.0, 97.6]	92.9 [91.7, 94.0]	4.4	6	-1
6	shake_shake_64d_cutc	97.1 [96.8, 97.4]	93.0 [91.8, 94.1]	4.1	5	1
7	shake_shake_26_2x96c	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_11(	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_11(	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_1(	95.0 [94.6, 95.4]	88.5 [87.0, 89.9]	6.5	19	1
21	resnet_v2_bottleneck	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
29	resnet_basic_32	92.5 [92.0, 93.0]	84.9 [83.2, 86.4]	7.7	26	3
31	random_features_256k	85.6 [84.9, 86.3]	73.1 [71.1, 75.1]	12.5	31	0
32	random_features_32k_	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

Table 2.12: 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, ΔRank = −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 2.7.3.2.

CIFAR-10						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	`autoaug_pyramid_net_`	98.4 [98.1, 98.6]	95.5 [94.5, 96.4]	2.9	1	0
2	`autoaug_shake_shake_`	98.1 [97.8, 98.4]	94.0 [92.9, 95.0]	4.1	2	0
3	`autoaug_shake_shake_`	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_`	97.3 [97.0, 97.6]	93.2 [92.0, 94.2]	4.2	4	1
6	`shake_shake_64d_cutc`	97.1 [96.8, 97.4]	93.1 [91.9, 94.2]	4.0	5	1
7	`shake_shake_26_2x96c`	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_11(`	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_11(`	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_1(`	95.0 [94.6, 95.4]	88.8 [87.4, 90.2]	6.2	19	1
21	`resnet_v2_bottleneck`	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
29	`resnet_basic_32`	92.5 [92.0, 93.0]	85.2 [83.6, 86.7]	7.3	25	4
31	`random_features_256k`	85.6 [84.9, 86.3]	73.6 [71.6, 75.5]	12.0	31	0
32	`random_features_32k_`	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

2.8 Appendix: 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 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$\times$ more training images than CIFAR-10 with roughly 100$\times$ 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 [109]. 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 [32, 132].

ImageNet. ImageNet [32, 132] 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 [112], 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 considered valid. Overall, the researchers employed over 49,000 workers from 167 countries [99].

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 [11].

2.8.1 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.

- 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 1.1.1). 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 2.5 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.

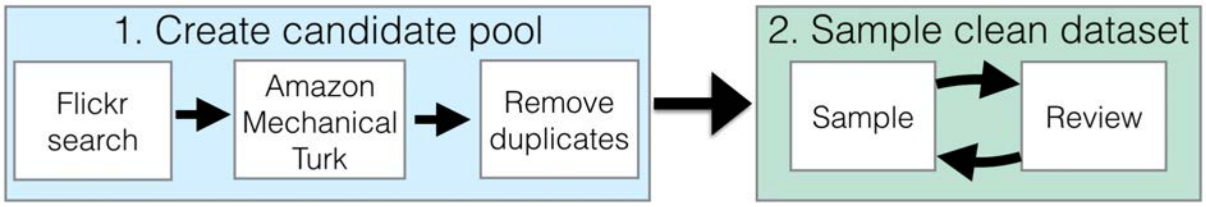


Figure 2.5: The pipeline for the new ImageNet test set. It consists of two parts: creating the candidate pool and sampling the final dataset from this candidate pool.

2.8.1.1 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 [32]).

- **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 [32, 132]. 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 2.8.4.1 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 [32]. 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 [137] model (fc7), and SSIM (structural similarity) [151], 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 [11]. 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.

2.8.1.2 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.[10]

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 2.3 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

[10]To be precise: Jia Deng's PhD thesis [31] 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.

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 2.8.4.2 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 2.8.3.2). 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.

2.8.2 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 2.8.4.3 contains a list of all models we evaluated with corresponding references and links to source code repositories. Tables 2.14 and 2.15 show the top-1 and top-5 accuracies for our main test set MatchedFrequency. Figure 2.12 visualizes the top-1 and top-5 accuracies on all three test sets.

In the main text of the paper and Figure 2.12, 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 2.6 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 2.6, the probit

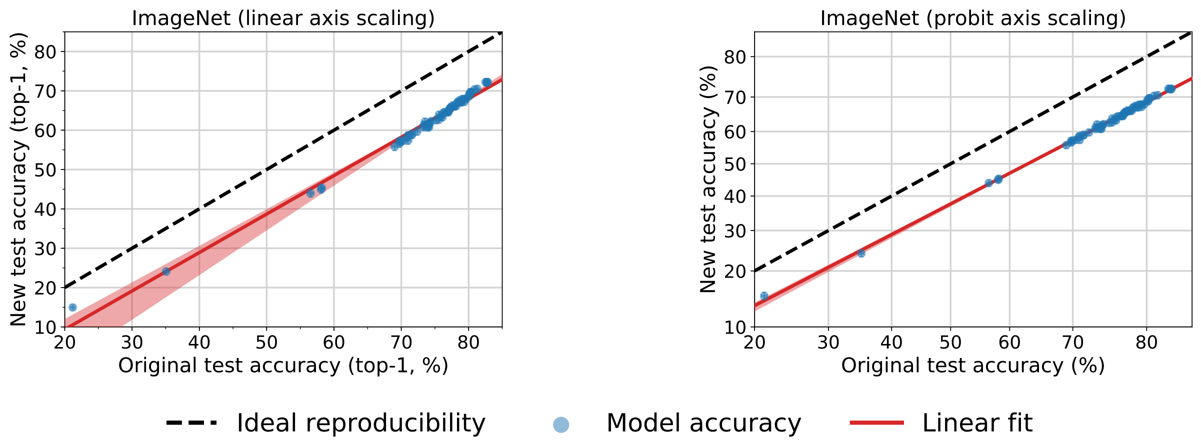


Figure 2.6: Model accuracy on the original ImageNet validation set vs. our new test set MatchedFrequency. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals), and we now also include the Fisher Vector models. The left plot shows the model accuracies 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.

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 2.12, we also show the top-1 and top-5 accuracies for all three datasets in the probit domain in Figure 2.13. Section 2.4.3 describes a possible generative model that leads to a linear fit in the probit domain as exhibited by the plots in Figures 2.6 and 2.13.

2.8.3 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.

2.8.3.1 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.[11] Moreover, recall that the ImageNet validation set is not a strictly i.i.d. sample from the same distribution as the training set (see the beginning of Section 2.3). 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[12] [68] for each of the five training sets and evaluated them on the corresponding held-out data. Table 2.13 shows the resulting accuracies for each split.

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 2.13: `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.

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.

[11]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.

[12]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 2.15.

2.8.3.2 Impact of Dataset Revisions

As mentioned in Section 2.8.1.2, 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 2.8.4.2) 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-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 2.7 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.

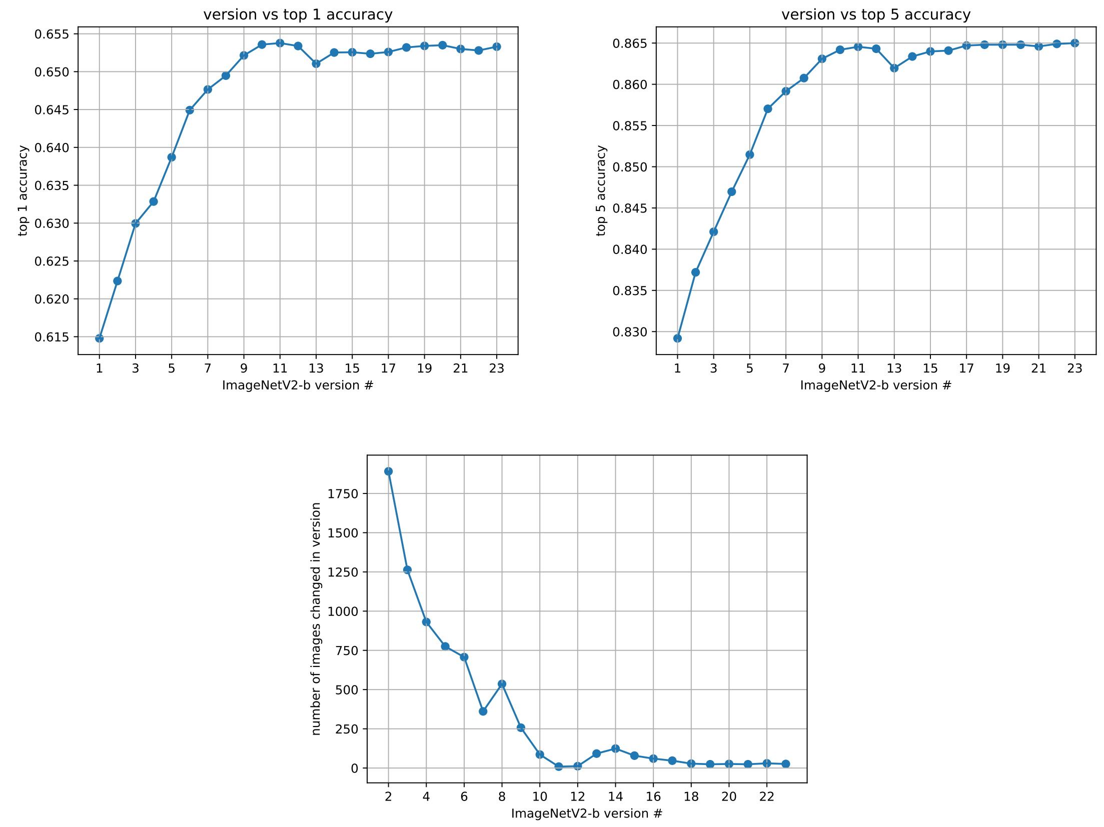


Figure 2.7: Impact of the reviewing passes on the accuracy of a `resnet152` on our new MatchedFrequency test set. The revision numbers correspond to the chronological ordering in which we created the dataset revisions

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 2.8, the resulting accuracies still show a good linear fit that is of similar quality as in Figure 2.12. This shows that the linear relationship between the test accuracies is not a result of our reviewing intervention.

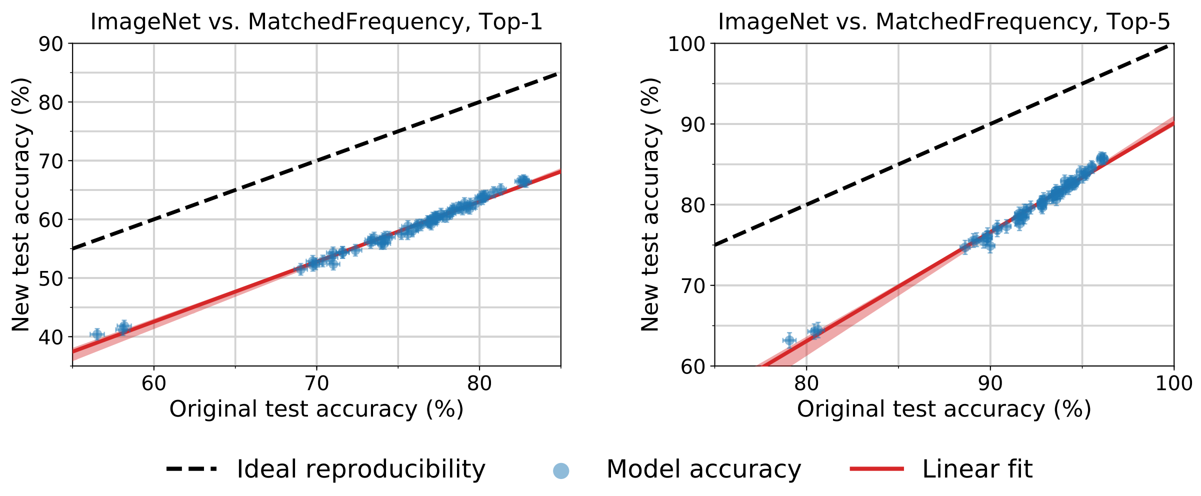


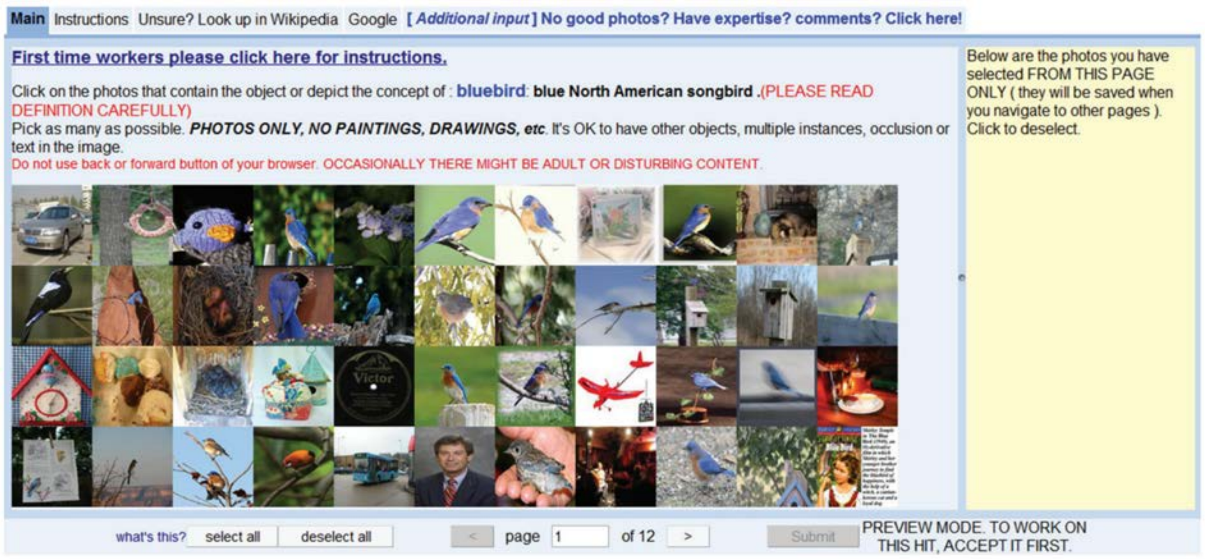
Figure 2.8: Model accuracy on the original ImageNet validation set vs. accuracy on *the first revision* of our MatchedFrequency test set. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). The red shaded region is a 95% confidence region for 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 2.12.

2.8.4 Additional Figures, Tables, and Lists

In this appendix we provide content pertaining to ImageNetV2 that did not fit into the preceding sections about our ImageNet experiments.

2.8.4.1 MTurk User Interfaces

For comparison, we include the original ImageNet MTurk user interface (UI) in Figure 2.9 and the MTurk UI we used in our experiments in Figure 2.10. 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).


Main
Instructions
Unsure? Look up in Wikipedia
Google
[Additional input] No good photos? Have expertise? comments? Click here!
First time workers please click here for instructions.
what's this?
select all
deselect all
page 1 of 12
Submit

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

2.8.4.2 User Interface for our Final Reviewing Process

Figure 2.11 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.

2.8.4.3 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` [92] `https://github.com/Cadene/pretrained-models.pytorch`
2. `bninception` [78] `https://github.com/Cadene/pretrained-models.pytorch`
3. `cafferesnet101` [68] `https://github.com/Cadene/pretrained-models.pytorch`
4. `densenet121` [76] `https://github.com/Cadene/pretrained-models.pytorch`
5. `densenet161` [76]`https://github.com/Cadene/pretrained-models.pytorch`
6. `densenet169` [76] `https://github.com/Cadene/pretrained-models.pytorch`
7. `densenet201` [76] `https://github.com/Cadene/pretrained-models.pytorch`
8. `dpn107` [17] `https://github.com/Cadene/pretrained-models.pytorch`
9. `dpn131` [17] `https://github.com/Cadene/pretrained-models.pytorch`
10. `dpn68b` [17]`https://github.com/Cadene/pretrained-models.pytorch`

11. `dpn68` [17] `https://github.com/Cadene/pretrained-models.pytorch`
12. `dpn92` [17] `https://github.com/Cadene/pretrained-models.pytorch`
13. `dpn98` [17] `https://github.com/Cadene/pretrained-models.pytorch`
14. `fbresnet152` [68] `https://github.com/tensorflow/models/tree/master/research/slim/`
15. `fv_4k` [19, 117] `https://github.com/modestyachts/nondeep` FisherVector model using SIFT, local color statistic features, and 16 GMM centers.
16. `fv_16k` [19, 117] `https://github.com/modestyachts/nondeep` FisherVector model using SIFT, local color statistic features, and 64 GMM centers.
17. `fv_64k` [19, 117] `https://github.com/modestyachts/nondeep` FisherVector model using SIFT, local color statistic features, and 256 GMM centers.
18. `inception_resnet_v2_tf` [143] `https://github.com/tensorflow/models/tree/master/research/slim/`
19. `inception_v1_tf` [141] `https://github.com/tensorflow/models/tree/master/research/slim/`
20. `inception_v2_tf` [78] `https://github.com/tensorflow/models/tree/master/research/slim/`
21. `inception_v3_tf` [142] `https://github.com/tensorflow/models/tree/master/research/slim/`
22. `inception_v3` [142] `https://github.com/Cadene/pretrained-models.pytorch`
23. `inception_v4_tf` [143] `https://github.com/tensorflow/models/tree/master/research/slim/`
24. `inceptionresnetv2` [78] `https://github.com/Cadene/pretrained-models.pytorch`

25. inceptionv3 [142] https://github.com/Cadene/pretrained-models.pytorch
26. inceptionv4 [143] https://github.com/Cadene/pretrained-models.pytorch
27. mobilenet_v1_tf [74] https://github.com/tensorflow/models/tree/master/research/slim/
28. nasnet_large_tf [166] https://github.com/tensorflow/models/tree/master/research/slim/
29. nasnet_mobile_tf [166] https://github.com/tensorflow/models/tree/master/research/slim/
30. nasnetalarge [166] https://github.com/Cadene/pretrained-models.pytorch
31. nasnetamobile [166] https://github.com/Cadene/pretrained-models.pytorch
32. pnasnet5large [103] https://github.com/Cadene/pretrained-models.pytorch
33. pnasnet_large_tf [103] https://github.com/tensorflow/models/tree/master/research/slim/
34. pnasnet_mobile_tf [103] https://github.com/tensorflow/models/tree/master/research/slim/
35. polynet [161] https://github.com/Cadene/pretrained-models.pytorch
36. resnet101 [68] https://github.com/Cadene/pretrained-models.pytorch
37. resnet152 [68] https://github.com/Cadene/pretrained-models.pytorch
38. resnet18 [68] https://github.com/Cadene/pretrained-models.pytorch

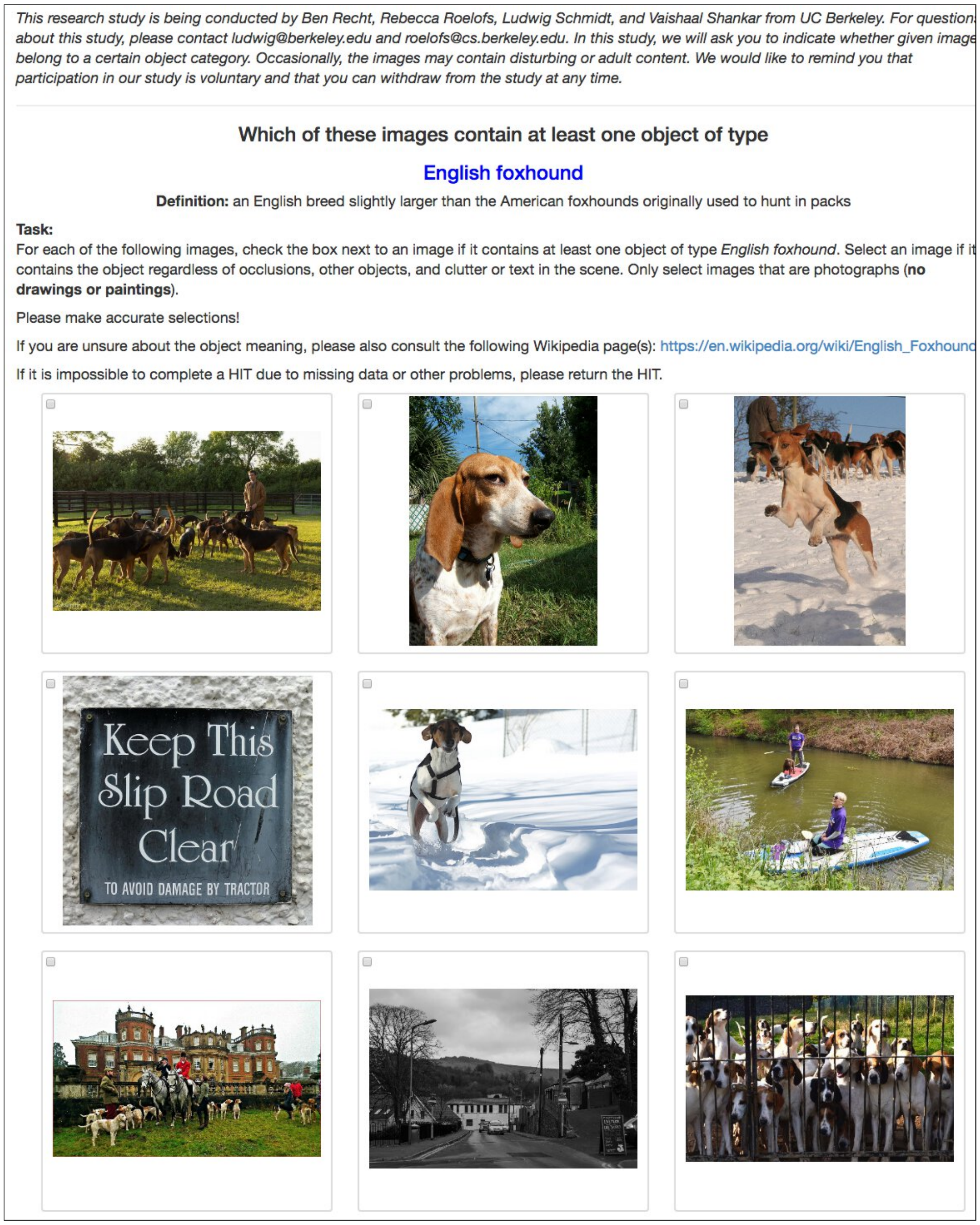
39. `resnet34` [68] `https://github.com/Cadene/pretrained-models.pytorch`
40. `resnet50` [68] `https://github.com/Cadene/pretrained-models.pytorch`
41. `resnet_v1_101_tf` [68] `https://github.com/tensorflow/models/tree/master/research/slim/`
42. `resnet_v1_152_tf` [68] `https://github.com/tensorflow/models/tree/master/research/slim/`
43. `resnet_v1_50_tf` [68] `https://github.com/tensorflow/models/tree/master/research/slim/`
44. `resnet_v2_101_tf` [69] `https://github.com/tensorflow/models/tree/master/research/slim/`
45. `resnet_v2_152_tf` [69] `https://github.com/tensorflow/models/tree/master/research/slim/`
46. `resnet_v2_50_tf` [69] `https://github.com/tensorflow/models/tree/master/research/slim/`
47. `resnext101_32x4d` [157] `https://github.com/Cadene/pretrained-models.pytorch`
48. `resnext101_64x4d` [157] `https://github.com/Cadene/pretrained-models.pytorch`
49. `se_resnet101` [75] `https://github.com/Cadene/pretrained-models.pytorch`
50. `se_resnet152` [75] `https://github.com/Cadene/pretrained-models.pytorch`
51. `se_resnet50` [75] `https://github.com/Cadene/pretrained-models.pytorch`
52. `se_resnext101_32x4d` [75] `https://github.com/Cadene/pretrained-models.pytorch`

53. se_resnext50_32x4d [75] https://github.com/Cadene/pretrained-models.pytorch
54. senet154 [75] https://github.com/Cadene/pretrained-models.pytorch
55. squeezenet1_0 [77] https://github.com/Cadene/pretrained-models.pytorch
56. squeezenet1_1 [77] https://github.com/Cadene/pretrained-models.pytorch
57. vgg11_bn [78] https://github.com/Cadene/pretrained-models.pytorch
58. vgg11 [137] https://github.com/Cadene/pretrained-models.pytorch
59. vgg13_bn [78] https://github.com/Cadene/pretrained-models.pytorch
60. vgg13 [137] https://github.com/Cadene/pretrained-models.pytorch
61. vgg16_bn [78] https://github.com/Cadene/pretrained-models.pytorch
62. vgg16 [137] https://github.com/Cadene/pretrained-models.pytorch
63. vgg19_bn [78] https://github.com/Cadene/pretrained-models.pytorch
64. vgg19 [137] https://github.com/Cadene/pretrained-models.pytorch
65. vgg_16_tf [137] https://github.com/tensorflow/models/tree/master/research/slim/
66. vgg_19_tf [137] https://github.com/tensorflow/models/tree/master/research/slim/

67. `xception` [18] `https://github.com/Cadene/pretrained-models.pytorch`

2.8.4.4 Full Results Tables

Tables 2.14 and 2.15 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 2.16 – 2.19 contain the accuracy scores for our Threshold0.7 and TopImages test sets.



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

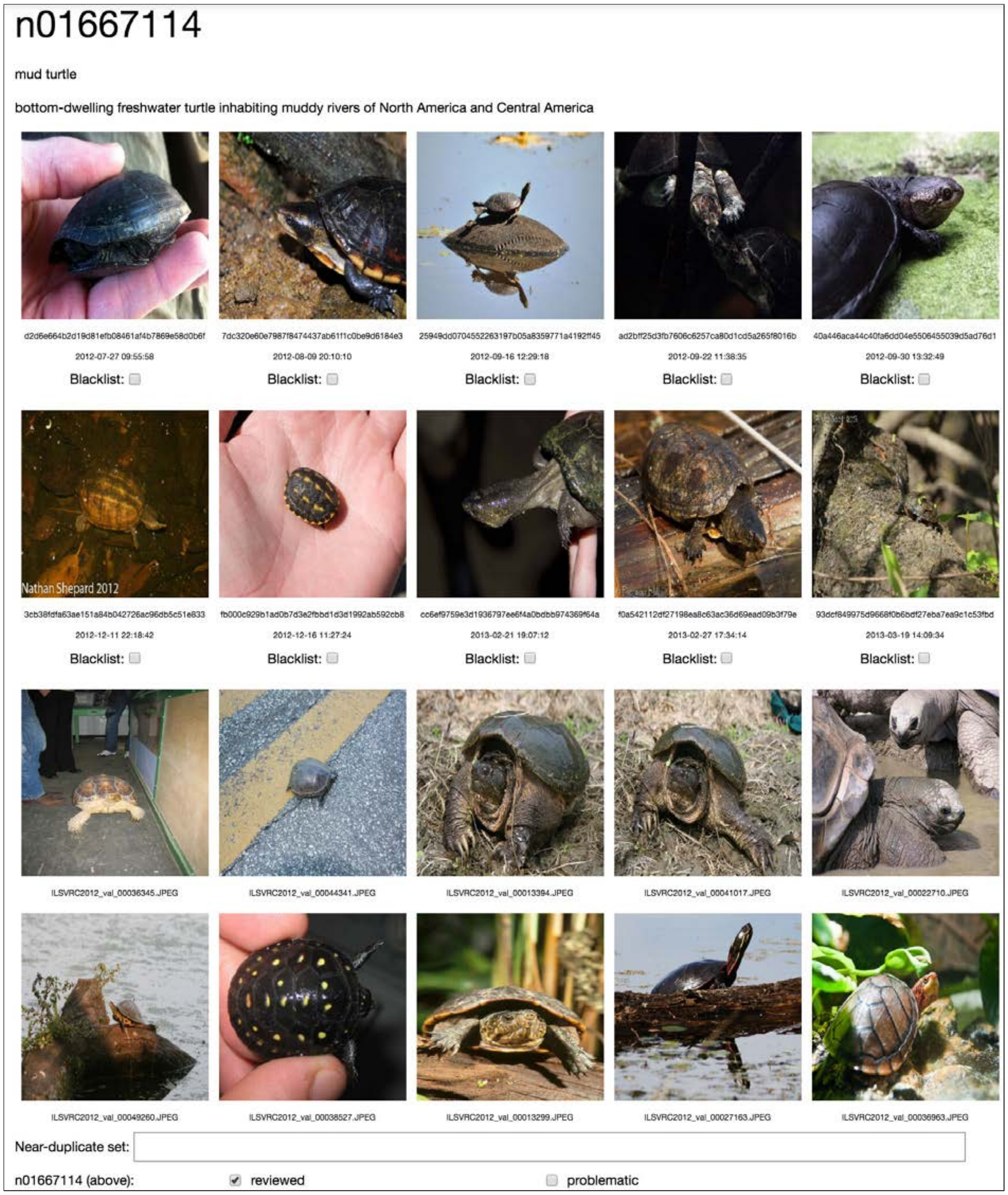
Task:
For each of the following images, check the box next to an image if it contains at least one object of type *English foxhound*. Select an image if it contains the object regardless of occlusions, other objects, and clutter or text in the scene. Only select images that are photographs (**no drawings 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 2.10: 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.



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 2.11: 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.

Table 2.14: Top-1 model accuracy on the original ImageNet validation set and our new test set MatchedFrequency.

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_`	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 2.15: Top-5 model accuracy on the original ImageNet validation set and our new test set MatchedFrequency.

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_	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 2.16: Top-1 model accuracy on the original ImageNet validation set and our new test set Threshold0.7.

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_`	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 2.17: Top-5 model accuracy on the original ImageNet validation set and our new test set Threshold0.7.

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_	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 2.18: Top-1 model accuracy on the original ImageNet validation set and our new test set TopImages.

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_	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 2.19: Top-5 model accuracy on the original ImageNet validation set and our new test set TopImages.

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_	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

2.8.4.5 Accuracy Plots for All ImageNet Test Sets

Figure 2.12 shows the top-1 and top-5 accuracies for our three test sets and all convolutional networks in our model testbed. Figure 2.13 shows the accuracies for all models (including Fisher Vector models) with a probit scale on the axes.

2.8.4.6 Example Images

Figure 2.14 shows randomly selected images for three randomly selected classes for both the original ImageNet validation set and our three new test sets.

2.8.4.7 Effect of Selection Frequency on Model Accuracy

To better understand how the selection frequency of an image impacts the model accuracies, Figures 2.15, 2.16, and 2.17 show model accuracies stratified into five selection frequency bins.

2.8.4.8 Ambiguous Class Examples

Figure **??** 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)”

[12]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.

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”

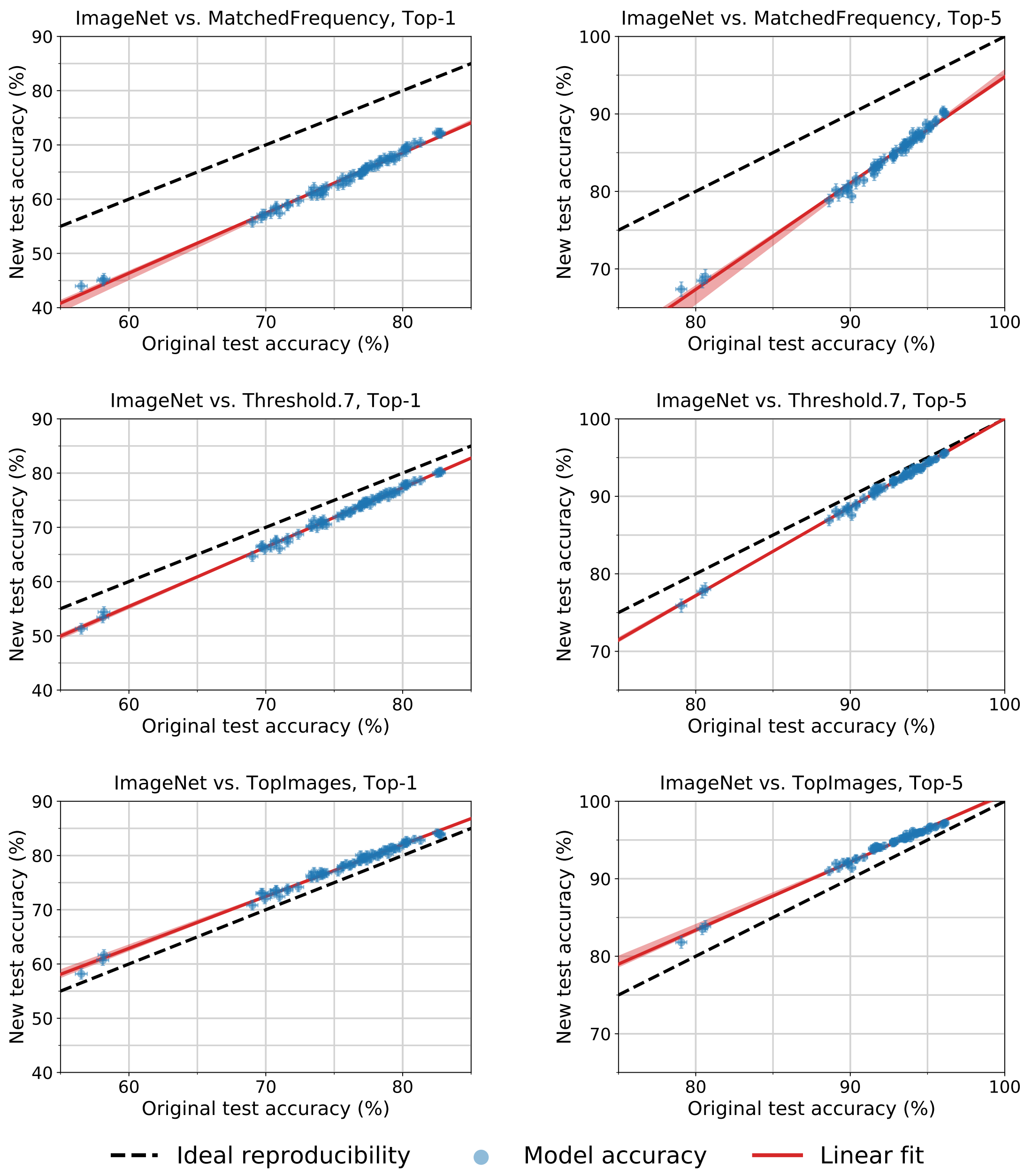


Figure 2.12: Model accuracy on the original ImageNet validation set vs. our new test sets. See Section 2.3 for a description of these test sets. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). The red shaded region is a 95% confidence region 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.

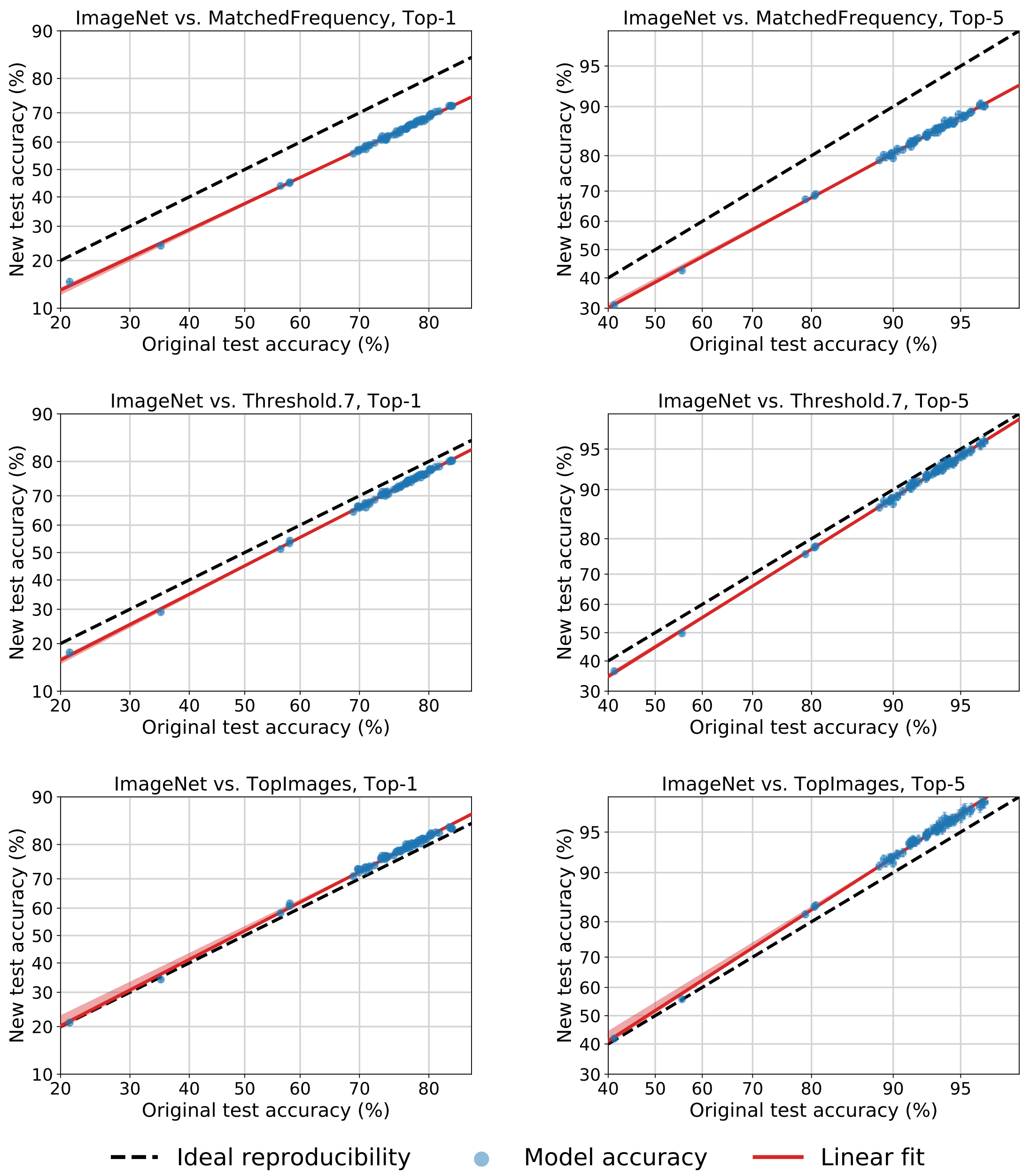


Figure 2.13: Model accuracy on the original ImageNet validation set vs. our new test sets. The structure of the plots is similar to Figure 2.12 and we refer the reader to the description there. In contrast to Figure 2.12, the plots here contain also the Fisher Vector models. Moreover, the axes are scaled 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 2.12, 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 2.14: Randomly selected images from the original ImageNet validation set and our new ImageNet test sets.

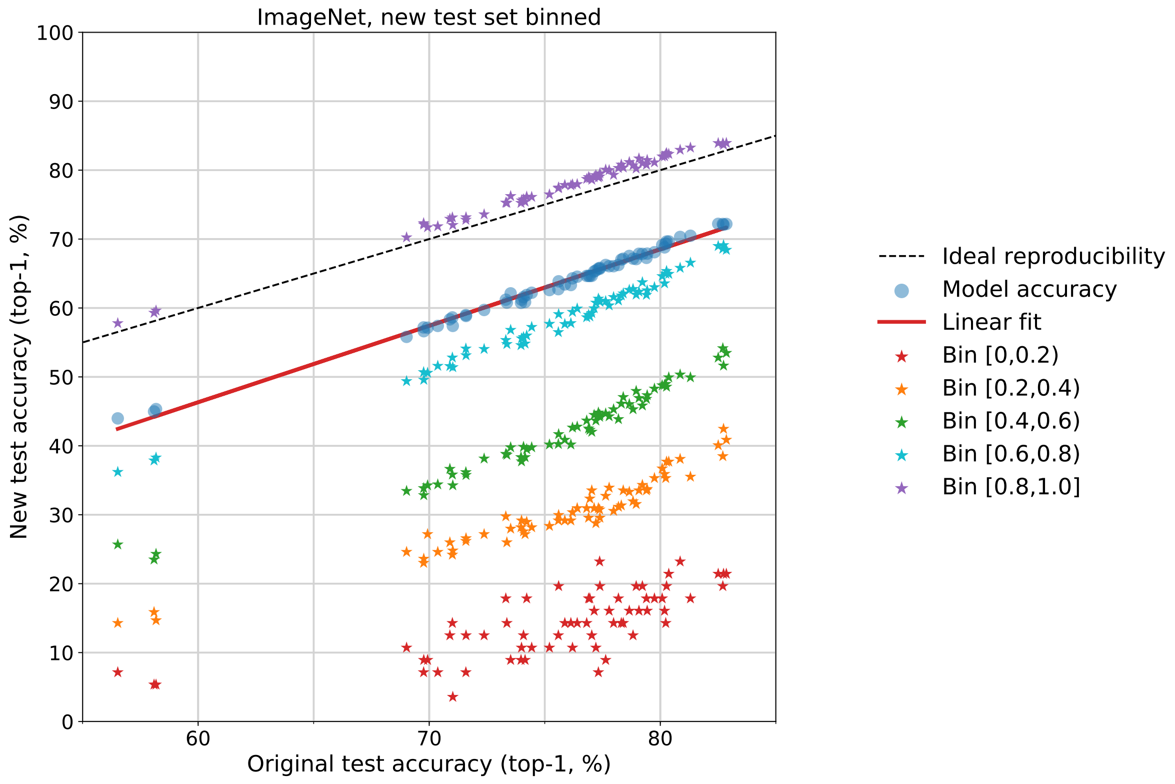


Figure 2.15: Model accuracy on the original ImageNet validation set vs. accuracy on our new test set MatchedFrequency, stratified into five selection frequency bins. Every bin contains the images with MTurk selection frequency falling into the corresponding range. Each data point corresponds to one model 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.

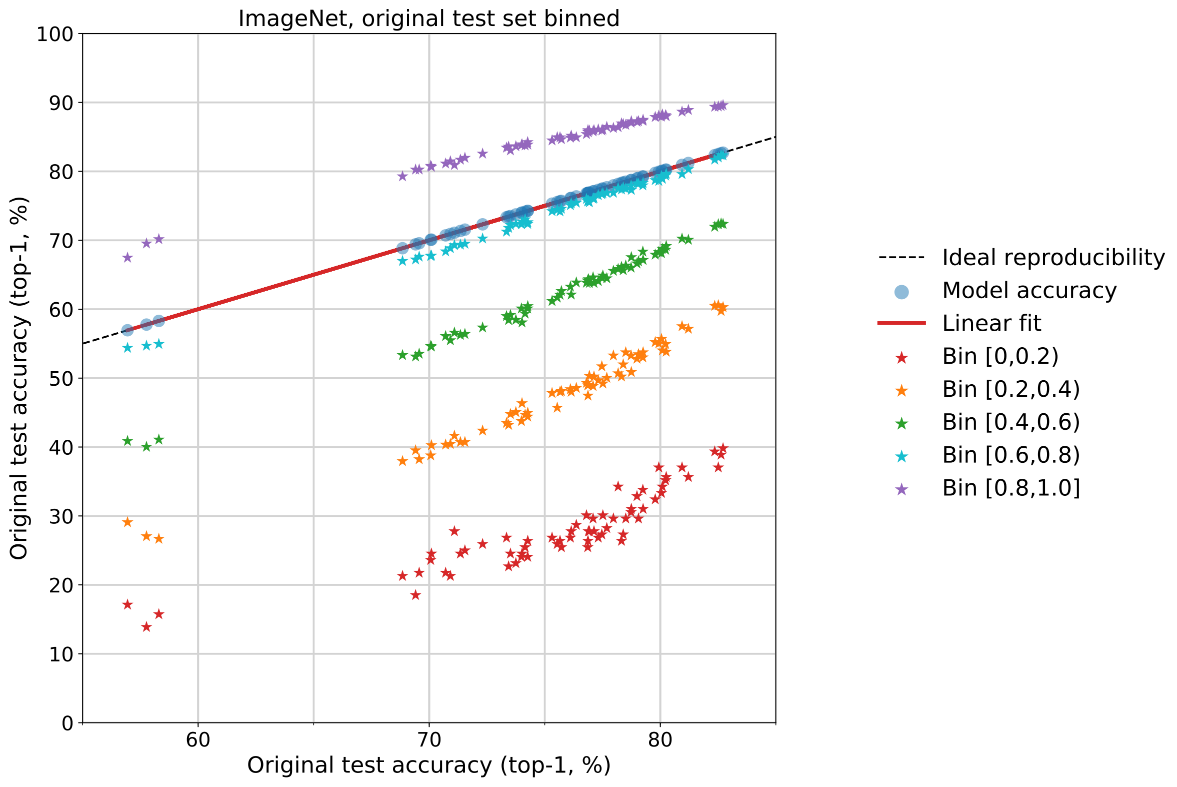


Figure 2.16: Model accuracy on the original ImageNet validation set stratified into five selection frequency bins. This plot has a similar structure as Figure 2.15 above, but contains the original validation set accuracy on both axes (as before, the images are binned on the y-axis and not binned on the 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.

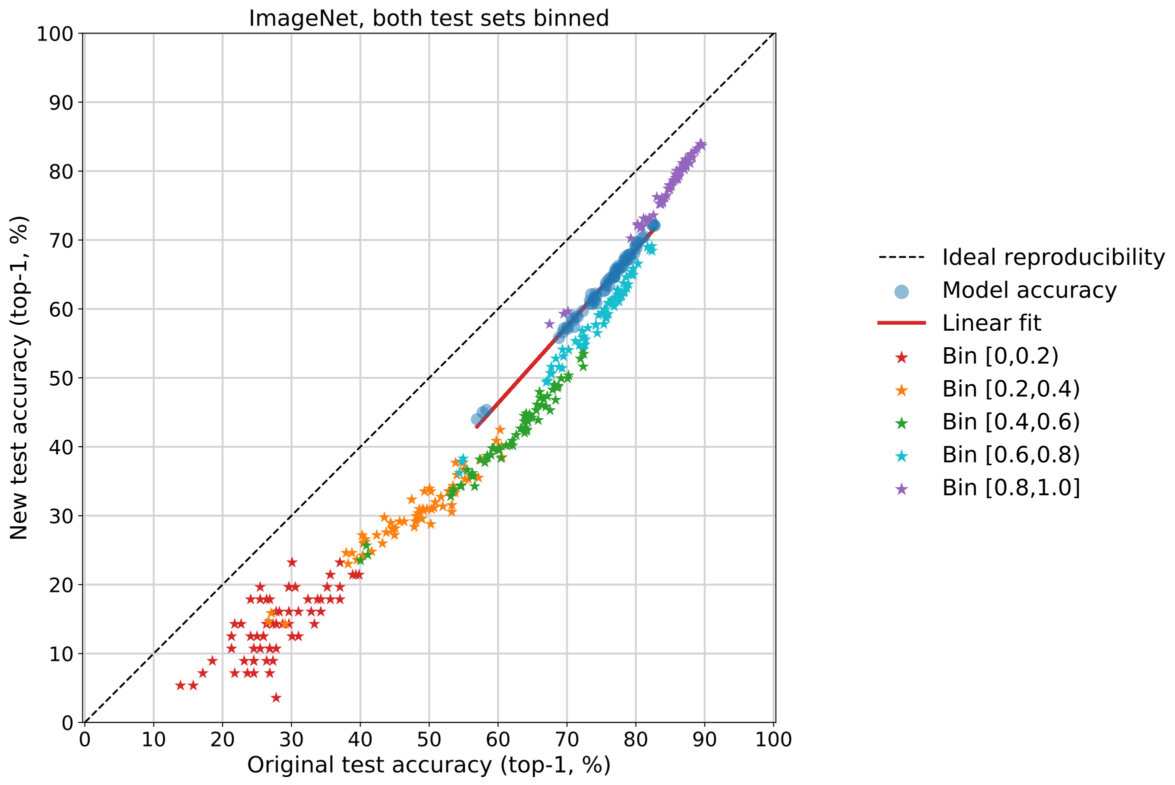


Figure 2.17: Model accuracy on the original ImageNet validation set vs. accuracy on our new test set MatchedFrequency. In contrast to the preceding Figures 2.15 and 2.16, both original and new test accuracy is now stratified into five selection frequency bins. Each data point corresponds to the accuracy 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).

Chapter 3

Human Accuracy on ImageNetV2

3.1 Introduction

In this chapter, we take a step towards a more comprehensive understanding of machine performance relative to human generalization capabilities. We focus on the ImageNet dataset since it has been a key benchmark in the past decade of machine learning and has a widely cited human vs. machine comparison [2, 66, 131]. The core part of our study is an extensive experimental comparison of human and machine behavior on ImageNet not only in terms of absolute accuracy, but also in terms of robustness to small distribution shifts. As the ultimate goal of a classification model is to process images beyond the benchmark test set, robustness is a crucial property of a trained model. We hope that our multi-dimensional approach will lead to improved evaluation methodology for trained classifiers more broadly. In addition, our results provide context for earlier claims about super-human performance on ImageNet.

To enable a thorough and semantically coherent evaluation of machine robustness, our experiment addresses the following questions concerning classification accuracy on ImageNet:

- *What is a meaningful evaluation metric for ImageNet?* Initially, most ImageNet models were evaluated using the `top-5` metric, which was also employed in the aforementioned human vs. machine comparison [131]. Since then, the field has moved to measuring `top-1` accuracy, which leads to a harder task and may affect the comparison of models and humans. However, a non-trivial fraction of images in ImageNet has more than one correct label, which makes the `top-1` metric overly stringent.

To address this issue, we re-annotate 40,000 images with multi-label annotations specific to each image.

- *How widely do trained humans vary on ImageNet?* The 2014 study only compared CNNs against two humans, and only one human was evaluated on more than 300 images [131]. As with many tasks, there may be substantial variation in human behavior on ImageNet as well.

  To gain a broader picture, we evaluate five trained labelers on 2,000 images each.

- *How robust are humans and models to small distribution shifts?* The 2014 study was conducted only on the ImageNet test set, which is drawn from the same distribution as the ImageNet training set. This leaves out important aspects of human generalization as humans reliably recognize objects in a variety of scenarios.

  To measure robustness, we utilize two separate test sets and evaluate humans and models on both.

Our main result is that robustness to small, naturally occurring distribution shifts is a performance dimension on which humans are still substantially better than all ImageNet classifiers. To establish this fact, we compare humans and machines not only on the standard ImageNet test set,[1] but also on the test set from the ImageNetV2 replication study [127].[2] The authors of ImageNetV2 followed the ImageNet creation process to construct a new test set that is close to the original, e.g., they used the same data source (Flickr) and the same data cleaning process (Mechanical Turk). Nevertheless, all current ImageNet models have substantially lower accuracy on the ImageNetV2 test set.

We find that this gap in model performance persists even after we re-labeled both test sets consistently with multi-label annotations. In contrast, all of our five human labelers see at most a 1% difference between the two datasets. Moreover, our five labelers show substantially less variation in robustness than in classification accuracy. These findings demonstrate that robustness to small distribution shift is a performance dimension not captured by most current benchmark evaluations, although humans still substantially outperform trained classifiers on this metric.

[1]We use the terms "test set" and "validation set" interchangeably in the context of ImageNet so that we can use the same term for both ImageNet and ImageNetV2. While ImageNet has distinct validation and test sets, the labels for the test set were never released and most papers report scores on the validation set.

[2]Specifically, we utilized the MatchedFrequency test set from Recht et al. [127]. For conciseness, we simply refer to this test set as ImageNetV2.

Table 3.1 summarizes the outcome of our experiment. On both test sets, there is substantial variation among humans: the gap between the highest and lowest accuracy achieved by a human is 5.4% on the original test set and 5.6% on the new test set. On the original dataset, three of our five labelers outperform the currently best published ImageNet model, with the median human being 0.2% more accurate than the best model. On the ImageNetV2 test set, all five labelers outperform all trained models, and the median human is 5.2% better than the best model.[3] Importantly, humans see almost no drop between the two test sets, while all models suffer a substantial accuracy difference. This trend is also visualized in Figure 3.1, which shows the accuracies of 71 ImageNet models and five human labelers. All humans are close to the $y = x$ diagonal, while the models follow a linear trend substantially below this diagonal.

Table 3.1: Human and model multi-label accuracy on the ImageNet validation dataset and ImageNetV2. The gap is measured as multi-label accuracy on ImageNet validation minus multi-label accuracy on ImageNetV2. The confidence intervals are 95% Clopper-Pearson intervals. The AdvProp model [156] is an EfficentNet-B8 trained with AdvProp and the FixRes model [107, 147] is a ResNext-32x48d trained on one billion images from Instagram.

ImageNet multi-label accuracy (%)			
Participant	Original	V2	**Gap**
`resnet50`	84.2 [81.8, 86.4]	75.7 [73.2, 78.7]	**8.4**
`AdvProp`	93.6 [91.9, 95.0]	88.3 [86.5, 90.6]	**5.3**
`FixRes`	95.5 [94.0, 96.7]	89.6 [87.9, 91.8]	**5.9**
Human A	91.9 [90.0, 93.5]	91.1 [89.6, 93.2]	**0.8**
Human B	94.7 [93.1, 96.0]	93.9 [92.6, 95.6]	**0.8**
Human C	96.2 [94.9, 97.3]	96.7 [95.9, 98.1]	**-0.5**
Human D	95.7 [94.3, 96.9]	94.8 [93.7, 96.4]	**0.9**
Human E	97.3 [96.0, 98.2]	96.5 [95.6, 97.9]	**0.7**

While the majority of our labelers outperforms the currently best published ImageNet model, we emphasize that we do not see our numbers as a definite human baseline in terms of absolute accuracy. First, we believe that more thoroughly trained humans will achieve higher accuracy than the best humans in our evaluation. Most human errors are currently in fine-grained class distinctions among the animal classes and particularly dog

[3]Here we only describe the point estimates. Section 3.5 contains a discussion of statistical significance.

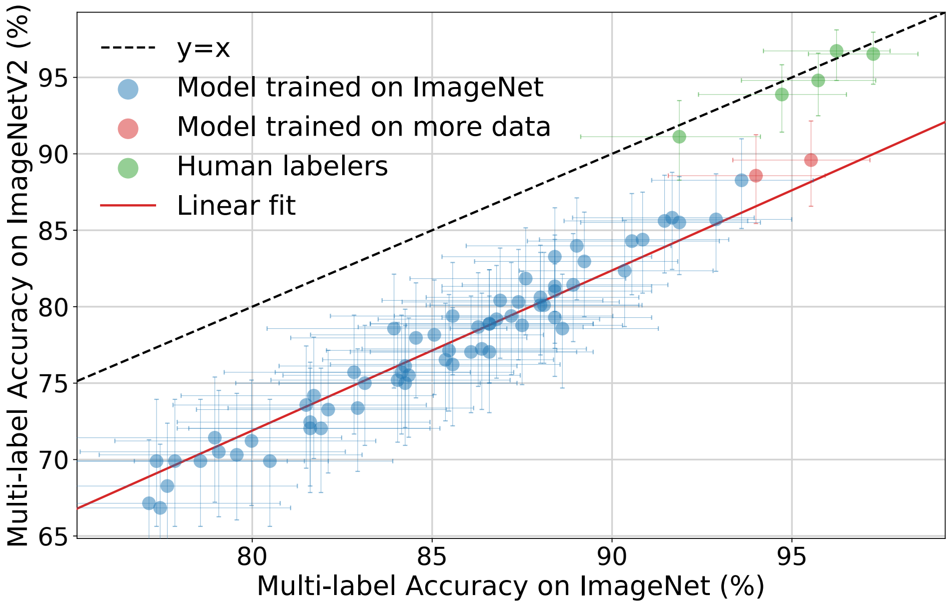


Figure 3.1: Multi-label accuracies for both CNN models and human participants on the ImageNet validation set versus their accuracies on the ImageNetV2 test set. The confidence intervals are 95% Clopper-Pearson confidence intervals.

breeds, while our best labelers already achieve more than 99.5% accuracy on the object classes (substantially outperforming the best models on this subset).

More importantly, we view robustness to small, naturally occuring distribution shift as the main metric of interest in our evaluation. Humans show substantially less variation in robustness than in accuracy, and the past decade of model development has led to little progress in this metric. We believe that addressing this gap is an important research direction for reliable machine learning.

3.2 Experiment setup

We conducted our experiment in four phases: (i) initial multi-label annotation, (ii) human labeler training, (iii) human labeler evaluation, and (iv) final annotation review. Figure 3.2 provides a detailed timeline of the experiment. In total, five human labelers participated in the experiment, denoted A through E. All five participants are anonymized authors of this

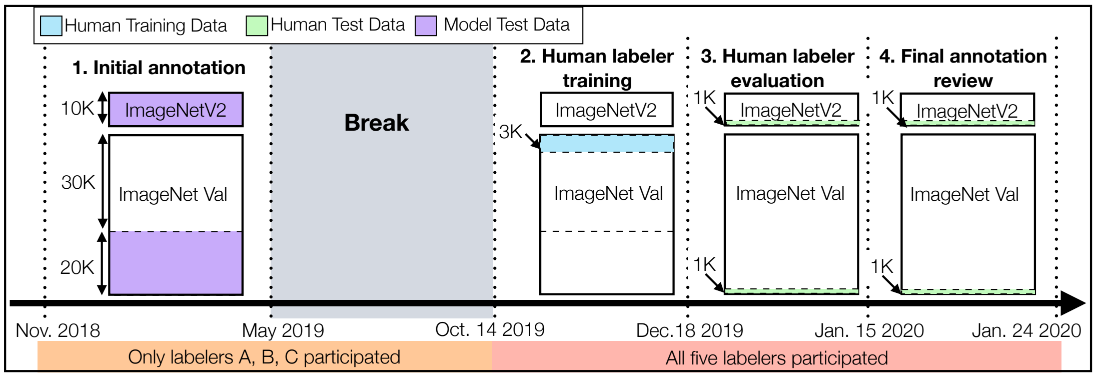


Figure 3.2: Timeline for the four phases of our experiment. 1) *Initial multi-label annotation:* First, starting in November 2018, human labelers A, B, and C annotated a set of images from ImageNetV2 and the ImageNet validation set with multi-label annotations. 2) *Human labeler training:* Then, after a long break, on October 14, 2019, all five participants began training on a 3,000 image subset of the original ImageNet validation set. (Humans were *not* trained on ImageNetV2.) 3) *Human labeler evaluation:* Next, starting on December 18, 2019, humans labeled 2,000 images from a random, class-balanced sample including 1,000 images from the ImageNet validation dataset and 1,000 images from ImageNetV2. The evaluation dataset did not include any of the images used in training. 4) *Final annotation review:* Finally, all five labelers reviewed the annotations collected for the 2,000 image evaluation dataset.

manuscript. While evaluating more humans would have provided additional information, the scale of the experiment made it difficult to incentivize others to invest the time and effort required to familiarize themselves with the 1,000 ImageNet classes and to label thousands of images.

In detail, the four phases of the experiment were:

1. Initial multi-label annotation. Labelers A, B, and C provided *multi-label* annotations for a subset of size 20,000 from the ImageNet validation set and 20,683 images from all three ImageNetV2 test sets collected by Recht et al. [127]. At this point, labelers A, B, and C already had extensive experience with the ImageNet dataset. We further discuss the annotation process in Section 3.3.

2. Human labeler training. Using a subset of the remaining 30,000 unannotated images in the ImageNet validation set, labelers A, B, C, D, and E underwent extensive training to understand the intricacies of fine-grained class distinctions in the ImageNet class hierarchy. The exact training process is detailed in Section 3.4.

3. Human labeler evaluation. For the human labeler evaluation, we generated a class-balanced random sample containing 1,000 images from the 20,000 annotated images of the ImageNet validation set and 1,000 images from ImageNetV2. We combined the two sets and randomly shuffled the resulting 2,000 images. Then, the five participants labeled these images over the course of 28 days.
4. Final annotation review. Lastly, all labelers reviewed the additional annotations generated in the human labeler evaluation phase. We discuss the main results from our evaluation in Section 3.5.

3.3 Multi-label annotations

In this section, we describe the details of the multi-label annotation process for the ImageNet validation dataset and ImageNetV2. We first explain why multi-label annotations are necessary for proper accuracy evaluation on ImageNet by outlining the pitfalls of the two most widely used accuracy metrics, `top-1` and `top-5`.

`Top-1` accuracy. `Top-1` accuracy is the standard accuracy measure used in the classification literature. It measures the proportion of examples for which the predicted label matches the single target label. However, the assumption that each image has a single ground truth label from a fixed set of classes is often incorrect. ImageNet images, such as Figure 3.3a, often contain multiple objects belonging to different classes (e.g. `desk`, `laptop`, `keyboard`, `space bar`, `screen`, and `mouse` frequently all appear in the same image). Moreover, even for images for which a class is prominent the ImageNet label might refer to another class present in the image. For example, in Figure 3.3b the class `gown` is central and appears in the foreground, but the ImageNet label is `picket fence`. As a result, one is not guaranteed to achieve high `top-1` accuracy by identifying the main objects in images. In other words, `top-1` accuracy can be overly stringent by penalizing predictions that appear in the image but do not correspond to the target label.

`Top-5` accuracy. To partially remedy issues with `top-1`, the organizers of the ImageNet challenge [131] measured `top-5` accuracy, which considers a classification correct if *any* of the five predictions matches the target label. However, allowing five guesses on *all* images on fine-grained classification tasks such as ImageNet can make certain class distinctions trivial. For example, there are five turtles in the ImageNet class hierarchy (`mud turtle`, `box turtle`, `loggerhead turtle`, `leatherback turtle`, and `terrapin`), which can be difficult to distinguish, but given an image of a turtle, a classifier can guess all five turtle classes to ensure that it predicts the correct label.

Multi-label accuracy. For multi-label accuracy, every image has a set of target labels and a prediction is marked correct if it corresponds to *any* of the target labels for that image. Due to the limitations of `top-1` and `top-5` accuracy, as well as ambiguity in the target class for many images, multi-label annotations are necessary for rigorous accuracy evaluation on ImageNet.

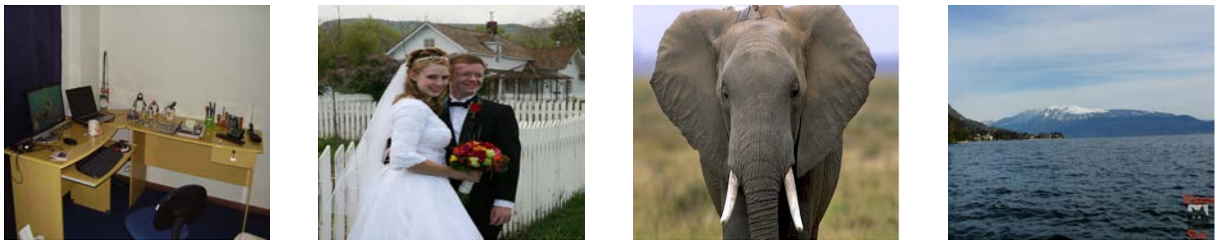
3.3.1 Types of multi-label annotations

Next, we discuss three categories of multi-label annotations that arose in our study, exemplified in Figure 3.3.

Multiple objects or organisms. For images that contain multiple objects or organisms corresponding to classes in the ImageNet hierarchy, we added an additional target label for each entity in the scene. For example, Figure 3.3a shows an image with target label `desk` that also contains multiple different objects corresponding to ImageNet classes. When there are multiple correct objects or organisms, the target class does not always correspond to the most central or largest entity in the scene. For example, in Figure 3.3b, the target class `picket fence` appears in the background of the image, but classes `groom`, `bow tie`, `suit`, `gown`, and `hoopskirt` all appear in the foreground.

Synonym or subset relationships. If two classes are synonyms of each other, or a class is a subset of another class, we considered both classes to be correct target labels. For example, the ImageNet class `tusker` is defined as any animal with visible tusks. Since `warthog`, `African elephant` and `Indian elephant` all have prominent tusks, these classes are all technically subsets of `tusker`. Figure 3.3c shows an `African elephant` that additionally has `tusker` as a correct label.

Unclear images. In certain cases, we could not ascertain whether a label was correct due to ambiguities in the image or in the class hierarchy. Figure 3.3d shows a scene which could arguably be either a `lakeshore` or a `seashore`.



(a) `desk` **(b)** `picket fence` **(c)** `African elephant` **(d)** `lakeshore`

Figure 3.3: Examples from the ImageNet validation of scenarios where multi-label annotations are necessary. *Multiple objects or organisms:* In Figure 3.3a, the ImageNet label is `desk` but `screen`, `monitor`, `coffee mug` and many more objects in the scene could count as correct labels. Figure 3.3b shows a scene where the target label `picket fence` is counterintuitive because it appears in the background of the image while classes `groom`, `bowtie`, `suit`, `gown`, and possibly `hoopskirt` are more prominently displayed in the foreground. b) *Synonym or subset relationships:* This image has ImageNet label `African elephant`, but can be labeled `tusker` as well, because every `African elephant` with tusks is a `tusker`. c) *Unclear images:* This image is labeled `lakeshore`, but could also be labeled `seashore` as there is not enough information in the scene to distinguish the water body between a lake or sea.

3.3.2 Collecting multi-label annotations

Next, we detail the process we used to collect multi-label annotations. We first collected the `top-1` predictions of 71 pre-trained ImageNet models published from 2012 to 2018. Then, over a period of three months, participants A, B and C reviewed all predictions made by the models on 40,683 images from ImageNet and ImageNetV2. Participants first researched class distinctions extensively – the details of this research are covered in 3.4. The three participants then categorized *every* unique prediction made by the 71 models on the 40,683 images (a total of 182,597 unique predictions) into `correct` or `incorrect`, thereby allowing each image to have multiple `correct` labels.

In total, we found that 18.2% of the ImageNet validation images have more than one correct label. Among images with multiple correct labels, the mean number of correct labels per image is 2.3.

The multi-label accuracy metric. Multi-label accuracy is computed by counting a prediction as correct if and only if it was marked correct by the expert reviewers during

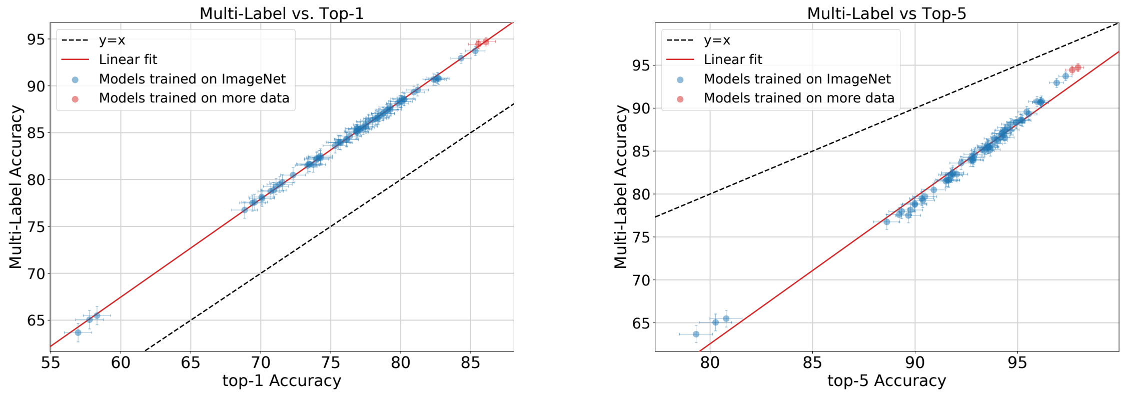


Figure 3.4: The relationship between `top-1`, `top-5`, and multi-label accuracy on ImageNet test for all 71 models in our test bed. The left figure plots multi-label vs. `top-1` accuracy accuracy. Multi-label accuracy makes the task easier than `top-1` accuracy, with a median improvement of 8.9% between `top-1` and multi-label scores. The right figure plots multi-label vs. `top-5` accuracy accuracy. Multi-label accuracy is more stringent than `top-5` accuracy, with a median drop of 7.4% between `top-5` and multi-label scores.

the annotation stage. We note that we performed a second annotation stage after the human labelers completed the experiment, as explained in Section 3.4.3.

In Figure 3.4, we plot each model's `top-5` and `top-1` accuracy versus its multi-label accuracy. *Every* model prediction was reviewed individually for correctness. Higher `top-1` and `top-5` accuracy correspond to higher multi-label accuracy with relatively few changes in model rankings across the different metrics. However, for all models, `top-1` accuracy underestimates multi-label accuracy (models see a median improvement of 8.9% when comparing multi-label to `top-1`) while `top-5` overestimates multi-label accuracy (models see a median drop of 7.4% when comparing multi-label accuracy to `top-5`). While multi-label accuracy is highly correlated with `top-1`and `top-5`accuracy, we assert that neither `top-1`nor `top-5`measure a semantically meaningful notion of accuracy.

3.4 Human accuracy measurement process

We now describe the human evaluation portion of our experiment. Annotators A, B, & C participated in the initial annotation review and thus saw all 40,683 evaluation images and labels from ImageNet and ImageNetV2. To remedy the possibility that annotators A, B & C unintentionally memorized the evaluation labels, two precautions were taken. First, annotators A, B, & C did not look at the data for six months. Second, we introduced annotators D & E, neither of whom had seen the test images prior to evaluation.

3.4.1 Human labeler training

After a six month period of inactivity, in October 2019, all five participants began a training regimen for the labeling task. Previously, participants A, B, C undertook a similar training for the initial multi-label annotation review. All training was carried out using a the 30,000 ImageNet validation images that would not be used for the final evaluation. The primary goal of training was to familiarize humans with the ImageNet class hierarchy.

The initial human accuracy study by Russakovsky et al. [131] details three main failure modes of humans: fine-grained distinctions, class unawareness, and insufficient training images. We address all three failure modes with our training regimen:

Fine-grained distinctions. There are many difficult class distinctions in ImageNet, but humans tend to struggle with fine-grained distinctions within the 410 animal classes and 118 dog classes. Even the scientific community disagrees about the exact taxonomy of specific species. For instance, while `tiger beetles` are often classified as a subfamily of `ground beetle`, this classification isn't universally accepted among entomologists [3, 152]. Similar issues arise in other animal families, such as the mustelines, monkeys, and wolves.

To help humans perform well on fine-grained class distinctions, we created training tasks containing only images from certain animal families. The training tasks gave labelers immediate feedback on whether they had made the correct prediction or not. These targeted training tasks were created after labelers identified classes for which they wanted additional training. Labelers trained on class-specific tasks for dogs, insects, monkeys, terriers, electric rays and sting rays, and marmots and beavers. After training, labelers reviewed each other's annotations as a group and discussed the class distinctions. Labelers also wrote a labeling guide containing useful information for distinguishing similar classes, discussed in more detail in Section 3.4.2.

Information from the American Kennel Club [1] was frequently used to understand and disambiguate difficult dog breeds. We also reached out to a member of the local chapter of the club for aid with dog identification. Since some dogs may be mixed-breeds, it may be impossible to disambiguate between similar dog breeds from pictures alone. Fortunately, the ImageNet dog labels are of high quality as they are derived from the Flickr image description, which are often authored by the owner of the dog.

Class unawareness. For the 590 object categories in ImageNet, *recall* is the primary difficulty for untrained humans. To address this, we built a labeling user interface that allowed annotators to either search for a specific ImageNet class or explore a graphical representation of the ImageNet classes based on the WordNet [111] hierarchy.

Insufficient training images. The two annotators in [131] trained on 500 and 100 images respectively, and then had access to 13 training images per class while labeling. In our experiment, human labelers had access to 100 training images per class while labeling.

3.4.2 Labeling guide

During training, the participants constructed a *labeling guide* that distilled class specific analysis learned during training into key discriminative traits that could be referenced by the labelers during the final labeling evaluation. The labeling guide contained detailed entries for 431 classes.

3.4.3 Final evaluation and annotation review.

On December 18th 2019, 1,000 images were sampled from ImageNet Validation and 1,000 images were sampled from ImageNetV2 and shuffled together. The datasets were sampled in a class balanced manner.

Between December 19th 2019 and January 16th 2020 all 5 participants labeled 2,000 images in order to produce the main results of this work. The only resources the labelers had access to during evaluation were 100 randomly sampled images from the ImageNet training set for each class, and the labeling guide. The participants spent a median of 26 seconds per image, with a median labeling time of 36 hours for the entire labeling task.

After the labeling task was completed, an additional multi-label annotation session was necessary. Since each image only contained reviewed labels for classes predicted by *models*, to ensure a fair multi-label accuracy, the human predictions for the 2,000 images had to be manually reviewed. To minimize bias, participants were not allowed to view their predicted labels after the task, and random model predictions were seeded into the annotation review such that every image had both model and human predictions to be reviewed. Compared to labels from the initial annotation review from November 2018, after the final annotation review, labels were unchanged for 1320 images, added for 531 images, and modified for 239 images. The modifications were due to a much greater knowledge of fine-grained class distinctions by the participants after the training phase.

Table 3.2: Human and model multi-label accuracy on three subsets of the ImageNet and ImageNetV2 test sets. These results suggest that human labelers have an easier time identifying objects than dogs and organisms. Moreover, human labelers are highly accurate on images on which they spent little time to assign a label.

ImageNet multi-label accuracy (%)								
	All Images		**Without Dogs**		**Objects Only**		**Fast Images**	
Participant	Original	V2	Original	V2	Original	V2	Original	V2
`resnet50`	84.2	75.7	84.9	76.8	82.5	72.8	86.8	79.6
`AdvProp`	93.6	88.3	94.1	89.3	92.3	86.7	94.9	91.3
`FixResNeXt`	95.5	89.6	96.0	90.1	95.0	89.1	96.2	92.3
Human A	91.9	91.1	94.2	93.4	97.0	96.7	97.6	97.5
Human B	94.7	93.9	96.9	96.0	98.3	97.8	98.5	98.5
Human C	96.2	96.7	98.4	98.6	99.1	99.8	99.1	99.7
Human D	95.7	94.8	97.3	96.6	98.8	98.4	99.3	98.3
Human E	97.2	96.5	98.7	97.3	98.8	97.0	99.5	98.6

3.5 Main Results

In this section we discuss two key facets of our experimental findings: a comparison of human and machine accuracies on ImageNet, and a comparison of human and machine robustness to the distribution shift between the ImageNet validation set and the ImageNet-V2 test set. We also consider these comparisons on three restricted sets of images.

The main results of our work are illustrated in Figure 3.1. We can see that all the human labelers fall close to the dotted line, indicating they their accuracies on the two datasets are within 1%. Moreover, we can see that the accuracies of three of the human labelers are better than the performance of the best model on both the original ImageNet validation set and on the ImageNet-V2 test set. Importantly, we note that labelers D and E, who did not participate in the initial annotation period, performed better than the best model.

Figure 3.1 shows that the ImageNet validation set confidence intervals of the best 4 humans labelers and of the best model overlap. However, McNemar's paired test rejects the null hypothesis that the `FixResNeXt` model (the best model) and Human E (the best human labeler) have the same accuracy on the ImageNet validation set distribution with a p-value of 0.037. In Figure 3.1 we observe that the confidence intervals of Humans C, D, and E on the ImageNetV2 test set do not overlap with the confidence interval of the best model. McNemar's test between Human B and the `FixResNeXt` model on ImageNetV2 yields a p-value of 2×10^{-4}.

Difficult images: One of the benefits of our experiments is the potential insight into the failure modes of image classification models. To have a point of comparison let us start with the human labelers. There were 10 images which were misclassified by *all* human labelers. These images consisted of one image of a monkey and nine images of dogs. On the other hand, there were 27 images misclassified by *all* 71 models considered by us. Interestingly, 19 out of these images correspond to object classes and 8 correspond to organism classes. We note that there are only two images that were misclassified by all models and human labelers, both of them containing dogs. Four of the 27 images which were difficult for the models are displayed in Figure 3.5. It is interesting that the failure cases of the models consist of many images of objects while the failure cases of human labelers are exclusively images of animals.

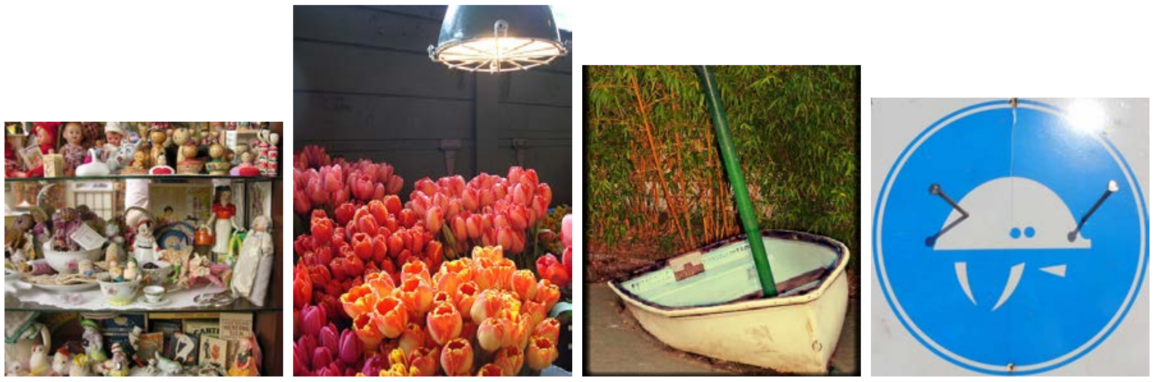


Figure 3.5: Four images which were misclassified by all 71 models, two from ImageNet (first two) and two from ImageNetv2. The correct target labels for these images are `cup`, `spotlight`, `yawl`, `nail`

Accuracies without dogs: To understand the extent to which models are better than the human labelers at classifying dogs and animals, we compute their accuracies on two restricted sets of images. First, we computed accuracies by excluding the 118 dog classes. In this case, Table 3.2 shows an increase in the accuracy of the best model ([147]) by 0.6% on ImageNet images and by 1.1% on ImageNetV2 images. However, the mean increase of the human labelers' accuracies is 1.9% on ImageNet and 1.8% on ImageNetV2. Before we interpret this result, we must establish whether the changes in accuracies shown in Table 3.2 are meaningful. There are 882 non-dog classes in ImageNet. We use the bootstrap to estimate changes in accuracies when the data is restricted to 882 classes. We compute accuracies over 1000 trials as follows: we sample without replacement 882 classes and compute the accuracies of the human labelers on the images whose main labels are in the sampled classes. All trials yield smaller changes in accuracy than those shown in Table 3.2. This simulation indicates that the increase in human performance on non-dog images is significant.

Therefore, the relative gap between human labelers and models increases on both ImageNet and ImageNetV2 when we remove the images containing dogs. This suggests that the dog images are more difficult for the human labelers participating in our experiment than for the models.

Accuracies on objects: To further understand the strengths and weaknesses of the models and human labelers, we compute their accuracies on the subset of data which have objects as their main labels, as opposed to organisms. There are 590 object classes. In Table 3.2 we can see the stark contrast in performance between human labelers and models on images of objects. The mean increase of the human labelers' accuracies is 3.3% on ImageNet and 3.4% on ImageNetV2, whereas the accuracy of the best model decreased by 0.5% on both ImageNet and ImageNetV2. A bootstrap simulation similar to the one described for the "Without Dogs" comparison reveals that human accuracy increase is significant. This result suggests that images of objects are substantially easier for the human labelers than the models.

Accuracies on fast images: Whereas CNN models spend the same amount of time classifying different images, the human labelers spent anywhere from a couple of seconds to 40 minutes labeling one image. What does the amount of time spent by humans labeling an image say about that image? We compute accuracies of all models and human labelers on the subset of images for which the median time spent by the human labelers to label it was at most 60 seconds. Out of a total of 2000 images used in the evaluation, there are 756 such images from ImageNet (77% of images) and 714 such images from ImageNetV2 (73% of images). We observe a dramatic increase in the accuracies of the human labelers, suggesting that human labelers know when an image is difficult for them and spend more time labeling it. The accuracies of the models also increase on "Fast Images." This result is intuitive, suggesting that images that humans label quickly are more likely to be correctly classified by models. We present results for these images in Table 3.2.

3.5.1 Accuracies on three disjoint subsets

To gain further insights in the capabilities of both machine and human labelers we compute their accuracies on three disjoint sets of classes: dogs, animals without dogs, and inanimate objects. The results can be found in Table 3.3

Table 3.3: Human and model multi-label accuracy on three subsets of the ImageNet and ImageNetV2 test sets. These results suggest that human labelers have an easier time identifying objects than dogs and organisms.

ImageNet multi-label accuracy (%)								
	All Images		**Dogs**		**Animals(No dogs)**		**Objects**	
Participant	Original	V2	Original	V2	Original	V2	Original	V2
`resnet50`	84.2	75.7	78.8	67.8	90.4	84.0	82.5	72.8
`AdvProp`	93.6	88.3	89.8	80.0	97.4	93.6	92.3	86.7
`FixResNeXt`	95.5	89.6	92.4	79.1	97.4	93.6	95.0	89.1
Human A	91.9	91.1	74.5	73.9	89.4	86.9	97.0	96.7
Human B	94.7	93.9	78.8	78.2	94.2	92.6	98.3	97.8
Human C	96.2	96.7	80.5	82.6	97.1	96.4	99.1	99.8
Human D	95.7	94.8	83.9	80.8	94.5	93.0	98.8	98.4
Human E	97.2	96.5	86.4	90.4	98.7	97.7	98.8	97.0

3.5.2 Top-1 Accuracies

In the main text, we measured the *multi-label* accuracy of both models and humans on both the ImageNet and ImageNetV2 test sets. For completeness, in Table **??** we now provide both the *top-1* and *multi-label* accuracy of models and humans on the same test sets. Figure 3.6 shows the scatters plot of `top-1`accuracies on ImageNet and ImageNetV2 test sets.

We note that under the `top-1`metric, we observe a substantially larger median accuracy drop of 4.3% between ImageNet and ImageNetV2 when compared to a median accuracy drop of 0.8% between the two datasets under the multi-label metric. As shown in Table 3.6, a similar accuracy drop accuracy drop exists on all three subsets studied in the main text.

While in Section 3.3 we address major issues with `top-1`accuracy for human evaluation, it is nonetheless interesting that such humans exhibit such a large performance drop in the `top-1`metric. In addition to the reasons mentioned in Section 3.3, we investigate two additional conjectures for the difference between `top-1`and multi-label result:

1. **Escape Hatches.** One potential failure mode of the multi-label metric would be an excess of images where the correct human prediction is a small or common object that is present in the scene but is presumably not the intended subject of the image. We denote these class labels “escape-hatch” labels as they allow the human to punt

on a difficult classification task such as the difference between a `French bull dog` vs a `Boston terrier` to classify an easier background object such as `pole`.

Since the notion of escape-hatches could substantially inflate accuracies on the classification task, all human subjects provided alternative labels for any image for which they believed they used an escape hatch. To preserve our blind analysis this was done *before* the subjects viewed the true labels for the images. While this process isn't perfect as it relied on each subjects own judgement on what constitutes an escape-hatch, we believe this is unavoidable since the notion of an escape hatch label is highly subjective in nature.

In table 3.7 we present accuracies on the predictions with the escape hatch alternatives imputed, that is for each image the subject marked as an escape hatch we replaced his or her prediction with the secondary prediction provided. We note these numbers vary significantly across each of the participants as the set of escape hatch images for each participants varies. We see that this induces an approximately 3% accuracy drop in 4/5 participants.

2. **Multi-label proportion in ImageNet vs ImageNet V2.** Another possible explanation for the `top-1`accuracy discrepancy between ImageNet and ImageNetV2 could be a higher proportion of images with multiple labels in ImageNetV2. Among the 1000 images labeled, 30.0% (292/984) images in ImageNet contained multiple labels compared to 34.4% (337/980) images in ImageNetV2.

   If humans have a significantly lower `top-1` accuracy on images with multiple semantically correct labels, the higher proportion of multi-label images in ImageNetV2 could partially explain the accuracy drop between the two datasets. Figure 3.7 illustrates precisely this notion, we measure the `top-1` accuracy on two mutually exclusive subsets of ImageNet and ImageNetV2, those with exactly one correct label, and those with multiple correct labels. We find that human accuracy can degrade over 40% between images with a single correct label and those with multiple correct labels.

Table 3.4: Original ImageNet validation `top-1`, `top-5`, and multi-label accuracy for every model in the testbed evaluated on the 20,000 image subset. The models are sorted by their `top-1` accuracy. The confidence intervals are 95% Clopper-Pearson intervals. The second part of the table can be found on the following page.

Original ImageNet accuracy (%)			
Model	Multi-label	`Top-1`	`Top-5`
`FixResNeXt101_32x48d_v2`	94.7 [94.4, 95.0]	86.1 [85.6, 86.6]	97.9 [97.7, 98.1]
`instagram-48d`	94.4 [94.1, 94.8]	85.6 [85.1, 86.0]	97.7 [97.4, 97.9]
`tf_efficientnet_b8_ap`	93.7 [93.4, 94.1]	85.3 [84.8, 85.8]	97.3 [97.1, 97.6]
`efficientnet-b7`	92.9 [92.6, 93.3]	84.3 [83.8, 84.8]	96.9 [96.7, 97.1]
`pnasnet_large_tf`	90.8 [90.4, 91.2]	82.6 [82.1, 83.2]	96.2 [95.9, 96.4]
`pnasnet5large`	90.8 [90.3, 91.2]	82.7 [82.2, 83.2]	95.9 [95.7, 96.2]
`nasnetalarge`	90.7 [90.3, 91.1]	82.4 [81.8, 82.9]	96.1 [95.8, 96.4]
`nasnet_large_tf`	90.6 [90.2, 91.1]	82.5 [82.0, 83.0]	96.1 [95.9, 96.4]
`senet154`	89.6 [89.1, 90.0]	81.2 [80.7, 81.8]	95.4 [95.1, 95.7]
`polynet`	89.3 [88.9, 89.8]	80.9 [80.4, 81.5]	95.5 [95.2, 95.8]
`inception_v4_tf`	88.6 [88.2, 89.1]	80.1 [79.5, 80.6]	95.2 [94.9, 95.5]
`inceptionresnetv2`	88.6 [88.1, 89.1]	80.2 [79.7, 80.8]	95.1 [94.8, 95.4]
`inception_resnet_v2_tf`	88.5 [88.1, 89.0]	80.3 [79.7, 80.8]	95.2 [94.9, 95.5]
`se_resnext101_32x4d`	88.3 [87.9, 88.8]	80.0 [79.5, 80.6]	94.9 [94.6, 95.2]
`inceptionv4`	88.3 [87.9, 88.8]	79.9 [79.4, 80.5]	94.9 [94.6, 95.2]
`dpn107`	88.1 [87.7, 88.6]	79.8 [79.2, 80.3]	94.7 [94.3, 95.0]
`dpn131`	87.7 [87.3, 88.2]	79.3 [78.7, 79.8]	94.5 [94.2, 94.8]
`dpn92`	87.5 [87.0, 87.9]	79.2 [78.7, 79.8]	94.6 [94.3, 94.9]
`dpn98`	87.4 [87.0, 87.9]	79.1 [78.5, 79.6]	94.4 [94.1, 94.7]
`se_resnext50_32x4d`	87.4 [86.9, 87.8]	79.0 [78.4, 79.6]	94.4 [94.0, 94.7]
`xception`	87.1 [86.6, 87.5]	78.7 [78.2, 79.3]	94.2 [93.9, 94.6]
`resnext101_64x4d`	87.0 [86.5, 87.5]	78.7 [78.2, 79.3]	94.2 [93.9, 94.5]
`se_resnet152`	86.9 [86.4, 87.4]	78.5 [77.9, 79.1]	94.3 [94.0, 94.6]
`resnet152`	86.6 [86.1, 87.1]	78.4 [77.8, 79.0]	94.2 [93.8, 94.5]
`se_resnet101`	86.6 [86.1, 87.0]	78.3 [77.7, 78.9]	94.4 [94.1, 94.7]
`resnext101_32x4d`	86.5 [86.0, 87.0]	78.2 [77.6, 78.7]	93.9 [93.5, 94.2]
`inception_v3_tf`	86.3 [85.8, 86.8]	78.0 [77.4, 78.5]	94.0 [93.6, 94.3]
`resnet_v2_152_tf`	86.2 [85.7, 86.7]	77.7 [77.1, 78.3]	94.0 [93.7, 94.4]
`se_resnet50`	85.8 [85.3, 86.3]	77.5 [76.9, 78.0]	93.8 [93.4, 94.1]
`resnet101`	85.7 [85.2, 86.2]	77.5 [76.9, 78.1]	93.5 [93.2, 93.9]
`inception_v3`	85.5 [85.0, 86.0]	77.1 [76.5, 77.7]	93.6 [93.3, 93.9]
`inceptionv3`	85.5 [85.0, 86.0]	77.3 [76.7, 77.9]	93.6 [93.2, 93.9]

Original ImageNet accuracy (%)			
Model	Multi-label	`Top-1`	`Top-5`
`resnet_v2_101_tf`	85.5 [85.0, 86.0]	76.9 [76.3, 77.5]	93.6 [93.2, 93.9]
`densenet161`	85.4 [84.9, 85.9]	76.9 [76.3, 77.5]	93.5 [93.2, 93.9]
`fbresnet152`	85.4 [84.8, 85.9]	77.1 [76.5, 77.7]	93.5 [93.1, 93.8]
`resnet_v1_152_tf`	85.2 [84.7, 85.7]	76.9 [76.3, 77.5]	93.3 [92.9, 93.6]
`dpn68b`	85.2 [84.6, 85.7]	76.8 [76.2, 77.4]	93.7 [93.3, 94.0]
`densenet201`	85.1 [84.5, 85.6]	76.9 [76.3, 77.4]	93.4 [93.0, 93.7]
`resnet_v1_101_tf`	84.6 [84.1, 85.2]	76.4 [75.8, 77.0]	92.9 [92.5, 93.3]
`cafferesnet101`	84.3 [83.8, 84.8]	76.1 [75.5, 76.7]	92.8 [92.4, 93.1]
`resnet50`	84.2 [83.7, 84.8]	76.1 [75.5, 76.7]	92.9 [92.5, 93.2]
`dpn68`	84.0 [83.5, 84.5]	75.7 [75.1, 76.3]	92.7 [92.3, 93.0]
`resnet_v2_50_tf`	84.0 [83.5, 84.5]	75.5 [74.9, 76.1]	92.8 [92.4, 93.2]
`densenet169`	83.9 [83.4, 84.5]	75.7 [75.1, 76.3]	92.8 [92.5, 93.2]
`resnet_v1_50_tf`	83.6 [83.1, 84.1]	75.3 [74.7, 75.9]	92.2 [91.9, 92.6]
`vgg19_bn`	82.5 [82.0, 83.0]	74.3 [73.6, 74.9]	91.8 [91.5, 92.2]
`densenet121`	82.3 [81.8, 82.9]	74.2 [73.5, 74.8]	91.8 [91.4, 92.2]
`pnasnet_mobile_tf`	82.3 [81.8, 82.8]	74.3 [73.7, 74.9]	92.0 [91.7, 92.4]
`inception_v2_tf`	82.2 [81.7, 82.8]	74.0 [73.4, 74.6]	91.7 [91.3, 92.1]
`nasnetamobile`	82.1 [81.6, 82.7]	74.0 [73.4, 74.6]	91.8 [91.4, 92.2]
`vgg16_bn`	81.7 [81.1, 82.2]	73.4 [72.8, 74.0]	91.5 [91.2, 91.9]
`bninception`	81.6 [81.1, 82.2]	73.5 [72.9, 74.1]	91.6 [91.2, 92.0]
`nasnet_mobile_tf`	81.6 [81.0, 82.1]	73.8 [73.1, 74.4]	91.6 [91.2, 92.0]
`resnet34`	81.5 [80.9, 82.1]	73.3 [72.7, 74.0]	91.5 [91.1, 91.9]
`vgg19`	80.5 [79.9, 81.0]	72.3 [71.7, 72.9]	90.9 [90.5, 91.3]
`vgg16`	79.7 [79.1, 80.3]	71.5 [70.9, 72.2]	90.5 [90.1, 90.9]
`vgg13_bn`	79.5 [78.9, 80.1]	71.4 [70.7, 72.0]	90.3 [89.9, 90.7]
`mobilenet_v1_tf`	79.3 [78.7, 79.9]	71.1 [70.5, 71.7]	90.4 [90.0, 90.8]
`vgg_16_tf`	78.9 [78.3, 79.4]	70.9 [70.3, 71.5]	90.0 [89.6, 90.4]
`vgg_19_tf`	78.8 [78.2, 79.3]	70.7 [70.1, 71.3]	90.0 [89.6, 90.4]
`vgg11_bn`	78.1 [77.5, 78.7]	70.1 [69.4, 70.7]	89.8 [89.3, 90.2]
`vgg13`	78.0 [77.4, 78.6]	70.1 [69.5, 70.7]	89.3 [88.9, 89.8]
`resnet18`	77.6 [77.0, 78.2]	69.6 [68.9, 70.2]	89.2 [88.8, 89.6]
`inception_v1_tf`	77.5 [76.9, 78.1]	69.4 [68.8, 70.0]	89.7 [89.2, 90.1]
`vgg11`	76.8 [76.1, 77.4]	68.8 [68.2, 69.5]	88.6 [88.2, 89.1]
`squeezenet1_1`	65.5 [64.8, 66.2]	58.3 [57.6, 59.0]	80.8 [80.2, 81.3]
`squeezenet1_0`	65.0 [64.4, 65.7]	57.8 [57.1, 58.5]	80.3 [79.7, 80.8]
`alexnet`	63.7 [63.0, 64.4]	56.9 [56.3, 57.6]	79.3 [78.8, 79.9]

Table 3.5: ImageNetV2 `top-1`, `top-5`, and multi-label accuracy for every model in the testbed. The confidence intervals are 95% Clopper-Pearson intervals. The models are sorted by their `top-1` accuracy. The second part of the table can be found on the following page.

ImageNetV2 accuracy (%)			
Model	Multi-label	`Top-1`	`Top-5`
`FixResNeXt101_32x48d_v2`	90.3 [89.7, 90.9]	86.1 [85.4, 86.8]	97.9 [97.6, 98.2]
`instagram-48d`	89.5 [88.9, 90.2]	85.6 [84.9, 86.2]	97.7 [97.3, 97.9]
`tf_efficientnet_b8_ap`	87.7 [87.1, 88.4]	85.3 [84.6, 86.0]	97.3 [97.0, 97.7]
`efficientnet-b7`	86.1 [85.4, 86.8]	84.3 [83.6, 85.0]	96.9 [96.5, 97.2]
`pnasnet5large`	83.0 [82.2, 83.8]	82.7 [82.0, 83.5]	95.9 [95.5, 96.3]
`pnasnet_large_tf`	83.0 [82.2, 83.7]	82.6 [81.9, 83.4]	96.2 [95.8, 96.5]
`nasnetalarge`	82.9 [82.1, 83.7]	82.4 [81.6, 83.1]	96.1 [95.7, 96.5]
`nasnet_large_tf`	82.8 [82.1, 83.6]	82.5 [81.8, 83.3]	96.1 [95.7, 96.5]
`senet154`	81.6 [80.8, 82.4]	81.2 [80.4, 82.0]	95.4 [95.0, 95.8]
`polynet`	81.4 [80.6, 82.2]	80.9 [80.2, 81.7]	95.5 [95.1, 95.9]
`inceptionresnetv2`	80.7 [79.9, 81.5]	80.2 [79.4, 81.0]	95.1 [94.7, 95.5]
`inception_resnet_v2_tf`	80.6 [79.8, 81.4]	80.3 [79.5, 81.0]	95.2 [94.8, 95.6]
`se_resnext101_32x4d`	80.3 [79.5, 81.1]	80.0 [79.3, 80.8]	94.9 [94.5, 95.4]
`inceptionv4`	80.1 [79.3, 80.9]	79.9 [79.1, 80.7]	94.9 [94.5, 95.3]
`inception_v4_tf`	79.5 [78.7, 80.3]	80.1 [79.3, 80.9]	95.2 [94.8, 95.6]
`dpn107`	79.2 [78.4, 80.0]	79.8 [79.0, 80.6]	94.7 [94.2, 95.1]
`se_resnext50_32x4d`	78.8 [77.9, 79.6]	79.0 [78.2, 79.8]	94.4 [93.9, 94.8]
`dpn98`	78.6 [77.7, 79.4]	79.1 [78.2, 79.8]	94.4 [93.9, 94.8]
`dpn131`	78.6 [77.7, 79.4]	79.3 [78.4, 80.0]	94.5 [94.0, 94.9]
`se_resnet152`	78.5 [77.7, 79.3]	78.5 [77.7, 79.3]	94.3 [93.9, 94.8]
`xception`	78.2 [77.4, 79.0]	78.7 [77.9, 79.5]	94.2 [93.8, 94.7]
`se_resnet101`	78.2 [77.3, 79.0]	78.3 [77.5, 79.1]	94.4 [93.9, 94.8]
`dpn92`	78.1 [77.2, 78.9]	79.2 [78.4, 80.0]	94.6 [94.2, 95.1]
`resnet152`	77.7 [76.9, 78.6]	78.4 [77.6, 79.2]	94.2 [93.7, 94.6]
`resnext101_64x4d`	77.7 [76.9, 78.6]	78.7 [77.9, 79.5]	94.2 [93.7, 94.7]
`resnet_v2_152_tf`	77.0 [76.1, 77.9]	77.7 [76.9, 78.5]	94.0 [93.6, 94.5]
`se_resnet50`	76.9 [76.0, 77.7]	77.5 [76.6, 78.3]	93.8 [93.3, 94.2]
`resnext101_32x4d`	76.9 [76.0, 77.7]	78.2 [77.4, 79.0]	93.9 [93.4, 94.3]

ImageNetV2 accuracy (%)			
Model	Multi-label	Top-1	Top-5
inception_v3_tf	76.7 [75.8, 77.6]	78.0 [77.1, 78.8]	94.0 [93.5, 94.4]
resnet101	76.6 [75.7, 77.5]	77.5 [76.7, 78.3]	93.5 [93.0, 94.0]
inceptionv3	76.4 [75.5, 77.3]	77.3 [76.5, 78.1]	93.6 [93.1, 94.0]
fbresnet152	76.4 [75.5, 77.2]	77.1 [76.3, 77.9]	93.5 [92.9, 93.9]
densenet161	76.2 [75.4, 77.1]	76.9 [76.1, 77.8]	93.5 [93.0, 94.0]
inception_v3	75.9 [75.0, 76.8]	77.1 [76.3, 78.0]	93.6 [93.1, 94.1]
resnet_v2_101_tf	75.9 [75.0, 76.8]	76.9 [76.0, 77.7]	93.6 [93.1, 94.0]
densenet201	75.4 [74.5, 76.3]	76.9 [76.0, 77.7]	93.4 [92.9, 93.9]
dpn68b	75.4 [74.5, 76.3]	76.8 [76.0, 77.6]	93.7 [93.2, 94.1]
resnet_v1_152_tf	75.2 [74.4, 76.1]	76.9 [76.1, 77.7]	93.3 [92.8, 93.7]
resnet_v1_101_tf	75.2 [74.4, 76.1]	76.4 [75.5, 77.2]	92.9 [92.4, 93.4]
cafferesnet101	74.9 [74.0, 75.8]	76.1 [75.3, 76.9]	92.8 [92.2, 93.3]
densenet169	74.1 [73.2, 75.0]	75.7 [74.9, 76.6]	92.8 [92.3, 93.3]
resnet50	74.1 [73.2, 74.9]	76.1 [75.3, 77.0]	92.9 [92.3, 93.4]
dpn68	74.0 [73.1, 74.9]	75.7 [74.8, 76.5]	92.7 [92.2, 93.2]
resnet_v2_50_tf	73.5 [72.6, 74.4]	75.5 [74.7, 76.4]	92.8 [92.3, 93.3]
resnet_v1_50_tf	73.1 [72.2, 74.0]	75.3 [74.5, 76.2]	92.2 [91.7, 92.8]
densenet121	72.8 [71.9, 73.7]	74.2 [73.3, 75.0]	91.8 [91.3, 92.4]
bninception	72.7 [71.8, 73.6]	73.5 [72.6, 74.4]	91.6 [91.1, 92.2]
vgg19_bn	72.0 [71.0, 72.9]	74.3 [73.4, 75.1]	91.8 [91.3, 92.4]
resnet34	71.8 [70.9, 72.7]	73.3 [72.5, 74.2]	91.5 [90.9, 92.0]
inception_v2_tf	71.8 [70.8, 72.7]	74.0 [73.1, 74.9]	91.7 [91.2, 92.2]
nasnetamobile	71.7 [70.8, 72.6]	74.0 [73.1, 74.8]	91.8 [91.2, 92.3]
pnasnet_mobile_tf	71.3 [70.4, 72.2]	74.3 [73.4, 75.1]	92.0 [91.5, 92.6]
vgg16_bn	71.1 [70.1, 72.0]	73.4 [72.6, 74.3]	91.5 [91.0, 92.1]
nasnet_mobile_tf	71.0 [70.0, 71.9]	73.8 [72.9, 74.6]	91.6 [91.1, 92.2]
vgg19	69.7 [68.7, 70.6]	72.3 [71.4, 73.2]	90.9 [90.3, 91.5]
vgg13_bn	69.2 [68.3, 70.1]	71.4 [70.5, 72.2]	90.3 [89.7, 90.9]
vgg16	68.6 [67.7, 69.6]	71.5 [70.6, 72.4]	90.5 [89.9, 91.0]
vgg_19_tf	68.3 [67.4, 69.3]	70.7 [69.8, 71.6]	90.0 [89.4, 90.6]
vgg_16_tf	68.0 [67.1, 69.0]	70.9 [70.0, 71.8]	90.0 [89.4, 90.6]
vgg11_bn	67.6 [66.7, 68.6]	70.1 [69.2, 71.0]	89.8 [89.2, 90.4]
resnet18	67.2 [66.3, 68.2]	69.6 [68.7, 70.5]	89.2 [88.6, 89.8]
mobilenet_v1_tf	67.2 [66.3, 68.2]	71.1 [70.2, 72.0]	90.4 [89.8, 90.9]
inception_v1_tf	66.8 [65.8, 67.8]	69.4 [68.5, 70.3]	89.7 [89.1, 90.3]
vgg13	66.7 [65.7, 67.7]	70.1 [69.2, 71.0]	89.3 [88.7, 89.9]
vgg11	65.2 [64.2, 66.2]	68.8 [67.9, 69.8]	88.6 [88.0, 89.2]
squeezenet1_1	53.6 [52.6, 54.6]	58.3 [57.3, 59.3]	80.8 [80.0, 81.6]
squeezenet1_0	53.2 [52.2, 54.2]	57.8 [56.8, 58.7]	80.3 [79.5, 81.0]
alexnet	51.6 [50.6, 52.7]	56.9 [56.0, 57.9]	79.3 [78.5, 80.1]

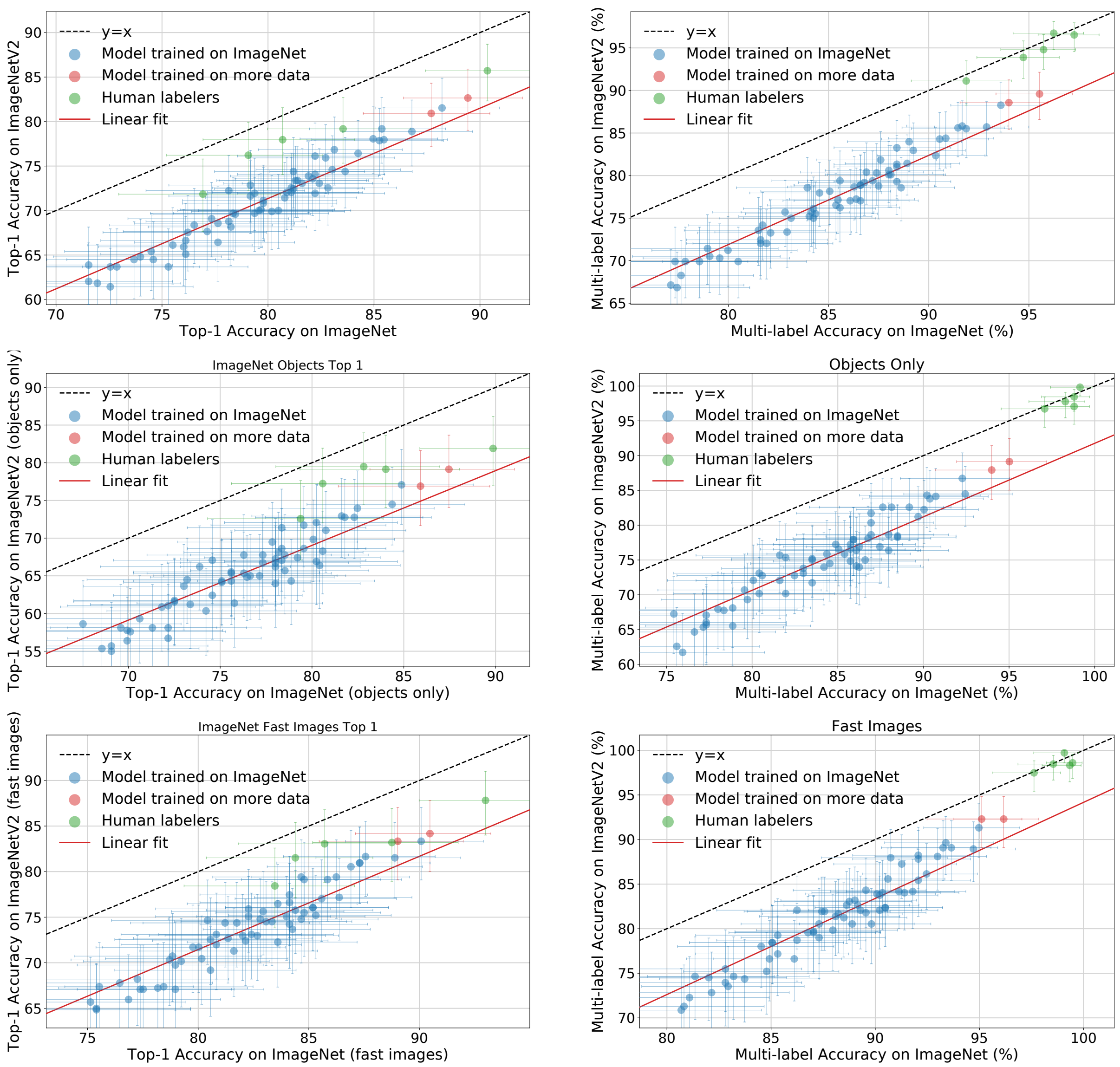


Figure 3.6: (left)`top-1`accuracies and (b) multi-label accuracies for both convolutional neural networks and five human labelers on the ImageNet validation set versus their accuracies on the ImageNetV2 test set. The confidence intervals are 95% Clopper-Pearson confidence intervals.

Table 3.6: Human and model `top-1` accuracy on three subsets of the ImageNet and ImageNetV2 test sets. The values shown in this table suggest that human labelers have an easier time identifying objects than dogs and organisms. Moreover, human labelers are highly accurate on images on which they spent little time to assign a label.

ImageNet Top-1 accuracy (%)								
	All Images		**Without Dogs**		**Objects Only**		**Fast Images**	
Participant	Original	V2	Original	V2	Original	V2	Original	V2
`resnet50`	78.1	68.8	78.4	69.0	74.6	62.4	80.8	72.0
`AdvProp`	88.2	81.5	88.2	81.8	84.9	77.1	90.1	83.3
`FixResNeXt`	89.4	82.7	89.5	83.4	87.5	79.1	90.5	84.2
`Human A`	76.9	71.8	79.8	74.7	79.4	72.6	83.5	78.4
`Human B`	79.1	76.2	82.7	80.0	80.6	77.2	84.4	81.5
`Human C`	80.7	78.0	84.5	82.1	82.8	79.5	85.7	83.1
`Human D`	83.5	79.2	85.7	81.7	84.0	79.1	88.8	83.2
`Human E`	90.3	85.7	91.6	85.9	89.9	81.9	93.0	87.8

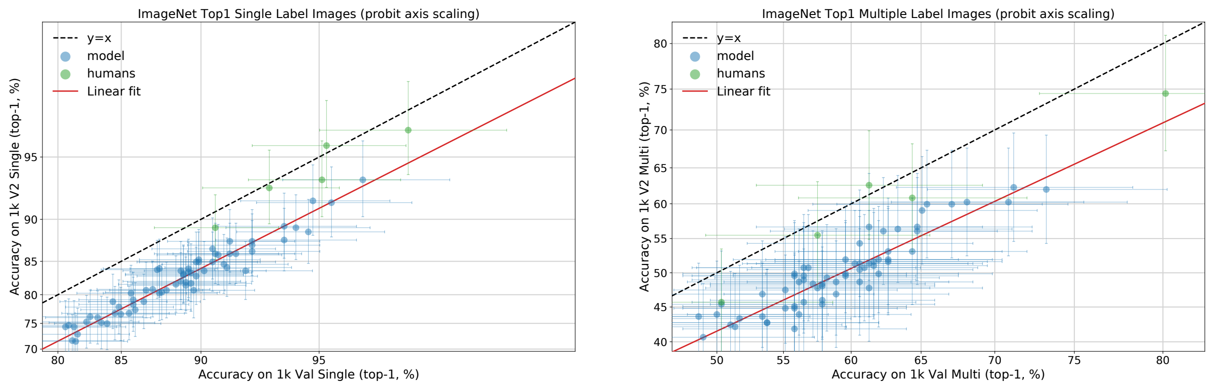


Figure 3.7: Scatter plot of `top-1`accuracies on subsets of ImageNet and ImageNetV2 with (left) A single "correct label (right) multiple "correct" labels. Note humans do *significantly* worse on images with multiple labels. The confidence intervals are 95% Clopper-Pearson confidence intervals.

Table 3.7: Human and model multi-label accuracy on ImageNet and ImageNetV2 test sets after escape hatch imputation. We also provide the original accuracies for reference.

ImageNet multi-label accuracy (%)				
	With Escape Hatch Imputation		**Without Escape Hatch Imputation**	
Participant	Original	V2	Original	V2
Human A	82.7	79.8	91.9	91.1
Human B	87.1	84.1	94.7	93.9
Human C	91.1	91.6	96.2	96.7
Human D	87.5	84.0	95.7	94.8
Human E	88.0	86.6	97.2	96.5

3.5.3 Examples of training effective Images

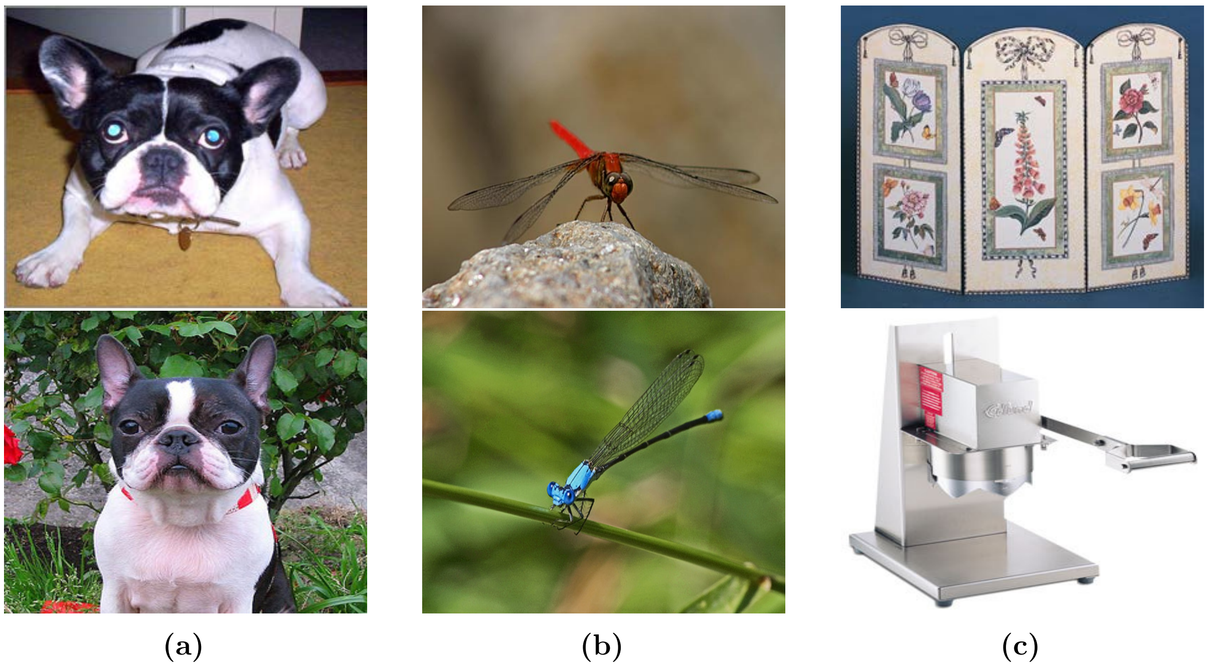


Figure 3.8: Examples of ImageNet classes that are difficult to distinguish or identify for untrained human labelers. a) A `French bull dog` (top) versus a `Boston bull terrier` (bottom). Though the dogs appear similar, there are key differences between the two breeds. `French bulldogs` have a more muscular build and weigh more than `Boston terriers`. b) A `dragonfly` (top) versus a `damselfly` (bottom). Again, the classes may initially look similar, but dragonflies have wings perpendicular to their body at rest while damselflies have wings parallel to their body at rest. c) A `fire screen` (top) and an industrial `can opener` (bottom). Both images are not prototypical examples for their class. Hence knowing the class hierarchy in detail to recognize the most likely classes is helpful.

3.5.4 Time Spent Per Image

Figure 3.9 details the median time spent by a human on each image. We note all the time measured is *active* time the kparticipants spent searching the UI for a potential label.

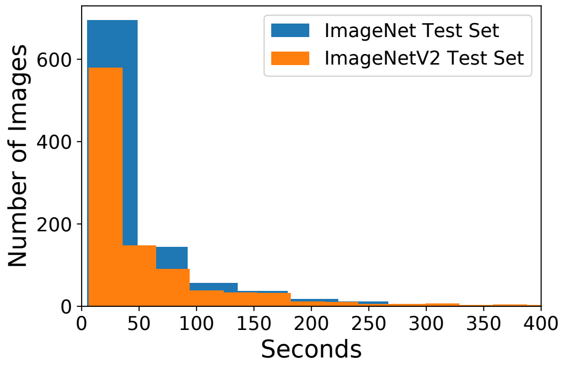


Figure 3.9: A histogram of the median time, measured in seconds, spent by the human labelers on images. We omitted are 24 outliers for which the median time spent by the human labelers was longer than 400 seconds: 13 outliers from the ImageNet test set and 11 outliers from the ImageNetV2 test set.

3.5.5 Problematic Image removal

One key step of the initial annotation procedure was removing *problematic images*. An image was problematic if any of the below criteria held:

- The original ImageNet label (`top-1`label) was incorrect
- Image was a cartoon or drawing
- Image was excessively edited
- Image had inappropriate content

Out of the 40,683 reviewed there were a total of 1206 images from ImageNet that were marked problematic and 686 images from ImageNetV2 marked problematic. In Table 3.8 we compute multi-label accuracies on both the problematic and non problematic subsets. Note accuracies are *substantially* lower on the problematic subset. Curiously accuracies

Table 3.8: Model multi-label accuracies on problematic and non-problematic subsets

	ImageNet multi-label accuracy (%)			
	Non Problematic Images		**Problematic Images**	
Model	Original	V2	Original	V2
`resnet50`	84.2	74.0	66.6	68.4
`AdvProp`	93.7	87.7	72.1	73.1
`FixRes`	94.7	90.2	74.0	75.7

are slightly higher on the problematic images from ImageNetV2 compared to ImageNet, we believe this is due to subjective decisions during the problematic image removal process and not some fundamental phenomena.

3.5.6 Ensembling Humans

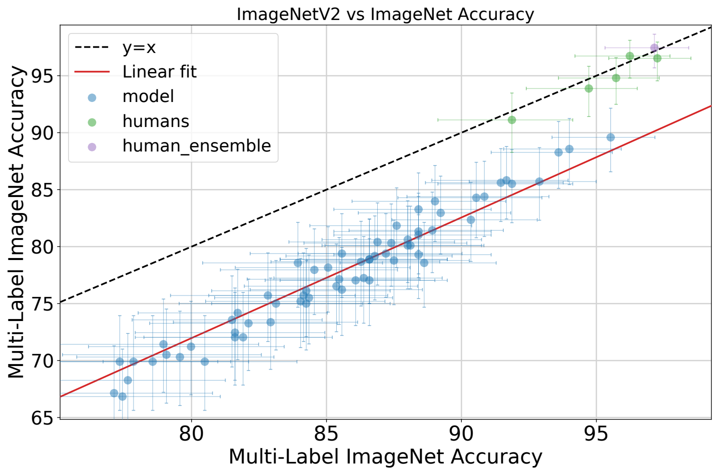


Figure 3.10: /
Accuracies of 71 convolutional neural networks, the five human labelers, and an ensemble of the same five human labelers. The confidence intervals are 95% Clopper-Pearson confidence intervals.

3.6 Related Work

Human accuracy on ImageNet. In the context of the ImageNet challenge, Russakovsky et al. [131] studied the accuracy of two trained humans on 1,500 and 258 ImageNet images, respectively. The widely publicized human baseline on ImageNet is the top-5 accuracy of the labeler who labeled 1,500 images. As mentioned in the introduction, our study goes beyond their comparison in three ways: multi-label accuracy, more labelers, and a focus on robustness to small distribution shift. While some of our findings differ, other results from [131] are consistent with ours. For instance, both experiments found that the time spent per image was approximately one minute, with a long tail due to difficult images.

Human performance in computer vision broadly. There have been several recent studies of humans in various areas of computer vision. For instance, Elsayed et al. [38] construct adversarial examples that fool both models and *time-limited* humans. Geirhos et al. [52] conducted psychophysical trials to investigate human robustness to synthetic distribution shfits, and Geirhos et al. [54] studied characteristics used by humans to make object recognition decisions. In a similar vein, Zhu et al. [165] contrast the effect of foreground and background objects on performance by humans and trained models.

Multi-label annotations. In this work, we contribute multi-label annotations for ImageNet and ImageNetV2. Previously, Stock and Cissé [139] studied the multi-label nature of ImageNet and found that `top-1` accuracy can underestimate multi-label by as much as 13.2%. The results of this work largely agree with our study. We hope the public release of our multi-label annotations will allow an accurate evaluation of all future models.

ImageNet inconsistencies and label error. During our annotation review, we recorded all incorrectly classified images we found in ImageNet and ImageNetV2. With the help of experts from the Cornell Lab of Ornithology,Van Horn et al. [148] estimate that at least 4% of birds are misclassified in ImageNet. Within the bird classes, [148] also observe inconsistencies in ImageNet's taxonomic structure which lead to weak class boundaries. We found that these taxonomic issues are present in the majority of the fine-grained organism classes.

Distribution shift. There is a growing body of work studying methods for addressing the challenge of distribution shift. For instance, the goal of distributionally robust optimization (DRO) is to find models that minimize the worst case expected error over a set of probability distributions [4, 10, 29, 37, 41, 133, 138]. A similar yet different line of work has focused on finding models that have low error rates on adversarial examples (worst case small perturbations to data points in the test set) [13, 14, 60, 106]. The work surrounding both DRO and adversarial examples has developed valuable ideas, but neither line of work has been shown to resolve the drop in accuracy between ImageNet and ImageNetV2.

3.7 Conclusion & Future Work

Achieving truly reliable machine learning will require a deeper understanding of the input changes a model should be robust to. Such understanding can likely guide research on more robust methods and is essential for developing meaningful tests of reliable performance. For tasks where human-like generalization is the ultimate goal, comparing model performance to human generalization can provide valuable information about the desired robustness properties. Our work is a step in this direction. Our results highlight that robustness to small, naturally occuring distribution shifts is a performance dimension not addressed by current benchmarks, but easily handled by humans. Besides the obvious direction of improving model robustness to such distribution shifts, there are further avenues for future work:

Robustness of non-expert labelers. A natural question is whether labelers with less training exhibit similar robustness to the distribution shift from ImageNet to ImageNetV2. Since untrained labelers will likely be in a lower accuracy regime, this would further illustrate that human robustness is a more stable property than direct accuracy measurements.

Other generalization dimensions. What further dimensions of human generalization are current models clearly lacking? Other forms of natural distribution shift such as robustness to temporal changes could be one candidate [62, 135].

Chapter 4

Do ImageNet Classifiers Generalize Across Time?

4.1 Introduction

Convolutional neural networks (CNNs) still exhibit many troubling failure modes. At one extreme, ℓ_p-adversarial examples cause large drops in accuracy for state-of-the-art models while relying only on visually imperceptible changes to the input image [13, 60]. However, this failure mode usually does not pose a problem outside a fully adversarial context because carefully crafted ℓ_p-perturbations are unlikely to occur naturally in the real world.

To study more realistic failure modes, researchers have investigated benign image perturbations such as rotations & translations, colorspace changes, and various image corruptions [39, 45, 71, 72]. However, it is still unclear whether these perturbations reflect the robustness challenges arising in real data since the perturbations also rely on synthetic image modifications.

Recent work has therefore turned to videos as a source of *naturally occurring* perturbations of images [9, 61, 162]. In contrast to other failure modes, the perturbed images are taken from existing image data without further modifications to make the task more difficult. As a result, robustness to such perturbations directly corresponds to performance improvements on real data.

However, it is currently unclear to what extent such video perturbations pose a robustness challenge. Azulay and Weiss [9] and Zheng et al. [162] only provide anecdotal evidence from a small number of videos. Gu et al. [61] go beyond individual videos and

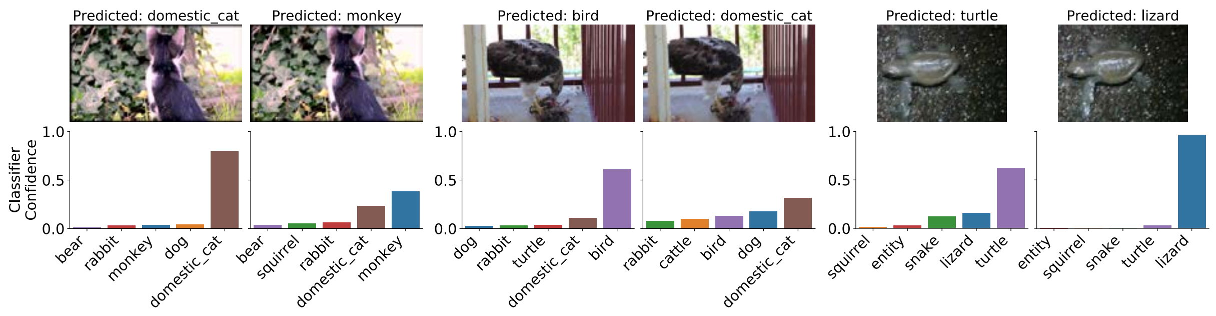


Figure 4.1: Three examples of natural perturbations from nearby video frames and resulting classifier predictions from a ResNet-152 model fine-tuned on ImageNet-Vid. While the images appear almost identical to the human eye, the classifier confidence changes substantially.

utilize a large video dataset [125] in order to measure the effect of video perturbations more quantitatively. In their evaluation, the best image classifiers lose about 3% accuracy for video frames up to 0.3 seconds away. However, the authors did not employ humans to review the frames in their videos. Hence the accuracy drop could also be caused by significant changes in the video frames (e.g., due to fast camera or object motion). Since the 3% accuracy drop is small to begin with, it remains unclear whether video perturbations are a robustness challenge for current image classifiers.

We address these issues by conducting a thorough evaluation of robustness to natural perturbations arising in videos. As a cornerstone of our investigation, we introduce two test sets for evaluating model robustness: `ImageNet-Vid-Robust` and `YTBB-Robust`, carefully curated from the ImageNet-Vid and Youtube-BB datasets, respectively [125, 130]. All images in the two datasets were screened by a set of expert labelers to ensure high annotation quality and minimize selection biases that arise when filtering a dataset with CNNs. To the best of our knowledge these are the first datasets of their kind, containing tens of thousands of images that are *human reviewed* and grouped into thousands of perceptually similar sets. In total, our datasets contain 3,139 sets of temporally adjacent and visually similar images (57,897 images total).

We then utilize these datasets to measure the robustness of current CNNs to small, naturally occurring perturbations. Our testbed contains over 45 different models, varying both architecture and training methodology (adversarial training, data augmentation, etc.). To better understand the drop in accuracy due to natural perturbations, we also introduce a robustness metric that is more stringent than those employed in prior work.

Under this metric, we find that natural perturbations from `ImageNet-Vid-Robust` and `YTBB-Robust` induce a median accuracy drop of 16% and 10% respectively for classification tasks and a median 14 point drop in mAP for detection tasks[1]. Even for the best-performing classification models, we observe an accuracy drop of 14% for `ImageNet-Vid-Robust` and 8% for `YTBB-Robust`.

Our results show that robustness to natural perturbations in videos is indeed a significant challenge for current CNNs. As these models are increasingly deployed in safety-critical environments that require both high accuracy and low latency (e.g., autonomous vehicles), ensuring reliable predictions on *every frame* of a video is an important direction for future work.

4.2 Constructing a test set for robustness

`ImageNet-Vid-Robust` and `YTBB-Robust` are sourced from videos in the ImageNet-Vid and Youtube-BB datasets [125, 130]. All object classes in ImageNet-Vid and Youtube-BB are from the WordNet hierarchy [112] and direct ancestors of ILSVRC-2012 classes. Using the WordNet hierarchy, we construct a canonical mapping from ILSVRC-2012 classes to ImageNet-Vid and Youtube-BB classes, which allows us to evaluate off-the-shelf ILSVRC-2012 models on `ImageNet-Vid-Robust` and `YTBB-Robust`. We provide more background on the source datsets in the appendix.

4.2.1 Source Dataset Overview

4.2.2 ImageNet-Vid

The 2015 ImageNet-Vid dataset is widely used for training video object detectors [65] as well as trackers [12]. We chose to work with the 2017 ImageNet-Vid dataset because it is a superset of the 2015 dataset. In total, the 2017 ImageNet-Vid dataset consists of 1,181,113 training frames from 4,000 videos and 512,360 validation frames from 1,314 videos. The videos have frame rates ranging from 9 to 59 frames per second (fps), with a median fps of 29. The videos range from 0.44 to 96 seconds in duration with a median duration of 12 seconds. Each frame is annotated with labels indicating the presence or absence of 30

[1]We only evaluated detection on `ImageNet-Vid-Robust` as bounding-box annotations in Youtube-BB are available only at a temporal resolution of 1 frame-per-second and hence not dense enough for our evaluation.

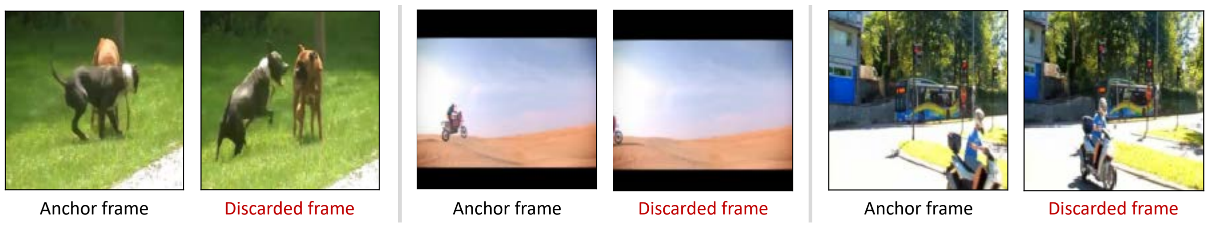


Figure 4.2: Temporally adjacent frames may not be visually similar. We show three randomly sampled frame pairs where the nearby frame was marked as "dissimilar" to the anchor frame during human review and then discarded from our dataset.

object classes and corresponding bounding boxes for any label present in the frame. The 30 classes are ancestors of 293 of the 1,000 ILSVRC-2012 classes.

4.2.3 Youtube-BB

The 2017 Youtube-BB is a a large scale dataset with 8,146,143 annotated training frames 253,569 unique videos and with 1,013,246 validation frames from 31,829 videos. The video segments are approximately 19 seconds long on average. Each frame is annotated with exactly one label indicating the presence of 22 object classes, all of which are ancestors of 229 out of the ILSVRC-2012 classes.

4.2.4 Constructing **`ImageNet-Vid-Robust`** and **`YTBB-Robust`**

Next, we describe how we extracted sets of naturally perturbed frames from ImageNet-Vid and Youtube-BB to create `ImageNet-Vid-Robust` and `YTBB-Robust`. A straightforward approach would be to select a set of anchor frames and use temporally adjacent frames in the video with the assumption that such frames contain only small perturbations from the anchor. However, as Figure 4.2 illustrates, this assumption is frequently violated, especially due to fast camera or object motion.

Instead, we first collect *preliminary* datasets of natural perturbations following the same approach, and then manually review each of the frame sets. For each video, we randomly sample an anchor frame and take $k = 10$ frames before and after the anchor frame as candidate perturbation images[2]. This results in two datasets containing one

[2]For `YTBB-Robust` we use a subset of the anchor frames used by Gu et al. [61].

anchor frame each from 3,139 videos, with approximately 20 candidate perturbation per anchor frame[3].

Next, we curate the dataset with the help of four expert human annotators. The goal of the curation step is to ensure that each anchor frame and its nearby frames are correctly labeled with the same ground truth class, and that the anchor frame and the nearby frames are visually similar.

Denser labels for Youtube-BB. As Youtube-BB contains only a single category label per frame at 1 frame per second, annotators first inspected each anchor frame individually and added any missing labels. In total, annotators corrected the labels for 834 frames, adding an average of 0.5 labels per anchor frame. These labels are then propagated to nearby, unlabeled frames at the native frame rate and verified in the next step. ImageNet-Vid densely labels all classes per frame, so we skipped this step for this dataset.

Frame pairs review. Next, for each pair of anchor and nearby frames, a human annotates (i) whether the pair is correctly labeled in the dataset, and (ii) whether the pair is similar. We took several steps to mitigate the subjectivity of this task and ensure high annotation quality. First, we trained reviewers to mark frames as dissimilar if the scene undergoes any of the following transformations: significant motion, significant background change, or significant blur change. We asked reviewers to mark each dissimilar frame with one of these transformations, or "other", and to mark a pair of images as dissimilar if a distinctive feature of the object is only visible in one of the two frames (such as the face of a dog). If an annotator was unsure about the correct label, she could mark the pair as "unsure". Second, we present only a single pair of frames at a time to reviewers because presenting videos or groups of frames could cause them to miss large changes due to the phenomenon of change blindness [116].

Verification. In the previous stage, all annotators were given identical labeling instructions and individually reviewed a total of 71,660 image pairs. To increase consistency in annotation, annotators jointly reviewed all frames marked as dissimilar, incorrectly labeled, or "unsure". A frame was only considered similar to its anchor if a strict majority of the annotators marked the pair as such.

After the reviewing was complete, we discarded all anchor frames and candidate perturbations that annotators marked as dissimilar or incorrectly labeled. The final datasets contain a combined total of 3,139 anchor frames with a median of 20 similar frames each.

[3]Anchor frames near the start or end of the video may have less than 20 candidate frames.

		`ImageNet-Vid-Robust`	`YTBB-Robust`
Anchor frames	Reviewed	1,314	2,467
	Accepted	1,109 (84%)	2,030 (82%)
	Labels updated	-	834 (41%)
Frame pairs	Reviewed	26,029	45,631
	Accepted	21,070 (81%)	36,827 (81%)

Table 4.1: Dataset statistics of `ImageNet-Vid-Robust` and `YTBB-Robust`. For `YTBB-Robust`, we updated the labels from for 41% (834) of the accepted anchors due to incomplete labels in Youtube-BB.

4.2.5 The `pm-k` evaluation metric

Given the datasets introduced above, we propose a metric to measure a model's robustness to natural perturbations. In particular, let $A = \{a_1, ..., a_n\}$ be the set of valid anchor frames in our dataset. Let $Y = \{y_1, ..., y_n\}$ be the set of labels for A. We let $\mathcal{N}_k(a_i)$ be the set of frames marked as similar to anchor frame a_i. In our setting, $\mathcal{N}_k$ is a subset of the $2k$ temporally adjacent frames (plus/minus k frames from the anchor).

Classification. The standard classification accuracy on the anchor frame is $\text{acc}_{\text{orig}} = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{0/1}(f(a_i), y_i)$, where $\mathcal{L}_{0/1}$ is the standard 0-1 loss function. We define the `pm-k` analog of accuracy as

$$\text{acc}_{\text{pmk}} \;=\; 1 - \frac{1}{N}\sum_{i=1}^{N} \max_{b \in \mathcal{N}_k(a_i)} \mathcal{L}_{0/1}(f(b), y_i) \;, \tag{4.2.1}$$

which corresponds to picking the worst frame from each set $\mathcal{N}_k(a_i)$ before computing accuracy. We note the similarity of the `pm-k` metric to standard ℓ_p adversarial robustness. If we let $\mathcal{N}_k(a_i)$ be the set of *all* images within an ℓ_p ball of radius ϵ around a_i, then the two notions of robustness are identical.

4.3 Detection

We briefly introduce the mAP metric for detection here and refer the reader to [101] for further details. The standard detection metric proceeds by first determining whether each predicted bounding box in an image is a true or false positive, based on the intersection over union (IoU) of the predicted and ground truth bounding boxes. The metric then computes the per-category average precision (AP, averaged over recall thresholds) of the predictions across all images. The final metric is reported as the mean of these per-category APs (mAP).

We define the pm-k analog of mAP by replacing each anchor frame in the dataset with a nearby frame that minimizes the per-image average precision. Since the category-specific average precision is undefined for categories not present in an image, we minimize the average precision across categories present in each frame rather than the mAP.

4.4 Experimental results

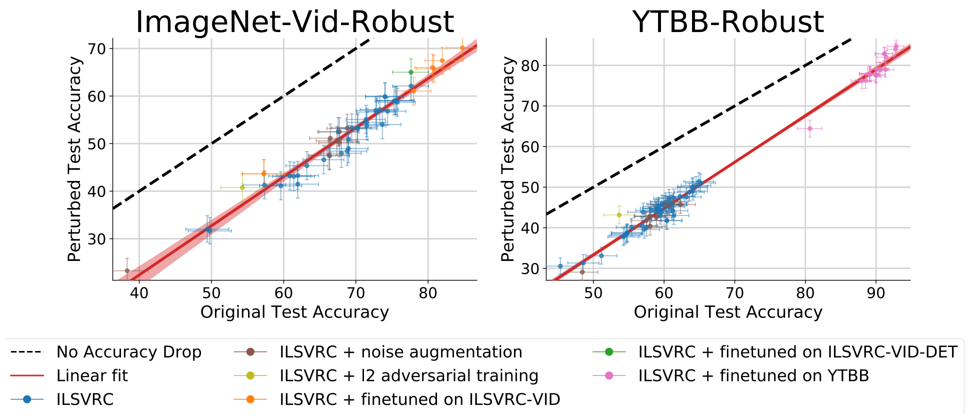


Figure 4.3: Model accuracy on original vs. perturbed images. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). If models were robust to perturbations, we would expect them to fall on the dashed line ($y = x$). Instead, we find they all lie significantly below this ideal line, consistently exhibiting a significant accuracy drop to perturbed frames. Each perturbed frame was taken from a ten frame neighborhood (approximately 0.3 seconds) of the original frame, and reviewed by experts to confirm visual similarity to the original frame.

We evaluate a testbed of 45 classification and three detection models on ImageNet-Vid-Robust and YTBB-Robust. We first discuss the various types of classification models evaluated with the pm-k classification metric. Second, we evaluate the performance of detection models on ImageNet-Vid-Robust using use the bounding box annotations inherited from ImageNet-Vid and using a variant of the pm-k metric for detection. We then analyze the errors made on the detection adversarial examples to isolate the effects of *localization* errors vs. *classification* errors. Finally, we analyze the impact of dataset review on the accuracy drop.

4.4.1 Classification

The classification robustness metric is $\mathrm{acc}_{\mathrm{pmk}}$ defined in Equation (4.2.1). For frames with multiple labels, we count a prediction as correct if the model predicts *any* of the correct classes for a frame. In Figure 4.3, we plot the benign accuracy, $\mathrm{acc}_{\mathrm{orig}}$, versus the robust accuracy, $\mathrm{acc}_{\mathrm{pmk}}$, for all classification models in our test bed and find a consistent drop from $\mathrm{acc}_{\mathrm{orig}}$ to $\mathrm{acc}_{\mathrm{pmk}}$. Further, we note that the relationship between $\mathrm{acc}_{\mathrm{orig}}$ and $\mathrm{acc}_{\mathrm{pmk}}$ is approximately linear, indicating that while improvements in the benign accuracy do result in improvements in the worst-case accuracy, they do not suffice to resolve the accuracy drop due to natural perturbations.

Our test bed consists of five model types with increasing levels of supervision. We present results for representative models from each model type in Table 4.2 and defer the full classification results table to the appendix.

ILSVRC Trained The WordNet hierarchy enables us to repurpose models trained for the 1,000 class ILSVRC-2012 dataset on `ImageNet-Vid-Robust` and `YTBB-Robust` (see Appendix for further details) We evaluate a wide array of ILSVRC-2012 models (available from [16]) against our natural perturbations. Since these datasets present a substantial distribution shift from the original ILSVRC-2012 validation set, we expect the *benign* accuracy $\mathrm{acc}_{\mathrm{orig}}$ to be lower than the comparable accuracy on the ILSVRC-2012 validation set. However, our main interest here is in the *difference* between the original and perturbed accuracies $\mathrm{acc}_{\mathrm{orig}}$ - $\mathrm{acc}_{\mathrm{pmk}}$. A small drop in accuracy would indicate that the model is robust to small changes that occur naturally in videos. Instead, we find significant median drops of 15.0% and 13.2% in accuracy on our two datasets, indicating sensitivity to such changes.

Noise augmentation One hypothesis for the accuracy drop from original to perturbed accuracy is that subtle artifacts and corruptions introduced by video compression schemes could degrade performance when evaluating on these corrupted frames. The worst-case nature of the `pm-k` metric could then be focusing on these corrupted frames. One model for these corruptions are the perturbations introduced in [71]. To test this hypothesis, we evaluate models augmented with a subset of the perturbations (exactly one of: Gaussian

Table 4.2: Accuracies of five model types and the best performing model (shown with 95% Clopper-Pearson confidence intervals). Δ denotes accuracy drop between evaluation on anchor frame (acc_{orig}) and worst frame in similarity set (acc_{pmk}). The model architecture is ResNet-50 unless noted otherwise. ‘FT’ denotes ‘fine-tuning.’ See Section 4.4.1 for details.

Model Type	Accuracy Original	Accuracy Perturbed	Δ
`ImageNet-Vid-Robust`			
Trained on ILSVRC	67.5 [64.7, 70.3]	52.5 [49.5, 55.5]	15.0
+ Noise Augmentation	68.8 [66.0, 71.5]	53.2 [50.2, 56.2]	15.6
+ ℓ_∞ robustness (ResNext-101)	54.3 [51.3, 57.2]	40.8 [39.0, 43.7]	12.4
+ FT on ImageNet-Vid	80.8 [78.3, 83.1]	65.7 [62.9, 68.5]	15.1
+ FT on ImageNet-Vid (ResNet-152)	84.8 [82.5, 86.8]	70.2 [67.4, 72.8]	14.6
+ FT on ImageNet-Vid-Det	77.6 [75.1, 80.0]	65.4 [62.5, 68.1]	12.3
`YTBB-Robust`			
Trained on ILSVRC	57.0 [54.9, 59.2]	43.8 [41.7, 46.0]	13.2
+ Noise Augmentation	62.3 [60.2, 64.4]	45.7 [43.5, 47.9]	16.6
+ ℓ_∞ robustness (ResNext-101)	53.6 [51.4, 55.8]	43.2 [41.0, 45.3]	10.4
+ FT on Youtube-BB	91.4 [90.1, 92.6]	82.0 [80.3, 83.7]	9.4
+ FT on Youtube-BB (ResNet-152)	92.9 [91.6, 93.9]	84.7 [83.0, 86.2]	8.2

Table 4.3: Detection and localization mAP for Faster R-CNN and R-FCN models. Both detection and localization suffer from significant mAP drops due to perturbations. (R-FCN was trained on ILSVRC Det and VID 2015, and evaluated on the 2015 subset of ILSVRC-VID 2017, indicated by *.)

Task	Model	mAP Original	mAP Perturbed	mAP Δ
Detection	FRCNN, ResNet 50	62.8	48.8	14.0
	FRCNN, ResNet 101	63.1	50.6	12.5
	R-FCN, ResNet 101 [154]*	79.4*	63.7*	15.7*
Localization	FRCNN, ResNet 50	76.6	64.2	12.4
	FRCNN, ResNet 101	77.8	66.3	11.5
	R-FCN, ResNet 101*	80.9*	70.3*	10.6*

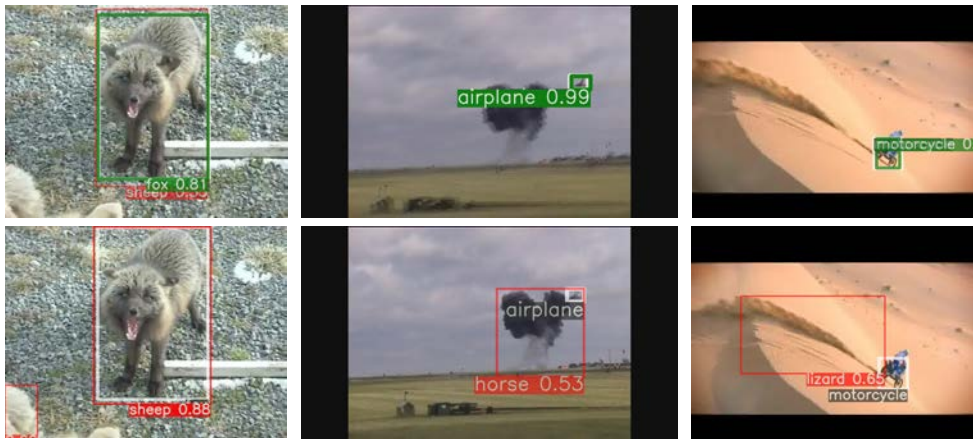


Figure 4.4: Naturally perturbed examples for detection. Red boxes indicate false positives; green boxes indicate true positives; white boxes are ground truth. Classification errors are common failures, such as the fox on the left, which is classified correctly in the anchor frame, and misclassified as a sheep in a nearby frame. However, detection models also have *localization* errors, where the object of interest is not correctly localized in addition to being misclassified, such as the airplane (middle) and the motorcycle (right). All visualizations show predictions with confidence greater than 0.5.

noise, Gaussian blur, shot noise, contrast change, impulse noise, or JPEG compression). We found that these augmentation schemes did not improve robustness against our perturbations substantially, and still result in a median accuracy drop of 15.6% and 16.6% on the two datasets.

ℓ_∞-robustness. We evaluate the model from [155], which currently performs best against ℓ_∞-attacks on ImageNet. We find that this model has a smaller accuracy drop than the two aforementioned model types on both datasets. However, the robust model achieves substantially lower original and perturbed accuracy than either of the two model types above, and the robustness gain is modest (3% compared to models of similar benign accuracy).

Fine-tuning on video frames. To adapt to the new class vocabulary and the video domain, we fine-tune several network architectures on the ImageNet-Vid and Youtube-BB training sets. For Youtube-BB, we train on the anchor frames used for training in [61], and for ImageNet-Vid we use all frames in the training set. We provide hyperparameters for all models in the appendix.

The resulting models significantly improve in accuracy over their ILSVRC pre-trained counterparts (e.g., 13% on `ImageNet-Vid-Robust` and 34% on `YTBB-Robust` for ResNet-50). This improvement in accuracy results in a modest improvement in the accuracy drop for `YTBB-Robust`, but a finetuned ResNet-50 still suffers from a substantial 9.4% drop. On `ImageNet-Vid-Robust`, there is almost no change in the accuracy drop from 15.0% to 15.1%.

Fine-tuning for detection on video frames. We further analyze whether additional supervision in the form of bounding box annotations improves robustness. To this end, we train the Faster R-CNN *detection* model [129] with a ResNet-50 backbone on ImageNet-Vid. Following standard practice, the detection backbone is pre-trained on ILSVRC-2012. To evaluate this detector for classification, we assign the class with the most confident bounding box as label to the image. We find that this transformation reduces accuracy compared to the model trained for classification (77.6% vs. 80.8%). While there is a slight reduction in the accuracy drop caused by natural perturbations, the reduction is well within the error bars for this test set. We leave an in-depth investigation of additional supervision to induce robustness for future work.

4.4.2 Detection

We further study the impact of natural perturbations on object detection. Specifically, we report results for two related tasks: object localization and detection. Object detection is the standard computer vision task of correctly classifying an object and finding the coordinates of a tight bounding box containing the object. "Object localization", meanwhile, refers to only the subtask of finding the bounding box, *without* attempting to correctly classify the object.

We provide our results on `ImageNet-Vid-Robust`, which contains dense bounding box labels unlike Youtube-BB, which only labels boxes at 1 frame per second. We use the popular Faster R-CNN [129] and R-FCN [26, 154] architectures for object detection and localization and report results in Table 4.3. For the R-FCN architecture, we use the model from [154][4]. We first note the significant drop in mAP of 12 to 15 points

[4]This model was originally trained on the 2015 subset of ImageNet-Vid. We evaluated this model on the 2015 validation set because the method requires access to pre-computed bounding box proposals which are available only for the 2015 subset of ImageNet-Vid.

Table 4.4: Impact of human review on `ImageNet-Vid-Robust` and `YTBB-Robust` on original and perturbed accuracy, using ResNet-152 fine-tuned on ImageNet-Vid and Youtube-BB, respectively.

		Accuracy		
	Reviewed	Original	Perturbed	Δ
`ImageNet-Vid-Robust`	✗	80.3	64.1	16.2
	✓	84.8	70.2	14.4
`YTBB-Robust`	✗	88.1	78.1	10.0
	✓	92.9	84.7	8.9

for object detection due to perturbed frames for both the Faster R-CNN and R-FCN architectures. Next, we show that localization is indeed easier than detection, as the mAP is higher for localization than for detection (e.g., 76.6 vs 62.8 for Faster R-CNN with a ResNet-50 backbone). Perhaps surprisingly, however, switching to the localization task does *not* improve the drop between original and perturbed frames, indicating that natural perturbations induce both classification and localization errors. We show examples of detection failures in Figure 4.4.

4.4.3 Impact of Dataset Review

We analyze the impact of our human review, described in Section 4.2.4, on the classifiers in our testbed. First, we compare the original and perturbed accuracies of a representative classifier (ResNet-152 finetuned) with and without review in Table 4.5. Our review improves the original accuracy by 3 to 4% by discarding mislabeled or blurry anchor frames, and improves perturbed accuracy by 5 to 6% by discarding dissimilar frame pairs. Our review reduces the accuracy drop by 1.8% on `ImageNet-Vid-Robust` and 1.1% on `YTBB-Robust`. These results indicate that the changes in model predictions are indeed due to a lack of robustness, rather than due to significant differences between adjacent frames.

To further analyze the impact of our review on model errors, we plot how frequently each offset distance from the anchor frame results in a model error across all model types in Figure 4.12. For both datasets, larger offsets (indicating pairs of frames further apart in time) lead to more frequent model errors. Our review reduces the fraction of errors across offsets, especially for large offsets, which are more likely to display large changes from the anchor frame.

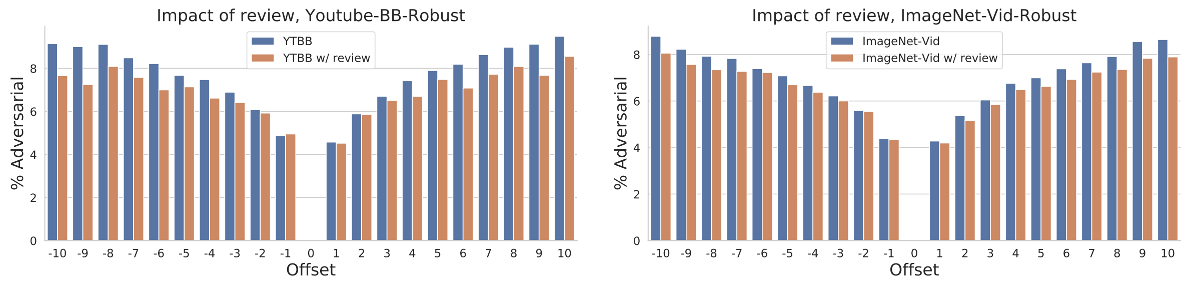


Figure 4.5: We plot how often each frame offset caused an error, across all evaluated models, for frames with and without review. Frames further away more frequently cause classifiers to misfire. Our review reduces the number of errors, especially for frames further in time, by removing dissimilar frames.

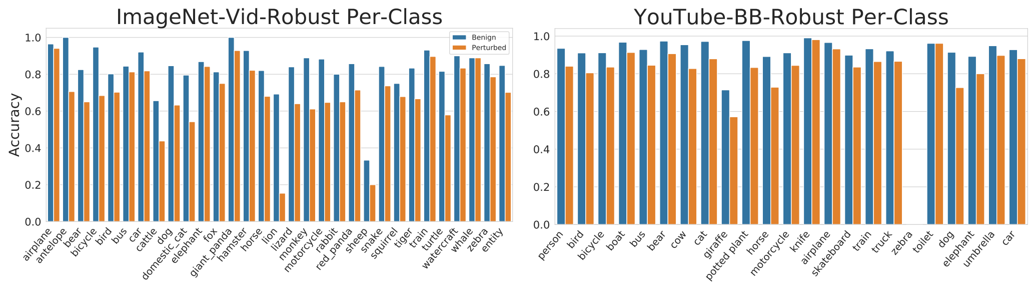


Figure 4.6: Per-class accuracy statistics for our best performing classification model (fine-tuned ResNet152) on `ImageNet-Vid-Robust` and `YTBB-Robust`. For Youtube-BB, note that 'zebra' is the least common label, present in only 24 anchor frames sampled by [61], of which 4 are included in our dataset.

4.4.4 Per class analysis

We study the effect of our perturbations on the 30 classes in `ImageNet-Vid-Robust` and `YTBB-Robust` to determine whether the performance drop was concentrated in a few "hard" classes. Figure 4.6 shows the original and perturbed accuracies across classes for our best performing model (a fine-tuned ResNet-152). Although there are a few particularly difficult classes for perturbed accuracy (e.g., lion or monkey on `ImageNet-Vid-Robust`), the accuracy drop is spread across most classes. On `ImageNet-Vid-Robust`, this model saw a total drop of 14.4% between original and perturbed images and a median drop of 14.0% in per-class accuracy. On `YTBB-Robust`, the total drop was 8.9% and the median drop was 6.7%.

4.4.5 Per-frame conditional robustness metric introduced in [61]

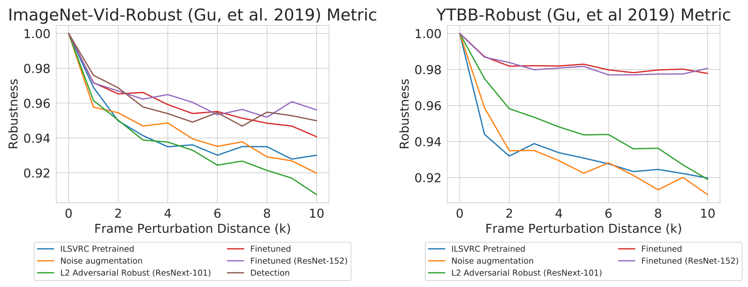


Figure 4.7: Conditional robustness metric from [61] on perturbed frames as a function of perturbation distance on `ImageNet-Vid-Robust` and `YTBB-Robust`. Model accuracies from five different model types and the best performing model are shown. The model architecture is ResNet-50 unless otherwise mentioned.

In concurrent work, the authors of [61] considered a different metric of robustness. In this section, we compute this metric on all models in our test bed to compare our findings to [61]. There are two main differences between PM-k and the robustness metric in [61].

1. For two visually similar "neighbor" frames I_0 and I_1 with true label Y and classifier f, [61] studies the conditional probability $P(f(I_1) = y | f(I_0) = y)$
2. While PM-k looks for errors in all neighbor frames in a neighborhood of k frames away from the anchor frame (so this would include frames 1, 2, . . . , k frames away), [61] only considers errors from **exactly** k frames away.

In Fig. 4.8 we illustrate simple example where two videos can have the same behavior for the metric introduced by [61] but drastically different behavior for the PM-kmetric.

4.5 ℓ_∞ distance vs PM-k Accuracy

ℓ_∞ adversarial examples are well studied in the robustness community, yet the connection between ℓ_∞ and other forms of more "natural" robustness is unclear. Here, we plot the cumulative distribution of the ℓ_∞ distance between pairs of nearby frames in our datasets. In Figure 4.9, we show the CDF of ℓ_∞ distance for all pairs, all reviewed pairs, and mistakes made by 3 indicative models. Note the `fbrobust` model is trained specifically to be robust to ℓ_∞ adversaries.

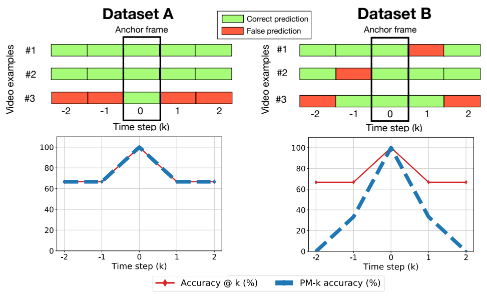


Figure 4.8: For the two example videos above the score from [61] metric (Accuracy @ K) is identical, but the PM-k metric behaves substantially differently when the errors are spread across many independent videos, as shown in the right example

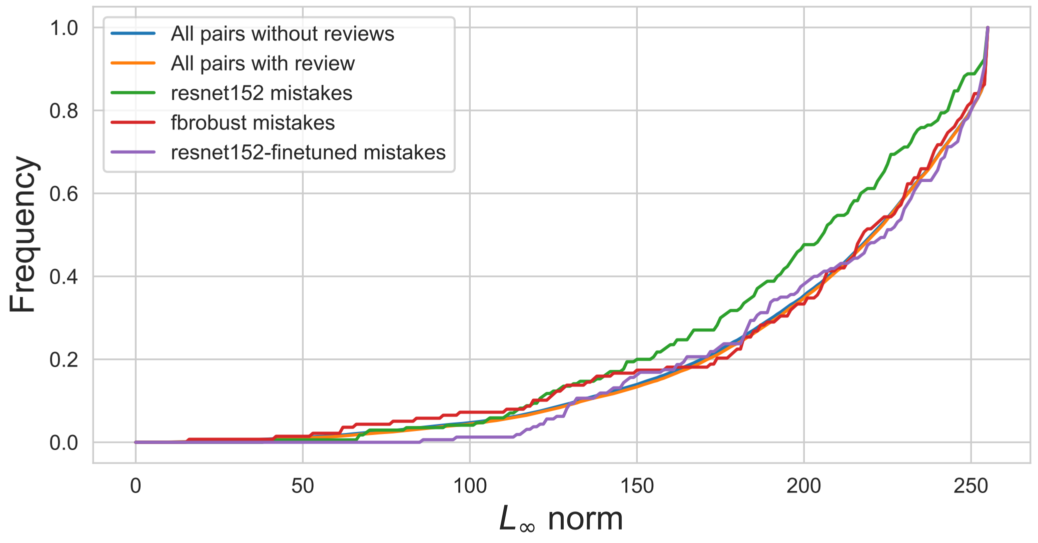


Figure 4.9: CDF showing the ℓ_∞ distance between pairs of frames from different distributions.

4.5.1 PM-k Accuracy with varying k

4.5.2 `ImageNet-Vid-Robust`

In Figure 4.10, we plot the relationship between acc_{pmk} and perturbation distance (i.e., the `k` in the `pm-k` metric). The entire x-axis in Figure 4.10 corresponds to a temporal

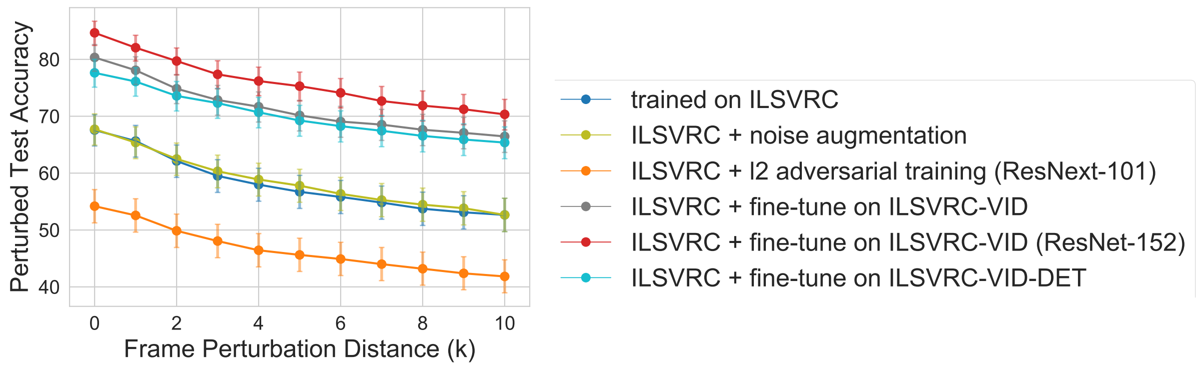


Figure 4.10: Model classification accuracy on perturbed frames as a function of perturbation distance (shown with 95% Clopper-Pearson confidence intervals). Model accuracies from five different model types and the best performing model are shown. The model architecture is ResNet-50 unless otherwise mentioned.

distance of at most 0.3 seconds between the original and perturbed frames.

4.5.3 Model independent perturbed frame selection

We have so far considered *model dependent* perturbations, as we selected the worst neighbor frame *for each model*. Here, we study the same problem but impose a static set of perturbed frames across all models. In Figure 4.11, we study a static set of perturbations across all models and still see a substantial (but smaller) drop in accuracy for both models. The static set of perturbations were chosen by choosing the neighbor frame that the largest number of models misclassified.

4.5.4 Impact of video compression

One concern with analyzing performance on video frames is the impact of video compression on model robustness. In particular, the 'mp4' videos in `ImageNet-Vid-Robust` contain 3 frame types: 'i-', 'p-', and 'b-' frames. 'p-frames' are compressed by referencing pixel content from previous frames, while 'b-frames' are compressed via references to previous and future frames. 'i-frames' are stored without references to other frames.

We compute the original and perturbed accuracies, and the drop in accuracy for a subset of the dataset without each frame type in Table 4.6. While there are modest differences in accuracy due to compression, our analysis suggests that the sensitivity

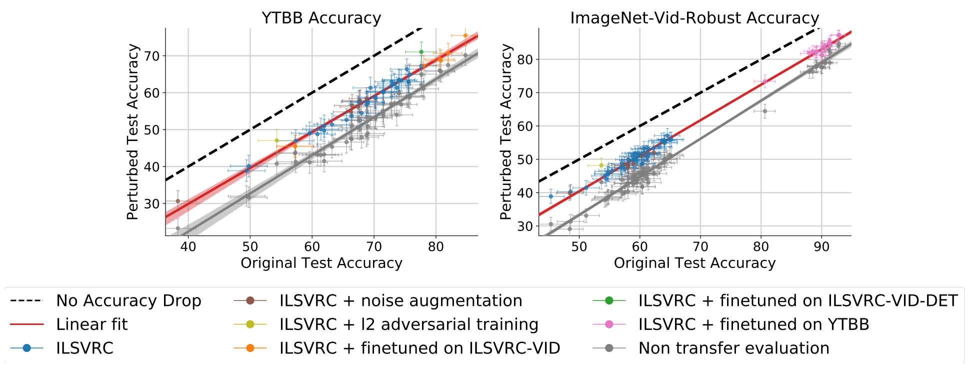


Figure 4.11: Model accuracy on original vs. perturbed images for a static, model-*independent* set of perturbed frames. The grey points and grey linear fit correspond to the perturbed accuracies of models evaluated on per model perturbations studied in Figure 4.3. See Section 4.5.3 for details.

of models is not significantly due to the differences in quality of frames due to video compression.

4.5.5 Impact of dataset review

Table 4.5: Impact of human review on original and perturbed accuracies for `ImageNet-Vid-Robust` and `YTBB-Robust`, numbers come from a ResNet-152 fine-tuned on ImageNet-Vid and Youtube-BB, respectively

		Accuracy		
	Reviewed	Original	Perturbed	Δ
`ImVid-Robust`	✗	80.3	64.1	16.2
	✓	84.8	70.2	14.4
`YTBB-Robust`	✗	88.1	78.1	10.0
	✓	92.9	84.7	8.9

We analyze the impact of our human review, described in Section 4.2.4, on the classifiers in our testbed. First, we compare the original and perturbed accuracies of a representative

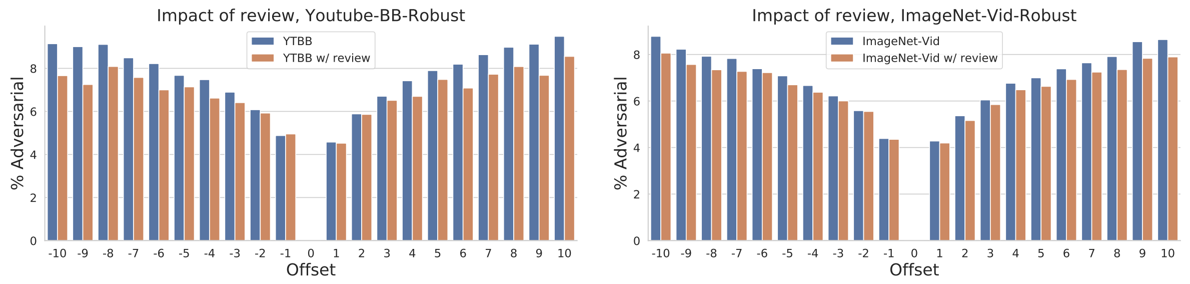


Figure 4.12: We plot the fraction of times each frame offset caused an error, across all evaluated models, for frames with and without review. Frames further away more frequently cause classifiers to misfire. Our review process reduces the number of errors, especially for frames further in time, by removing dissimilar frames.

classifier (ResNet-152 finetuned) with and without review in Table 4.5. Our review improves the original accuracy by 3 to 4% by discarding mislabeled or blurry anchor frames, and improves perturbed accuracy by 5 to 6% by discarding dissimilar frame pairs. Our review reduces the accuracy drop by 1.8% on `ImageNet-Vid-Robust` and 1.1% on `YTBB-Robust`. These results indicate that the changes in model predictions are indeed due to a lack of robustness, rather than due to significant differences between adjacent frames.

To further analyze the impact of our review on model errors, we plot how frequently each offset distance from the anchor frame results in a model error across all model types in Figure 4.12. For both datasets, larger offsets (indicating pairs of frames further apart in time) lead to more frequent model errors. Our review reduces the fraction of errors across offsets, especially for large offsets, which are more likely to display large changes from the anchor frame.

4.5.6 FPS analysis

`ImageNet-Vid-Robust` To analyze the impact of frame-rate on accuracy, we show results on subsets of videos with fixed fps (25, 29, and 30, which cover 89% of the dataset) using a fine-tuned ResNet-152 model in Table 4.7. The accuracy drop is similar across the subsets, and similar to the drop for the whole dataset.

Table 4.6: Analyzing results based on frame-type in video compression. See Section 4.5.4 for details.

	Accuracy			
	Original	Perturbed	Δ	# anchors
All frames	84.8	70.2	14.6	1109
w/o 'i-frames'	84.7	70.3	14.4	1104
w/o 'p-frames'	83.9	73.7	10.2	415
w/o 'b-frames'	85.4	73.2	12.2	699

FPS	Acc. Orig.	Acc. Perturbed	Drop	# Videos
25	87.3 [83.0, 90.9]	73.3 [67.8, 78.3]	14.0	292
29	87.7 [84.0, 90.8]	74.9 [70.3, 79.2]	12.8	383
30	78.3 [73.3, 82.7]	61.7 [56.0, 67.1]	16.6	313

Table 4.7: Results on subsets of ImageNet-Vid-Robust with fixed FPS.

4.5.7 ILSVRC training with `ImageNet-Vid-Robust` classes

We trained ResNet-50 from scratch on ILSVRC using the 30 ImageNet-Vid classes. We also fine-tuned the model on ImageNet-Vid. In Table 4.8, we show the accuracy drops are consistent with models in our submission. We hypothesize that the lower accuracy is due to coarser supervision on ILSVRC.

Model	Acc. Orig.	Acc. Perturbed	Drop
ILSVRC-30	61.0	44.9	15.1
ILSVRC-30 + FT	77.8	59.9	17.9

Table 4.8: Results of training ResNet-50 on ILSVRC with 30 classes from `ImageNet-Vid-Robust`.

4.5.8 Detection pm-k

We briefly introduce the mAP metric for detection here and refer the reader to [101] for further details. The standard detection metric proceeds by first determining whether each predicted bounding box in an image is a true or false positive, based on the intersection over union (IoU) of the predicted and ground truth bounding boxes. The metric then computes the per-category average precision (AP, averaged over recall thresholds) of the predictions across all images. The final metric is reported as the mean of these per-category APs (mAP).

We define the `pm-k` analog of mAP by replacing each anchor frame in the dataset with a nearby frame that minimizes the per-image average precision. Since the category-specific average precision is undefined for categories not present in an image, we minimize the average precision across categories present in each frame rather than the mAP.

4.5.9 Experimental Details & Hyperparameters

All classification experiments were carried out using PyTorch version 1.0.1 on an AWS p3.2xlarge with the NVIDIA V100 GPU. All pretrained models were downloaded from [16] at commit hash `021d97897c9aa76ec759deff43d341c4fd45d7ba`. Evaluations in Tables 4.6.1 and 4.6.2 all use the default settings for evaluation. The hyperparameters for the *fine-tuned* models are presented in Table 4.9. We searched for learning rates between 10^{-3} and 10^{-5} for all models.

We additionally detail hyperparameters for detection models in Table 4.10. Detection experiments were conducted with PyTorch version 1.0.1 on a machine with 4 Titan X GPUs, using the Mask R-CNN benchmark repository[110]. We used the default learning rate provided in [110]. For R-FCN, we used the model trained by [154].

Table 4.9: Hyperparameters for models finetuned on ImageNet-Vid,

Model	Base Learning Rate	Learning Rate Schedule	Batch Size	Epochs
resnet152	10^{-4}	`Reduce LR On Plateau`	32	10
resnet50	10^{-4}	`Reduce LR On Plateau`	32	10
alexnet	10^{-5}	`Reduce LR On Plateau`	32	10
vgg16	10^{-5}	`Reduce LR On Plateau`	32	10

Table 4.10: Hyperparameters for detection models.

Model	Base Learning Rate	Learning Rate Schedule	Batch Size	Iterations
F-RCNN ResNet-50	10^{-2}	`Step 20k, 30k`	8	40k
F-RCNN ResNet-101	10^{-2}	`Step 20k, 30k`	8	40k

4.6 Full Original vs Perturbed Accuracies

4.6.1 `ImageNet-Vid-Robust`

4.6.2 `YTBB-Robust`

4.7 Related work

Adversarial examples. Various forms of adversarial examples have been studied, the majority of research focuses on ℓ_p robustness [13, 60]. However, it is unclear whether adversarial examples pose a problem for robustness outside of a truly worst case context. It is an open question whether perfect robustness against a ℓ_p adversary will induce robustness to realistic image distortions such as those studied in this paper. Recent work has proposed less adversarial image modifications such as small rotations & translations [9, 39, 45, 83], hue and color changes [72], image stylization [53] and synthetic image corruptions such as Gaussian blur and JPEG compression [56, 71]. Even though the above examples are more realistic than the ℓ_p model, they still synthetically modify the input images to generate perturbed versions. In contrast, our work performs no synthetic modification and instead uses images that naturally occur in videos.

Utilizing videos to study robustness. In recent work, Gu et al. [61] exploit the temporal structure in videos to study robustness. However, their experiments suggest a substantially smaller drop in accuracy. The primary reason for this is a less stringent metric used in [61]. By contrast, our PM-k metric is inspired by the "worst-of-k" metric used in prior work [39], highlighting the sensitivity of models to natural perturbations. In the appendix, we study the differences between the two metrics in more detail. Furthermore, the lack of human review and the high label error-rate we discovered in Youtube-BB (Table 4.1) presents a potentially troubling confounding factor that we resolve in our work.

Distribution shift. Small, benign changes in the test distribution are often referred to as *distribution shift*. Recht et al. [128] explore this phenomenon by constructing new test sets for CIFAR-10 and ImageNet and observe substantial performance drops for a

Table 4.11: Classification model perturbed and original accuracies for all models in our test bed evaluated on the ImageNet-Vid-Robust dataset.

Model	Accuracy Original	Accuracy Perturbed	Δ
resnet152_finetuned	84.8 [82.5, 86.8]	70.2 [67.4, 72.8]	14.6
resnet50_finetuned	80.8 [78.3, 83.1]	65.7 [62.9, 68.5]	15.1
vgg16bn_finetuned	78.0 [75.4, 80.4]	61.0 [58.1, 63.9]	17.0
nasnetalarge_imagenet_pretrained	77.6 [75.1, 80.1]	62.1 [59.2, 65.0]	15.5
resnet50_detection	77.6 [75.1, 80.1]	65.0 [62.1, 67.8]	12.6
inceptionresnetv2_imagenet_pretrained	75.7 [73.1, 78.2]	58.7 [55.7, 61.6]	17.0
dpn107_imagenet_pretrained	75.6 [72.9, 78.1]	59.1 [56.1, 62.0]	16.5
inceptionv4_imagenet_pretrained	75.3 [72.6, 77.8]	59.0 [56.0, 61.9]	16.3
dpn92_imagenet_pretrained	74.4 [71.7, 76.9]	56.8 [53.8, 59.7]	17.6
dpn131_imagenet_pretrained	74.0 [71.3, 76.6]	59.9 [56.9, 62.8]	14.1
dpn68b_imagenet_pretrained	73.7 [71.0, 76.2]	54.0 [51.0, 57.0]	19.7
resnext101_32x4d_imagenet_pretrained	73.3 [70.6, 75.9]	57.2 [54.2, 60.1]	16.1
resnext101_64x4d_imagenet_pretrained	72.9 [70.1, 75.5]	56.6 [53.7, 59.6]	16.3
resnet152_imagenet_pretrained	72.8 [70.0, 75.4]	57.0 [54.0, 59.9]	15.8
resnet101_imagenet_pretrained	71.5 [68.7, 74.1]	53.7 [50.8, 56.7]	17.8
fbresnet152_imagenet_pretrained	71.5 [68.7, 74.1]	54.5 [51.5, 57.4]	17.0
densenet161_imagenet_pretrained	71.4 [68.7, 74.1]	55.1 [52.1, 58.1]	16.3
densenet169_imagenet_pretrained	70.2 [67.5, 72.9]	53.1 [50.1, 56.1]	17.1
densenet201_imagenet_pretrained	70.2 [67.5, 72.9]	53.4 [50.4, 56.4]	16.8
dpn68_imagenet_pretrained	69.4 [66.6, 72.1]	53.3 [50.3, 56.3]	16.1
bninception_imagenet_pretrained	69.0 [66.2, 71.7]	49.0 [46.0, 51.9]	20.0
densenet121_imagenet_pretrained	69.0 [66.2, 71.7]	50.9 [47.9, 53.8]	18.1

large suite of models on the newly constructed test sets. Similar to our Figure 4.3, the relationship between their original and new test set accuracies is also approximately linear. However, the images in their test set bear little visual similarity to images in the original test set, while all of our failure cases are on perceptually similar images. In a similar vein of study, [146] studies distribution shift *across* different computer vision data sets such as Caltech-101, PASCAL, and ImageNet.

Temporal Consistency in Computer Vision A common issue when applying image based models to videos is *flickering*, where object detectors spuriously produce

Table 4.12: (Continued) Classification model perturbed and original accuracies for all models in our test bed evaluated on the ImageNet-Vid-Robust dataset.

Model	Accuracy Original	Accuracy Perturbed	Δ
nasnetamobile_imagenet_pretrained	68.8 [66.0, 71.5]	48.4 [45.4, 51.4]	20.4
resnet50_augment___jpeg_compression	68.8 [66.0, 71.5]	53.2 [50.2, 56.2]	15.6
resnet34_imagenet_pretrained	68.0 [65.2, 70.7]	48.0 [45.0, 51.0]	20.0
resnet50_augment___impulse_noise	67.7 [64.9, 70.5]	50.2 [47.2, 53.2]	17.5
resnet50_augment__gaussian_blur	67.7 [64.9, 70.5]	52.5 [49.5, 55.5]	15.2
resnet50_imagenet_pretrained	67.5 [64.7, 70.3]	52.5 [49.5, 55.5]	15.0
resnet50_augment___gaussian_noise	67.4 [64.5, 70.1]	50.6 [47.6, 53.6]	16.8
resnet50_augment___shot_noise	66.5 [63.6, 69.2]	51.1 [48.1, 54.1]	15.4
vgg16_bn_imagenet_pretrained	66.4 [63.5, 69.1]	47.4 [44.5, 50.4]	19.0
resnet50_augment___defocus_blur	66.3 [63.4, 69.1]	47.6 [44.6, 50.6]	18.7
vgg19_bn_imagenet_pretrained	65.6 [62.7, 68.4]	46.6 [43.6, 49.6]	19.0
vgg19_imagenet_pretrained	63.2 [60.3, 66.1]	45.4 [42.4, 48.3]	17.8
resnet18_imagenet_pretrained	61.9 [59.0, 64.8]	41.5 [38.6, 44.4]	20.4
vgg13_bn_imagenet_pretrained	61.9 [59.0, 64.8]	43.3 [40.3, 46.3]	18.6
vgg16_imagenet_pretrained	61.4 [58.5, 64.3]	43.1 [40.2, 46.1]	18.3
vgg11_bn_imagenet_pretrained	60.9 [57.9, 63.8]	43.2 [40.3, 46.2]	17.7
vgg13_imagenet_pretrained	59.6 [56.6, 62.5]	41.1 [38.2, 44.1]	18.5
vgg11_imagenet_pretrained	57.3 [54.4, 60.3]	41.3 [38.4, 44.3]	16.0
alexnet_finetuned	57.3 [54.3, 60.2]	43.6 [40.7, 46.6]	13.7
ResNeXtDenoiseAll-101_robust_pgd	54.3 [51.3, 57.2]	40.8 [37.8, 43.7]	13.5
squeezenet1_1_imagenet_pretrained	49.8 [46.8, 52.8]	31.7 [28.9, 34.5]	18.1
alexnet_imagenet_pretrained	49.4 [46.4, 52.4]	32.0 [29.3, 34.8]	17.4
resnet50_augment___contrast_change	38.3 [35.5, 41.3]	23.3 [20.8, 25.9]	15.0

false-positives or false-negatives in isolated frames or groups of frames. [81] explicitly identify such failures and use a technique reminiscent of adversarially robust training to improve image-based models. A similar line of work focuses on improving object detection in videos as objects become occluded or move quickly [48, 85, 154, 164]. The focus in this line of work has generally been on improving object detection when objects transform in a way that makes recognition difficult from a single frame, such as fast motion or occlusion. In this work, we document a broader set of failure cases for image-based classifiers and detectors and show that failures occur when the neighboring frames are imperceptibly different.

4.8 Conclusion

Our study quantifies the sensitivity of image classifiers to naturally occuring temporal perturbations. These perturbations cause significant drops in accuracy for a wide range of models in both classification and detection. Our work on analyzing this failure mode opens multiple avenues for future research:

Building more robust models. Our `YTBB-Robust` and `ImageNet-Vid-Robust` datasets provide a standard measure for robustness that can be used to evaluate to any classification or detection model. In Table 4.2, we evaluated several commonly used models and found that all of them suffer from substantial accuracy drops due to natural perturbations. In particular, we found that model improvements with respect to artificial perturbations (such as image corruptions or ℓ_∞ adversaries) induce at best modest improvements in robustness. We hope that our standardized datasets and evaluation metric will enable future work to quantify improvements in natural robustness directly.

Further natural perturbations. Videos provide a straightforward method for collecting natural perturbations of images, enabling the study of realistic forms of robustness for machine learning methods. Other methods for generating these natural perturbations are likely to provide additional insights into model robustness. As an example, photo sharing websites contain a large number of near-duplicate images: pairs of images of the same scene captured at different times, viewpoints, or from a different camera [128]. More generally, devising similar, domain-specific strategies to collect, verify, and measure robustness to natural perturbations in domains such as natural language processing or speech recognition is a promising direction for future work.

Table 4.13: Classification model perturbed and original accuracies for all models in our test bed evaluated on the YTBB-robust dataset..

Model	Accuracy Original	Accuracy Perturbed	Δ
resnet152_finetuned	92.9 [91.2, 94.3]	84.7 [82.4, 86.8]	8.2
resnet50_finetuned	91.4 [89.6, 93.0]	82.0 [79.6, 84.2]	9.4
inceptionresnetv2_finetuned	91.3 [89.5, 92.9]	79.0 [76.4, 81.3]	12.3
vgg19_finetuned	90.5 [88.6, 92.2]	79.1 [76.5, 81.4]	11.4
vgg16_finetuned	89.1 [87.1, 90.8]	78.0 [75.4, 80.4]	11.1
inceptionv4_finetuned	88.5 [86.5, 90.3]	76.3 [73.6, 78.7]	12.2
resnet18_finetuned	88.0 [85.9, 89.8]	76.2 [73.6, 78.7]	11.8
alexnet_finetuned	80.6 [78.2, 82.9]	64.4 [61.5, 67.3]	16.2
pnasnet5large_imagenet_pretrained	65.2 [62.3, 68.0]	51.0 [48.0, 54.0]	14.2
nasnetalarge_imagenet_pretrained	64.9 [62.0, 67.7]	51.4 [48.4, 54.4]	13.5
inceptionresnetv2_imagenet_pretrained	64.5 [61.6, 67.4]	50.4 [47.5, 53.4]	14.1
dpn98_imagenet_pretrained	64.1 [61.2, 66.9]	49.0 [46.0, 52.0]	15.1
dpn107_imagenet_pretrained	64.1 [61.2, 66.9]	50.1 [47.2, 53.1]	14.0
dpn131_imagenet_pretrained	64.0 [61.1, 66.8]	49.9 [46.9, 52.9]	14.1
inceptionv4_imagenet_pretrained	63.6 [60.7, 66.4]	48.8 [45.8, 51.8]	14.8
xception_imagenet_pretrained	63.2 [60.2, 66.0]	47.6 [44.6, 50.6]	15.6
dpn92_imagenet_pretrained	62.3 [59.3, 65.1]	47.7 [44.8, 50.7]	14.6
resnet50_augment__jpeg_compressioon	62.3 [59.4, 65.2]	45.7 [42.8, 48.7]	16.6
polynet_imagenet_pretrained	61.4 [58.4, 64.3]	47.3 [44.4, 50.3]	14.1
nasnetamobile_imagenet_pretrained	61.4 [58.4, 64.3]	43.0 [40.1, 46.0]	18.4
resnet50_augment__shot_noise	61.3 [58.3, 64.2]	46.4 [43.4, 49.3]	14.9
dpn68_imagenet_pretrained	61.2 [58.3, 64.1]	44.2 [41.2, 47.2]	17.0
fbresnet152_imagenet_pretrained	61.1 [58.1, 64.0]	45.9 [42.9, 48.8]	15.2
resnet152_imagenet_pretrained	60.8 [57.8, 63.7]	46.5 [43.5, 49.5]	14.3
resnet101_imagenet_pretrained	60.8 [57.8, 63.7]	45.2 [42.2, 48.2]	15.6
senet154_imagenet_pretrained	60.7 [57.7, 63.6]	47.2 [44.3, 50.2]	13.5
resnet50_augment__impulse_noise	60.6 [57.7, 63.5]	45.5 [42.6, 48.5]	15.1
se_resnet101_imagenet_pretrained	60.5 [57.6, 63.4]	45.6 [42.6, 48.6]	14.9
bninception_imagenet_pretrained	60.4 [57.4, 63.3]	41.8 [38.9, 44.7]	18.6
densenet161_imagenet_pretrained	60.2 [57.3, 63.1]	46.4 [43.4, 49.4]	13.8
resnet50_augment__gaussian_noise	60.2 [57.3, 63.1]	45.7 [42.8, 48.7]	14.5
se_resnext50_32x4d_imagenet_pretrained	59.9 [56.9, 62.8]	45.7 [42.7, 48.6]	14.2
dpn68b_imagenet_pretrained	59.7 [56.7, 62.6]	45.9 [42.9, 48.8]	13.8

Table 4.14: (Continued) Classification model perturbed and original accuracies for all models in our test bed evaluated on the YTBB-robust dataset..

Model	Accuracy Original	Accuracy Perturbed	Δ
inceptionv3_imagenet_pretrained	59.6 [56.6, 62.5]	43.8 [40.8, 46.8]	15.8
densenet121_imagenet_pretrained	59.5 [56.5, 62.4]	43.1 [40.1, 46.0]	16.4
se_resnext101_32x4d_imagenet_pretrained	59.2 [56.3, 62.1]	45.2 [42.3, 48.2]	14.0
densenet201_imagenet_pretrained	59.2 [56.2, 62.1]	44.8 [41.8, 47.8]	14.4
densenet169_imagenet_pretrained	59.2 [56.2, 62.1]	44.6 [41.7, 47.6]	14.6
resnet50_augment__brightness_change	58.9 [56.0, 61.8]	42.6 [39.6, 45.5]	16.3
se_resnet50_imagenet_pretrained	58.8 [55.9, 61.7]	44.1 [41.1, 47.1]	14.7
se_resnet152_imagenet_pretrained	58.8 [55.9, 61.7]	44.8 [41.9, 47.8]	14.0
cafferesnet101_imagenet_pretrained	58.2 [55.2, 61.1]	44.3 [41.3, 47.3]	13.9
resnet50_augment__regular	58.0 [55.1, 61.0]	42.9 [39.9, 45.8]	15.1
resnet34_imagenet_pretrained	57.9 [55.0, 60.9]	42.8 [39.8, 45.7]	15.1
vgg19_imagenet_pretrained	57.5 [54.6, 60.5]	40.1 [37.2, 43.1]	17.4
resnet50_augment__gaussian_blur	57.5 [54.5, 60.4]	41.8 [38.9, 44.7]	15.7
vgg16_bn_imagenet_pretrained	57.2 [54.2, 60.1]	39.6 [36.7, 42.6]	17.6
resnet50_imagenet_pretrained	57.0 [54.1, 60.0]	43.8 [40.9, 46.8]	13.2
vgg19_bn_imagenet_pretrained	56.8 [53.9, 59.8]	40.6 [37.7, 43.5]	16.2
vgg16_imagenet_pretrained	55.4 [52.4, 58.4]	40.1 [37.2, 43.1]	15.3
vgg13_bn_imagenet_pretrained	54.8 [51.8, 57.7]	38.6 [35.7, 41.6]	16.2
vgg11_bn_imagenet_pretrained	54.8 [51.8, 57.7]	38.8 [35.9, 41.8]	16.0
vgg11_imagenet_pretrained	54.7 [51.7, 57.6]	38.4 [35.5, 41.3]	16.3
resnet18_imagenet_pretrained	54.4 [51.4, 57.4]	38.1 [35.2, 41.0]	16.3
vgg13_imagenet_pretrained	54.2 [51.3, 57.2]	37.7 [34.9, 40.7]	16.5
ResNeXtDenoiseAll-101_robust_pgd	53.6 [50.7, 56.6]	43.2 [40.2, 46.1]	10.4
squeezenet1_0_imagenet_pretrained	51.1 [48.1, 54.1]	33.1 [30.3, 36.0]	18.0
squeezenet1_1_imagenet_pretrained	48.6 [45.6, 51.6]	31.3 [28.6, 34.2]	17.3
resnet50_augment__defocus_blur	48.4 [45.4, 51.4]	29.1 [26.4, 31.8]	19.3
alexnet_imagenet_pretrained	45.3 [42.4, 48.3]	30.5 [27.8, 33.3]	14.8

Part II

Kernels

Chapter 5

Neural Kernels Without Tangents

5.1 Introduction

In this chapter, we aim to understand empirically if there are computationally tractable kernels that approach the expressive power of neural networks, and if there are any practical links between kernel and neural network architectures. We take inspiration from both the recent literature on "neural tangent kernels" (NTK) and the classical literature on compositional kernels, such as ANOVA kernels. We describe a set of three operations in feature space that allow us to turn data examples presented as collections of small feature vectors into a single expressive feature-vector representation. We then show how to compute these features directly on kernel matrices, obviating the need for explicit vector representations. We draw connections between these operations, the compositional kernels of Daniely et al. [27], and the Neural Tangent Kernel limits of Jacot et al. [80]. These connections allow us to relate neural architectures to kernels in a transparent way, with appropriate simple analogues of convolution, pooling, and nonlinear rectification (Sec. 5.3).

Our main investigation, however, is not in establishing these connections. Our goal is to test whether the analogies between these operations hold in practice: is there a correlation between neural architecture performance and the performance of the associated kernel? Inspired by simple networks proposed by David Page [115], we construct neural network architectures for computer vision tasks using only 3×3 convolutions, 2×2 average pooling, and ReLU nonlinearities. We show that the performance of these neural architectures on CIFAR-10 strongly predicts the performance of the associated kernel. The best architecture achieves 96% accuracy on CIFAR-10 when trained with SGD on a mean squared error (MSE) loss. The corresponding compositional kernel achieves 90% accuracy, which is, to our knowledge, the highest accuracy achieved thus far by a kernel machine on CIFAR-10. We emphasize here that we compute an *exact kernel* directly from pixels, and do not rely on random feature approximations often used in past work.

On CIFAR-10, we observe that compositional kernels provide dramatically better results than Neural Tangent Kernels. We also demonstrate that this trend holds in the "small data" regime [8]. Here, we find that compositional kernels outperform NTKs and neural networks outperform both kernel methods when properly tuned and trained. On a benchmark of 90 UCI tabular datasets, we find that simple, properly tuned Gaussian kernels perform, on aggregate, slightly better than NTKs. Taken together, our results provide a promising starting point for designing practical, high performance, domain specific kernel functions. We suggest that while some notion of compositionality and hierarchy may be necessary to build kernel predictors that match the performance of neural networks, NTKs themselves may not actually provide particularly useful guides to the practice of kernel methods.

5.2 Related Work

We build upon many prior efforts to design specialized kernels that model specific types of data. In classical work on designing kernels for pattern analysis, Shawe-Taylor et al. [136] establishes an algebra for constructing kernels on structured data. In particular, we recall the construction of the ANOVA kernels, which are defined recursively using a set of base kernels. Many ideas from ANOVA kernels transfer naturally to images, with operations that capture the similarities between different patches of data.

More recent work [108] proposes a multi-layer "convolutional kernel network" (CKN) for image classification that iterates convolutional and nonlinear operations in kernel space. While the structure of our compositional kernels is similar to CKNs, Mairal et al. [108] are unable to explore networks deeper than two layers due to the inefficiency of their construction, which involves approximating a kernel feature map using optimization. In contrast, we compute our kernel *exactly*, and the complexity of our compositional kernel functions is worst-case linear in depth, enabling us to explore much deeper kernel compositions.

Another line of recent work investigates the connection between kernel methods and infinitely wide neural networks. Jacot et al. [80] posits that least squares regression with respect to the neural tangent kernel (NTK) is equivalent to optimizing an infinitely wide neural network with gradient flow. Similarly, it has been shown that optimizing just the *last* layer of an infinitely wide neural network is equivalent to a Gaussian process

(NNGP) based on the neural network architecture [96]. Both of these equivalences extend to convolutional neural networks (CNNs) [100, 113]. The compositional kernels we explore can be expressed as NNGPs.

To construct our compositional kernel functions, we rely on key results from Daniely et al. [27], which explicitly studies the duality between neural network architectures and compositional kernels.

5.3 Compositional kernels for bags of features

A variety of data formats are naturally represented by collections of related vectors. For example, an image can be considered a spatially arranged collection of 3-dimensional vectors. A sentence can be represented as a sequence of word embeddings. Audio can be represented as temporally ordered short-time Fourier transforms. In this section, we propose a generalization of these sorts of data types, and a set of operations that allow us to compress these representations into vectors that can be fed into a downstream prediction task. We then show how these operations can be expressed as kernels and describe how to compute them. None of the operations described here are novel, but they form the basic building blocks that we use to build classifiers to compare to neural net architectures.

A *bag of features* is simply a generalization of a matrix or tensor: whereas a matrix is a list of vectors indexed by the natural numbers, a bag of features is a collection of elements in a Hilbert space $\mathcal{H}$ with a finite, structured index set $\mathcal{B}$. As a canonical example, we can consider an image to be a bag of features where the index set $\mathcal{B}$ is the pixel's row and column location and $\mathcal{H}$ is $\mathbb{R}^3$: at every pixel location, there is a corresponding vector in $\mathbb{R}^3$ encoding the color of that pixel. In this section we will denote a generic bag of features by a bold capital letter, e.g., $\boldsymbol{X}$, and the corresponding feature vectors by adding subscripts, e.g., $\boldsymbol{X}_b$. That is, for each index $b \in \mathcal{B}$, $\boldsymbol{X}_b \in \mathcal{H}$.

If our data is represented by a bag of features, we need to map it into a single Hilbert space to perform linear (or nonlinear) predictions. We describe three simple operations to compress a bag of features into a single feature vector.

Concatenation. Let $\mathcal{S}_1, \dots, \mathcal{S}_L \subseteq \mathcal{B}$ be *ordered* subsets with the same cardinality, s. We write each subset as an ordered set of indices: $\mathcal{S}_j = \{i_{j1}, \dots, i_{js}\}$. Then we can define a new bag of features $c(\boldsymbol{X})$ with index set $\{1, \dots, L\}$ and Hilbert space $\mathcal{H}^s$ as follows. For each $j = 1, \dots, L$, set

$$c(\boldsymbol{X})_j = (\boldsymbol{X}_{i_{j1}}, \boldsymbol{X}_{i_{j2}}, \dots, \boldsymbol{X}_{i_{js}})\,.$$

The simplest concatenation is setting $\mathcal{S}_1 = \mathcal{B}$, which corresponds to vectorizing the bag of features. As we will see, more complex concatenations have strong connections to convolutions in neural networks.

Downsampling. Again let $\mathcal{S}_1, \ldots, \mathcal{S}_L \subseteq \mathcal{B}$ be subsets, but now let them have arbitrary cardinality and order. We can define a new bag of features $p(\boldsymbol{X})$ with index set $\{1, \ldots, L\}$ and Hilbert space $\mathcal{H}$. For each $j = 1, \ldots, L$ set

$$p(\boldsymbol{X})_j = \frac{1}{|\mathcal{S}_j|} \sum_{i \in \mathcal{S}_j} \boldsymbol{X}_i \,.$$

This is a useful operation for reducing the size of $\mathcal{B}$. Here we use the letter p for the operation as downsampling is commonly called "pooling" in machine learning.

Embedding. Embedding simply means a isomorphism of one Hilbert space to another. Let $\varphi : \mathcal{H} \to \mathcal{H}'$ be a map. Then we can define a new bag of features $\Phi(\boldsymbol{X})$ with index set $\mathcal{B}$ and Hilbert Space $\mathcal{H}'$ by setting

$$\Phi(\boldsymbol{X})_b = \varphi(\boldsymbol{X}_b)\,.$$

Embedding functions are useful for increasing the expressiveness of a feature space.

5.3.1 Kernels on bags of features

Each operation on a bag of features can be performed directly on the kernel matrix of all feature vectors. Given two bags of features with the same $(\mathcal{B}, \mathcal{H})$, we define the kernel function

$$k(\boldsymbol{X}, a, \boldsymbol{Z}, b) = \langle \boldsymbol{X}_a, \boldsymbol{Z}_b \rangle\,.$$

Note that this implicitly defines a *kernel matrix* between two bags of features: we compute the kernel function for each pair of indices in $\mathcal{B} \times \mathcal{B}$ to form a $|\mathcal{B}| \times |\mathcal{B}|$ matrix. Let us now describe how to implement each of the above operations introduced in Section 5.3.

Concatenation. Since

$$\langle c(\boldsymbol{X})_j, c(\boldsymbol{Z})_k \rangle = \sum_{\ell=1}^{s} \langle \boldsymbol{X}_{i_{j\ell}}, \boldsymbol{Z}_{i_{k\ell}} \rangle\,,$$

we have

$$k(c(\boldsymbol{X}), j, c(\boldsymbol{Z}), k) = \sum_{\ell=1}^{s} k(\boldsymbol{X}, i_{j\ell}, \boldsymbol{Z}, i_{k\ell}) .$$

Downsampling. Similarly, for downsampling, we have

$$k(c(\boldsymbol{X}), j, c(\boldsymbol{Z}), k) = \frac{1}{|\mathcal{S}_j||\mathcal{S}_k|} \sum_{i\in\mathcal{S}_j} \sum_{\ell\in\mathcal{S}_k} k(\boldsymbol{X}, i, \boldsymbol{Z}, \ell) .$$

Embedding. Note that the embedding function φ induces a kernel on $\mathcal{H}$. If $\boldsymbol{x}$ and $\boldsymbol{z}$ are elements of $\mathcal{H}$, define

$$k_\varphi(\boldsymbol{x}, \boldsymbol{z}) = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{z}) \rangle .$$

Then, we don't need to materialize the embedding function to compute the effect of embedding a bag of features. We only need to know k_φ:

$$k(\Phi(\boldsymbol{X}), j, \Phi(\boldsymbol{Z}), k) = k_\varphi(\boldsymbol{X}_j, \boldsymbol{Z}_k) . \tag{5.3.1}$$

We will restrict our attention to φ where we can compute $k_\varphi(\boldsymbol{x}, \boldsymbol{z})$ only from $\langle \boldsymbol{x}, \boldsymbol{z} \rangle$, $\|\boldsymbol{x}\|$ and $\|\boldsymbol{z}\|$. This will allow us to iteratively use Equation (5.3.1) in cascades of these primitive operations.

5.3.2 Kernel operations on images

In this section, we specialize kernel operations to operations on images. As described in Section 5.3, images are bags of three dimensional vectors indexed by two spatial coordinates. Assuming that our images have $D_1 \times D_2$ pixels, we create a sequence of kernels by composing the three operations described above.

Input kernel. The input kernel function k_0 relates all pixel vectors between all pairs of images in our dataset. Computationally, given N images, we can use an image tensor $\boldsymbol{T}$ of shape $N \times D_1 \times D_2 \times 3$ to represent the whole dataset of images, and map this into a kernel tensor $\boldsymbol{K}_{out}$ of shape $N \times D_1 \times D_2 \times N \times D_1 \times D_2$. The elements of $\boldsymbol{K}_{out} = k_0(\boldsymbol{T})$ can be written as:

$$K_{out}[i, j, k, \ell, m, n] = \langle T[i, j, k], T[\ell, m, n] \rangle .$$

All subsequent operations operate on 6-dimensional tensors with the same indexing scheme.

Convolution. The convolution operation c_w maps an input tensor $\boldsymbol{K}_{in}$ to an output tensor $\boldsymbol{K}_{out}$ of the same shape: $N \times D_1 \times D_2 \times N \times D_1 \times D_2$. w is an integer denoting the size of the convolution (e.g. $w = 1$ denotes a 3×3 convolution).

The elements of $\boldsymbol{K}_{out} = c_w(\boldsymbol{K}_{in})$ can be written as:

$$\begin{aligned} &K_{out}[i, j, k, \ell, m, n] = \\ &\sum_{dx=-w}^{w} \sum_{dy=-w}^{w} K_{in}[i, j + dx, k + dy, \ell, m + dx, n + dy] \end{aligned}$$

For out-of-bound location indexes, we simply zero pad the $\boldsymbol{K}_{in}$ so all out-of-bound accesses return zero.

Average pooling. The average pooling operation p_w downsamples the spatial dimension, mapping an input tensor $\boldsymbol{K}_{in}$ of shape $N \times D_1 \times D_2 \times N \times D_1 \times D_2$ to an output tensor $\boldsymbol{K}_{out}$ of shape $N \times (D_1/w) \times (D_2/w) \times N \times (D_1/w) \times (D_2/w)$. We assume D_1 and D_2 are divisible by w.

The elements of $\boldsymbol{K}_{out} = p_w(\boldsymbol{K}_{in})$ can be written as:

$$\begin{aligned} K_{out}[i, j, k, \ell, m, n] = \frac{1}{w^4} \sum_{a=1}^{w} \sum_{b=1}^{w} \sum_{c=1}^{w} \sum_{d=1}^{w} \\ \Big(K_{in}[i, wj + a, wk + b, \ell, wm + c, wn + d] \Big) \end{aligned}$$

Embedding. The nonlinearity layers add crucial nonlinearity to the kernel function, without which the entire map would be linear and much of the benefit of using a kernel method would be lost. We first consider the kernel counterpart of the ReLU activation.

The ReLU embedding, k_{relu}, is shape preserving, mapping an input tensor $\boldsymbol{K}_{in}$ of shape $N \times D_1 \times D_2 \times N \times D_1 \times D_2$ to an output tensor $\boldsymbol{K}_{out}$ of shape $N \times D_1 \times D_2 \times N \times D_1 \times D_2$. To ease the notation, we define two auxiliary tensors: $\boldsymbol{A}$ with shape $N \times D_1 \times D_2$ and $\boldsymbol{B}$ with shape $N \times D_1 \times D_2 \times N \times D_1 \times D_2$, where the elements of each are:

$$\begin{aligned} A[i, j, k] &= \sqrt{K_{in}[i, j, k, i, j, k]} \\ B[i, j, k, \ell, m, n] &= \arccos\left(\frac{K_{in}[i, j, k, \ell, m, n]}{A[i, j, k] A[\ell, m, n]} \right) \end{aligned}$$

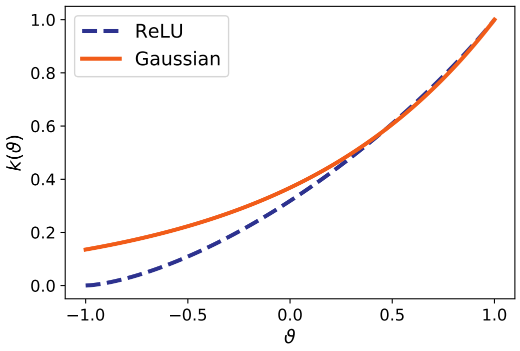


Figure 5.1: Comparison of the ReLU (arccosine) and Gaussian kernels ($\gamma = 1$), as a function of the angle ϑ between two examples.

The elements of $\boldsymbol{K}_{out} = k_{relu}(\boldsymbol{K}_{in})$ can be written as:

$$\begin{aligned}&K_{out}[i,j,k,\ell,m,n]\\ &= \frac{1}{\pi}\Big(A[i,j,k]A[\ell,m,n]\sin(B[i,j,k,\ell,m,n])+\\ &(\pi - B[i,j,k,\ell,m,n])\cos(B[i,j,k,\ell,m,n])\Big)\end{aligned}$$

The relationship between the ReLU operator and the ReLU kernel is covered in Subsection 5.3.3.

In addition to the ReLU kernel, we also work with a normalized Gaussian kernel. The elements of $\boldsymbol{K}_{out} = k_{gauss}(\boldsymbol{K}_{in})$ can be written as:

$$\begin{aligned}&K_{out}[i,j,k,\ell,m,n]\\ &= A[i,j,k]A[\ell,m,n]\exp(B[i,j,k,\ell,m,n]-1)\end{aligned}$$

The normalized Gaussian kernel has a similar output response to the ReLU kernel (shown in Figure 5.1). Experimentally, we find the Gaussian kernel to be marginally faster and more numerically stable.

5.3.3 Relating compositional kernels to neural network architectures

Each of these compositional kernel operations is closely related to neural net architectures, with close ties to the literature on random features [121]. Consider two tensors: $\boldsymbol{U}$ of shape $N \times D_1 \times D_2 \times D_3$ and $\boldsymbol{W}$ of shape $(2w+1) \times (2w+1) \times D_3 \times D_4$. $\boldsymbol{U}$ is the input, which can be N images, w is an integer denoting the size of the convolution (e.g. $w = 1$ denotes a 3×3 convolution), and $\boldsymbol{W}$ is a tensor contains the "weights" of a convolution. Consider a simple convolutional layer followed by a ReLU layer in a neural network:

$$\Psi(\boldsymbol{U}) = \mathrm{relu}(\boldsymbol{W} * \boldsymbol{U})$$

where "$*$" denotes the convolution operation and relu denotes elementwise ReLU nonlinearity.

A convolution operation can be rewritten as a matrix multiplication with a reshaping of input tensors. We first flatten the weights tensor $\boldsymbol{W}$ to a matrix $\boldsymbol{W'}$ of D_4 rows and $D_3(2w+1)^2$ columns. For the input tensor $\boldsymbol{U}$, given the convolution size $(2w+1)\times(2w+1)$, we consider the "patch" of each entry $U[n, d_1, d_2, c]$, which includes the $(2w+1) \times (2w+1)$ entries $U[n, i, j, c]$, where $i \in [d_1 - w, d_1 + w],\ j \in [d_2 - w, d_2 + w]$. Therefore, we can flatten the input tensor $\boldsymbol{U}$ to a matrix $\boldsymbol{U'}$ of size $D_3(2w + 1)^2 \times D_1 D_2 N$ by padding all out-of-bounds entries in the patches to zero.

The ReLU operation is shape preserving, applying the ReLU nonlinearity $\varphi(x)$ elementwise to the tensor. Thus we can rewrite the above convolution and ReLU operations into

$$\Psi(\boldsymbol{U}) = \mathrm{relu}(\boldsymbol{W'}\boldsymbol{U'}) = \mathrm{relu}(\boldsymbol{W} * \boldsymbol{U})$$

Therefore, a simple convolution layer and a ReLU layer give us an output tensor $\Psi(\boldsymbol{U})$ of shape $N \times D_1 \times D_2 \times D_4$.

With the help of random features, we are able to relate the above neural network architecture to kernel operations. Suppose the entries of $\boldsymbol{W}$ are appropriately scaled random Gaussian variables. We can evaluate the following expectation according to the calculation in Daniely et al. [27], thereby relating our kernel construction to inner products between the outputs of *random* neural networks:

$$\begin{aligned}\mathbb{E}\left[\sum_{c=1}^{D_4} \Psi(\boldsymbol{U})[i,j,k,c]\Psi(\boldsymbol{U})[\ell,m,n,c]\right] = \\ k_{relu}\Big(c_w\big(k_0(\boldsymbol{U})\big)\Big)[i,j,k,\ell,m,n]\end{aligned} \tag{5.3.2}$$

where k_0 is the input kernel defined in Subsection 5.3.2. We include the proof for the above equality in the appendix.

Similar calculations can be made for the pooling operation, and for any choice of nonlinearity for which the above expectation can be computed. Moreover, since in Eq (5.3.2), the term inside the expectation only depends on inner products, this relation can be generalized to arbitrary depths.

5.3.4 Implementation

Now we actualize the above formulations into a procedure to generate a kernel matrix from the input data. Let $\mathcal{A}$ be a set of valid neural network operations. A given network architecture $\mathcal{N}$ is represented as an ordered list of operations from $\mathcal{A}$. Let $\mathcal{K}$ denote a mapping from elements of $\mathcal{A}$ to their corresponding operators as defined in Subsection 5.3.2.

Algorithm 1 defines the procedure for constructing a compositional kernel from a given architecture $\mathcal{N}$ and an input tensor $\boldsymbol{X}$ of N RGB images of shape $N \times D \times D \times 3$. We note that the output kernel is only a $N \times N$ matrix if there exist exactly $\log D$ pooling layers. We emphasize that this procedure is a deterministic function of the input images and network architecture.

Due to memory limitations, in practice we compute the compositional kernel in batches on a GPU. Implementation details are given in Section 5.4.

Algorithm 1 Compositional Kernel

```
  Input
      $\mathcal{N}$   Input architecture of $m$ layers from $\mathcal{A}$
      $\mathcal{K}$   Map from $\mathcal{A}$ to layerwise operators
      $\boldsymbol{X}$   Tensor of input images, shape $(N \times D \times D \times 3)$
  Output
      $\boldsymbol{K}_m$ Compositional kernel matrix, shape $(N \times N)$
  $\boldsymbol{K}_0 = k_0(\boldsymbol{X})$
  for $i = 1$ to $m$ do
      $k_i \leftarrow \mathcal{K}(\mathcal{N}_i)$
      $\boldsymbol{K}_i \leftarrow k_i(\boldsymbol{K}_{i-1})$
  end for
```

5.4 Experiments

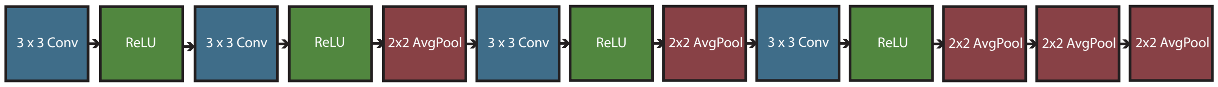


Figure 5.2: A 5 layer network from the "Myrtle" family (Myrtle5).

In this section, we first provide an overview of the architectures used in our experiments. We then present comparison results between neural networks, NTKs, and compositional kernels on a variety of datasets, including MNIST, CIFAR-10 (Krizhevsky [91]), CIFAR-10.1 (Recht et al. [128]), CIFAR-100 (Krizhevsky [91]) and 90 UCI datasets (Fernández-Delgado et al. [49]).

5.4.1 Architectures

We design our deep convolutional kernel based on the non-residual convolutional "Myrtle" networks introduced in Page [115]. We choose this particular network because of its rare combination of simplicity and high performance. Many components commonly used in neural networks, including residual connections, are intended to ease training but have little or unclear effect in terms of the function of the trained network. It is unclear how to model these neural network components in the corresponding kernels, but equally unclear what benefit this might offer. We further simplify the architecture by removing batch normalization and swapping out max pooling with average pooling, for similar reasons. The remaining components are exclusively 3×3 convolutions, 2×2 average pools, and ReLUs. More generally, we refer to all architectures that can be represented as a list of operations from the set {`conv3`, `pool2`, `relu`} as the "Myrtle" family.

We work with 3 networks from this family: Myrtle5, Myrtle7 and Myrtle10, denoting the depth of each network. An example of the Myrtle5 architecture is shown in Figure 5.2. The deeper variants have more convolution and ReLU layers; we refer the reader to the appendix for an illustration of the exact architectures. Next we show convolutional neural networks from this family can indeed achieve high accuracy on CIFAR-10, as can their kernel counterparts.

5.4.2 Experimental setup.

We implemented all the convolutional kernels in the tensor comprehensions framework [149] and executed them on V100 GPUs using Amazon Web Services (AWS) P3.16xlarge instances. For image classification tasks (MNIST, CIFAR-10, CIFAR-10.1, and CIFAR-100), we used compositional kernels based on the Myrtle family described above. For tabular datasets (90 UCI datasets), we used simpler Gaussian kernels. All experiments on CIFAR-10, CIFAR-10.1 and CIFAR-100 used ZCA whitening as a preprocessing step, except for the comparison experiments explicitly studying preprocessing. We apply “flip" data augmentation to our kernel method by flipping every example in the training set across the vertical axis and constructing a kernel matrix on the concatenation of the flipped and standard datasets.

For all image classification experiments (MNIST, CIFAR-10, CIFAR-10.1, and CIFAR-100) we perform kernel ridge regression with respect to one-hot labels, and solve the optimization problem exactly using a Cholesky factorization. More details are provided in the appendix. For experiments on the UCI datasets, we minimize the hinge loss with libSVM to appropriately compare with prior work [8, 49].

5.4.3 MNIST

As a “unit test," we evaluate the performance of the compositional kernels in comparison to several baseline methods, including the Gaussian kernel, on the MNIST dataset of handwritten digits [94]. Results are presented in Table 5.1. We observe that all convolutional methods show nearly identical performance, outperforming the three non-convolutional methods (NTK, arccosine kernel, and Gaussian kernel).

Table 5.1: Classification performance on MNIST. All methods with convolutional structure have essentially the same performance.

Method	MNIST Accuracy
NTK	98.6
ArcCosine Kernel	98.8
Gaussian Kernel	98.8
Gabor Filters + Gaussian Kernel	99.4
LeNet-5 [93]	99.2
CKN [108]	99.6
Myrtle5 Kernel	99.5
Myrtle5 CNN	99.5

5.4.4 CIFAR-10

Table 5.3 compares the performance of neural networks with various depths and their corresponding compositional kernels on both the 10,000 test images from CIFAR-10 and the additional 2,000 “harder" test images from CIFAR-10.1[1] [91, 128]. We include the performance of the Gaussian kernel and a standard ResNet32 as baselines. We train all the Myrtle CNNs on CIFAR-10 using SGD and the mean squared error (MSE) loss with multi-step learning rate decay. The exact hyperparameters are provided in the appendix.

We observe that a simple neural network architecture built exclusively from 3×3 convolutions, 2×2 average pooling layers, and ReLU nonlinearities, and trained with only flip augmentation, achieves 93% accuracy on CIFAR-10. The corresponding fixed compositional kernel achieves 90% accuracy on the same dataset, outperforming all previous kernel methods. We note the previous best-performing kernel method from Li et al. [100] heavily relies on a data dependent feature extraction before data is passed into the kernel function [20]. When additional sources of augmentation are used, such as cutout and random crops, the accuracy of the neural network increases to 96%. Unfortunately due to the quadratic dependence on dataset size, it is currently intractable to augment the compositional kernel to the same extent. For all kernel results[2] on CIFAR-10, we gained a performance improvement of roughly 0.5% using two techniques: Leave-One-Out tilting and ZCA augmentation we detail these techniques in appendix 5.6.

Effect of preprocessing. For all of our primary CIFAR-10 experiments, we begin with ZCA pre-processing [59]. Table 5.3 also shows the accuracy of our baseline CNN and its corresponding kernel when we replace ZCA with a simpler preprocessing of mean subtraction and standard deviation normalization. We find a substantial drop in accuracy for the compositional kernel without ZCA preprocessing, compared to a much more modest drop in accuracy for the CNN. This result underscores the importance of proper preprocessing for kernel methods; we leave improvements in this area for future work.

5.4.5 CIFAR-100

For further evaluation, we compute the compositional kernel with the best performance on CIFAR-10 on CIFAR-100. We report our results in Table 5.2. We find the compositional kernel to be modestly performant on CIFAR-100, matching the accuracy of a CNN of the same architecture when no augmentation is used. However we note this might be due to

[1]As this dataset was only recently released, some works do not report accuracy on this dataset.
[2]with the exception of the experiment performed without ZCA processing

training instability as the network performed more favorably after flip augmentation was used. Accuracy further increased when batch normalization was added, lending credence to the training instability hypothesis. We also note cross entropy loss was used to achieve the accuracies in Table 5.2, as we had difficulty optimizing MSE loss on this dataset. We leave further investigations on the intricacies of achieving high accuracy on CIFAR-100 for future work.

Table 5.2: Accuracy on CIFAR-100. All CNNs were trained with cross entropy loss.

Method	CIFAR-100 Accuracy
Myrtle10-Gaussian Kernel	65.3
Myrtle10-Gaussian Kernel + Flips	68.2
Myrtle10 CNN	64.7
Myrtle10 CNN + Flips	71.4
Myrtle10 CNN + BatchNorm	70.3
Myrtle10 CNN + Flips + BatchNorm	74.7

5.4.6 Subsampled CIFAR-10

In this section, we present comparison results in the small dataset regime using subsamples of CIFAR-10, as investigated in Arora et al. [8]. Results are shown in Figure 5.3. Subsampled datasets are class balanced, and standard deviations are computed over 20 random subsamples, as in Arora et al. [8]. More details are provided in the appendix.

Results. We demonstrate that in the small dataset regime explored in Arora et al. [8], our convolutional kernels significantly outperform the NTK on subsampled training sets of CIFAR-10. We find a network with the same architecture as our kernel severely underperforms both the compositional kernel and NTK in the low data regime. As with CIFAR-100 we suspect this is a training issue as once we add batch normalization the network outperforms both our kernel and the NTK from Arora et al. [8].

5.4.7 UCI datasets

In this section, we present comparison results between the Gaussian kernel and NTK evaluated on 90 UCI datasets, following the setup used in Arora et al. [8]. Arora et al. [8]

Table 5.3: Classification performance on CIFAR-10.

Method	CIFAR-10 Accuracy	CIFAR-10.1 Accuracy
Gaussian Kernel	57.4	-
CNTK + Flips [100]	81.4	-
CNN-GP + Flips [100]	82.2	-
CKN [108]	82.2	-
Coates-NG + Flips [128]	85.6	73.1
Coates-NG + CNN-GP + Flips [100]	88.9	-
ResNet32	92.5	84.4
Myrtle5 Kernel + No ZCA	77.7	62.2
Myrtle5 Kernel	85.8	71.6
Myrtle7 Kernel	86.6	73.1
Myrtle10 Kernel	87.5	74.5
Myrtle10-Gaussian Kernel	88.2	75.1
Myrtle10-Gaussian Kernel + Flips	89.8	78.3
Myrtle5 CNN + No ZCA	87.8	75.8
Myrtle5 CNN	89.8	79.0
Myrtle7 CNN	90.2	79.7
Myrtle10 CNN	91.2	79.9
Myrtle10 CNN + Flips	93.4	84.8
Myrtle10 CNN + Flips + CutOut + Crops	96.0	89.8

identifies that the NTK outperforms a variety of classifiers, including the Gaussian kernel, random forests (RF), and polynomial kernels, evaluated in Fernández-Delgado et al. [49] on 90 UCI datasets.

Results. For appropriate comparison, we use the same set of 90 “small" UCI datasets (containing no more than 5000 data points) as in Arora et al. [8] for the evaluations. For the tuning and evaluation procedure we make one crucial modification to the evaluation procedure posed in Arora et al. [8] and Fernández-Delgado et al. [49]. We compute the optimal hyperparameters for each dataset (for both NTK and Gaussian kernel) by averaging performance over *four* cross-validation folds, while both Arora et al. [8] and Fernández-Delgado et al. [49] choose optimal hyper parameters on a *single* cross validation

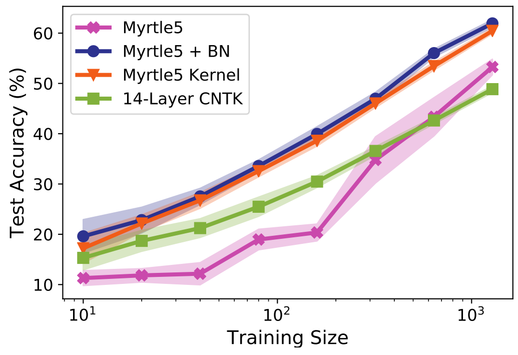


Figure 5.3: Accuracy results on random subsets of CIFAR-10, with standard deviations over 20 trials. The 14-layer CNTK results are from Arora et al. [8].

Table 5.4: Results on 90 UCI datasets for the NTK and Gaussian kernel (both tuned over 4 eval folds).

Classifier	Friedman Rank	Average Accuracy (%)	P90 (%)	P95 (%)	PMA (%)
SVM NTK	14.3	83.2 ± 13.5	96.7	83.3	97.3 ± 3.8
SVM Gaussian kernel	11.6	83.4 ± 13.4	95.6	83.3	97.5 ± 3.7

fold. Using a single cross validation fold can lead to high variance in final performance, especially when evaluation is done purely on small datasets. A single fold was used in the original experimental setup of Fernández-Delgado et al. [49] for purely computational reasons, and the authors point out the issue of high variance hyperparameter optimization. Table 5.4 reports the average cross-validation accuracy over the 90 datasets for the NTK and Gaussian kernel. Compared to results in Arora et al. [8], the modified evaluation protocol increases the performance of both methods, and the gap between the NTK and Gaussian kernel disappears.

We compute the same metrics used in Arora et al. [8]: Friedman rank, P90, P95 and PMA, where a better classifier is expected to have lower Friedman rank and higher P90, P95, and PMA. The average accuracy is reported together with its standard deviation. Friedman rank denotes the ranking metric introduced to compare classifiers across multiple datasets in Demšar [30], and reports the average ranking of a given classifier compared to all other classifiers. P90/P95 denotes the percentage of datasets on which the classifier achieves at least 90%/95% of the maximum accuracy across all classifiers for this dataset.

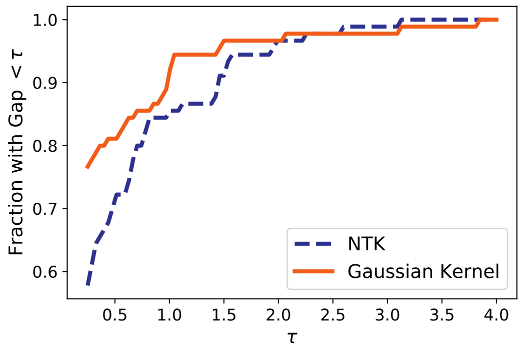


Figure 5.4: Performance profiles for NTK and tuned Gaussian kernel on 90 UCI datasets.

PMA denotes the average percentage of the maximum accuracy across all classifiers for each dataset.

On all metrics reported by Arora et al. [8], the Gaussian kernel has comparable or better performance relative to the NTK. Figure 5.4 shows a performance profile to visually compare the two classifiers [34]. For a given τ, the y axis denotes the fraction of instances where a classifier either has the highest accuracy or has accuracy within τ of the best accuracy. The performance profile reveals that the Gaussian kernel and NTK perform quite comparably on the 90 UCI datasets.

5.5 Limitations and Future Work

The compositional kernels proposed in this manuscript significantly advance the state of the art of kernel methods applied to pattern recognition tasks. However, these kernels still have significant limitations that must be addressed before they can be applied in practice.

Computational cost. The compositional kernels we study compare all pairs of input pixels for two images with D pixels each, so the cost of evaluating the kernel function on two data points is $\tilde{O}(D^2)$. In addition, $O(N^2)$ kernel evaluations must be computed to construct the full kernel matrix, creating a total complexity of $\tilde{O}(N^2D^2)$. Even with heavily optimized GPU code, this requires significant computation time. We therefore limited our scope to image datasets with a small pixel count and modest number of examples: CIFAR-10/CIFAR-100 consist of $60,000$ $32 \times 32 \times 3$ images and MNIST consists of $70,000$ 28×28 images. Even with this constraint, the largest compositional kernel matrices we study took approximately 1000 GPU hours to compute. Thus, we believe an imperative direction of future work is reducing the complexity of each kernel evaluation. Random feature methods or other compression schemes could play a significant role here.

Once a kernel matrix is constructed, exact minimization of empirical risk often scales as $O(N^3)$. For datasets with less than 100,000 examples, these calculations can be performed relatively quickly on standard workstations with sufficient RAM. However, even these solves are expensive for larger datasets. Fortunately, recent work on kernel optimization [25, 105, 134, 150] paves a way to scale our approach to larger datasets.

Data augmentation. A major advantage of neural networks is that data augmentation can be added essentially for free. For kernel methods, data augmentation requires treating each augmented example as if it was part of the data set, and hence computation scales superlinearly with the amount of augmentation: if one wants to perform 100 augmentations per example, then the final kernel matrix will be 10,000 times larger, and solving the prediction problem may be one million times slower. Finding new paths to cheaply augment kernels [28, 124] or to incorporate the symmetries implicit in data augmentation explicitly in kernels should dramatically improve the effectiveness of kernel methods on contemporary datasets. One promising avenue is augmentation via kernel ensembling, e.g. by forming many smaller kernels with augmented data and averaging their predictions appropriately.

Architectural modifications. We consider a simple set of architectural building blocks (convolution, average pool, and ReLU) in this work, but there exist several commonly used primitives in deep networks that have no clear analogues for kernel machines (e.g residual connections, max pool, batch normalization, etc.). While it is unclear whether these primitives are necessary, the question remains open whether the performance gap between kernels and neural networks indicates a fundamental limitation of kernel methods or merely an engineering hurdle that can be overcome (e.g. with improved architectures or by additional subunits).

5.6 Appendix: LOO Tilting and ZCA Augmentation

Two additional techniques were used for the Cifar-10 experiments for an additional 0.5% performance improvement in test accuracy.

5.6.1 ZCA Augmentation

As mentioned in Section 5.5, incorporating augmentation directly is difficult for kernel methods. To capture a small fraction of the benefit of the augmentation in the preprocessing method itself, we first augment the data 20 times using the random augment method proposed in Cubuk et al. [24].

We then learn the ZCA preprocessing matrix by computing the eigendecomposition of the augmented training data as described in [59]. We then use just the portion of the preprocessed data matrix corresponding to the regular unaugmented training data (or corresponding to the unaugmented training data and its horizontal flips) to compute the kernel matrix.

We find this technique offered a small performance boost of around 0.2% across CIFAR-10 and CIFAR-10.

5.6.2 Leave-One-Out Tilting

We additionally found a minor improvement in prediction by averaging the predictions from the true labels and from the labels imputed by leave-one-out prediction. With $\boldsymbol{K}$ and $\boldsymbol{Y}$ defined as they are in section 1.2.1, let $\boldsymbol{Q}$ be $(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}$ and $\boldsymbol{\alpha}$ be $(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\boldsymbol{Y}$. $\boldsymbol{\alpha}$ are the coefficients for standard ridge regression.

Let $\boldsymbol{Y_{loo}}$ be a $N \times C$ matrix of leave-one-out predictions. Here, the ith row of $\boldsymbol{Y}_{loo}$ is the output of ridge regression where we predict example i using every element in the entire training set *except* example i. For kernel ridge regression, we can actually compute the leave-one-out prediction matrix in closed form:

$$Y_{loo_{ic}} = Y_{ic} - \frac{Q_{ii}}{\alpha_{ic}}$$

Our titled prediction uses an affine combination of the true labels $\boldsymbol{Y}$ and the imputed leave-one-out predictions $\boldsymbol{Y}_{loo}$:

$$\boldsymbol{\alpha}_{loo} = (\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}(\boldsymbol{Y} - t\boldsymbol{Y}_{loo})$$

Where t is chosen to maximize test accuracy on CIFAR-10. We empirically find the optimal value of t to be 0.3. Though we do not yet have a theoretical justification for this method, we found that this solution never reduced test error, and always performed well on the test set CIFAR-10.1. For our best model, we found this technique offered a modest performance boost of around 0.3% on both CIFAR-10 and CIFAR-10.1. We leave an analysis of the efficacy of this technique to future work.

5.7 Appendix: Supplementary proof

For completeness, we present the proof details for section 3.3 using results from [27].

We aim to prove the following equality:

$$\begin{aligned}&\mathbb{E}\left[\sum_{c=1}^{D_4}\Psi(\boldsymbol{U})[i,j,k,c]\Psi(\boldsymbol{U})[\ell,m,n,c]\right]=\\&k_{relu}\Big(c_w\big(k_0(\boldsymbol{U})\big)\Big)[i,j,k,\ell,m,n]\end{aligned}\qquad(5.7.1)$$

where Ψ, $\boldsymbol{U}$ and k_0 are as defined in the main text.

Daniely et al. [27] (Section 4.2) presents the concepts of dual activation and kernel:

$$\hat{\sigma}(\rho)=\mathbb{E}_{(X,Y)\sim N_\rho}[\sigma(X)\sigma(Y)]$$

where we denote by N_ρ the multivariate Gaussian distribution with mean 0 and covariance matrix $\begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}$.

In our case, we have the *relu* activation $\sigma(\cdot)=\max(x,0)$, whose dual activation function takes the form $\hat{\sigma}(\rho)=\frac{\sqrt{1-\rho^2}+\rho(\pi-\cos^{-1}(\rho))}{\pi}$ [27].

Now we show how the above result translates to equation 5.3.2. Recall that for a convolutional layer followed by a ReLU layer, we have tensors $\boldsymbol{U}$ with shape $N\times D_1\times D_2\times D_3$, and $\boldsymbol{W}$ with shape $(2w+1)\times(2w+1)\times D_3\times D_4$. To ease the notation, denote $\boldsymbol{Z}$ as the tensor $\boldsymbol{W}*\boldsymbol{U}$ with shape $N\times D_1\times D_2\times D_4$. We begin with the LHS of equation 5.3.2:

$$\begin{aligned}&\mathbb{E}\left[\sum_{c=1}^{D_4}\Psi(\boldsymbol{U})[i,j,k,c]\Psi(\boldsymbol{U})[\ell,m,n,c]\right]\\=&\mathbb{E}\left[\sigma\Big(\sqrt{D_4}(\boldsymbol{Z}[i,j,k,1])\Big)\sigma\Big(\sqrt{D_4}(\boldsymbol{Z}[l,m,n,1])\Big)\right]\end{aligned}$$

Let $X = \sqrt{D_4}(\boldsymbol{Z}[i,j,k,1])$, $Y = \sqrt{D_4}(\boldsymbol{Z}[l,m,n,1])$, and choose entries of W to be independent and identically distributed Gaussian random variables with mean 0 and variance $\frac{1}{D_4}$. With normalization on every patch, we have that (X, Y) follow the multivariate Gaussian distribution with mean 0 and covariance matrix $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$, where

$$\rho = \sum_{\delta=-w}^{w} \sum_{\Delta=-w}^{w} \sum_{d=1}^{D_3} U[i, j+\delta, k+\Delta, d] U[l, m+\delta, n+\Delta, d]$$

Following Daniely et al. [27], we have:

$$\begin{aligned} &\mathbb{E}\Big[\sum_{c=1}^{D_4} \Psi(\boldsymbol{U})[i,j,k,c]\Psi(\boldsymbol{U})[\ell,m,n,c]\Big] \\ =&\frac{\sqrt{1-\rho^2} + \rho(\pi - \cos^{-1}(\rho))}{\pi} \end{aligned} \tag{5.7.2}$$

Now we show that the RHS of equation 5.3.2 is indeed $\frac{\sqrt{1-\rho^2}+\rho(\pi-\cos^{-1}(\rho))}{\pi}$.
As defined in Subsection 5.3.2,

$$k_0(\boldsymbol{U})[i,j,k,l,m,n] = \sum_{d=1}^{D_3} U[i,j,k,d]U[l,m,n,d]$$

.

Let $\boldsymbol{C}$ be the tensor $c_w\Big(k_0(\boldsymbol{U})\Big)$,

$$\begin{aligned} &C[i,j,k,l,m,n] \\ =&\sum_{\delta=-w}^{w} \sum_{\Delta=-w}^{w} \sum_{d=1}^{D_3} U[i, j\pm\delta, k\pm\Delta, d] U[l, m\pm\delta, n\pm\Delta, d] \end{aligned}$$

With normalization for every patch, $\sqrt{C[i,j,k,i,j,k]} = 1$, and

$$\begin{aligned} &k_{relu}\Bigg(c_w\Big(k_0(\boldsymbol{U})\Big)\Bigg) \\ =\ &\frac{1}{\pi}\Bigg(\sqrt{1-\rho^2} + \rho(\pi - \cos^{-1}(\rho))\Bigg) \end{aligned} \tag{5.7.3}$$

Combining (5.7.2) and (5.7.3) completes the proof.

5.8 Appendix: Neural Network Parameters

The parameters used to train neural networks for the experiments in this paper are as follows:

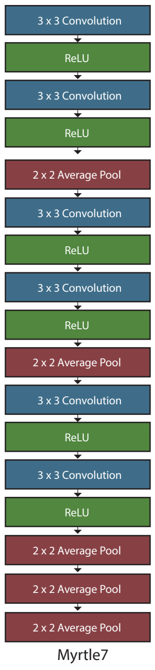
For Myrtle5 on MNIST, we used a width of 1,024 filters for all layers and trained for 20 epochs using MSE loss and Adam as the optimizer with a learning rate of 0.001, without weight decay, and without any form of data preprocessing. For the Myrtle5 Kernel on MNIST we used a regularization value (λ) of 1e-4.

For CIFAR-10, all experiments are trained using MSE loss and SGD with Nesterov momentum, setting weight decay to 0.0005, momentum to 0.9, and minibatch size to 128. All experiments using Myrtle5 used 1,024 filters for all layers and trained for 60 epochs at half-precision with an initial learning rate of 0.1, which is decayed by 0.1 at 15, 30, and 45 epochs. Myrtle7, Myrtle10 without augmentation, and Myrtle10 with flips used 1,024 filters and are trained for 200 epochs with an initial learning rate of 0.05, which is decayed by 0.1 at 80, 120, and 160 epochs. Myrtle10 with flips, cutout, and random crops used 2,048 filters and is trained for 400 epochs with an initial learning rate of 0.1, which is decayed by 0.1 at 80, 160, 240, and 320 epochs. For all CIFAR-10 kernel experiments we used a regularization value (λ) of 0.

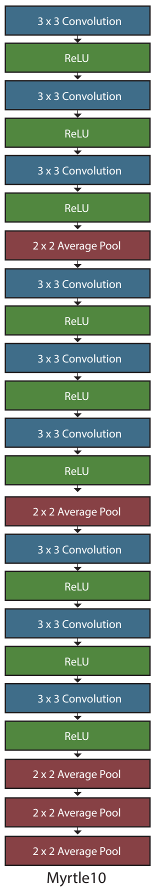
For CIFAR-100, all experiments use a width of 2,048 filters for all layers and are trained for 200 epochs using cross entropy loss and SGD with Nesterov momentum, setting weight decay to 0.0005, momentum to 0.9, and minibatch size to 128. The learning rate is decayed by 0.2 at 60, 120, and 160 epochs. The initial learning rate is set to 0.1 for both experiments with batch normalization, 0.05 for Myrtle10 CNN with flips, and 0.01 for Myrtle10 CNN without augmentation. For all CIFAR-100 kernel experiments we used a regularization value (λ) of 1e-4.

5.9 Appendix: Neural Network Architectures

In Figure 5.5 we illustrate the two “deeper" Myrtle architectures used. The architectures are similar to the 5 layer variant illustrated in main text, except with more convolution and nonlinearity layers.



(a)



(b)

Figure 5.5: a) 7 layer b) 10 layer variants of the Myrtle architectures

5.10 Appendix: Subsampled CIFAR-10 experiments details

Subsets of CIFAR-10 were selected uniformly at random without replacement, and each experiment was repeated in 20 independent trials (over which we report standard deviations). This procedure, and the sizes of training sets we consider, match the setup in [8]. Table 5.5 compares the performance (on all 10,000 test examples) of the CNTK from [8] with that of our Myrtle5 kernel and CNN (with and without batch normalization), our Myrtle10-Gaussian kernel, and a baseline linear classifier. Each linear model used a regularization parameter λ tuned on a log scale between 10^{-4} and 10^{6}. The optimal values for λ from top to bottom were: $10^2, 10^{-2}, 10^2, 10^3, 10^3, 10^3, 10^5, 10^4$. The Myrtle10-Gaussian kernel is our highest-performing unaugmented kernel on the full CIFAR-10 dataset; here we confirm that it retains high performance in the small-data regime.

Training Size	CNTK	Myrtle5 CNN	Myrtle5 CNN + BN	Myrtle5 Kernel	Myrtle10-G Kernel	Linear
10	15.33 ± 2.43	11.29 ± 1.48	19.60 ± 3.32	17.22 ± 2.95	19.15 ± 1.94	12.94 ± 0.74
20	18.79 ± 2.13	11.83 ± 1.34	22.82 ± 2.56	22.16 ± 1.69	21.65 ± 2.97	13.54 ± 0.69
40	21.34 ± 1.91	12.16 ± 2.20	27.53 ± 1.61	26.74 ± 1.56	27.20 ± 1.90	14.66 ± 0.60
80	25.48 ± 1.91	18.96 ± 2.04	33.58 ± 1.22	32.56 ± 1.12	34.22 ± 1.08	15.54 ± 0.61
160	30.48 ± 1.17	20.36 ± 1.68	39.96 ± 1.44	38.61 ± 1.06	41.89 ± 1.34	17.15 ± 0.64
320	36.57 ± 0.88	34.79 ± 4.60	46.96 ± 1.29	46.03 ± 0.82	50.06 ± 1.06	19.18 ± 0.71
640	42.63 ± 0.68	43.36 ± 3.80	56.03 ± 0.80	53.45 ± 0.80	57.60 ± 0.48	22.30 ± 0.57
1280	48.86 ± 0.68	53.27 ± 1.55	61.94 ± 0.74	60.46 ± 0.58	64.40 ± 0.48	25.64 ± 0.61

Table 5.5: Accuracy results (%) on random subsets of CIFAR-10, with standard deviations over 20 resamplings. Myrtle10-G denotes the Myrtle10-Gaussian kernel, our best-performing kernel on the full CIFAR-10 dataset which retains its high performance in the small-data regime. The shallower Myrtle5 CNN trained with batch normalization has similar performance to the corresponding compositional kernel, both of which outperform the CNTK and the Myrtle5 CNN trained without batch normalization.

Chapter 6

Transcription Factor Binding Site Prediction

6.1 Problem Setup

Understanding binding affinity between proteins and DNA sequence is a crucial step in deciphering the regulation of gene expression. Specifically, characterizing the binding affinity of transcription factor proteins (TFs) to DNA sequence determines the relative expression of genes downstream from a TF binding site.

The recent advent of sequencing technologies, such as chromatin immunoprecipitation with massively parallel DNA sequencing (ChIP-seq), provides us with genome-wide binding specificities for 187 TFs across 98 cellular contexts of interest from the ENCODE consortium [22]. These specificities can be thresholded to define high-confidence bound and unbound regions for a given TF. Given the location of these binding sites, we can formulate a binary sequence classification problem, classifying regions bound and unbound by a TF as positive and negative, respectively. Using a binary sequence classification model, we can predict binding sites in new cellular contexts, learning regulatory behavior without the expense of ChIP-seq experiments.

String kernel methods are well understood and have been extensively used for sequence classification [42, 79, 98]. Specifically, Fletez-Brant et al. and Lee et al. [50, 95] have applied string kernel methods to the prediction of transcription factor binding sites. However, kernel methods require pairwise comparison between all n training sequences and thus incur an expensive $\mathcal{O}(n^2)$ computational and storage complexity, making them computationally intractable for large data sets.

Recently, convolutional neural networks (CNN) have been successful for prediction of TF binding sites [5, 88, 163]. CNNs generalize well by encoding spatial invariance during training. Fast convolutions on a Graphical Processing Unit (GPU) allows CNNs to train on large datasets. However, the actual design of the neural network greatly impacts model performance, yet there is no clear understanding of how to design a network for a particular task. Furthermore there is no generally accepted network architecture for the task of TF binding site prediction from DNA sequence.

In this work, we present a convolutional kernel approximation algorithm that maintains the spatial invariance and computational efficiency of CNNs. Dubbed Convolutional Kitchen Sinks (CKS), our algorithm learns a model from the output of a 1 layer random convolutional neural network [122]. All the parameters of the network are independent and identically distributed (IID) random samples from a gaussian distribution with a specified variance. We then train a linear model on the output of this network. Our results show that for five out of six transcription factors, CKS outperform current state-of-the art CNN implementations, while maintaining a simple architecture and training eight times faster than a CNN.

6.2 Method

The task of transcription factor (TF) binding site prediction from DNA sequence reduces to binary sequence classification. We present a randomized algorithm for finding an embedding of sequence data apt for linear classification (Algorithm 2). Our algorithm is closely related to the work of convolutional kernel networks, which approximates a convolutional kernel feature map via a nonconvex optimization objective [108]. However, unlike Mairal et al. [108], we approximate the convolutional kernel feature map via random projections in the style of Rahimi et al. [120, 122].

We will first define the convolutional n-gram kernel, and then analyze why it has desired properties for the task of string classification. Note that we use the term n-gram to refer to a contiguous sequence of n characters, whereas computational biology literature refers to the same concept as a k-mer.

Definition 1 (Convolutional n-gram kernel)**.** *Let x, y be strings of length d from an underlying alphabet $\mathcal{A}$, and let $\mathbb{H}(x, y)$ denote the Hamming distance between the two strings. Let $x_{i:j}$ denote the substring of x from index i to $j-1$. Let n be an integer less than d and let γ be a real valued positive number denoting the width of the kernel. The kernel function $K_{n,\gamma}(x, y)$ is defined as:*

$$K_{n,\gamma}(x,y) = \sum_{i=0}^{d-n}\sum_{j=0}^{d-n} \exp(-\gamma \mathbb{H}^2(x_{i:i+n}, y_{j:j+n})) \tag{6.2.1}$$

To gain intuition for the behavior of this kernel, take γ to be a large value. It follows that $\exp(-\gamma\mathbb{H}^2(x_{i:i+n}, y_{j:j+n})) \approx \mathbb{1}[x_{i:i+n} = y_{j:j+n}]$.

This combinatorial reformulation results in the following well studied Spectrum Kernel (Definition 2).

Definition 2 (Spectrum Kernel)**.** *Let $\mathcal{S}_n(\mathcal{A})$ be the set of all length n contiguous substrings in $\mathcal{A}$, and $\#(x,s)$ count the occurrences of $s \in x$ [98].*

$$K_{spec}(x,y) = \sum_{s\in\mathcal{S}_n(\mathcal{A})} \#(x,s)\#(y,s) \tag{6.2.2}$$

Other string kernel methods such as the mismatch [42] and gapped n-gram kernel [57] allow for partial mismatches between n-grams. We note that decreasing γ in Equation 6.2.1 relaxes the penalty of n-gram mismatches between disappoints, thereby capturing the behavior of the mismatch and gapped n-gram kernels [42, 57]. Note that Equation 6.2.1 is computationally prohibitive, as it takes $\Omega(nd^2)$ to compute each of the N^2 entries in the kernel matrix. Furthermore, the feature map induced by the kernel in Equation 6.2.1 is infinite dimensional, so the kernel matrix is necessary.

Instead, we turn to a random approximation of Equation 6.2.1 (see Algorithm 2). Since our kernel is a sum of non linear functions it suffices to define a feature map $\hat{\phi}$ on sequences x and y that approximates each term in the sum from Equation 6.2.1:

$$\exp(-\gamma\mathbb{H}(x_{i:i+n}, y_{j:j+n})) \approx \hat{\phi}(x_{i:i+n})^T\hat{\phi}(y_{j:j+n}) \tag{6.2.3}$$

Claim 1 from Rahimi et al. [120] states that for $j \in \{0 \ldots M-1\}$, if we choose $\hat{\phi}(x_{i:i+n})_j = \sqrt{\frac{2}{M}}\cos(w_j^T x_{i:i+n} + b_j)$, where $w_j \sim \mathcal{N}(0,\gamma)$, $b_j \sim U(0, 2\pi)$, then $\hat{\phi}(x_{i:i+n})$ satisfies Equation 6.2.3. Note that to use Claim 1, we represent Hamming distance in Equation 6.2.1 as an L2 distance. We refer to each w_j as a “random kitchen sink". The result in in Rahimi et al. [120] (Claim 1) gives strong guarantees that $\hat{\phi}(x)^T\hat{\phi}(y)$ concentrates exponentially fast to Equation 6.2.1, which means we can set M, the number of kitchen sinks, to be small.

Algorithm 2 details the kernel approximation. Note that in Algorithm 2, line 16 we reuse w_j across all $x_{i:i+n}$ in Equation 6.2.1 by a convolution. Algorithm 2 is a finite dimensional approximation of the feature map induced by the kernel in Equation 6.2.1

directly, circumventing the need for a kernel matrix. The computational complexity of Algorithm 2 is $\mathcal{O}(NMdn)$.

For the task of TF binding site prediction we let alphabet $\mathcal{A} = \{A, T, C, G\}$, and set $n = 8$, similar to common parameter configuration for DNA sequence [5, 50, 57].

6.3 Results

We compare our CKS to DeepBind, a state-of-the-art CNN approach for predicting transcription factor (TF) binding sites. We compare to DeepBind over other CNN methods [88, 163] due to its primary attention to DNA sequence specificity and ability to identify fine grained (101 bp) locations of binding affinity.

6.3.1 Datasets

We train and evalute on datasets preprocessed from the ENCODE consortium. Because binding affinity is TF specific, we use separate train and evaluation sets for each TF.

We use the same training sets as DeepBind's publically available models. We then evaluate on DeepBind's test sets as well as a larger dataset processed directly from ENCODE.

Table 6.1: Comparison of ROC Area under Curve values (AUC) between DeepBind and CKS tested on 500 bound regions from ENCODE and 500 synthetic unbound regions.

TF	**Train Cell Type**	**Test Cell Type**	**Train Size**	**Train Time**	**DeepBind AUC**	**CKS AUC**
ATF2	H1-hESC	GM12878	10998	154s	0.72	0.77
ATF3	H1-hESC	HepG2	8616	139s	0.94	0.95
ATF3	H1-hESC	K562	8616	139s	0.83	0.84
CEBPB	HeLa-S3	A549	121010	1620s	0.99	0.99
CEBPB	HeLa-S3	K562	121010	1620s	0.99	0.98
EGR1	K562	GM12878	72996	772s	0.94	0.96
EGR1	K562	H1-hESC	72996	772s	0.87	0.92
EP300	HepG2	SK-N-SH	54828	519s	0.67	0.70
EP300	HepG2	K562	54828	519s	0.66	0.81
STAT5A	GM12878	K562	13846	199s	0.65	0.79

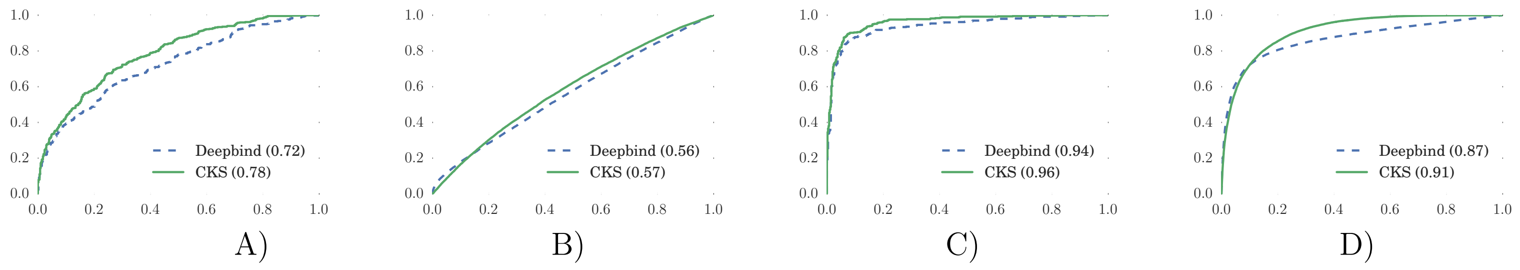


Figure 6.1: Comparison of ROC between DeepBind on CKS on EGR1 and ATF2 for GM12878. A) ROC for ATF2 on the DeepBind's test set. B) ROC for ATF2 on the ENCODE set. C) ROC for EGR1 on the DeepBind's test set. D) ROC for EGR1 on the the ENCODE set.

DeepBind's test sets consist of 1000 regions for each cell type over six TFs. Each set consists of 500 positive sequences extracted from regions of high ChIP-seq signal and 500 synthetic negative sequences generated from dinucleotide shuffle of positive sequences [5].

The second test dataset consists of 100,000 regions extracted from ChIP-seq datasets for TFs ATF2 and EGR1 across multiple cell types. Positive sequences are extracted from regions of high ChIP-seq signal. Negative sequences are extracted from regions of low ChIP-seq signal with exposed chromatin.

Table 6.2: Comparison of ROC Area under Curve values (AUC) between DeepBind and CKS tested on 100,000 bound and unbound regions from ENCODE. Because both experiments trained on the same dataset, the train cell types, train times, and train sizes are the same as in Table 6.1.

TF	**Test Cell Type**	**DeepBind AUC**	**CKS AUC**
ATF2	GM12878	0.56	0.57
ATF2	MCF7	0.93	0.76
EGR1	GM12878	0.87	0.91
EGR1	H1-hESC	0.77	0.85
EGR1	HCT116	0.77	0.82
EGR1	MCF7	0.84	0.86

6.3.2 Experimental Setup

Experiments for DeepBind and CKS were run on one machine with 24 Xeon processors, and 256 GB of ram and 1 Nvidia Tesla K20c GPU.

We train a linear model minimizing squared loss with an L2 penalty of λ on the output of the CKS defined in Algorithm 2. We do not tune the hyper-parameters n (convolution size) and M (number of kitchen sinks), and leave them constant at 8 and 8192 respectively. We tune the hyper paraemters γ (kernel width) and λ on held out data from the train set. To assess generalization across cellular contexts, we train and evaluate on separate cell types.

6.3.3 Evaluation

We compare DeepBind against CKS using area under the curve (AUC) of Receiver Operating Characteristic (ROC). We choose AUC as a metric for binary classifcation due to its ability to measure both TF binding site detection and false positive rates.

We detail our experimental results and compare to DeepBind's pretrained models in Tables 6.1 and 6.2. We also show ROCs for ATF2 and EGR1 on both datasets in Figure 6.1.

Our AUC is competitive (within 0.01) or superior to that of DeepBind except for ATF2 on MCF7 cell type. Furthermore on five out of six large ENCODE test sets, our AUC is strictly greater than DeepBind.

We measure DeepBind's training time on TF EGR1, trained on K562 with $72,996$ train sequences. DeepBind's training procedure takes 6497 seconds to learn 2123 parameters. For comparison, training time for CKS takes 712 seconds (Table 6.1) to learn 16384 parameters, which is approximately eight times faster than DeepBind's runtime.

6.4 Conclusion & Future Work

In this paper, we show that Convolutional Kitchen Sinks train eight times faster and has superior predictive performance to CNNs. We note that our current work focuses on binding affinity in the context of DNA sequence, making this model agnostic to specific cell contexts of interest. Because Algorithm 1 is not specific to DNA sequence, positional counts of chromatin accessibility and gene expression data can be aggregated with current implementation to account for cell type specific information. We leave this extension for future work.

Algorithm 2 Convolutional Kitchen Sink for sequences

```
Input
    $x_i \dots x_N \in \mathbb{R}^d$ (input sequences)
    $\gamma$ (width of kernel)
    $n$ (convolution size)
    $M$ (the approximation dimension, number of kitchen sinks)
Output
    $\phi(x_i) \dots \phi(x_N)$
for $j \in \{0 \dots M\}$ do
    $w_j \sim \mathcal{N}(0, \gamma I_n)$          Sample kitchen sink from gaussian
    $b_j \sim U(0, 2\pi)$          Sample phase from uniform disk

    for $i \in \{0 \dots N\}$ do
        $z_{ij} = w_j * x_i$          Convolve filter with input sequence

        $c_{ij} = \cos(z_{ij} + b_j)$          Add phase and compute element-wise cosine
                    Note $z_{ij}$ and $c_{ij}$ are vectors in $\mathcal{R}^{d-n+1}$

        $\phi(x_i)_j = \sqrt{\frac{2}{M}} \sum_{k=0}^{d-n} c_{ijk}$          Average to get the $j$th output feature value for sequence
    $x_i$

    end for
end for
```

Bibliography

[1] American kennel club. URL `https://www.akc.org/`.

[2] *Economic Report of the President.* 2019. `https://www.govinfo.gov/app/collection/erp/2019`.

[3] Subfamily cicindelinae - tiger beetles, Oct 2019. URL `https://bugguide.net/node/view/375`.

[4] Soroosh Shafieezadeh Abadeh, Peyman Mohajerin Mohajerin Esfahani, and Daniel Kuhn. Distributionally robust logistic regression. In *Advances in Neural Information Processing Systems*, pages 1576–1584, 2015.

[5] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology*, 2015.

[6] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in Neural Information Processing Systems*, 2019.

[7] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, 2019.

[8] Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020.

[9] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, 2018.

[10] Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

[11] Alex Berg. Personal communication, 2018.

[12] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[13] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 2018. `https://arxiv.org/abs/1712.03141`.

[14] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases*, 2013. `https://link.springer.com/chapter/10.1007/978-3-642-40994-3_25`.

[15] Avrim Blum and Moritz Hardt. The Ladder: A reliable leaderboard for machine learning competitions. In *International Conference on Machine Learning (ICML)*, 2015. `http://arxiv.org/abs/1502.04585`.

[16] Remi Cadene. Pretrained models for pytorch. `https://github.com/Cadene/pretrained-models.pytorch`. Accessed: 2019-05-20.

[17] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *Neural Information Processing Systems (NIPS)*, 2017. `https://arxiv.org/abs/1707.01629`.

[18] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. `https://arxiv.org/abs/1610.02357`.

[19] Stephane Clinchant, Gabriela Csurka, Florent Perronnin, and Jean-Michel Renders. XRCE's participation to ImagEval. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.6670&rep=rep1&type=pdf`, 2007.

[20] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the Trade*, pages 561–580. Springer, 2012.

[21] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. `http://proceedings.mlr.press/v15/coates11a.html`.

[22] ENCODE Project Consortium et al. The ENCODE (Encyclopedia of DNA elements) project. *Science*, 306(5696):636–640, 2004.

[23] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning augmentation policies from data. `https://arxiv.org/abs/1805.09501`, 2018.

[24] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019.

[25] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, 2014.

[26] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[27] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances in Neural Information Processing Systems*, 2016.

[28] Tri Dao, Albert Gu, Alexander J. Ratner, Virginia Smith, Christopher De Sa, and Christopher Re. A kernel theory of modern data augmentation. In *International Conference on Machine Learning (ICML)*, 2019.

[29] Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations research*, 58(3):595–612, 2010.

[30] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.

[31] Jia Deng. *Large Scale Visual Recognition.* PhD thesis, Princeton University, 2012. `ftp://ftp.cs.princeton.edu/techreports/2012/923.pdf`.

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. `http://www.image-net.org/papers/imagenet_cvpr09.pdf`.

[33] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with Cutout. `https://arxiv.org/abs/1708.04552`, 2017.

[34] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming, Series A*, 91:201–213, 2002.

[35] Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, 2019.

[36] Simon S. Du, Xiyu Zhai, Barnab's Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[37] John C Duchi, Tatsunori Hashimoto, and Hongseok Namkoong. Distributionally robust losses against mixture covariate shifts. *Under review*, 2019.

[38] Gamaleldin Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In *Advances in Neural Information Processing Systems*, pages 3910–3920, 2018.

[39] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *arXiv preprint arXiv:1712.02779*, 2017.

[40] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling CNNs with simple transformations. `http://arxiv.org/abs/1712.02779`, 2017.

[41] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171(1-2):115–166, 2018.

[42] Eleazar Eskin, Jason Weston, William S Noble, and Christina S Leslie. Mismatch string kernels for SVM protein classification. In *Advances in Neural Information Processing Systems*, pages 1417–1424, 2002.

[43] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 2010. `http://dx.doi.org/10.1007/s11263-009-0275-4`.

[44] Alhussein Fawzi and Pascal Frossard. Manitest: Are classifiers really invariant? In *British Machine Vision Conference (BMVC)*, 2015. `https://arxiv.org/abs/1507.06535`.

[45] Alhussein Fawzi and Pascal Frossard. Manitest: Are classifiers really invariant? In *British Machine Vision Conference (BMVC)*, 2015.

[46] Alhussein Fawzi and Pascal Frossard. Manitest: Are classifiers really invariant? In *British Machine Vision Conference (BMVC)*, 2015. `https://arxiv.org/abs/1507.06535`.

[47] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 2007. `http://dx.doi.org/10.1016/j.cviu.2005.09.012`.

[48] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046, 2017.

[49] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[50] Christopher Fletez-Brant, Dongwon Lee, Andrew S McCallion, and Michael A Beer. Kmer-SVM: a web server for identifying predictive regulatory sequence features in genomic data sets. *Nucleic acids research*, 41(W1):W544–W556, 2013.

[51] Xavier Gastaldi. Shake-shake regularization. `https://arxiv.org/abs/1705.07485`, 2017.

[52] Robert Geirhos, David HJ Janssen, Heiko H Schütt, Jonas Rauber, Matthias Bethge, and Felix A Wichmann. Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969*, 2017.

[53] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[54] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *CoRR*, abs/1811.12231, 2018. URL `http://arxiv.org/abs/1811.12231`.

[55] Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schütt, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2018. `http://papers.nips.cc/paper/7982-generalisation-in-humans-and-deep-neural-networks.pdf`.

[56] Robert Geirhos, Carlos RM Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. In *Advances in Neural Information Processing Systems*, pages 7538–7550, 2018.

[57] Mahmoud Ghandi, Dongwon Lee, Morteza Mohammad-Noori, and Michael A Beer. Enhanced regulatory sequence prediction using gapped k-mer features. *PLOS Computational Bioliology*, 10(7):e1003711, 2014.

[58] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension, 2019.

[59] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. Maxout networks. In *International Conference on Machine Learning*, 2013.

[60] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[61] Keren Gu, Brandon Yang, Jiquan Ngiam, Quoc Le, and Jonathan Shlens. Using videos to evaluate image model robustness. *arXiv preprint arXiv:1904.10076*, 2019.

[62] Keren Gu, Brandon Yang, Jiquan Ngiam, Quoc V. Le, and Jonathon Shlens. Using videos to evaluate image model robustness. *CoRR*, abs/1904.10076, 2019. URL `http://arxiv.org/abs/1904.10076`.

[63] Ben Hamner. Popular datasets over time. `https://www.kaggle.com/benhamner/popular-datasets-over-time/data`, 2017.

[64] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. `https://arxiv.org/abs/1610.02915`.

[65] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, 2015. `https://arxiv.org/abs/1502.01852`.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. `https://arxiv.org/abs/1512.03385`.

[69] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, 2016. `https://arxiv.org/abs/1603.05027`.

[70] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*, 2019. `https://arxiv.org/abs/1807.01697`.

[71] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[72] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1614–1619, 2018.

[73] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018. `https://arxiv.org/abs/1804.00499`.

[74] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. `https://arxiv.org/abs/1704.04861`, 2017.

[75] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. `https://arxiv.org/abs/1709.01507`.

[76] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. `https://arxiv.org/abs/1608.06993`.

[77] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$0.5MB model size. `https://arxiv.org/abs/1602.07360`, 2016.

[78] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. `https://arxiv.org/abs/1502.03167`.

[79] Tommi S Jaakkola, Mark Diekhans, and David Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of ISMB*, volume 99, pages 149–158, 1999.

[80] Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[81] SouYoung Jin, Aruni RoyChowdhury, Huaizu Jiang, Ashish Singh, Aditya Prasad, Deep Chakraborty, and Erik Learned-Miller. Unsupervised hard example mining from videos for improved object detection. In *ECCV*, 2018.

[82] Eric Jonas, Monica Bobra, Vaishaal Shankar, J Todd Hoeksema, and Benjamin Recht. Flare prediction using photospheric and coronal image data. *Solar Physics*, 293(3):48, 2018.

[83] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Geometric robustness of deep networks: analysis and improvement. *arXiv preprint arXiv:1711.09115*, 2017.

[84] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Geometric robustness of deep networks: Analysis and improvement. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. `https://arxiv.org/abs/1711.09115`.

[85] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object detection in videos with tubelet proposal networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 727–735, 2017.

[86] Andrej Karpathy. Lessons learned from manually classifying CIFAR-10. `http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/`, 2011.

[87] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. `https://arxiv.org/abs/1710.05468`, 2017.

[88] David R Kelley, Jasper Snoek, and John L Rinn. Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research*, 2016.

[89] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better ImageNet models transfer better? `http://arxiv.org/abs/1805.08974`, 2018.

[90] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy.

Openimages: A public dataset for large-scale multi-label and multi-class image classification. https://storage.googleapis.com/openimages/web/index.html, 2017.

[91] Alex Krizhevsky. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

[92] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

[93] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[94] Yann LeCun, Corinna Cortes, and CJC Burges. The mnist dataset of handwritten digits, 1998.

[95] Dongwon Lee, David U Gorkin, Maggie Baker, Benjamin J Strober, Alessandro L Asoni, Andrew S McCallion, and Michael A Beer. A method to predict the impact of regulatory variants from DNA sequence. *Nature Genetics*, 47(8):955–961, 2015.

[96] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[97] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, 2019.

[98] Christina S Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Pacific Symposium on Biocomputing*, volume 7, pages 566–575, 2002.

[99] Fei-Fei Li and Jia Deng. ImageNet: Where have we been? where are we going? http://image-net.org/challenges/talks_2017/imagenet_ilsvrc2017_v1.0.pdf, 2017.

[100] Zhiyuan Li, Ruosong Wang, Dingli Yu, Simon S. Du, Wei Hu, Ruslan Salakhutdinov, and Sanjeev Arora. Enhanced convolutional neural tangent kernels, 2019.

[101] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. MS COCO detection evaluation. `http://cocodataset.org/#detection-eval`. Accessed: 2019-05-16.

[102] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014. `https://arxiv.org/abs/1405.0312`.

[103] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *European Conference on Computer Vision (ECCV)*, 2018. `https://arxiv.org/abs/1712.00559`.

[104] Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *Asian Conference on Pattern Recognition (ACPR)*, 2015. `https://ieeexplore.ieee.org/document/7486599/`.

[105] Siyuan Ma and Mikhail Belkin. Kernel machines that adapt to gpus for effective large batch training, 2018.

[106] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2018.

[107] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[108] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, pages 2627–2635, 2014.

[109] Jitendra Malik. Technical perspective: What led computer vision to deep learning? *Communications of the ACM*, 2017. `http://doi.acm.org/10.1145/3065384`.

[110] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. `https://github.com/facebookresearch/maskrcnn-benchmark`, 2018. Accessed: 2019-05-20.

[111] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38 (11):39–41, 1995.

[112] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38 (11):39–41, 1995.

[113] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019.

[114] Yaniv Ovadia, Jasper Snoek, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D. Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980, 2019.

[115] David Page. myrtle.ai, 2018. URL `https://myrtle.ai/how-to-train-your-resnet-4-architecture/`.

[116] Harold Pashler. Familiarity and visual change detection. *Perception & psychophysics*, 44 (4):369–378, 1988.

[117] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the Fisher kernel for large-scale image classification. In *European Conference on Computer Vision (ECCV)*, 2010. URL `https://www.robots.ox.ac.uk/~vgg/rg/papers/peronnin_etal_ECCV10.pdf`.

[118] Jean Ponce, Tamara L. Berg, Mark Everingham, David A. Forsyth, Martial Hebert, Sveltana Lazebnik, Marcin Marszalek, Cordelia Schmid, Bryan C. Russell, Antionio Torralba, Chris. K. I. Williams, Jianguo Zhang, and Andrew Zisserman. *Dataset issues in object recognition.* 2006. `https://link.springer.com/chapter/10.1007/11957959_2`.

[119] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning.* The MIT Press, 2009.

[120] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

[121] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[122] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems*, pages 1313–1320, 2009.

[123] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. URL `https://www.aclweb.org/anthology/D16-1264`.

[124] A. Ratner, H. Ehrenberg, Z. Hussain, J. Dunnmon, and Christopher Re. Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems*, 2017.

[125] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5296–5305, 2017.

[126] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. `http://arxiv.org/abs/1802.01548`, 2018.

[127] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? 2019. URL `http://arxiv.org/abs/1902.10811`.

[128] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019.

[129] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[130] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[131] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Fei-Fei Li. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015. `https://arxiv.org/abs/1409.0575`.

[132] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Fei-Fei Li. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015. `https://arxiv.org/abs/1409.0575`.

[133] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[134] Vaishaal Shankar, Karl Krauth, Qifan Pu, Eric Jonas, Shivaram Venkataraman, Ion Stoica, Benjamin Recht, and Jonathan Ragan-Kelley. numpywren: Serverless linear algebra. *CoRR*, abs/1810.09679, 2018. URL `http://arxiv.org/abs/1810.09679`.

[135] Vaishaal Shankar, Achal Dave, Rebecca Roelofs, Deva Ramanan, Benjamin Recht, and Ludwig Schmidt. A systematic framework for natural perturbations from videos. *CoRR*, abs/1906.02168, 2019. URL `http://arxiv.org/abs/1906.02168`.

[136] John Shawe-Taylor, Nello Cristianini, et al. *Kernel Methods for Pattern Analysis*. Cambridge university press, 2004.

[137] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. `https://arxiv.org/abs/1409.1556`, 2014.

[138] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571*, 2017.

[139] Pierre Stock and Moustapha Cissé. Convnets and imagenet beyond accuracy: Explanations, bias detection, adversarial examples and model criticism. *CoRR*, abs/1711.11443, 2017. URL `http://arxiv.org/abs/1711.11443`.

[140] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2013. `http://arxiv.org/abs/1312.6199`.

[141] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. `https://arxiv.org/abs/1409.4842v1`.

[142] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. `https://arxiv.org/abs/1512.00567`.

[143] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, Inception-Resnet and the impact of residual connections on learning. In *Conference On Artificial Intelligence (AAAI)*, 2017. `https://arxiv.org/abs/1602.07261`.

[144] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. `http://people.csail.mit.edu/torralba/publications/datasets_cvpr11.pdf`.

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

[146] Antonio Torralba, Alexei A Efros, et al. Unbiased look at dataset bias. In *CVPR*, volume 1, page 7. Citeseer, 2011.

[147] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy, 2019.

[148] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL `https://vision.cornell.edu/se3/wp-content/uploads/2015/05/Horn_Building_a_Bird_2015_CVPR_paper.pdf`.

[149] Nicolas Vasilache, Oleksandr Zinenko, Theodoros Theodoridis, Priya Goyal, Zachary DeVito, William S Moses, Sven Verdoolaege, Andrew Adams, and Albert Cohen. Tensor comprehensions: Framework-agnostic high-performance machine learning abstractions. *arXiv preprint arXiv:1802.04730*, 2018.

[150] Ke Alexander Wang, Geoff Pleiss, Jake Gardner, Stephen Tyree, Kilian Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, 2019.

[151] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004. `http://www.cns.nyu.edu/pub/lcv/wang03-preprint.pdf`.

[152] Wikipedia contributors. Tiger beetle — Wikipedia, the free encyclopedia, 2019. URL `https://en.wikipedia.org/w/index.php?title=Tiger_beetle&oldid=932794435`. [Online; accessed 1-February-2020].

[153] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018. `https://arxiv.org/abs/1801.02612`.

[154] Fanyi Xiao and Yong Jae Lee. Video object detection with an aligned spatial-temporal memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 485–501, 2018.

[155] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. *arXiv preprint arXiv:1812.03411*, 2018.

[156] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan Yuille, and Quoc V. Le. Adversarial examples improve image recognition, 2019.

[157] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. `https://arxiv.org/abs/1611.05431`.

[158] Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. `https://arxiv.org/abs/1802.02375`, 2018.

[159] Benjamin Z. Yao, Xiong Yang, and Song-Chun Zhu. Introduction to a large-scale general purpose ground truth database: methodology, annotation tool and benchmarks. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2007. `https://link.springer.com/chapter/10.1007/978-3-540-74198-5_14`.

[160] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016. `https://arxiv.org/abs/1605.07146`.

[161] Xingcheng Zhang, Zhizhong Li, Chen Change Loy, and Dahua Lin. Polynet: A pursuit of structural diversity in very deep networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. `https://arxiv.org/abs/1611.05725`.

[162] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. doi: 10.1109/cvpr.2016.485. URL `http://dx.doi.org/10.1109/cvpr.2016.485`.

[163] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature methods*, 12(10):931–934, 2015.

[164] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017.

[165] Zhuotun Zhu, Lingxi Xie, and Alan L. Yuille. Object recognition with and without objects. *CoRR*, abs/1611.06596, 2016. URL `http://arxiv.org/abs/1611.06596`.

[166] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. `https://arxiv.org/abs/1707.07012`.