Neural Kernels and ImageNet Multi-Label Accuracy

Alex Fang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-93
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-93.html

May 29, 2020

Acknowledgement

The work presented here is joint work with others, including Vaishaal Shankar, Wenshuo Guo, Sara Fridovich-Keil, Becca Roelofs, Horia Mania, Ludwig Schmidt, and Professors Jonathan Ragan-Kelley and Ben Recht. I would like to especially thank Vaishaal for his mentorship, Professor Jonathan Ragan-Kelley for being my research advisor, and Professor Ben Recht for additional research guidance.

Neural Kernels and ImageNet Multi-Label Accuracy

by Alex Fang

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

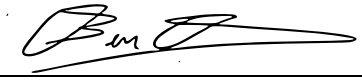
Committee:

Professor Jonathan Ragan-Kelley
Research Advisor

5/26/2020

(Date)

* * * * * * *



Professor Benjamin Recht
Second Reader

5/27/2020

(Date)

Neural Kernels and ImageNet Multi-Label Accuracy

Alex Fang

Abstract

Inspired by the success of deep learning, we investigate the connections between neural networks and simple building blocks in kernel space in order to create a kernel that achieves performance closer to state of the art on CIFAR10. Additionally, we evaluate human accuracy on ImageNet with a multi-label accuracy metric in order to better understand the robustness of machine models. By looking at both, we can better understand different failure modes of various machine learning models to improve performance in the field.

1 Introduction to Neural Kernels

Kernel methods have fallen out of favor due to the superiority of neural networks on modern benchmark tasks. However, kernel methods have advantages such as nice properties and being well understood. Recent work show a connection between neural networks trained with gradient descent and kernel methods [4, 7, 9, 10, 12, 16, 20]. Specifically, they relate "infinitely wide" neural networks to particular kernel spaces, showing that infinite limits of random initializations of neural networks have equivalent kernels. Recent work has explored using such kernels on benchmark tasks [5, 21], but has not been able to match the performance of neural networks on the majority of tasks.

Here 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. [7], and the Neural Tangent Kernel limits of Jacot et al. [16]. 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. 2).

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 [26], 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 [5]. Here, we find that compositional kernels outperform NTKs and neural networks outperform both kernel methods when properly tuned and trained. 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.

2 Kernel Formation

To construct our compositional kernel functions, we rely on key results from Daniely et al. [7], which explicitly studies the duality between neural network architectures and compositional kernels.

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 $\mathcal{S}_1, \ldots, \mathcal{S}_L \subseteq \mathcal{B}$ be *ordered* subsets with the same cardinality, s. We write each subset as an ordered set of indices: $\mathcal{S}_j = \{i_{j1}, \ldots, i_{js}\}$. 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 $\mathcal{S}_1,\dots,\mathcal{S}_L \subseteq \mathcal{B}$ 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 $\{1,\dots,L\}$ and Hilbert space $\mathcal{H}$. For each $j=1,\dots,L$ 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.

2.1 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 2.

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:

$$k(\Phi(\boldsymbol{X}), j, \Phi(\boldsymbol{Z}), k) = k_\varphi(\boldsymbol{X}_j, \boldsymbol{Z}_k)\,. \tag{1}$$

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 (1) in cascades of these primitive operations.

2.2 Kernel operations on images

In this section, we specialize kernel operations to operations on images. As described in Section 2, 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:

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

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:

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

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:

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

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

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

The relationship between the ReLU operator and the ReLU kernel is covered in Subsection 2.3.

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 1). Experimentally, we find the Gaussian kernel to be marginally faster and more numerically stable.

2.3 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 [28]. Consider two tensors: $\boldsymbol{U}$ of shape $N \times D_1 \times D_2 \times D_3$

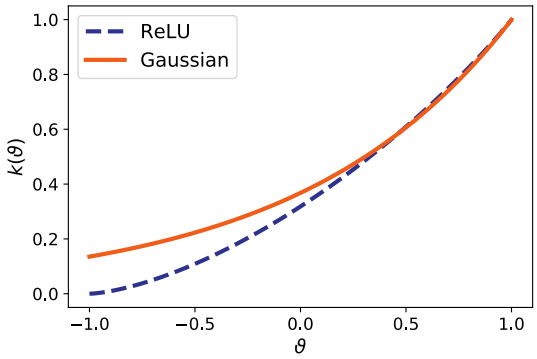


Figure 1: Comparison of the ReLU (arccosine) and Gaussian kernels ($\gamma = 1$), as a function of the angle ϑ between two examples.

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:

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

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

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

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. [7], thereby relating our kernel construction to inner products between the outputs of *random* neural networks:

$$\begin{aligned} &\mathbb{E}\left[\sum_{c=1}^{D_4} \Psi(\boldsymbol{U})[i,j,k,c]\Psi(\boldsymbol{U})[\ell,m,n,c]\right] = \\ &k_{relu}\Big(c_w\big(k_0(\boldsymbol{U})\big)\Big)[i,j,k,\ell,m,n] \end{aligned} \tag{2}$$

where k_0 is the input kernel defined in Subsection 2.2.

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 (2), the term inside the expectation only depends on inner products, this relation can be generalized to arbitrary depths.

2.4 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 2.2.

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 3.

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

3 Kernel Experiments

Figure 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 [17]), CIFAR-10.1 (Recht et al. [31]), and CIFAR-100 (Krizhevsky [17]).

3.1 Architectures

We design our deep convolutional kernel based on the non-residual convolutional "Myrtle" networks introduced in Page [26]. 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 2. The deeper variants have more convolution and ReLU layers. Next we show convolutional neural networks from this family can indeed achieve high accuracy on CIFAR-10, as can their kernel counterparts.

3.2 Experimental setup.

We implemented all the convolutional kernels in the tensor comprehensions framework [36] 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. 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.

3.3 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 [19]. Results are presented in Table 1. We observe that all convolutional methods show nearly identical performance, outperforming the three non-convolutional methods (NTK, arccosine kernel, and Gaussian kernel).

Table 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 [18]	99.2
CKN [23]	99.6
Myrtle5 Kernel	99.5
Myrtle5 CNN	99.5

3.4 CIFAR-10

Table 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] [17, 31]. 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.

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. [21] heavily relies on a data dependent feature extraction before data is passed into the kernel function [6]. 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.

Effect of preprocessing. For all of our primary CIFAR-10 experiments, we begin with ZCA pre-processing [13]. Table 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.

[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

3.5 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 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 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 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 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

3.6 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. [5]. Results are shown in Figure 3. Subsampled datasets are class balanced, and standard deviations are computed over 20 random subsamples, as in Arora et al. [5].

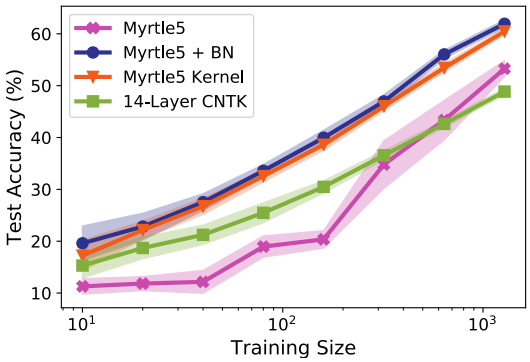


Figure 3: Accuracy results on random subsets of CIFAR-10, with standard deviations over 20 trials. The 14-layer CNTK results are from Arora et al. [5].

Table 3: Classification performance on CIFAR-10.

Method	CIFAR-10 Accuracy	CIFAR-10.1 Accuracy
Gaussian Kernel	57.4	-
CNTK + Flips [21]	81.4	-
CNN-GP + Flips [21]	82.2	-
CKN [23]	82.2	-
Coates-NG + Flips [31]	85.6	73.1
Coates-NG + CNN-GP + Flips [21]	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

Results. We demonstrate that in the small dataset regime explored in Arora et al. [5], 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. [5].

4 ImageNet Background

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

[11, 15, 25, 27, 30, 33, 34]. 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 [8, 32] and SQuAD [29] 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.

Here, 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, 14, 32]. 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 [32]. 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 [32]. 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,[3] but also on the test set from the ImageNetV2 replication study [30].[4] 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.

Table 4 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.[5] 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 4, 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.

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 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] 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.

[4] Specifically, we utilized the MatchedFrequency test set from Recht et al. [30]. For conciseness, we simply refer to this test set as ImageNetV2.

[5] Here we only describe the point estimates. Section 8 contains a discussion of statistical significance.

Table 4: 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 [38] is an EfficentNet-B8 trained with AdvProp and the FixRes model [22, 35] 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**

5 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 5 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 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. [30]. At this point, labelers A, B, and C already had extensive experience with the ImageNet dataset. We further discuss the annotation process in Section 6.

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 7.

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 8.

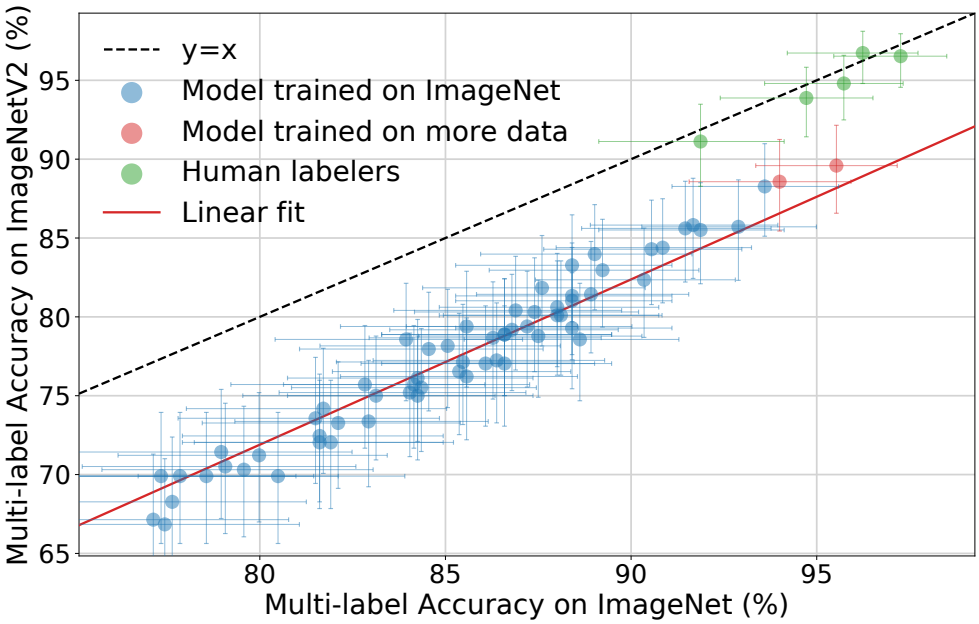


Figure 4: Multi-label accuracies for both CNN models and human participants on the ImageNet validation set versus their accuracies on the ImageNetV2 test set. The confidence intervals are 95% Clopper-Pearson confidence intervals.

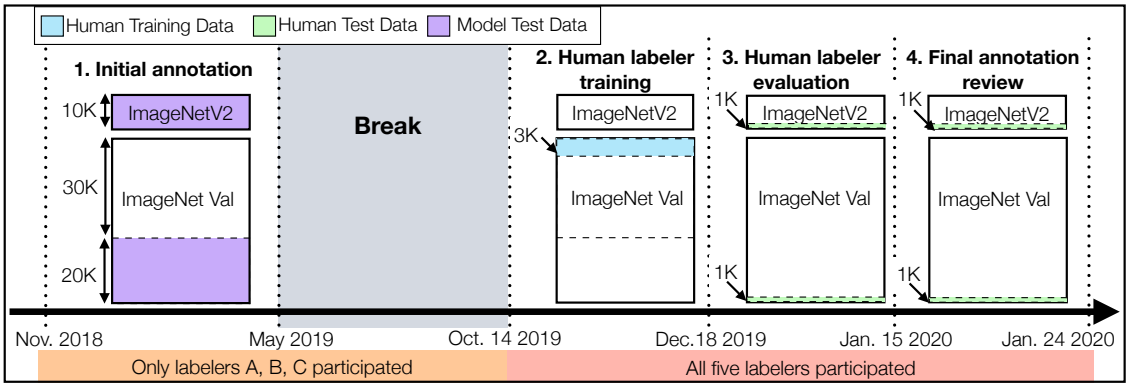


Figure 5: Timeline for the four phases of our experiment. 1) *Initial multi-label annotation:* First, starting in November 2018, human labelers A, B, and C annotated a set of images from ImageNetV2 and the ImageNet validation set with multi-label annotations. 2) *Human labeler training:* Then, after 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.

6 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 6a, 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 6b 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 [32] 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.

6.1 Types of multi-label annotations

Next, we discuss three categories of multi-label annotations that arose in our study, exemplified in Figure 6.

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 6a 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

(a) `desk`

(b) `picket fence`

(c) `African elephant`

(d) `lakeshore`

Figure 6: Examples from the ImageNet validation of scenarios where multi-label annotations are necessary. *Multiple objects or organisms:* In Figure 6a, the ImageNet label is `desk` but `screen`, `monitor`, `coffee mug` and many more objects in the scene could count as correct labels. Figure 6b 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.

scene. For example, in Figure 6b, 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 6c 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 6d shows a scene which could arguably be either a `lakeshore` or a `seashore`.

6.2 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 7. 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.

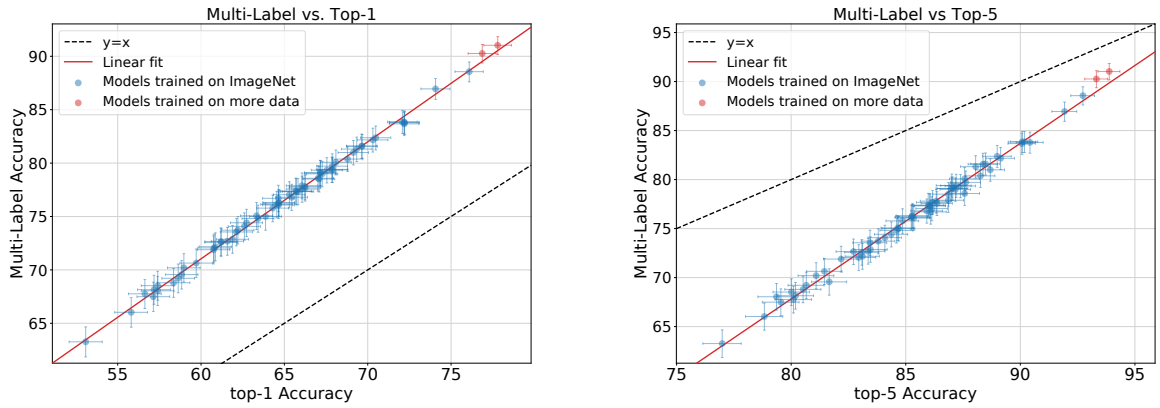


Figure 7: The relationship between `top-1`, `top-5`, and multi-label accuracy on ImageNet test for all 71 models in our test bed. The left figure plots multi-label vs. `top-1` accuracy accuracy. Multi-label accuracy makes the task easier than `top-1` accuracy, with a median improvement of 8.9% between `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 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 the annotation stage. We note that we performed a second annotation stage after the human labelers completed the experiment, as explained in Section 7.3.

In Figure 7, 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.

7 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.

7.1 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. [32] 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, 37]. 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 7.2.

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 [24] hierarchy.

Insufficient training images. The two annotators in [32] 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.

7.2 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.

7.3 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.

8 Human Accuracy 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 4. 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 4 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 4 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}.

Table 5: 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

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 8. 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 8: 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 5 shows an increase in the accuracy of the best model ([35]) 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 5 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 5. 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 5 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 5.

9 Future Work

The kernels in this work advance the state of the art of kernel methods for image classification, but they are still limited by computational cost. Important future work to make these kernels more viable include finding ways to reduce computational cost, allow for data augmentation, and search for better architectures. To tie together kernels with ImageNet, we are currently working on using these kernels to achieve deep neural network levels of accuracy on ImageNet by using approximation to reduce computational cost. Approximation allows these kernels for ImageNet to be computationally tractable, as well as make data augmentation more practical.

In addition to the above, we are also exploring whether these simpler approaches to image classification work well on different tasks. To do so, we are working on using random features as an approximation to kernels and applying them to various Kaggle image classification competitions to determine whether these approaches can match neural networks in a wide variety of tasks, ranging from classifying human protein patterns to classifying human driver behavior.

Acknowledgements

The work presented here is joint work with others, including Vaishaal Shankar, Wenshuo Guo, Sara Fridovich-Keil, Becca Roelofs, Horia Mania, Ludwig Schmidt, and Professors Jonathan Ragan-Kelley and Ben Recht. I would like to especially thank Vaishaal for his mentorship, Professor Jonathan Ragan-Kelley for being my research advisor, and Professor Ben Recht for additional research guidance.

References

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

[5] Arora, S., Du, S. S., Li, Z., Salakhutdinov, R., Wang, R., and Yu, D. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020.

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

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

[8] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. `http://www.image-net.org/papers/imagenet_cvpr09.pdf`.

[9] Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, 2019.

[10] Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

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

[12] Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. Linearized two-layers neural networks in high dimension, 2019.

[13] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., and Bengio, Y. Maxout networks. In *International Conference on Machine Learning*, 2013.

[14] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[15] Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[16] Jacot, A., Hongler, C., and Gabriel, F. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[17] Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

[18] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] LeCun, Y., Cortes, C., and Burges, C. The mnist dataset of handwritten digits, 1998.

[20] Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, 2019.

[21] Li, Z., Wang, R., Yu, D., Du, S. S., Hu, W., Salakhutdinov, R., and Arora, S. Enhanced convolutional neural tangent kernels, 2019.

[22] Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 181–196, 2018.

[23] Mairal, J., Koniusz, P., Harchaoui, Z., and Schmid, C. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, 2014.

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

[25] Ovadia, Y., Snoek, J., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J., Ren, J., and Nado, Z. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pp. 13969–13980, 2019.

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

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

[28] Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pp. 1177–1184, 2008.

[29] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. 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.

[30] Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do imagenet classifiers generalize to imagenet? 2019. URL http://arxiv.org/abs/1902.10811.

[31] Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, 2019.

[32] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Li, F.-F. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015. https://arxiv.org/abs/1409.0575.

[33] Shankar, V., Dave, A., Roelofs, R., Ramanan, D., Recht, B., and Schmidt, L. A systematic framework for natural perturbations from videos. *CoRR*, abs/1906.02168, 2019. URL http://arxiv.org/abs/1906.02168.

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

[35] Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. Fixing the train-test resolution discrepancy, 2019.

[36] Vasilache, N., Zinenko, O., Theodoridis, T., Goyal, P., DeVito, Z., Moses, W. S., Verdoolaege, S., Adams, A., and Cohen, A. Tensor comprehensions: Framework-agnostic high-performance machine learning abstractions. *arXiv preprint arXiv:1802.04730*, 2018.

[37] 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].

[38] Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A., and Le, Q. V. Adversarial examples improve image recognition, 2019.