Do Better ImageNet Models Transfer Better?

Simon Kornblith,* **Jonathon Shlens, and Quoc V. Le**
Google Brain
{skornblith,shlens,qvl}@google.com

Abstract

Transfer learning has become a cornerstone of computer vision with the advent of ImageNet features, yet little work has been done to evaluate the performance of ImageNet architectures across different datasets. An implicit hypothesis in modern computer vision research is that models that perform better on ImageNet necessarily perform better on other vision tasks. However, this hypothesis has never been systematically tested. Here, we compare the performance of 13 classification models on 12 image classification tasks in three settings: as fixed feature extractors, fine-tuned, and trained from random initialization. We find that, when networks are used as fixed feature extractors, ImageNet accuracy is only weakly predictive of accuracy on other tasks ($r^2 = 0.24$). In this setting, ResNets consistently outperform networks that achieve higher accuracy on ImageNet. When networks are fine-tuned, we observe a substantially stronger correlation ($r^2 = 0.86$). We achieve state-of-the-art performance on eight image classification tasks simply by fine-tuning state-of-the-art ImageNet architectures, outperforming previous results based on specialized methods for transfer learning. Finally, we observe that, on three small fine-grained image classification datasets, networks trained from random initialization perform similarly to ImageNet-pretrained networks. Together, our results show that ImageNet architectures generalize well across datasets, with small improvements in ImageNet accuracy producing improvements across other tasks, but ImageNet features are less general than previously suggested.

1 Introduction

The last decade of computer vision research has pursued academic benchmarks as a measure of progress. No benchmark has been as hotly pursued as ImageNet [1]. Network architectures measured against this dataset have fueled much progress in computer vision research across a broad array of problems, including transferring to new datasets [2, 3], object detection [4], image segmentation [5, 6] and perceptual metrics of images [7]. An implicit assumption behind this progress is that network architectures that perform better on ImageNet necessarily perform better on other vision tasks. Another assumption is that better network architectures learn better features that can be transferred across vision-based tasks. Although previous studies have provided some evidence for these hypotheses (e.g. [4, 5, 8–10]), they have never been systematically explored.

In the present work, we seek to test these hypotheses by investigating the transferability of both ImageNet features and ImageNet classification architectures. Specifically, we conduct a large-scale study of transfer learning across 13 top-performing convolutional neural networks for image classification on 12 image classification datasets in 3 different experimental settings, visualized in Figure 1: as fixed feature extractors [2, 3], fine-tuned from ImageNet initialization [8, 11, 12], and trained from random initialization. Our main contributions are as follows:

*Work done as a member of the Google AI Residency program (g.co/airesidency).

Preprint. Work in progress.

- The best ImageNet models do not provide the best fixed image features. Features from ResNet models [14, 15] trained on ImageNet consistently outperform features from networks that achieve higher accuracy on ImageNet.

- When networks are fine-tuned, ImageNet accuracy is a much stronger indicator of transfer task accuracy ($r^2 = 0.86$), with a state-of-the-art ImageNet architecture yielding state-of-the-art results across many tasks.

- Architectures transfer well across tasks even when weights do not. On 3 small fine-grained classification datasets, fine-tuning does not provide a substantial benefit over training from random initialization, but better ImageNet architectures nonetheless obtain higher accuracy.

2 Related work

ImageNet follows in a succession of progressively larger and more realistic benchmark datasets for computer vision. Each successive dataset was designed to address perceived issues with the size and content of previous datasets. Torralba and Efros [16] showed that many early datasets were heavily biased, with classifiers trained to recognize or classify objects on those datasets possessing almost no ability to generalize to images from other datasets.

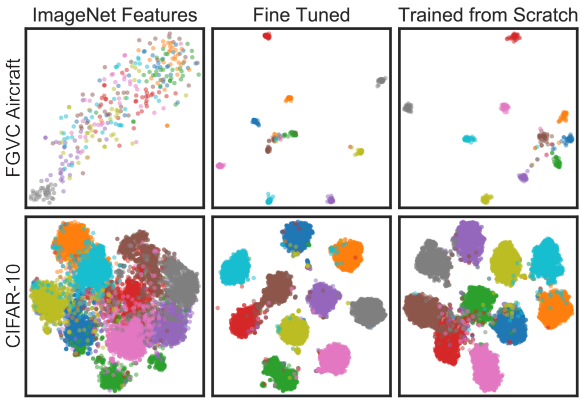

Figure 1: Fine-grained datasets (FGVC Aircraft) benefit significantly from fine-tuning or training from scratch. Datasets similar to ImageNet (CIFAR-10) benefit less. Low dimensional embedding using t-SNE [13] on features from the penultimate layer of Inception v4, for 10 classes from the test set.

Early works using convolutional neural networks (CNNs) for transfer learning extracted fixed features from ImageNet-trained networks and used these features to train SVMs and logistic regression classifiers for new tasks [2, 3, 8]. These works found that these features could outperform hand-engineered features even for tasks very distinct from ImageNet classification [2, 3]. Several such studies have compared the performance of AlexNet-like CNNs of varying levels of computational complexity in a transfer learning setting with no fine-tuning. Chatfield et al. [8] found that, out of three networks, the two more computationally expensive networks performed better on PASCAL VOC. Similar work concluded that deeper networks produce higher accuracy across many transfer tasks, but wider networks produce lower accuracy [17].

A number of studies have compared the accuracy of classifiers trained on fixed image features versus fine-tuning the image representations on a new dataset [8, 11, 12]. Fine-tuning typically achieves higher accuracy, especially for larger datasets or datasets with a larger domain mismatch from the training set [17–21]. In object detection, ImageNet-pretrained networks are used as backbone models for Faster R-CNN and R-FCN detection systems. Classifiers with higher ImageNet accuracy achieve higher overall object detection accuracy [4], although variability across network architectures is small compared to variability from other object detection architecture choices. A parallel story likewise appears in image segmentation models [6], although it has not been as systematically explored.

Several authors have investigated how properties of the original training dataset affect transfer accuracy. Examining performance of fixed image features drawn from networks trained on subsets of ImageNet, Azizpour et al. [17] reported that the number of classes is more important, while Huh, Agrawal, and Efros [20] reported that the number of images per class is more important, provided that the classes are sampled at random rather than split according to the WordNet hierarchy. Yosinski et al. [18] showed that the first layer of AlexNet can be frozen when transferring between natural and manmade subsets of ImageNet without performance impairment, but freezing later layers produces a substantial drop in accuracy.

3 Statistical methods

Much of the analysis in this work requires comparing accuracies across datasets of differing difficulty. When fitting linear models to accuracy values across multiple datasets, we consider effects of model and dataset to be additive. In this context, using untransformed accuracy as a dependent variable is problematic: The meaning of a 1% additive increase in accuracy is different if it is relative to a base accuracy of 50% vs. 99%. Thus, we consider the accuracy after the logit transformation $\text{logit}(p) = \log(p/(1-p)) = \text{sigmoid}^{-1}(p)$. The logit transformation is the most commonly used transformation for analysis of proportion data, and an additive change Δ in logit-transformed accuracy has a simple interpretation as a multiplicative change $\exp \Delta$ in the odds of correct classification:

$$\text{logit}\left(\frac{n_{\text{correct}}}{n_{\text{correct}} + n_{\text{incorrect}}}\right) + \Delta = \log\left(\frac{n_{\text{correct}}}{n_{\text{incorrect}}}\right) + \Delta = \log\left(\frac{n_{\text{correct}}}{n_{\text{incorrect}}} \exp \Delta\right)$$

After the logit transformation, results do not depend upon whether performance is measured in terms of accuracy or error rate, since $\text{logit}(p) = -\text{logit}(1-p)$. We plot results on logit-scaled axes.

We tested for significant differences between pairs of networks on the same dataset using a permutation test or equivalent binomial test of the null hypothesis that the predictions of the two networks are equally likely to be correct; see Appendix A.1.1 for further details. We tested for significant differences between networks in average performance across datasets using a Wilcoxon signed rank test.

When examining the strength of the correlation between ImageNet accuracy and accuracy on transfer datasets, we report r^2 for the correlation between the logit-transformed ImageNet accuracy and the logit-transformed transfer accuracy averaged across datasets. We also report the rank correlation (Spearman's ρ) in Appendix A.1.2. We computed error bars for model accuracy averaged across datasets using the procedure from Morey [22] to remove variability due to inherent differences in dataset difficulty. Briefly, given logit-transformed accuracies x_{md} of model $m \in \mathcal{M}$ on dataset $d \in \mathcal{D}$, we compute adjusted accuracies $\text{acc}(m, d) = x_{\text{md}} - \sum_{n \in \mathcal{M}} x_{\text{nd}}/|\mathcal{M}|$. For each model, we take the mean and standard error of the adjusted accuracy across datasets, and multiply the latter by a correction factor $\sqrt{|\mathcal{D}|/(|\mathcal{D}|-1)}$.

4 Results

We examined 13 networks ranging in ImageNet top-1 accuracy from 69.8% to 82.7%. These networks encompassed the widely used architectures VGG [9]; Inception v1-v4 and Inception-ResNet v2 [23–26]; ResNet-50, ResNet-101, and ResNet-152 [14]; MobileNet v1 [10]; and the mobile (4 @ 1056) and large (6 @ 4032) variants of NASNet [27]. Appendix A.3 describes models in further detail and lists the ImageNet top-1 accuracy, parameter count, dimension of the penultimate layer, and input image size for each network. For all experiments, we rescaled images to the same image size as was used for ImageNet training.

Table 1: Datasets examined in transfer learning

Dataset	Classes	Size (train/test)	Accuracy measure
Food-101 [28]	101	75,750/25,250	top-1
CIFAR-10 [29]	10	50,000/10,000	top-1
CIFAR-100 [29]	10	50,000/10,000	top-1
Birdsnap [30]	500	47,386/2,443	top-1
SUN397 [31]	397	19,850/19,850	top-1
Stanford Cars [32]	196	8,144/8,041	top-1
FGVC Aircraft [33]	100	6,667/3,333	mean per-class
PASCAL VOC 2007 Cls. [34]	20	5,011/4,952	11-point mAP
Describable Textures (DTD) [35]	47	3,760/1,880	top-1
Oxford-IIIT Pets [36]	37	3,680/3,369	mean per-class
Caltech-101 [37]	102	3,060/6,084	mean per-class
Oxford 102 Flowers [38]	102	2,040/6,149	mean per-class

We evaluated models on 12 image classification datasets ranging in training set size from 2,040 to 75,750 images (20 to 5,000 images per class; Table 1). These datasets covered a wide range of image

classification tasks, including superordinate-level object classification (CIFAR-10 [29], CIFAR-100 [29], PASCAL VOC 2007 [34], Caltech-101 [37]); fine-grained object classification (Food-101 [28], Birdsnap [30], Stanford Cars [32], FGVC Aircraft [33], Oxford-IIIT Pets [36]); texture classification (DTD [35]); and scene classification (SUN397 [31]).

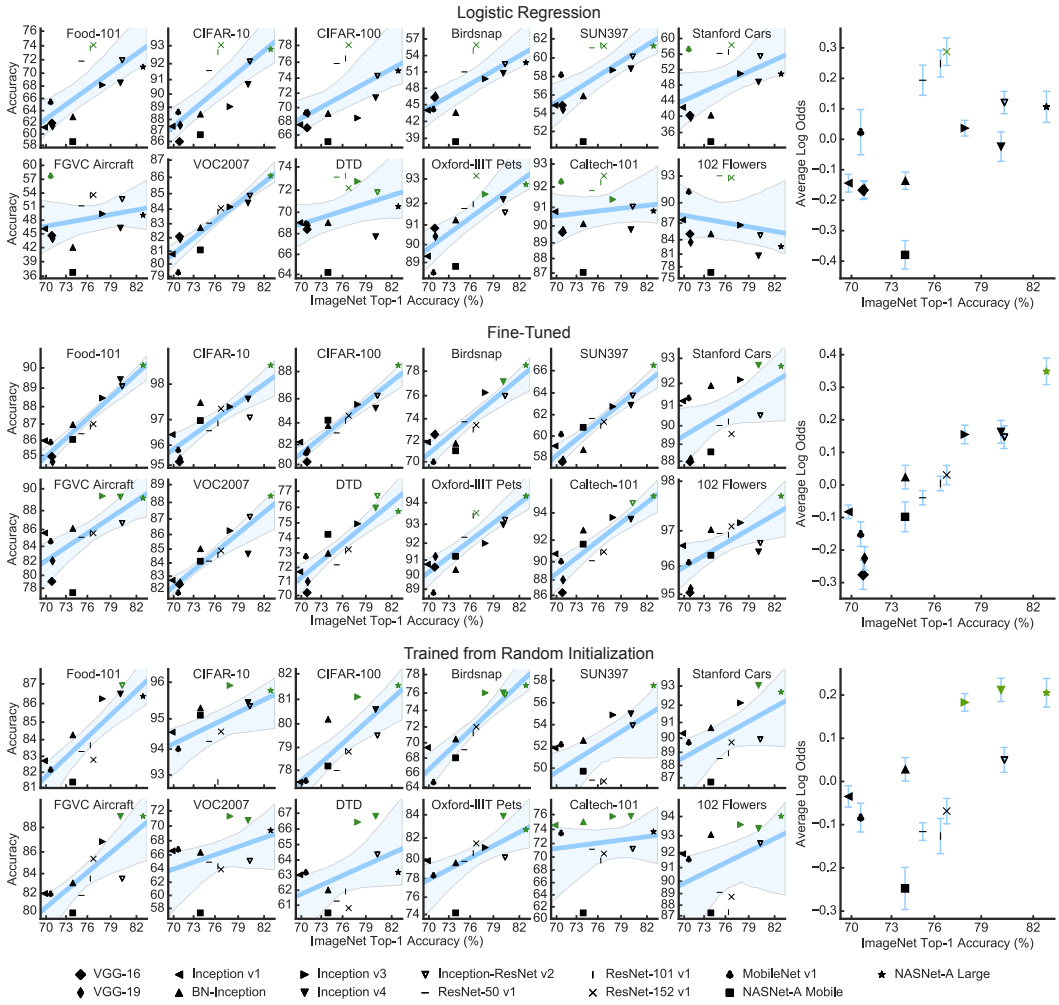

Figure 2: ImageNet accuracy is a strong predictor of transfer accuracy in fine-tuned models. Each set of panels measures correlations between ImageNet accuracy and transfer accuracy across fixed ImageNet features (top), fine-tuned networks (middle) and randomly initialized networks (bottom). Left: Relationship between classification accuracy on transfer datasets (y-axis) and ImageNet top-1 accuracy (x-axis) in different training settings. Axes are logit-scaled (see text). The regression line and a 95% bootstrap confidence interval are plotted in blue. Right: Average log odds of correct classification across datasets, relative to the mean of all classifiers on the dataset. Error bars are standard error. Points corresponding to models not significantly different from the best model ($p > 0.05$) are colored green.

Figure 2 presents correlations between the top-1 accuracy on ImageNet versus the performance of the same model architecture on new image tasks. We measure transfer learning performance in three settings: (1) training a logistic regression classifier on the *fixed* feature representation from the penultimate layer of the ImageNet-pretrained network, (2) fine-tuning the ImageNet-pretrained network, and (3) training the same CNN architecture from scratch on the new image task.

4.1 ResNets are the best fixed feature extractors

We first examined the performance of different networks when used as *fixed* feature extractors by training an L2-regularized logistic regression classifier on penultimate layer activations using L-BFGS [39] without data augmentation. In this setting, ImageNet accuracy accounted for only a small fraction of the differences in transfer accuracy among models ($r^2 = 0.24$). In particular, even though ResNet-50, ResNet-101, and ResNet-152 were the 5$^{\text{th}}$, 6$^{\text{th}}$ and 7$^{\text{th}}$ best models in terms of ImageNet top-1 accuracy, they were the top three models for transfer learning by logistic regression. ResNets were the best performing architectures on 9 of the 12 datasets, and insignificantly different from the best performing model on another. The surprisingly high performance of ResNets was not directly attributable to the dimension of the penultimate layer, as our experiments included networks with higher ImageNet top-1 performance that had equal (Inception v3), higher (NASNet Large) and lower (Inception-ResNet v2) feature dimension (Appendix A.3).

We performed several control experiments to verify that the superiority of ResNet-152 as a fixed feature extractor was due to the architecture rather than the training procedure or classifier. First, our results remained unchanged when we repeated the logistic regression experiments using ResNet models trained with different image preprocessing [24, 25, 27]. Second, we retrained one network (Inception v3) with less regularization in a setting similar to ResNets (see Appendix A.4). Although the resulting network was more accurate than ResNet-152 on ImageNet, ResNet-152 still performed better on all transfer datasets except for VOC 2007. Third, we trained SVM and k-NN classifiers, and logistic regression classifiers with data augmentation; results were similar (Appendix B).

4.2 ImageNet accuracy is a good predictor of fine-tuning performance

We next examined performance when fine-tuning ImageNet networks. We initialized each network from the ImageNet weights and fine-tuned for 19,531 steps with Nesterov momentum and a cosine decay learning rate schedule at a batch size of 256, searching across learning rate and weight decay on a validation set (for details, see Appendix A.5). Compared to logistic regression on penultimate layer features, fine-tuning produced much stronger correlations between ImageNet accuracy and transfer dataset accuracy ($r^2 = 0.86$). Correspondingly, the best performing model on ImageNet tested [27] was the best performing model on 9 transfer datasets and insignificantly different from the best model on the remainder.

When training from random initialization, correlations were more variable, but there was a tendency toward higher performance for models that achieved higher accuracy on ImageNet ($r^2 = 0.45$). For these experiments, we used a similar training setup as for fine-tuning (see Appendix A.6). The best network in this context was Inception v4, although Inception v3 and NASNet-A Large were insignificantly worse. These are three of the four highest performing networks on ImageNet tested.

4.3 Fine-tuning with better models outperforms specialized methods for transfer learning

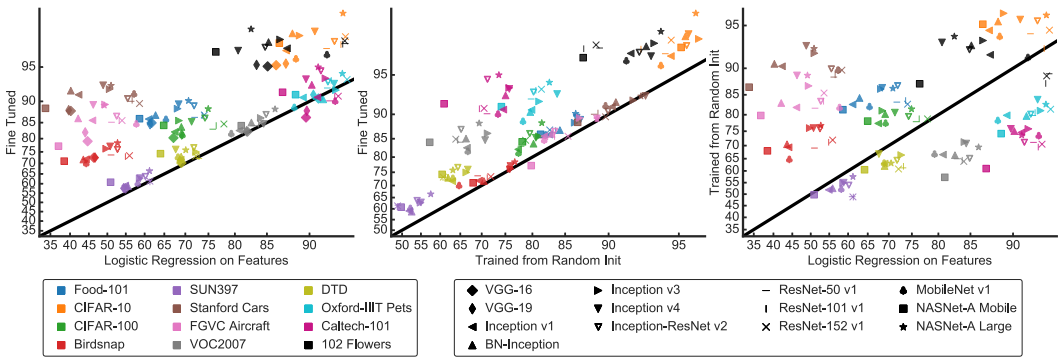

Figure 3: Fine-tuning consistently achieves higher accuracy than logistic regression on top of fixed ImageNet features or training from randomly initialized models. The performance of logistic regression on fixed ImageNet features vs. networks trained from random initialization depends heavily on the dataset. Axes are logit scaled.

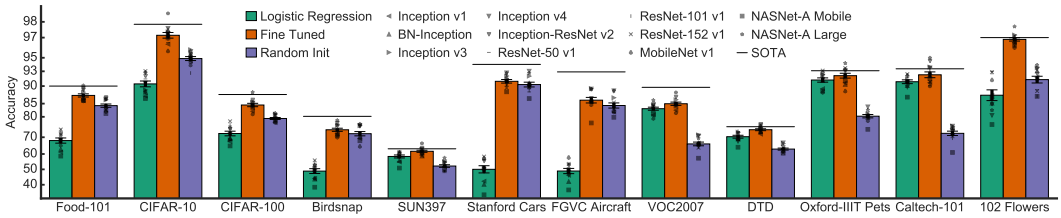

Figure 4: Fine-tuning achieves state-of-the-art performance. Bars reflect accuracy across models (excluding VGG) for logistic regression, fine-tuning, and training from random initialization. Error bars are standard error. Points represent individual models. Lines represent previous state-of-the-art.

Fine-tuning was substantially more accurate than classifiers trained on fixed features for most datasets. As shown in Figure 3 (left), fine-tuning improved performance over logistic regression in 143 out of 156 dataset and model combinations. For all datasets, the best fine-tuned model was better than the best model fit by logistic regression. When averaged across the tested architectures, fine-tuning yielded significantly better results on all datasets except Caltech-101 ($p < 0.05$, Wilcoxon signed rank test; Figure 4). The improvement was generally larger for larger datasets. However, fine-tuning produced substantial gains on the smallest dataset, 102 Flowers, with 102 classes and 2,040 training examples.

Table 2: Performance of best models.

	Previously reported		Current work	
Dataset	Acc.	Network	Acc.	Best network
Food-101	**90.0**	Deep layer aggregation [40]	**90.1**	NASNet-A Large, fine-tuned
CIFAR-10	97.9	AmoebaNet [41]	**98.4**[a]	NASNet-A Large, fine-tuned
CIFAR-100	87.8	ShakeDrop [42]	**88.2**[a]	NASNet-A Large, fine-tuned
Birdsnap	**80.2**[b]	Mask-CNN [43]	78.5	NASNet-A Large, fine-tuned
SUN397	63.2	Places-pretrained VGG [44]	**66.5**	NASNet-A Large, fine-tuned
Stanford Cars	**94.1**	Deep layer aggregation [40]	93.0	Inception v4, random init
FGVC Aircraft	**92.9**[b]	Deep layer aggregation [40]	89.4	Inception v3, fine-tuned
VOC 2007 Cls.	**89.7**	VGG [9]	88.4	NASNet-A Large, fine-tuned
DTD	75.5	FC+FV-CNN+D-SIFT [45]	**76.7**	Inception-ResNet v2, fine-tuned
Oxford-IIIT Pets	93.8	Object-part attention [46]	**94.3**	NASNet-A Large, fine-tuned
Caltech-101	93.4	Spatial pyramid pooling [47]	**95.0**	NASNet-A Large, fine-tuned
Oxford 102 Flowers	97.1	Object-part attention [46]	**97.7**	NASNet-A Large, fine-tuned

[a] Accuracy excludes images duplicated between the ImageNet training set and CIFAR test sets; see Appendix D.
[b] Krause et al. [48] achieve 85.4% on Birdsnap and 95.9% on Aircraft using bird and aircraft images collected from Google image search.

In Table 2, we compare the performance of our models against the best previously reported results on each dataset (see Appendix C for all numerical results). We achieve state-of-the-art performance on 8 of the 12 datasets. For CIFAR-10 and CIFAR-100, the best previously reported results were achieved without auxiliary training data, so the comparison is not strictly fair. All other benchmarks use ImageNet-pretrained networks. Our results suggest that architecture is a critical factor in transfer performance. Several papers have proposed methods to make better use of CNN features and thus improve the efficacy of transfer learning [19, 45, 46, 49–54]. On the we datasets examine (Table 2), we outperform all such methods simply by fine-tuning state-of-the-art CNNs. Moreover, in some cases, a better CNN can make up for dataset deficiencies: By fine-tuning ImageNet-pretrained NASNet Large, we obtain state-of-the-art performance on the SUN397 scene dataset, outperforming features extracted from VGG trained on the Places dataset [44], which more closely matches the domain of SUN397. However, the results on Birdsnap and FGVC Aircraft fall far short of Krause et al. [48], who augment these datasets with bird and aircraft images collected from the Internet.

Even after excluding methods that perform dataset-specific collection of additional training images, we do not achieve state-of-the-art on the Birdsnap, Cars, Aircraft, or VOC 2007 datasets. The best performing methods on these datasets evaluated at higher image resolutions (i.e., 448×448 [40] and 768×768 [9]) than the fine-tuned networks examined in this work. It is known that networks

perform better at larger image sizes at the expense of computational cost, both on ImageNet [10] and in transfer settings [49, 55].

4.4 ImageNet pretraining does not necessarily improve accuracy on fine-grained tasks

Fine-tuning was more accurate than training from random initialization for 129 out of 132 dataset/model combinations, but on Birdsnap, Stanford Cars, and FGVC Aircraft, the improvement was unexpectedly small (Figures 3 and 4). Fine-tuned models were only marginally better than models trained from random initialization (odds ratios of correct classification of 1.11, 1.09, and 1.15 respectively; Figure 4). For Stanford Cars, the best model trained from scratch (Inception v4) outperformed the best fine-tuned model, although the difference was not significant ($p = 0.23$, binomial test). This model obtained 93.0% accuracy, close to the state-of-the-art result of 94.1% [40].

ImageNet pretraining has marginal benefits for fine-grained classification tasks where labels are not well-represented in the ILSVRC2012 hierarchy. Stanford Cars and FGVC Aircraft are smaller than most datasets used to train CNNs [56] (8,144 training examples/196 classes and 6,667 examples/100 classes, respectively). Three other datasets (Pets, 102 Flowers, and Food-101) require similarly fine-grained classification. However, Pets comprises classes of cats and dogs, which are well-represented within ImageNet, and 102 Flowers is very small, with only 20 training images per class. Performance of training from random initialization on Food-101 approached ImageNet fine-tuning, but there was still a substantial gap (odds ratio 1.31).

4.5 ImageNet pretraining accelerates convergence

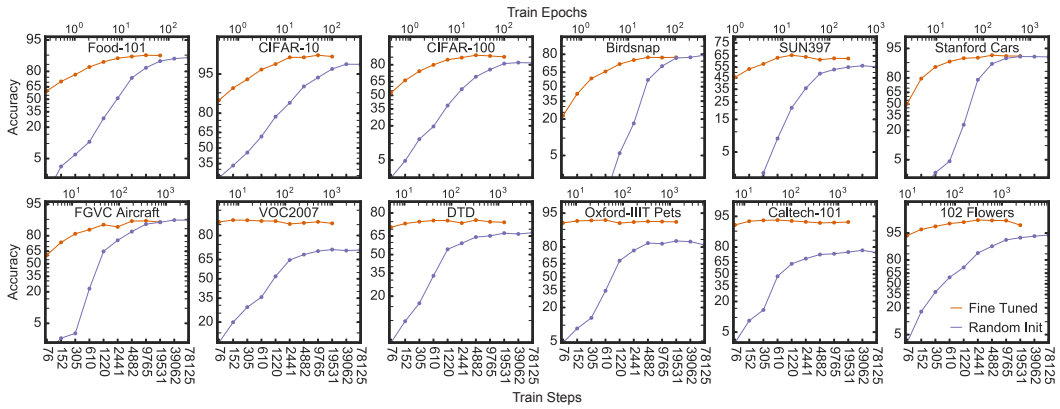

Figure 5: Networks pretrained on ImageNet converge faster. Each point represents an independent Inception v4 model trained with optimized hyperparameters. Axes are logit scaled.

Given that fine-tuning and training from random initialization achieved similar performance on Birdsnap, Stanford Cars, and FGVC Aircraft, we next asked whether fine-tuning still posed an advantage in terms of training time. In Figure 5, we examine performance of Inception v4 when fine-tuning or training from random initialization for different numbers of steps. Even when fine-tuning and training from scratch achieved similar final accuracy, we could fine-tune the model to this level of accuracy in an order of magnitude fewer steps. To quantify the acceleration provided by fine-tuning, we computed the number of epochs and steps required to reach 90% of the maximum odds of correct classification achieved at any number of steps, and computed the geometric mean across datasets. Fine-tuning reached this threshold level of accuracy in an average of 26 epochs/1151 steps (inter-quartile ranges 267-4882 steps, 12-58 epochs), whereas training from scratch required 444 epochs/19531 steps (inter-quartile ranges 9765-39062 steps, 208-873 epochs) corresponding to a 17-fold speedup on average.

4.6 Measuring when pretraining improves performance

We finally examined the behavior of logistic regression, fine-tuning, and training from random initialization in the regime of extremely limited data. In Figure 6, we show the accuracy achieved

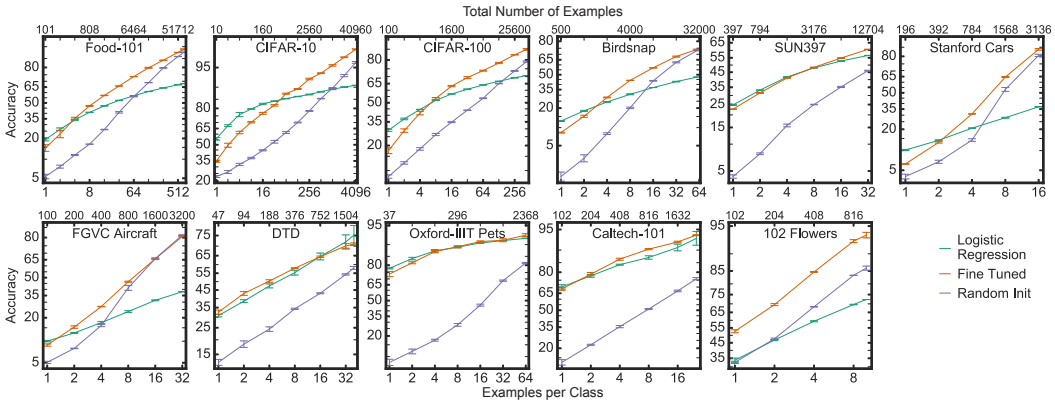

Figure 6: Pretraining on ImageNet improves performance on fine-grained tasks with small amounts of data. Performance of Inception v4 at different dataset sizes. Error bars reflect standard error over 3 subsets. Note that the maximum dataset size shown is not the full dataset.

by these methods for dataset subsets with different numbers of examples per class. When labeled data is sparse (47-800 total examples, or 1-32 per class), logistic regression is a strong baseline, often achieving accuracy comparable to or better than fine-tuning. At larger dataset sizes, fine-tuning achieves higher performance. However, the advantages of ImageNet pretraining can fade surprisingly quickly with more data: On FGVC Aircraft, training from random initialization achieved parity with fine-tuning at 1600 total examples (16 per class).

5 Discussion

The last decade of computer vision research has demonstrated the superiority of image features learned from data over generic, hand-crafted features. Before the rise of convolutional neural networks, most approaches to image understanding relied on manually engineered feature descriptors [57–59] combined with methods to aggregate these descriptors [60]. Krizhevsky, Sutskever, and Hinton [61] showed that, given the training data provided by ImageNet [1], features learned by convolutional neural networks could substantially outperform these hand-engineered features. Soon after, it became clear that intermediate representations learned from ImageNet also provided substantial gains over hand-engineered features when transferred to other tasks [2, 3].

Our results reveal clear limits to transferring features, even among natural image datasets. But we also show that, even when features are not transferable, better architectures consistently achieve higher performance. We found that the best *fixed* image features do not come from the best ImageNet models, as measured by ImageNet accuracies. However, *fine-tuning* better ImageNet models yields better performance. ImageNet pretraining accelerates convergence and improves performance on many datasets, but its value diminishes with greater training time, more training data, and greater divergence from ImageNet labels. For some fine-grained classification datasets, a few thousand labeled examples, or a few dozen per class, are all that are needed to make training from scratch perform competitively with fine-tuning. Surprisingly, however, the value of architecture persists.

Is the general enterprise of learning widely-useful features doomed to suffer the same fate as feature engineering in computer vision? Given differences among datasets [16], it is not entirely surprising that features learned on one dataset benefit from some amount of adaptation (i.e., fine-tuning) when applied to another. It is, however, surprising that features learned from a large dataset cannot always be profitably adapted to much smaller datasets. ImageNet weights provide a starting point for features on a new classification task, but perhaps what is needed is a way to learn *how* to adapt features. This problem is closely related to few-shot learning [62–67], but these methods are typically evaluated with training and test classes from the same distribution. It remains to be seen whether methods can be developed to adapt visual representations learned from ImageNet to provide larger benefits across all natural image tasks.

Acknowledgements

We thank George Dahl, Sara Hooker, Pieter-jan Kindermans, Jiquan Ngiam, Ruoming Pang, Daiyi Peng, Vishy Tirumalashetty, Vijay Vasudevan, and Emily Xue for comments on the experiments and manuscript, and Aliza Elkin and members of the Google Brain team for support and ideas.

References

[1] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 248–255.

[2] Jeff Donahue et al. "Decaf: A deep convolutional activation feature for generic visual recognition". In: *International Conference on Machine Learning*. 2014, pp. 647–655.

[3] Ali Sharif Razavian et al. "CNN features off-the-shelf: an astounding baseline for recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE. 2014, pp. 512–519.

[4] Jonathan Huang et al. "Speed/accuracy trade-offs for modern convolutional object detectors". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017. arXiv: 1611.10012.

[5] Kaiming He et al. "Mask R-CNN". In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 2980–2988.

[6] Liang-Chieh Chen et al. "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2018), pp. 834–848.

[7] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European Conference on Computer Vision*. Springer. 2016, pp. 694–711.

[8] Ken Chatfield et al. "Return of the devil in the details: delving deep into convolutional nets". In: *British Machine Vision Conference*. 2014. arXiv: 1405.3531 [cs].

[9] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *International Conference on Learning Representations* (2015).

[10] Andrew G Howard et al. "MobileNets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[11] Pulkit Agrawal, Ross B. Girshick, and Jitendra Malik. "Analyzing the performance of multilayer neural networks for object recognition". In: *European Conference on Computer Vision*. 2014. arXiv: 1407.1610.

[12] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 580–587.

[13] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[14] Kaiming He et al. "Deep residual learning for image recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[15] Kaiming He et al. "Identity mappings in deep residual networks". In: *European Conference on Computer Vision*. Springer. 2016, pp. 630–645.

[16] Antonio Torralba and Alexei A Efros. "Unbiased look at dataset bias". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2011, pp. 1521–1528.

[17] H. Azizpour et al. "Factors of Transferability for a Generic ConvNet Representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.9 (Sept. 2016), pp. 1790–1802. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2015.2500224.

[18] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems*. 2014, pp. 3320–3328.

[19] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. "Bilinear CNN models for fine-grained visual recognition". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1449–1457.

[20] Mi-Young Huh, Pulkit Agrawal, and Alexei A. Efros. "What makes ImageNet good for transfer learning?" In: *CoRR* abs/1608.08614 (2016). arXiv: 1608.08614. URL: http://arxiv.org/abs/1608.08614.

[21] Brian Chu et al. "Best Practices for Fine-Tuning Visual Classifiers to New Domains". In: *Computer Vision – ECCV 2016 Workshops*. Ed. by Gang Hua and Hervé Jégou. Cham: Springer International Publishing, 2016, pp. 435–442. ISBN: 978-3-319-49409-8.

[22] Richard D. Morey. "Confidence intervals from normalized data: A correction to Cousineau (2005)". In: *Tutorials in Quantitative Methods for Psychology* 4.2 (2008), pp. 61–64. DOI: 10.20982/tqmp.04.2.p061. URL: http://www.tqmp.org/RegularArticles/vol04-2/p061/p061.pdf.

[23] Christian Szegedy et al. "Going deeper with convolutions". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[24] Sergey Ioffe and Christian Szegedy. "Batch normalization: accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[25] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826.

[26] Christian Szegedy et al. "Inception-v4, Inception-ResNet and the impact of residual connections on learning." In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*. 2017.

[27] Barret Zoph et al. "Learning transferable architectures for scalable image recognition". In: *arXiv preprint arXiv:1707.07012* (2017).

[28] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101 — Mining discriminative components with random forests". In: *European Conference on Computer Vision*. Springer. 2014, pp. 446–461.

[29] Alex Krizhevsky and Geoffrey Hinton. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto, 2009. URL: `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[30] Thomas Berg et al. "Birdsnap: Large-scale fine-grained visual categorization of birds". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2014, pp. 2019–2026.

[31] Jianxiong Xiao et al. "SUN database: Large-scale scene recognition from abbey to zoo". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2010, pp. 3485–3492.

[32] Jonathan Krause et al. "Collecting a large-scale dataset of fine-grained cars". In: *Second Workshop on Fine-Grained Visual Categorization*. 2013.

[33] S. Maji et al. *Fine-Grained Visual Classification of Aircraft*. Tech. rep. 2013. arXiv: `1306.5151 [cs-cv]`.

[34] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) challenge". In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338.

[35] Mircea Cimpoi et al. "Describing textures in the wild". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2014, pp. 3606–3613.

[36] Omkar M Parkhi et al. "Cats and dogs". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2012, pp. 3498–3505.

[37] 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". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Generative-Model Based Vision*. 2004.

[38] Maria-Elena Nilsback and Andrew Zisserman. "Automated flower classification over a large number of classes". In: *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*. IEEE. 2008, pp. 722–729.

[39] Dong C Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1-3 (1989), pp. 503–528.

[40] Fisher Yu, Dequan Wang, and Trevor Darrell. "Deep Layer Aggregation". In: *CoRR* abs/1707.06484 (2017). arXiv: `1707.06484`. URL: `http://arxiv.org/abs/1707.06484`.

[41] Esteban Real et al. "Regularized Evolution for Image Classifier Architecture Search". In: *CoRR* abs/1802.01548 (2018). arXiv: `1802.01548`. URL: `http://arxiv.org/abs/1802.01548`.

[42] Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. "ShakeDrop regularization". In: *CoRR* abs/1802.02375 (2018). arXiv: `1802.02375`. URL: `http://arxiv.org/abs/1802.02375`.

[43] Xiu-Shen Wei et al. "Mask-CNN: Localizing parts and selecting descriptors for fine-grained bird species categorization". In: *Pattern Recognition* 76 (2018), pp. 704–714. ISSN: 0031-3203.

[44] Bolei Zhou et al. "Places: A 10 million image database for scene recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[45] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. "Deep filter banks for texture recognition and segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015, pp. 3828–3836.

[46] Yuxin Peng, Xiangteng He, and Junjie Zhao. "Object-part attention model for fine-grained image classification". In: *IEEE Transactions on Image Processing* 27.3 (2018), pp. 1487–1500.

[47] Kaiming He et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *European Conference on Computer Vision*. Springer. 2014, pp. 346–361.

[48] Jonathan Krause et al. "The unreasonable effectiveness of noisy data for fine-grained recognition". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 301–320. ISBN: 978-3-319-46487-9.

[49] Tsung-Yu Lin and Subhransu Maji. "Visualizing and understanding deep texture representations". In: *IEEE International Conference on Computer Vision (ICCV)*. 2016, pp. 2791–2799.

[50] Yang Gao et al. "Compact bilinear pooling". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 317–326.

[51] Hantao Yao et al. “Coarse-to-fine description for fine-grained visual categorization”. In: *IEEE Transactions on Image Processing* 25.10 (2016), pp. 4858–4872.

[52] Yang Song et al. “Locally-transferred Fisher vectors for texture classification”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4912–4920.

[53] Yin Cui et al. “Kernel pooling for convolutional neural networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[54] Zhichao Li et al. “Dynamic computational time for visual attention”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1199–1209.

[55] Yin Cui et al. “Large scale fine-grained categorization and domain-specific transfer learning”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[56] Blair Hanley Frank. “Google Brain chief: Deep learning takes at least 100,000 examples”. In: *VentureBeat* (2017). URL: https://venturebeat.com/2017/10/23/google-brain-chief-says-100000-examples-is-enough-data-for-deep-learning/.

[57] David G Lowe. “Object recognition from local scale-invariant features”. In: *IEEE International Conference on Computer Vision*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[58] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE. 2005, pp. 886–893.

[59] Herbert Bay et al. “Speeded-up robust features (SURF)”. In: *Computer Vision and Image Understanding* 110.3 (2008), pp. 346–359.

[60] Pedro F Felzenszwalb et al. “Object detection with discriminatively trained part-based models”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.9 (2010), pp. 1627–1645.

[61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105.

[62] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. “Human-level concept learning through probabilistic program induction”. In: *Science* 350.6266 (2015), pp. 1332–1338.

[63] Oriol Vinyals et al. “Matching networks for one shot learning”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[64] Sachin Ravi and Hugo Larochelle. “Optimization as a model for few-shot learning”. In: *International Conference on Machine Learning*. 2016.

[65] Jake Snell, Kevin Swersky, and Richard Zemel. “Prototypical networks for few-shot learning”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4080–4090.

[66] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *International Conference on Machine Learning*. 2017, pp. 1126–1135.

[67] Nikhil Mishra et al. “A simple neural attentive meta-learner”. In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=B1DmUzWAW.

A Supplementary experimental procedures

A.1 Supplementary statistical methods

A.1.1 Comparison of two models on the same dataset

To test for superiority of one model over another on a given dataset, we constructed permutations where, for each example, we randomly exchanged the predictions of the two networks. For each permutation, we computed the difference in accuracy between the two networks.[2] We computed a p-value as the proportion of permutations where the difference is at least as extreme as the observed difference in accuracy. For top-1 accuracy, this procedure is equivalent to a binomial test sometimes called the "exact McNemar test," and a p-value can be computed exactly. For mean per-class accuracy, we approximated a p-value based on 10,000 permutations. These tests assess whether one trained model performs better than another on data drawn from the test set distribution. However, they are tests between trained models, rather than tests between architectures, since we do not measure variability arising from training networks from different random initializations or from different orderings of the training data.

A.1.2 Measures of correlation

Table 3: Correlations between ImageNet accuracy and average transfer accuracy

Setting	Pearson r^2	Pearson r	Spearman ρ
Logistic regression	0.24	0.49	0.59
Fine-tuned	0.86	0.92	0.90
Trained from scratch	0.45	0.67	0.65

Table 3 shows the Pearson correlation (as r^2 and r) as well as the Spearman rank correlation (ρ) in each of the three transfer settings we examine. We believe that Pearson correlation is the more appropriate measure, given that it is less dependent on the specific CNNs chosen for the study and the effects are approximately linear, but our results are similar in either case.

A.2 Datasets

All datasets had a median image size on the shortest side of at least 331 pixels (the highest input image size out of all networks tested), except Caltech-101, for which the median size is 225 on the shortest side and 300 on the longer side, and CIFAR-10 and CIFAR-100, which consist of 32×32 pixel images.

For datasets with a provided validation set (FGVC Aircraft, VOC2007, DTD, and 102 Flowers), we used this validation set to select hyperparameters. For other datasets, we constructed a validation set by subsetting the original training set. For the DTD and SUN397 datasets, which provide multiple train/test splits, we used only the first provided split. For the Caltech-101 dataset, which specifies no train/test split, we trained on 30 images per class and tested on the remainder, as in previous works [1–4]. With the exception of dataset subset results (Figure 6), all results indicate the performance of models retrained on the combined training and validation set.

A.3 Networks

We used publicly available network checkpoints and models from the TF-Slim model repository (`https://github.com/tensorflow/models`). Table 4 lists the ImageNet top-1 accuracy, parameter count, penultimate layer feature dimension, and input image size for each checkpoint. For VGG and ResNets, we used the preprocessing and data augmentation described in Simonyan and Zisserman [2]. For other networks, we used the preprocessing and data augmentation from The TensorFlow Authors [5].

The majority of our networks were trained with crops after a non-aspect ratio preserving resize, following the data augmentation procedure described in Szegedy et al. [8]. ResNet and VGG

[2]For VOC2007, we considered the accuracy of predictions across labels.

Table 4: ImageNet classification networks

Model	ImageNet top-1[a]	Parameters[b]	Features	Image size
VGG-16 (D) [2]	70.9%	134.3M	4096	224
VGG-19 (E) [2]	71.0%	139.6M	4096	224
Inception v1[c] [6]	69.8%	5.6M	1024	224
BN-Inception[d] [7]	74.0%	10.1M	1024	224
Inception v3 [8]	78.0%	21.8M	2048	299
Inception v4 [9]	80.2%	41.1M	1536	299
Inception-ResNet v2 [9]	80.4%	54.3M	1536	299
ResNet-50 v1 [10]	75.2%	23.5M	2048	224
ResNet-101 v1 [10]	76.4%	42.5M	2048	224
ResNet-152 v1 [10]	76.8%	58.1M	2048	224
MobileNet v1 [11]	70.7%	3.2M	1024	224
NASNet-A Mobile [12]	74.0%	4.2M	1056	224
NASNet-A Large [12]	82.7%	84.7M	4032	331

[a] Accuracy of model implementation, which may differ slightly from the original reported result.

[b] Excludes logits layer.

[c] We used Inception models and checkpoints from `https://github.com/tensorflow/models`, which do not match the original papers.

[d] This model is called "Inception v2" in TF-Slim model repository, but matches the model described in Ioffe and Szegedy [7], rather than the model that Szegedy et al. [8] call "Inception v2."

models were converted from the original checkpoints of He et al. [10], and were trained with aspect ratio-preserving crops as in Simonyan and Zisserman [2].

A.4 Logistic regression

For each dataset, we extracted features from the penultimate layer of the network and normalized them by their L2 norm as in Simonyan and Zisserman [2] and Chatfield et al. [4]. We trained a multinomial logistic regression classifier using L-BFGS, with an L2 regularization parameter applied to the sum of the per-example losses, selected from a range of 45 logarithmically spaced values from 10^{-6} to 10^{5} on the validation set. Since the optimization problem is convex, we used the solution at the previous point along the regularization path can be used as a warm start for the next point, which greatly accelerated the search. For these experiments, we did not perform data augmentation or scale aggregation.

When investigating the effect of preprocessing upon performance, we tested the performance of ResNet-50 v1, ResNet-101 v1, and ResNet-152 v1 models retrained with the preprocessing from Szegedy et al. [6] with label smoothing of 0.1 and an input size of 299×299. This model was trained with momentum with an exponentially decaying learning rate schedule, and used an exponential moving average of the weights, as in Szegedy et al. [8]. These models achieved higher performance than non-ResNet models on 9 of 12 datasets.

When investigating the effect of training procedure, we trained an Inception v3 model with regularization and learning rate schedule more comparable to He et al. [10]. Compared to the original Inception v3 model, this model had no auxiliary classifier, no dropout, and no label smoothing, and was trained with momentum and a stepwise learning rate schedule rather than RMSProp with exponential decay. This model achieved an ImageNet top-1 accuracy of 77.2%, 0.8% worse than our original Inception v3 model, but still 0.4% better than ResNet-152, but achieved lower performance than ResNet-152 on all transfer datasets except VOC2007.

A.5 Fine tuning

For all fine-tuning experiments except dataset size experiments, we initialized networks with ImageNet-pretrained weights and trained for 19,531 steps at a batch size of 256 using Nesterov momentum with a momentum parameter of 0.9. We selected the optimal learning rate and weight decay on the validation set by grid search. Our early experiments indicated that the optimal weight decay at a given learning rate varied inversely with the learning rate, as has been recently reported [13]. Thus, our grid consisted of 7 logarithmically spaced learning rates between 0.0001 and 0.1

and 7 logarithmically spaced weight decay to learning rate ratios between $10^{-5.5}$ and $10^{-2.5}$, as well as zero. We found it useful to decrease the batch normalization momentum parameter from its ImageNet value to $\max(1 - 10/s, 0.9)$ where s is the number of steps per epoch. We left any other hyperparameters at their ImageNet settings. We found that the maximum performance on the validation set at any step during training was very similar to the maximum performance at the last step (presumably because we searched over both learning rate and weight decay), so we did not perform early stopping.

When examining the effect of dataset size (Section 4.6), we trained for at least 1000 steps or 100 epochs (following guidance from our analysis of training time in Section 4.5) at a batch size of 64, with the learning rate range scaled down by a factor of 4. Because we chose hyperparameters based on a large validation set, the results may not reflect what can be accomplished in practice when training on datasets of this size [14].

A.6 Training from random initialization

We used a similar training protocol for training from random initialization as for fine tuning, i.e., we trained for 19,531 steps at a batch size of 256 using Nesterov momentum with a momentum parameter of 0.9. Training from random initialization generally achieved optimal performance at higher learning rates and with greater weight decay, so we adjusted the learning rate range to span from 0.001 to 1.0 and the weight decay to learning rate ratio range to span from 10^{-5} to 10^{-2}.

We found that the best hyperparameters for training VGG tended to be very close to those that caused the network to diverge, so that it was not possible to train the network consistently using the hyperparameters that achieved the best performance on the validation set. We note that implementations of all other networks included batch normalization, which allows the networks to be trained with higher initial learning rates without divergence [7]. Rather than report potentially inaccurate results for VGG due to the extent of hyperparameter tuning required, we omitted it from these analyses.

When examining the effect of dataset size (Section 4.6), we trained from random initialization for at least 78,125 steps or 200 epochs at a batch size of 16, with the learning rate range scaled down by a factor of 16. Our investigation of effects of training time (Section 4.5) indicated that training from random initialization always benefited from increased training time, whereas fine tuning did not. Additionally, pilot experiments indicated that training from random initialization, but not fine tuning, benefited from a reduced batch size with very small datasets.

B Comparison of alternative classifiers

Although a logistic regression classifier trained on the penultimate layer activations has a natural interpretation as retraining the last layer of the neural network, previous studies have typically reported results with SVM [1, 2, 15]. Thus, we examine performance in this setting as well (Figure 7). SVM and logistic regression results were extremely highly correlated ($r = 0.9994$), and ResNets remained the top performing architectures on 9 datasets, and insignificantly different from the best on one other. SVM achieved higher performance than logistic regression on 102/156 dataset/model pairs, but on average, the improvement was marginal (odds ratio 1.006, $t(155) = 2.4, p = 0.02$, t-test).

We also examined performance of a k-nearest neighbor classifier, inspired by [16], which reported a reduction in test error for a nearest neighbor classifier trained on the penultimate layer of an MNIST classifier. We selected k on the validation set. Performance was universally worse. Nonetheless, ResNets were the top performing architecture on 7 datasets, and insignificantly different from the top performing architecture on one other.

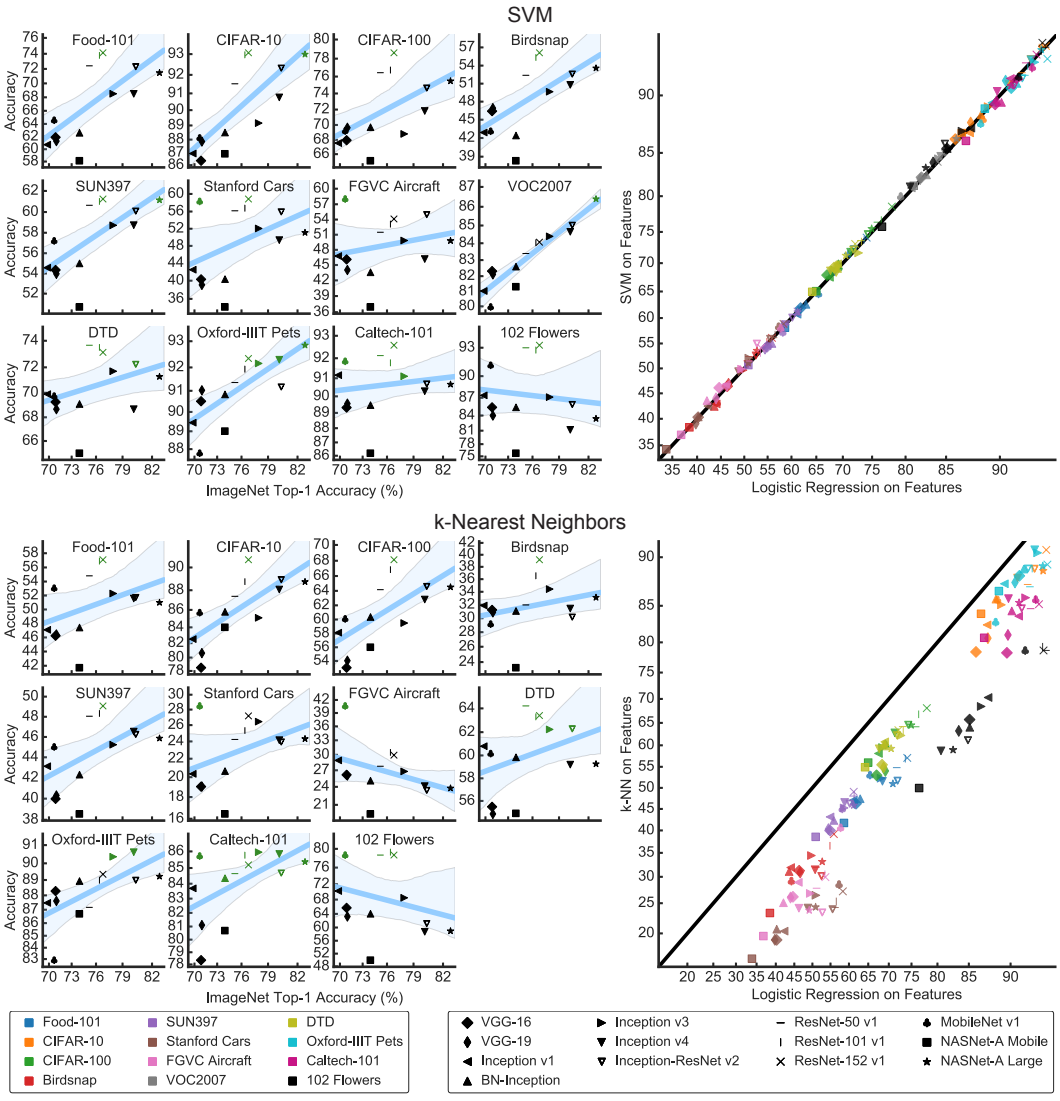

Figure 7: ResNet features are superior for SVM and k-NN classification. Left: Correlation between ImageNet and transfer accuracy, as in Figure 2. Right: Correlation between SVM/k-NN accuracy and logistic regression accuracy.

Finally, we trained a logistic regression classifier with data augmentation, in the same setting we use for fine tuning. We trained for 19,531 steps with Nesterov momentum and a batch size of 256. Because the optimization problem is convex, we did not optimize over learning rate, but instead fixed the learning rate at 0.1. We did optimize over L2 regularization parameters for the final layer, applied to the mean of the per-example losses, selected from a range of 11 logarithmically spaced values between 10^{-10} and 1. Results are shown in Figure 8. With data augmentation, ResNets outperformed other architectures on 7 datasets, and were insignificantly different from best performing models on 3 others. Fine-tuning remained clearly superior to logistic regression with data augmentation, achieving better results for 146/156 dataset/model pairs.

Averaged across all model/dataset combinations, logistic regression with data augmentation outperformed logistic regression without data augmentation (odds ratio 1.04), but the superiority was inconsistent. Logistic regression with data augmentation performed better for only 86/156 dataset/model pairs, and the best performing model without data augmentation was better than the best performing model with data augmentation on half of the 12 datasets. One weakness of this comparison is that, since we trained logistic regression without data augmentation in a full-batch setting with a second-order optimizer (albeit at significantly lower computational cost, since we needed to perform only a single forward pass through the network for each image), it is possible that these models reached parameters closer to the minimum of the loss function.

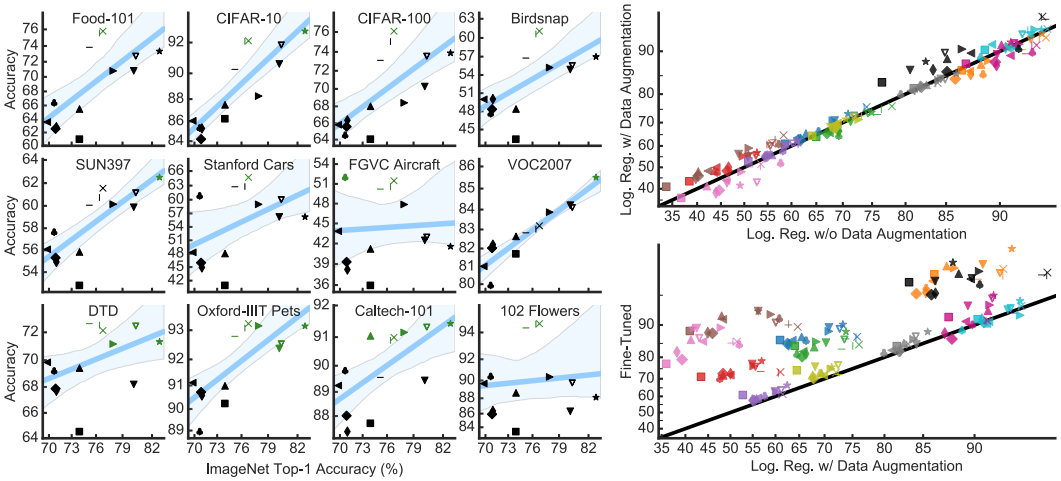

Figure 8: ResNet features are superior for logistic regression trained with data augmentation. Left: Correlation between ImageNet and transfer accuracy, as in Figure 2. Right: Correlation between accuracy of logistic regression without and with data augmentation (top) and between accuracy of logistic regression with data augmentation and fine-tuning (bottom).

C Numerical performance results

We present the numerical results for logistic regression, fine tuning, and training from random initialization in Table 5. Bold-faced numbers represent best models, or models insignificantly different from the best, in each training setting.

Table 5: Model performance

Logistic regression

Network	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech101	Flowers
VGG-16	61.9	86.0	67.1	46.3	54.9	40.2	44.6	82.1	68.4	90.8	89.6	85.1
VGG-19	61.4	87.5	69.2	46.6	54.4	39.7	44.0	81.9	68.8	90.4	89.7	83.5
Inception v1	61.2	87.4	67.5	44.0	54.9	42.2	46.1	80.8	69.0	89.4	90.8	87.5
BN-Inception	62.9	88.4	69.2	43.6	55.9	40.2	42.0	82.7	69.0	91.2	90.1	85.1
Inception v3	68.2	89.1	68.5	49.7	58.7	51.0	49.4	84.1	**72.8**	**92.4**	**91.4**	86.7
Inception v4	68.5	90.7	71.4	50.8	58.8	48.8	46.3	84.4	67.7	92.2	89.8	80.6
Inception-ResNet v2	71.9	92.1	74.3	52.4	60.1	55.5	52.7	84.9	**71.8**	91.6	91.0	84.8
ResNet-50 v1	71.8	91.6	75.8	51.0	**61.1**	56.2	51.2	83.0	**73.1**	91.8	**91.8**	**93.0**
ResNet-101 v1	**73.7**	**92.7**	76.5	**54.9**	**61.4**	56.6	53.5	83.8	**73.2**	92.0	**92.2**	**92.8**
ResNet-152 v1	**74.1**	**93.0**	**78.0**	**55.9**	**61.3**	**58.4**	53.6	84.1	**72.2**	**93.1**	**92.5**	**92.8**
MobileNet v1	65.4	88.6	69.3	44.2	58.2	**57.3**	**57.8**	79.3	68.9	88.4	**92.2**	91.4
NASNet-A Mobile	58.7	86.6	65.0	38.4	50.9	33.8	36.8	81.1	64.2	88.8	87.0	76.6
NASNet-A Large	70.9	**92.8**	74.9	52.7	**61.3**	50.9	49.1	**86.1**	70.5	**92.8**	90.8	82.6

Fine tuning

Network	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech101	Flowers
VGG-16	84.9	95.18	80.3	72.6	57.5	88.0	79.1	82.3	70.2	90.5	86.4	95.07
VGG-19	84.6	95.34	81.4	72.6	57.7	88.1	82.1	82.4	71.0	91.2	88.0	95.22
Inception v1	86.0	96.45	82.2	71.9	59.1	91.2	85.6	82.7	71.7	90.7	90.8	96.62
BN-Inception	87.0	97.53	83.8	71.8	58.7	91.9	86.1	85.0	73.0	90.4	92.8	97.03
Inception v3	88.5	97.41	85.5	76.3	62.8	92.2	**89.4**	86.3	74.9	92.0	93.7	97.19
Inception v4	89.4	97.63	85.2	**77.2**	62.9	**92.7**	**89.3**	84.7	**76.0**	93.0	93.6	96.44
Inception-ResNet v2	89.1	97.07	86.2	76.0	63.8	90.5	86.7	87.1	**76.7**	93.2	**94.6**	96.69
ResNet-50 v1	86.4	96.60	83.1	73.7	61.7	90.0	85.1	84.1	72.2	92.3	90.1	96.94
ResNet-101 v1	86.8	96.87	84.2	73.0	61.0	90.2	85.5	84.6	73.1	93.4	90.9	96.9
ResNet-152 v1	87.0	97.35	84.6	73.4	61.3	89.6	85.6	84.9	73.2	**93.5**	91.0	97.11
MobileNet v1	85.9	95.78	81.2	70.0	60.2	91.4	84.6	81.7	72.8	88.7	90.0	96.13
NASNet-A Mobile	86.1	96.97	84.2	71.1	60.8	88.6	77.3	84.1	74.3	91.2	91.7	96.34
NASNet-A Large	**90.1**	**98.39**	**88.4**	**78.5**	**66.5**	**92.7**	**89.2**	**88.4**	**75.7**	**94.3**	**95.0**	**97.74**

Random initialization

Network	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech101	Flowers
Inception v1	82.7	94.58	77.5	69.4	51.9	90.3	82.0	66.6	63.0	79.8	**74.6**	91.9
BN-Inception	84.3	95.32	80.2	70.5	52.6	90.7	83.1	66.4	62.0	79.6	**75.0**	93.1
Inception v3	86.2	**95.92**	**81.1**	**76.0**	54.9	92.1	86.9	**71.3**	**66.4**	81.1	**75.8**	**93.6**
Inception v4	86.5	95.47	80.6	**76.0**	55.0	**93.0**	**88.8**	**70.8**	**66.8**	**83.9**	**75.8**	**93.4**
Inception-ResNet v2	**86.9**	95.37	79.5	**75.8**	53.9	89.9	83.5	65.1	64.4	80.2	71.2	92.6
ResNet-50 v1	83.3	94.27	78.0	69.1	48.9	88.5	81.8	65.0	61.3	79.8	71.2	89.1
ResNet-101 v1	83.7	92.70	78.9	71.2	48.7	88.9	83.5	64.4	61.9	80.6	69.5	87.4
ResNet-152 v1	82.8	94.60	78.8	72.0	48.8	89.7	85.4	63.9	60.8	81.5	70.5	88.7
MobileNet v1	82.2	94.01	77.5	64.7	52.2	89.7	81.9	66.8	63.2	78.4	73.5	91.6
NASNet-A Mobile	81.4	95.11	78.2	68.0	49.7	86.6	79.8	57.3	60.5	74.2	61.0	87.3
NASNet-A Large	86.4	**95.79**	**81.6**	**76.8**	**57.6**	**92.7**	**88.8**	69.4	63.2	**82.7**	73.7	**94.0**

D Duplicate images

We used a CNN-based duplicate detector trained on synthesized image triplets to detect images that were present in both the ImageNet training set and the datasets we examine. Because the duplicate detector is optimized for speed, it is imperfect. We used a threshold that was conservative based on manual examination, i.e., it resulted in some false positives but very few false negatives. Thus, the results below represent a worst-case scenario for overlap in the datasets examined. Generally, there are relatively few duplicates. For most of these datasets, standard practice is to fine-tune an ImageNet pretrained network without special handling of duplicates, so the presence of duplicates does not affect the comparability of our results to previous work. However, for CIFAR-10 and CIFAR-100, we compare against networks trained from scratch, so we exclude duplicates from the test set.

On CIFAR-10, we achieve an accuracy of 98.39% when fine-tuning NASNet Large (the best model) on the full test set. We also achieve an accuracy of 98.39% on the 9,863 example test set that is disjoint with the ImageNet training set. We achieve an accuracy of 98.54% on the 137 duplicates. On CIFAR-100, we achieve an accuracy of 88.4% on the full test set. We achieve an accuracy of 88.2% on the 9,771 example test set that is disjoint from the ImageNet training set, and an accuracy of 96.94% on the 229 duplicates.

Table 6: Prevalence of images duplicated between the ImageNet training set and transfer datasets investigated

Dataset	Train size	Test size	Train dups	Test dups	Train dup %	Test dup %
Food-101	75,750	25,250	2	1	0.00%	0.00%
CIFAR-10	50,000	10,000	703	137	1.41%	1.37%
CIFAR-100	50,000	10,000	1,134	229	2.27%	2.29%
Birdsnap	47,386	2,443	431	23	0.91%	0.94%
SUN397	19,850	19,850	113	95	0.57%	0.48%
Stanford Cars	8,144	8,041	10	14	0.12%	0.17%
FGVC Aircraft	6,667	3,333	0	1	0.00%	0.03%
VOC2007	5,011	4,952	46	38	0.92%	0.77%
DTD	3,760	1,880	14	9	0.37%	0.48%
Oxford-IIIT Pets	3,680	3,669	227	58	6.17%	1.58%
Caltech-101	3,060	6,084	28	21	0.92%	0.35%
102 Flowers	2,040	6,149	1	0	0.05%	0.00%

References

[1] Jeff Donahue et al. "Decaf: A deep convolutional activation feature for generic visual recognition". In: *International Conference on Machine Learning*. 2014, pp. 647–655.

[2] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *International Conference on Learning Representations* (2015).

[3] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European Conference on Computer Vision*. Springer. 2014, pp. 818–833.

[4] Ken Chatfield et al. "Return of the devil in the details: delving deep into convolutional nets". In: *British Machine Vision Conference*. 2014. arXiv: `1405.3531 [cs]`.

[5] The TensorFlow Authors. *inception_preprocessing.py*. 2016. URL: `https://github.com/tensorflow/models/blob/0344c5503ee55e24f0de7f37336a6e08f10976fd/research/slim/preprocessing/inception_preprocessing.py`.

[6] Christian Szegedy et al. "Going deeper with convolutions". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[7] Sergey Ioffe and Christian Szegedy. "Batch normalization: accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[8] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826.

[9] Christian Szegedy et al. "Inception-v4, Inception-ResNet and the impact of residual connections on learning." In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*. 2017.

[10] Kaiming He et al. “Deep residual learning for image recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[11] Andrew G Howard et al. “MobileNets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

[12] Barret Zoph et al. “Learning transferable architectures for scalable image recognition”. In: *arXiv preprint arXiv:1707.07012* (2017).

[13] Ilya Loshchilov and Frank Hutter. “Fixing Weight Decay Regularization in Adam”. In: *CoRR* abs/1711.05101 (2017). arXiv: 1711.05101. URL: http://arxiv.org/abs/1711.05101.

[14] Augustus Odena et al. “Realistic Evaluation of Semi-Supervised Learning Algorithms”. In: *ICLR Workshops*. 2018. URL: https://openreview.net/pdf?id=ByCZsFyPf.

[15] Ali Sharif Razavian et al. “CNN features off-the-shelf: an astounding baseline for recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE. 2014, pp. 512–519.

[16] Chris Olah. *Neural networks, manifolds, and topology*. Mar. 2014. URL: http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/.