Supplementary Material for “Do Better ImageNet Models Transfer Better?"

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

A. Supplementary experimental procedures

A.1. 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. (For VOC2007, we considered the accuracy of predictions across labels.) 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

Setting	r^2	r	ρ	p-value
Logistic regression	0.97	0.99	0.99	$< 10^{-11}$
Fine-tuned	0.91	0.96	0.97	$< 10^{-8}$
Trained from scratch	0.30	0.55	0.59	0.03
Logistic regression (public checkpoints)	0.14	0.37	0.48	0.16

Table A.1. Correlations between ImageNet accuracy and average transfer accuracy (Pearson r and r^2 and Spearman’s ρ), as well as p-values for the null hypothesis that $r = 0$.

Table A.1 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 ImageNet-native 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 [6, 24, 31, 1]. With the exception of dataset subset results (Figure 9), all results indicate the performance of models retrained on the combined training and validation set.

A.3. Networks and ImageNet training procedure

Table A.2 lists the parameter count, penultimate layer feature dimension, and input image size for each network examined. Unless otherwise stated, our results were obtained with networks we trained, rather than publicly available checkpoints. We

Model	Parameters[a]	Features	Image Size	ImageNet Top-1 Accuracy Paper	Public Checkpoint[b]	Retrained
Inception v1[c] [26]	5.6M	1024	224	73.2	69.8	73.6
BN-Inception[d] [15]	10.2M	1024	224	74.8	74.0	75.3
Inception v3 [27]	21.8M	2048	299	78.8	78.0	78.6
Inception v4 [25]	41.1M	1536	299	80.0	80.2	79.9
Inception-ResNet v2 [25]	54.3M	1536	299	80.1	80.4	80.3
ResNet-50 v1[e] [11, 9, 8]	23.5M	2048	224	76.4	75.2	76.9
ResNet-101 v1 [11, 9, 8]	42.5M	2048	224	77.9	76.4	78.6
ResNet-152 v1 [11, 9, 8]	58.1M	2048	224	N/A	76.8	79.3
DenseNet-121 [14]	7.0M	1024	224	75.0	74.8	75.6
DenseNet-169 [14]	12.5M	1024	224	76.2	76.2	76.7
DenseNet-201 [14]	18.1M	1024	224	77.4	77.3	77.1
MobileNet v1 [13]	3.2M	1024	224	70.6	70.7	72.4
MobileNet v2 [22]	2.2M	1280	224	72.0	71.8	71.6
MobileNet v2 (1.4) [22]	4.3M	1792	224	74.7	75.0	74.7
NASNet-A Mobile [32]	4.2M	1056	224	74.0	74.0	73.6
NASNet-A Large [32]	84.7M	4032	331	82.7	82.7	80.8

[a] Excludes logits layer.

[b] Performance of checkpoint from TF-Slim repository (`https://github.com/tensorflow/models/tree/master/research/slim`), or, for DenseNets, from Keras applications (`https://keras.io/applications/`).

[c] We used Inception model code from the TF-Slim repository, which uses batch normalization layers for Inception v1. Additionally, the models in this repository contain minor modifications compared to the models described in the original papers. We cite the performance number for BN-GoogLeNet from Szegedy et al. [27].

[d] This model is called "Inception v2" in TF-Slim model repository, but matches the model described in Ioffe and Szegedy [15], rather than the model that Szegedy et al. [27] call "Inception v2."

[e] The ResNets we train incorporate two common modifications to the original ResNet v1 model: Stride-2 downsampling on the 3×3 convolution instead of the first 1×1 convolution in the block [9, 8] and initialization of the batch normalization γ to 0 in the last batch normalization layer of each block [8]. We report the numbers from Goyal et al. [8] as the original accuracy. No public TensorFlow checkpoints are available for these models, so, for public checkpoint results, we use the TF-Slim ResNet v1 checkpoints, which were converted from the original He et al. [11] model.

Table A.2. ImageNet classification networks

trained all networks with a batch size of 4096 using Nesterov momentum of 0.9 and weight decay of 8×10^{-5}, taking an exponential moving average of the weights with a decay factor of 0.9999. We performed linear warmup to a learning rate of 1.6 over the first 10 epochs, and then continuously decayed the learning rate by a factor of 0.975 per epoch. We used the preprocessing and data augmentation from [28]. To determine how long to train each network, we trained a separate model for up to 300 epochs with approximately 50,000 ImageNet training images held out as a validation set, and then trained a model on the full ImageNet training set for the number of steps that yielded the highest performance. Except in experiments explicitly studying the effects of these choices, for all networks, we used scale parameters for batch normalization layers, and did not use label smoothing, dropout, or an auxiliary head. For NASNet-A Large, we additionally disabled drop path regularization.

When training on ImageNet, we did not optimize hyperparameters for each network individually because we were able to achieve ImageNet top-1 performance comparable to publicly available checkpoints without doing so. (When fine-tuning and training from random initialization, we found that hyperparameters were more important and performed extensive tuning; see below.) For all networks except NASNet-A Large, our retrained models achieved accuracy no more than 0.5% lower than the original reported results and public checkpoint, and sometimes substantially higher (Table A.2). Given that we disabled the regularizers used in the original model, we expected a larger performance drop. Our experiments indicate that these regularizers further improve accuracy, but are evidently not necessary to achieve performance close to the published results.

For NASNet-A Large, there was a substantial gap between the performance of the published model and our retrained model (82.7% vs. 80.8%). As a sanity check, we enabled label smoothing, dropout, the auxiliary head, and drop path, and retrained NASNet-A Large with the same hyperparameters described above. This regularized model achieved 82.5% accuracy, suggesting that most of the loss in accuracy in our setup is due to disabling regularization. For other models, we could further improve ImageNet top-1 accuracy over published results by applying regularizers: A retrained Inception-ResNet v2 model with label smoothing, dropout, and the auxiliary head enabled achieved 81.4% top-1 accuracy, 1.1% better than the unregularized model and 1.3% better than the published result [25]. However, because these regularizers clearly hurt results in the logistic regression setting, and because our goal was to compare all models and settings fairly, we report results for models trained and fine-tuned without regularization unless otherwise specified.

A.4. Logistic regression

For each dataset, we extracted features from the penultimate layer of the network. 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 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, and we used the entire image, rather than cropping the central 87.5% as is common for testing on ImageNet.

A.5. Fine-tuning

For fine-tuning experiments in Figure 2, we initialized networks with ImageNet-pretrained weights and trained for 20,000 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 [17]. 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^{-6} and 10^{-3}, as well as no weight decay. 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 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 learning rate, so we did not perform early stopping. On the validation set, we evaluated on both uncropped images and images cropped to the central 87.5% and picked the approach that gave higher accuracy for evaluation on the test set. Cropped images typically yielded better performance, except on CIFAR-10 and CIFAR-100, where results differed by model.

When examining the effect of dataset size (Section 4.7), we fine-tuned for at least 1000 steps or 100 epochs (following guidance from our analysis of training time in Section 4.6) at a batch size of 64, with the learning rate range scaled down by a factor of 4. Otherwise, we used the same settings as above. 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 [18]. In Sections 4.7 and 4.6, we fine-tuned models from the publicly available Inception v4 checkpoint rather than using the model trained as above.

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 20,000 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}.

When examining the effect of dataset size (Section 4.7), 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. We chose these parameters because investigation of effects of training time (Section 4.6) 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. Logistic regression performance of public checkpoints

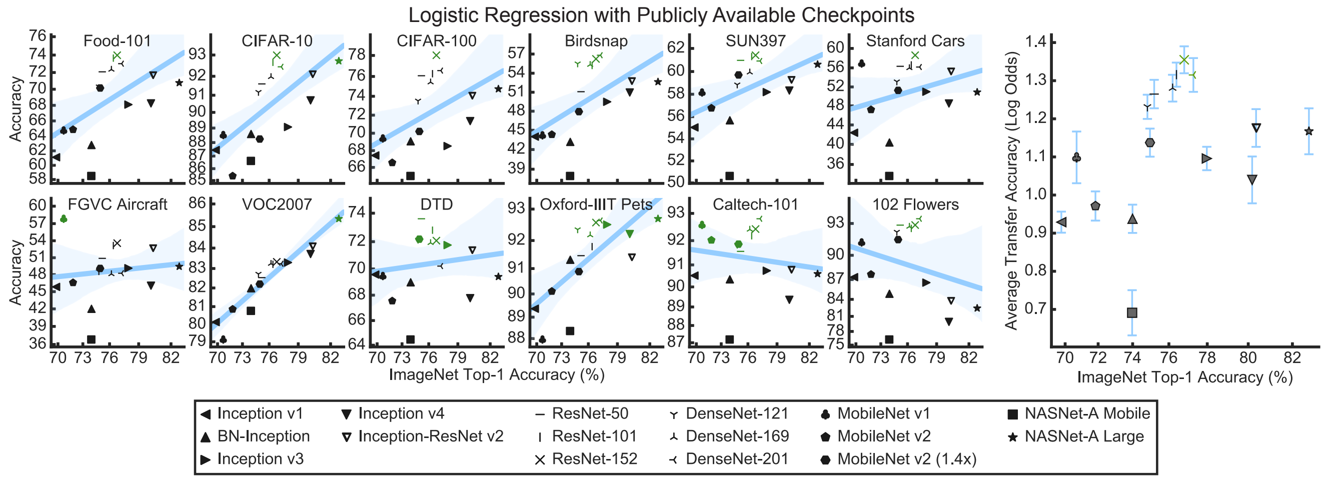

Figure B.1. Accuracy of logistic regression classifiers on fixed features from publicly available checkpoints, rather than retrained models. See also Figure 2.

We present results of logistic regression with features extracted from publicly available checkpoints in Figure B.1. With these checkpoints, ResNets and DenseNets were consistently among the top performing models. The correlation between ImageNet top-1 accuracy and accuracy across transfer tasks was weak and did not reach statistical significance ($r = 0.37$, $p = 0.16$). By contrast, the correlation with between ImageNet top-1 accuracy and accuracy across transfer tasks with retrained models ($r = 0.99$) was much higher ($p < 10^{-4}$, $z = 5.2$, test of equality of nonoverlapping correlations based on dependent groups [23, 5]).

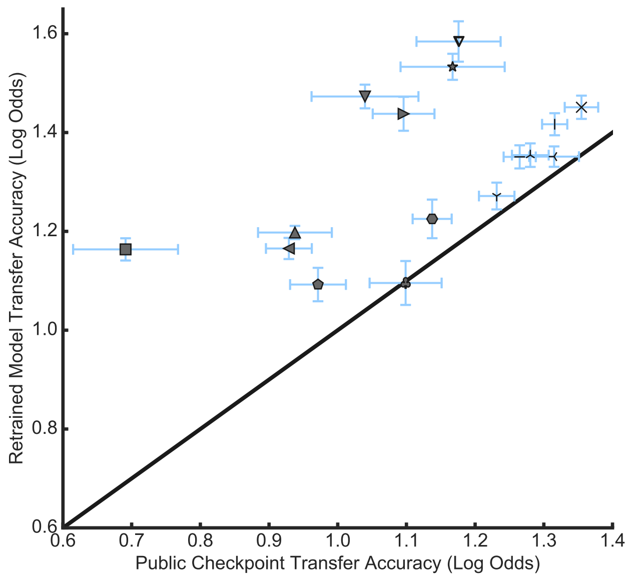

Figure B.2. Our retrained models consistently outperform public checkpoints for logistic regression. See Figure B.1 for legend.

Retrained models achieved higher transfer accuracy than publicly available checkpoints. For 11 of the 12 datasets investigated (all but Oxford Pets), features from the best retrained model achieved higher accuracy than features from the best publicly available checkpoint. Retrained models achieved higher accuracy for 84% of dataset/model pairs (162/192), and transfer accuracy averaged across datasets was higher for retrained models for all networks except MobileNet v1 (Figure B.2). The best retrained model, Inception-ResNet v2, achieved an average log odds of 1.58, whereas the best public checkpoint, ResNet v1 152, achieved an average log odds of 1.35 ($t(11) = 5.6$, $p = 0.0002$).

C. Extended analysis of effect of training/regularization settings

C.1. Performance of penultimate layer features

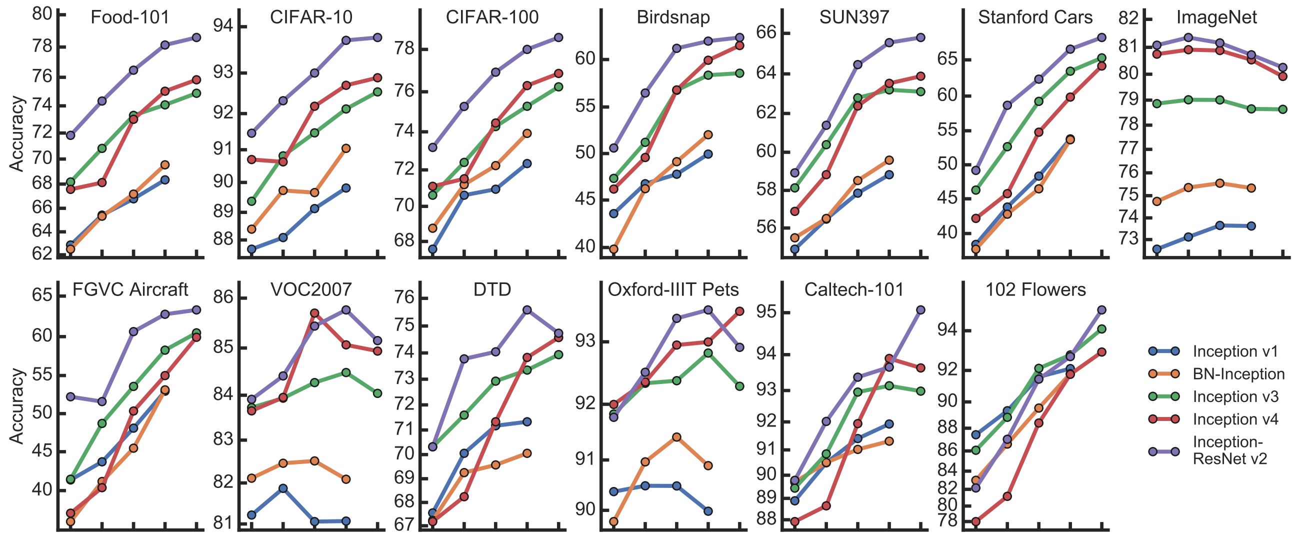

Figure C.1. When performing logistic regression on penultimate layer features, all datasets and models benefit from removal of regularization. Each subplot represents transfer performance on one of the datasets investigated. The top right plot shows ImageNet top-1 accuracy of the models. Points along the x-axis represent different training settings (presence/absence of batch normalization scale parameter, label smoothing, dropout, and presence/absence of auxiliary head), following the same convention as in Figure 3. The leftmost setting is the Inception default, and uses no batch normalization scale parameter, but includes label smoothing, dropout, and an auxiliary head. From left to right, we enable the batch normalization scale parameter; disable label smoothing; disable dropout; and disable the auxiliary head.

Figure C.1 shows performance of penultimate layer features in each of the training settings in Figure 3, broken down by dataset. Across nearly all datasets and models, the least-regularized models achieved the highest performance, even though these models were not the best in terms of ImageNet top-1 accuracy. We also experimented with removing weight decay, but found that this yielded substantially lower performance on both ImageNet and all transfer datasets except for FGVC Aircraft.

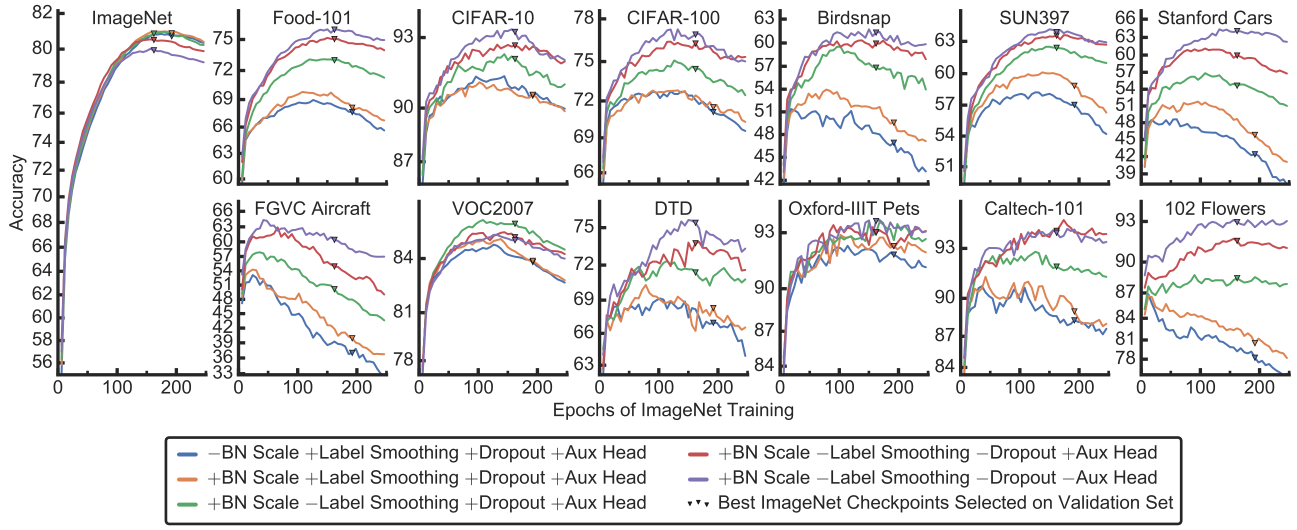

Figure C.2. Regularization affects transfer learning with fixed features earlier in training than ImageNet top-1 accuracy. Transfer learning performance of Inception v4 checkpoints evaluated every 12 epochs. Triangles represent the checkpoint that optimized performance on a validation set split from the training set, in a separate training run.

To investigate whether the effect of regularization upon the performance of fixed features was mediated by training time, rather than the regularization itself, we performed logistic regression on penultimate layer features from Inception v4 at different epochs over training (Figure C.2). For models with more regularizers, checkpoints from earlier in training typically performed better than the checkpoint that achieved the best accuracy on ImageNet. However, on most datasets, the best checkpoint without regularization outperformed all checkpoints with regularization. For most datasets, the best transfer accuracy was achieved at around the same number of training epochs as the best ImageNet top-1 accuracy, but on FGVC Aircraft, we found that a checkpoint early in training yielded much higher accuracy.

C.2. Fine-tuning performance

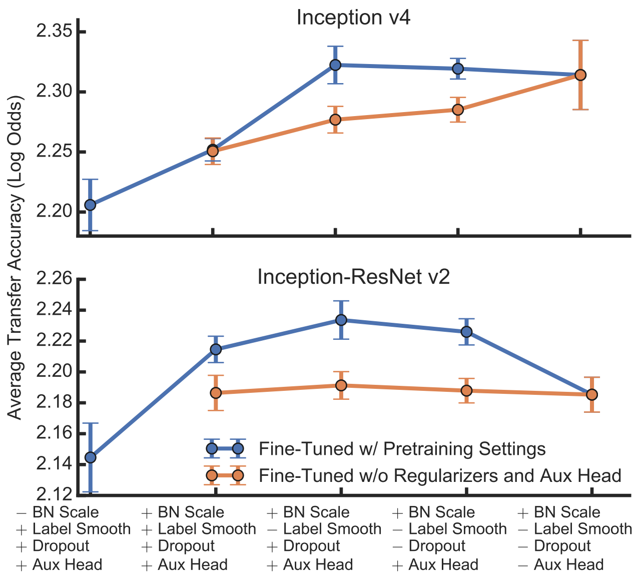

Figure C.3. Using regularizers at ImageNet pretraining time does not benefit fine-tuning performance unless the same regularizers are used to fine-tune. Blue points represent models pretrained and fine-tuned with the same training configuration, as in Figure 5. Orange points represent models pretrained with different configurations but fine-tuned without regularization or an auxiliary head (the rightmost configuration in the plot). Accuracy is averaged across 3 fine-tuning runs each for 2 rounds of hyperparameter tuning.

In this section, we present an expanded analysis of the effect of regularization upon fine-tuning analysis. Figure C.3 shows average fine-tuning both performance across datasets when fine-tuning with the same settings as used for pretraining (blue, same data as in Figure 5), and when pretraining with regularization but fine-tuning without any regularization (orange). For all regularization settings, benefits are only clearly observed when the regularization is used for both pretraining and fine-tuning. Figure C.4 shows results broken down by dataset for pretraining and fine-tuning with the same settings.

Overall, the effect of regularization upon fine-tuning performance was much smaller than the effect upon the performance of logistic regression on penultimate layer features. As in the logistic regression setting, enabling batch normalization scale parameters and disabling label smoothing improved performance. Effects of dropout and the auxiliary head were not entirely consistent across models and datasets (Figure C.4). Inception-ResNet v2 clearly performed better when the auxiliary head was present. For Inception v4, the auxiliary head improved performance on some datasets (Food-101, FGVC Aircraft, VOC2007, and Oxford Pets) but worsened performance on others (CIFAR-100, DTD, Oxford 102 Flowers). However, because improvements were only observed when the auxiliary head was used both for pretraining and fine-tuning, it is unclear whether the auxiliary head leads to better weights or representations. It may instead improve fine-tuning performance by acting as a regularizer at fine-tuning time.

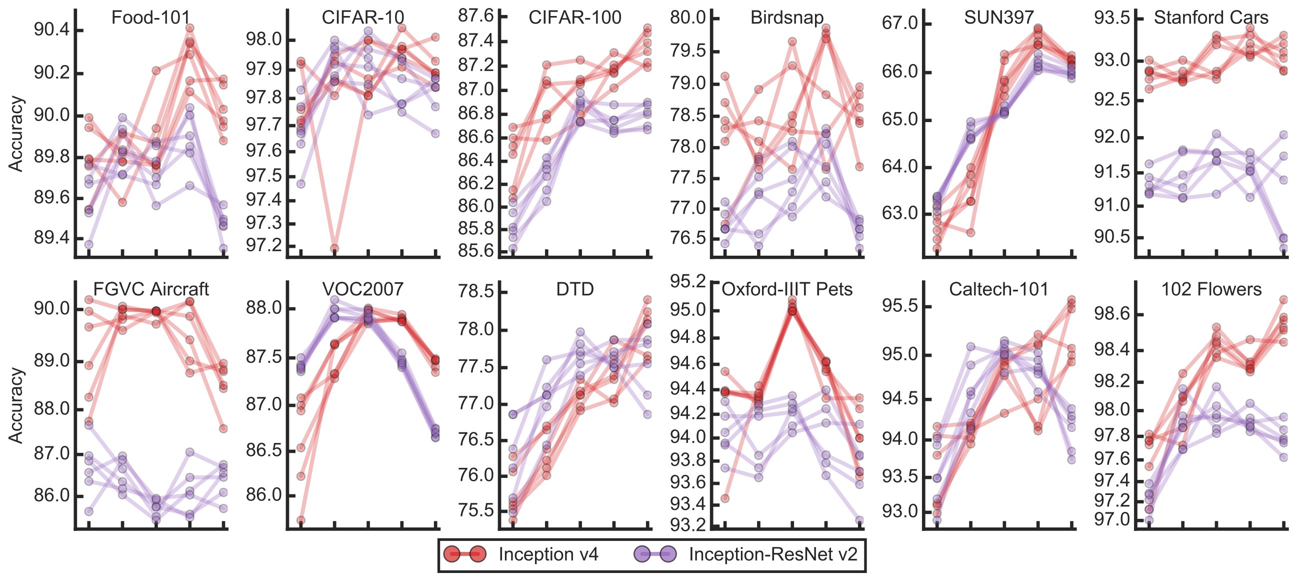

Figure C.4. For fine-tuning, different training settings are best on different datasets. Points along the x-axis represent different training settings (presence/absence of batch normalization scale parameter, label smoothing, dropout, and presence/absence of auxiliary head), following the same convention as in Figure C.3. The leftmost setting is the Inception default, and uses no batch normalization scale parameter, but includes label smoothing, dropout, and an auxiliary head. From left to right, we enable the batch normalization scale parameter; disable label smoothing; disable dropout; and disable the auxiliary head. Each line shows performance for a different fine-tuning run.

D. Relationship between dataset size and predictive power of ImageNet accuracy

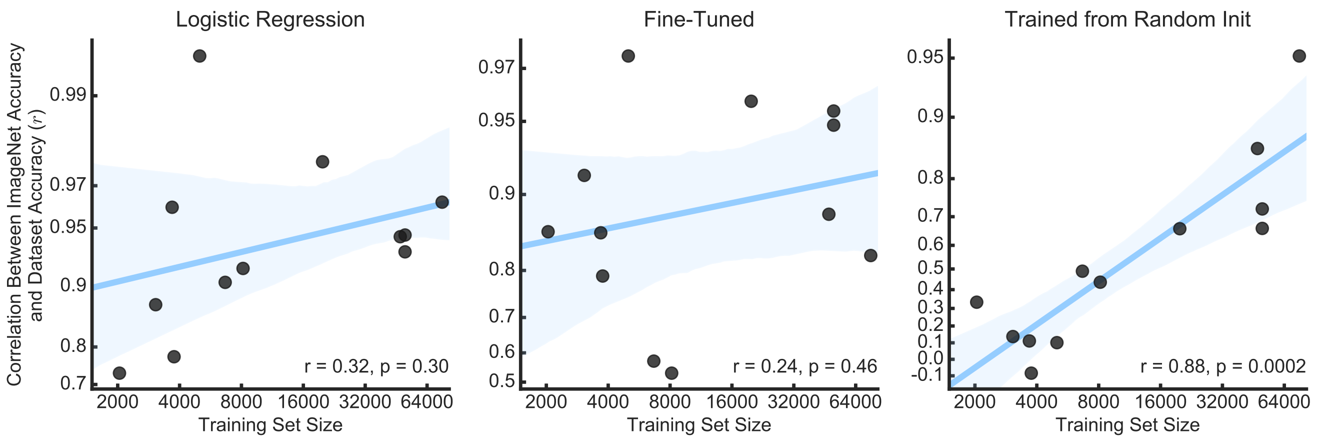

Figure D.1. When training from random initialization, the correlation between ImageNet top-1 accuracy and transfer accuracy is higher for larger datasets. Each point represents one of the 12 datasets investigated. The y-axis represents the Pearson correlation between ImageNet top-1 accuracy and accuracy for that dataset, based on the 16 ImageNet networks investigated, and is scaled according to the Fisher z-transformation. The x-axis is log-scaled. r and p values in bottom left reflect the correlation between the log-transformed training set size and Fisher z-transformed correlations.

As datasets get larger, ImageNet accuracy becomes a better predictor of the performance of models trained from scratch. Figure D.1 shows the relationship between the dataset size and the correlation between ImageNet accuracy and accuracy on other datasets, for each of the 12 datasets investigated. We found that there was a significant relationship when networks were trained from random initialization ($p = 0.0002$), but there were no significant relationships in the transfer learning settings.

One possible explanation for this behavior is that ImageNet performance measures both inductive bias and capacity. When training from scratch on smaller datasets, inductive bias may be more important than capacity.

E. Additional comparisons of logistic regression, fine-tuning, and training from random initialization

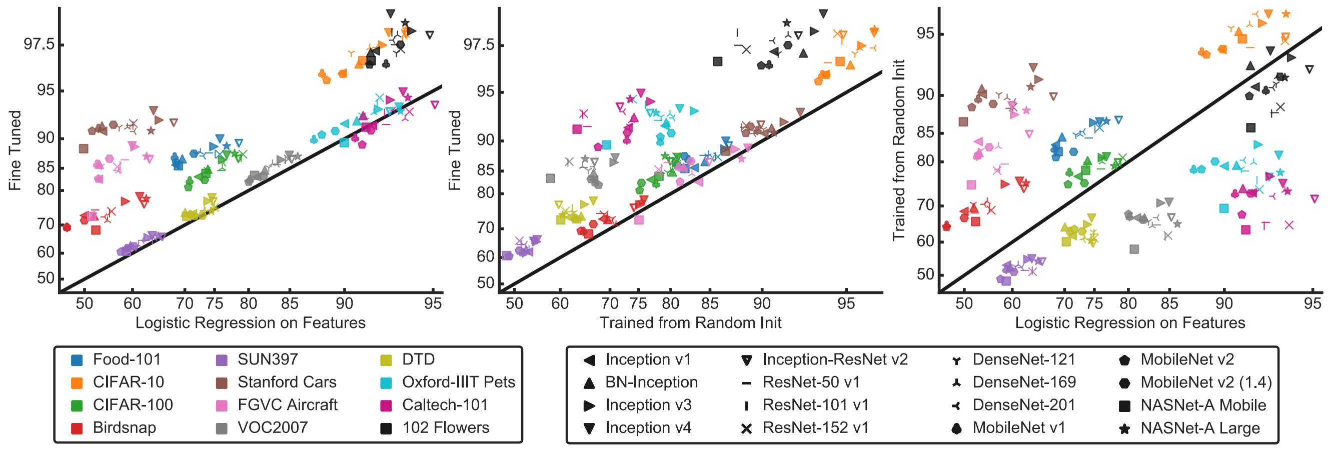

Figure E.1. Scatter plots comparing trained models in each pair of settings investigated. Axes are logit-scaled.

Figure E.1 presents additional scatter plots comparing performance in the three settings we investigated. Fine-tuning usually achieved higher accuracy than logistic regression on top of fixed ImageNet features or training from randomly initialized models, but for some datasets, the gap was small. The performance of logistic regression on fixed ImageNet features vs. networks trained from random initialization was heavily dependent on the dataset.

F. Comparison versus state-of-the-art

	Previously reported		Current work	
Dataset	Acc.	Method	Acc.	Best network
Food-101	90.4	Domain-specific transfer @ 448 [4]	90.0 (**90.8**[a])	Inception v4, fine-tuned
CIFAR-10	98.5	AutoAugment [3]	98.0[b]	NASNet-A Large, fine-tuned
CIFAR-100	89.3	AutoAugment [3]	87.5[b]	NASNet-A Large, fine-tuned
Birdsnap	80.2[c]	Mask-CNN @ 448 [29]	78.4 (**81.8**[a])	Inception v4, fine-tuned
SUN397	**70.2**	Places/ImageNet-pretrained multi-scale VGG ensemble [12]	66.4 (68.3[a])	Inception v4, fine-tuned
Stanford Cars	**94.8**	AutoAugment @ 448 [3]	93.3 (93.4[a])	Inception v4, fine-tuned
FGVC Aircraft	**92.9**[c]	Deep layer aggregation @ 448 [30]	89.0 (90.9[a])	Inception v4, fine-tuned
VOC 2007 Cls.	**89.7**	VGG multi-scale ensemble [24]	87.4 (88.2[a])	Inception v4, fine-tuned
DTD	75.5	FC+FV-CNN+D-SIFT [2]	**78.1**	Inception v4, fine-tuned
Oxford-IIIT Pets	93.8	Object-part attention [20]	**94.5**	ResNet-152 v1, fine-tuned
Caltech-101	93.4	Spatial pyramid pooling [10]	**95.1**	Inception-ResNet v2, log. regression
Oxford 102 Flowers	97.7	Domain-specific transfer [4]	**98.5**	Inception v4, fine-tuned

[a]For datasets where the best published result evaluated at 448 $\times$ 448 or at multiple scales, we provide results at 448 $\times$ 448 in parentheses.
[b]Accuracy excludes images duplicated between the ImageNet training set and CIFAR test sets; see Appendix H. A previous version of this paper achieved accuracies of 98.4% on CIFAR-10 and 88.2% on CIFAR-100 by fine-tuning the public NASNet checkpoint with the auxiliary head, dropout, and drop path. The difference in this version is due to the change in settings; the previous results remain valid.
[c]Krause et al. [16] achieve 85.4% on Birdsnap and 95.9% on Aircraft using bird and aircraft images collected from Google image search.

Table F.1. Performance of best models.

Table F.1 shows the best previously reported results of which we are aware on each of the datasets investigated. We achieve state-of-the-art performance on either 4 datasets at networks' native image sizes, or 6 if we retrain networks at 448 $\times$ 448, as some previous transfer learning works have done. For CIFAR-10, CIFAR-100, and Stanford Cars, the best result was trained from scratch; for all other datasets, the baselines use some form of ImageNet pretraining.

G. Comparison of alternative classifiers

In addition to the logistic regression without data augmentation setting described in the main text, we investigated transfer learning performance using support vector machines without data augmentation, and using logistic regression with data augmentation. Results are shown in Figure G.1.

G.1. SVM

Although a logistic regression classifier trained on the penultimate layer activations has a natural interpretation as retraining the last layer of the neural network, many previous studies have reported results with support vector machines [6, 21, 1, 24]. Thus, we examine performance in this setting as well (Figure G.1A-D). Following previous work [24, 1], we L_2-normalized the input to the model along the feature dimension. We used the SVM implementation from scikit-learn [7, 19], selecting the value of the hyperparameter C from 26 logarithmically spaced values between 0.001 and 100. SVM and logistic regression results were highly correlated ($r = 0.998$). For most (146/192) dataset/model pairs, logistic regression outperformed SVM, but differences were small (average log odds 1.32 vs. 1.28, $p < 10^{-19}$, t-test).

G.2. Logistic regression with data augmentation

Finally, we trained a logistic regression classifier with data augmentation, in the same setting we use for fine-tuning. We trained for 40,000 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 initial learning rate at 0.1 and used a cosine decay schedule. We optimized over L2 regularization parameters for the final layer, applied to the mean of the per-example losses, selected from a range of 10 logarthmically spaced values between 10^{-10} and 0.1. Results are shown in Figure G.1E-H. No findings changed. Transfer accuracy with data augmentation was highly correlated with ImageNet accuracy (Figure G.1F) and with results without data augmentation (Figure G.1G; $r = 0.99$ for both correlations). Fine-tuning remained clearly superior to logistic regression with data augmentation, achieving better results for 188/192 dataset/model pairs (Figure G.1H).

Logistic regression with data augmentation performed better for 100/192 dataset/model pairs. Data augmentation gave a slight improvement in average log odds (1.35 vs. 1.32), but the best performing model without data augmentation was better than the best performing model with data augmentation on half of the 12 datasets.

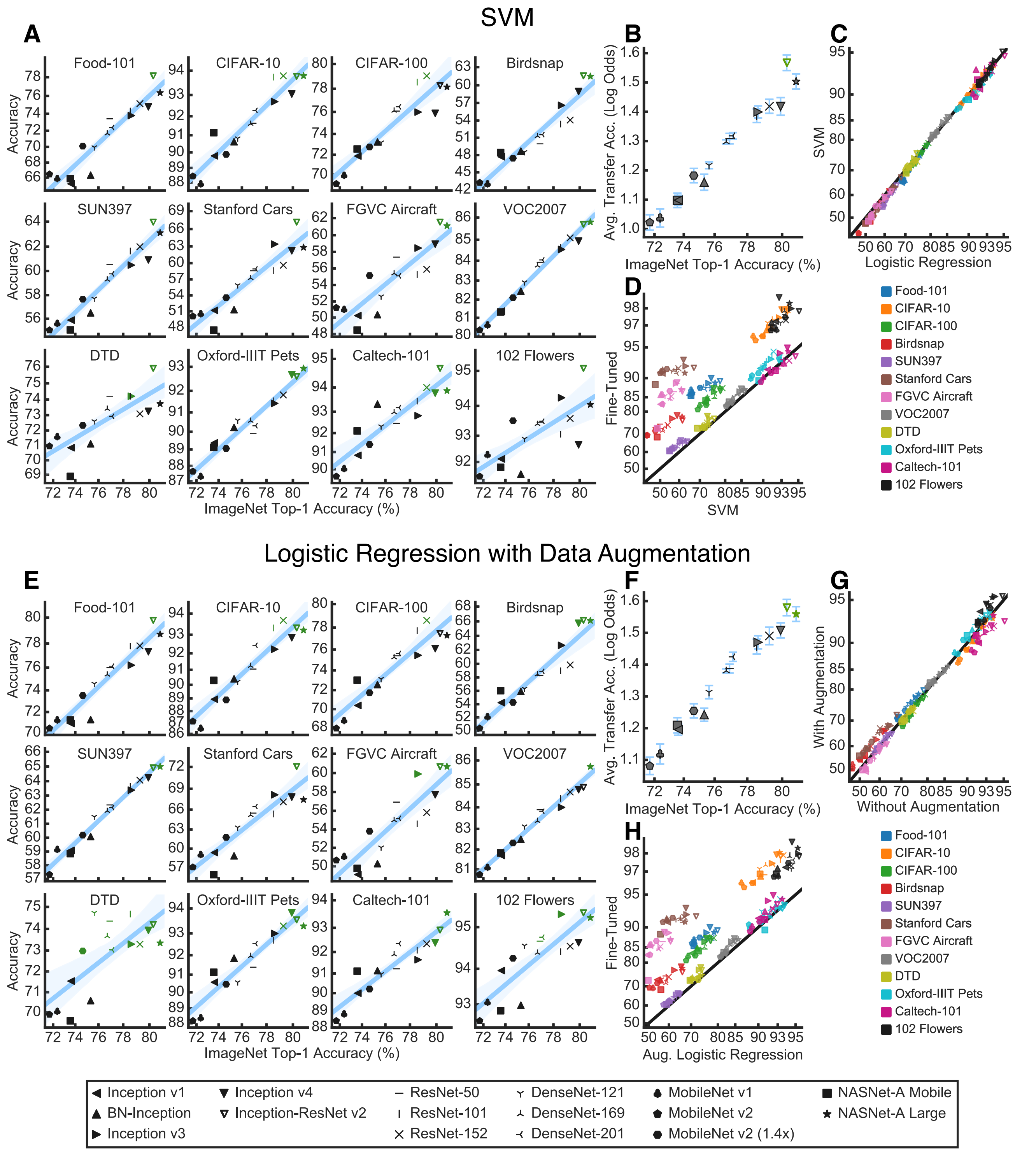

Figure G.1. Analysis of SVM and logistic regression with data augmentation, performed on fixed features. **A** and **E**: Scatter plots of ImageNet top-1 accuracy versus transfer accuracy on each of the 12 datasets examined. See also Figure 2. **B** and **F**: ImageNet top-1 accuracy versus average transfer accuracy for each network investigated. **C** and **G**: Performance of logistic regression without data augmentation versus SVM (**C**) or logistic regression with data augmentation (**G**). **D** and **H**: Performance of SVM (**D**) or logistic regression with data augmentation (**H**) versus fine-tuning. See also Figure E.1.

H. Duplicate images

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%

Table H.1. Prevalence of images duplicated between the ImageNet training set and datasets investigated for transfer.

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 and there are a substantial number of duplicates, so we exclude duplicates from the test set.

On CIFAR-10, we achieve an accuracy of 98.04% when fine-tuning NASNet Large (the best model) on the full test set. We also achieve an accuracy of 98.02% on the 9,863 example test set that is disjoint with the ImageNet training set. We achieve an accuracy of 99.27% on the 137 duplicates. On CIFAR-100, we achieve an accuracy of 87.7% on the full test set. We achieve an accuracy of 87.5% on the 9,771 example test set that is disjoint from the ImageNet training set, and an accuracy of 95.63% on the 229 duplicates.

I. Numerical performance results

We present the numerical results for logistic regression, fine-tuning, and training from random initialization in Table I.1. Bold-faced numbers represent best models, or models insignificantly different from the best, in each training setting.

Logistic regression

Network	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech101	Flowers
Inception v1	68.3	89.8	72.3	49.9	58.8	53.8	53.0	81.1	71.3	90.0	91.93	92.1
BN-Inception	69.5	91.0	73.9	52.0	59.6	53.7	53.1	82.1	70.1	90.9	91.32	91.8
Inception v3	74.9	92.5	76.2	58.6	63.1	65.3	60.5	84.0	**73.9**	92.3	92.98	94.1
Inception v4	75.8	92.9	76.9	**61.4**	63.9	64.2	59.9	84.9	**74.6**	**93.4**	93.65	93.0
Inception-ResNet v2	**78.6**	**93.8**	**78.5**	**62.3**	**65.8**	**67.9**	**63.3**	85.2	**74.7**	**92.9**	**95.06**	**94.9**
ResNet-50 v1	74.1	91.8	76.0	52.2	62.5	59.5	58.5	83.5	**74.9**	91.5	92.74	93.2
ResNet-101 v1	75.1	**93.6**	**78.9**	55.3	64.0	60.1	57.4	84.5	**74.9**	92.2	92.65	93.1
ResNet-152 v1	75.8	**93.8**	**79.2**	55.7	64.1	60.2	56.9	84.8	**75.0**	92.4	93.96	93.5
DenseNet-121	72.0	90.5	73.8	51.9	60.7	57.3	53.5	82.6	**74.8**	91.2	92.13	93.3
DenseNet-169	72.7	91.8	76.2	54.9	61.2	59.0	57.2	83.0	**73.4**	92.0	93.75	93.4
DenseNet-201	73.2	92.2	76.4	54.2	61.9	60.3	57.1	83.6	73.2	91.4	93.15	93.1
MobileNet v1	68.2	88.2	70.9	46.3	58.8	52.9	52.6	80.2	71.0	87.4	90.77	92.7
MobileNet v2	68.4	88.6	70.6	46.3	57.6	51.6	52.9	80.0	71.7	88.1	91.26	91.7
MobileNet v2 (1.4)	71.6	89.8	73.4	50.0	60.3	56.1	55.2	81.9	73.0	89.3	91.83	93.5
NASNet-A Mobile	68.9	91.3	73.6	52.4	58.8	49.8	51.5	80.8	70.3	90.0	91.52	91.8
NASNet-A Large	76.9	**93.8**	78.0	**62.8**	65.1	63.7	**62.8**	**85.8**	**74.5**	**93.5**	93.89	93.8

Fine-tuned

Network	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech101	Flowers
Inception v1	85.6	96.17	83.2	73.0	62.0	91.0	82.7	83.2	73.6	91.9	91.7	97.26
BN-Inception	86.8	96.67	84.8	72.9	62.8	91.7	85.8	84.6	73.9	92.3	92.8	97.2
Inception v3	88.8	97.5	86.6	**77.2**	65.7	92.3	**88.8**	86.6	**77.2**	93.5	**94.3**	97.98
Inception v4	**90.0**	**97.93**	**87.5**	**78.4**	**66.4**	**93.3**	**89.0**	**87.4**	**78.1**	93.7	**94.9**	**98.45**
Inception-ResNet v2	89.4	**97.87**	86.8	76.3	**65.9**	92.0	86.7	86.7	**77.1**	93.3	**93.9**	97.85
ResNet-50 v1	87.8	96.77	84.5	74.7	64.7	91.7	86.6	85.7	75.2	92.5	91.8	97.51
ResNet-101 v1	87.6	97.68	87.0	73.8	64.8	91.7	85.6	86.6	76.2	**94.0**	93.1	97.94
ResNet-152 v1	87.6	**97.91**	**87.6**	74.3	**66.0**	92.0	85.3	86.8	75.4	**94.5**	93.2	97.35
DenseNet-121	87.7	97.18	84.8	73.2	62.3	91.5	85.4	85.1	74.9	92.9	91.9	97.18
DenseNet-169	88.0	97.4	85.0	71.4	63.0	91.5	84.5	85.9	74.8	93.1	92.5	97.86
DenseNet-201	87.3	97.41	86.0	72.6	64.7	91.0	84.6	85.8	74.5	92.8	93.4	97.68
MobileNet v1	87.1	96.15	82.3	69.2	61.7	91.4	85.8	82.6	73.4	89.9	90.1	96.67
MobileNet v2	86.2	95.74	80.8	69.3	60.5	91.0	82.8	82.1	72.9	90.5	89.1	96.63
MobileNet v2 (1.4)	87.7	96.13	82.5	71.5	62.6	91.8	86.8	83.4	73.0	91.0	91.1	97.52
NASNet-A Mobile	85.5	96.83	83.9	68.3	60.7	88.5	72.8	83.5	72.8	89.4	91.5	96.83
NASNet-A Large	88.9	**98.04**	**87.7**	**77.9**	**66.2**	91.1	87.2	87.2	74.3	93.3	**94.5**	**98.22**

Trained from random initialization

Network	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech101	Flowers
Inception v1	83.1	94.03	77.0	68.6	53.1	90.1	83.7	66.9	61.9	79.1	73.3	90.9
BN-Inception	84.4	95.17	80.2	69.6	52.5	90.7	81.7	66.9	64.4	79.3	74.3	92.8
Inception v3	86.6	95.61	**80.8**	**75.3**	**54.9**	91.6	87.7	70.8	65.1	**83.2**	**77.0**	**93.5**
Inception v4	**86.7**	**96.05**	**81.0**	**75.9**	**55.0**	**92.7**	**88.8**	**70.9**	**66.8**	81.2	**75.4**	**93.9**
Inception-ResNet v2	**87.0**	94.85	79.9	74.2	54.2	89.9	84.9	67.0	59.6	76.9	71.8	92.5
ResNet-50 v1	84.3	94.17	78.6	68.2	51.5	88.5	79.6	64.9	62.3	78.1	65.6	87.9
ResNet-101 v1	85.6	94.81	79.9	69.5	51.5	88.2	78.6	61.6	62.6	76.2	64.6	87.8
ResNet-152 v1	85.9	94.61	**80.8**	68.9	51.1	88.6	78.2	61.9	61.1	74.0	64.9	88.7
DenseNet-121	84.8	95.35	79.5	70.4	52.6	90.1	82.1	65.9	62.9	78.6	73.5	91.2
DenseNet-169	84.8	95.53	80.0	71.1	53.2	89.7	82.8	64.6	61.3	79.9	73.8	91.9
DenseNet-201	85.3	**96.05**	**80.8**	70.4	52.4	89.3	78.4	66.9	60.7	80.3	72.4	90.8
MobileNet v1	82.4	93.88	77.9	64.8	51.8	89.6	81.1	67.2	63.1	78.5	73.0	90.5
MobileNet v2	80.9	93.68	75.2	64.3	48.8	88.6	81.3	67.8	63.7	78.5	67.7	89.9
MobileNet v2 (1.4)	81.9	94.07	75.5	66.8	51.1	89.0	82.7	66.3	63.1	80.1	73.1	91.9
NASNet-A Mobile	81.9	94.73	78.3	65.9	48.3	86.7	75.1	57.9	60.1	69.4	63.5	85.8
NASNet-A Large	**86.8**	**96.06**	79.2	**75.5**	54.3	90.9	**88.2**	65.2	60.5	77.8	73.6	91.8

Table I.1. Model performance

References

[1] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[2] 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)*, pages 3828–3836. IEEE, 2015.

[3] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[4] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] Birk Diedenhofen and Jochen Musch. cocor: A comprehensive solution for the statistical comparison of correlations. *PLOS ONE*, 10(4):1–12, 04 2015.

[6] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, pages 647–655, 2014.

[7] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(Aug):1871–1874, 2008.

[8] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[9] Sam Gross and Michael Wilber. Training and investigating residual nets. In *The Torch Blog*. http://torch.ch/blog/2016/02/04/resnets.html, 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, pages 346–361. Springer, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[12] Luis Herranz, Shuqiang Jiang, and Xiangyang Li. Scene recognition with CNNs: objects, scales and dataset bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 571–579, 2016.

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

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

[16] Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 301–320, Cham, 2016. Springer International Publishing.

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[18] Augustus Odena, Avital Oliver, Colin Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of semi-supervised learning algorithms. In *ICLR Workshops*, 2018.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[20] Yuxin Peng, Xiangteng He, and Junjie Zhao. Object-part attention model for fine-grained image classification. *IEEE Transactions on Image Processing*, 27(3):1487–1500, 2018.

[21] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 512–519. IEEE, 2014.

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[23] N. Clayton Silver, James B. Hittner, and Kim May. Testing dependent correlations with nonoverlapping variables: A monte carlo simulation. *Journal of Experimental Education*, 73(1):53–69, 2004.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[25] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 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.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[27] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[28] The TensorFlow Authors. inception_preprocessing.py.

[29] Xiu-Shen Wei, Chen-Wei Xie, Jianxin Wu, and Chunhua Shen. Mask-CNN: Localizing parts and selecting descriptors for fine-grained bird species categorization. *Pattern Recognition*, 76:704 – 714, 2018.

[30] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2403–2412, 2018.

[31] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

[32] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, 2018.