Revisiting Unreasonable Effectiveness of Data in Deep Learning Era

Chen Sun[1], Abhinav Shrivastava[1,2], Saurabh Singh[1], and Abhinav Gupta[1,2]

[1]Google Research
[2]Carnegie Mellon University

Abstract

The success of deep learning in vision can be attributed to: (a) models with high capacity; (b) increased computational power; and (c) availability of large-scale labeled data. Since 2012, there have been significant advances in representation capabilities of the models and computational capabilities of GPUs. But the size of the biggest dataset has surprisingly remained constant. What will happen if we increase the dataset size by $10\times$ *or* $100\times$*? This paper takes a step towards clearing the clouds of mystery surrounding the relationship between 'enormous data' and visual deep learning. By exploiting the JFT-300M dataset which has more than 375M noisy labels for 300M images, we investigate how the performance of current vision tasks would change if this data was used for representation learning. Our paper delivers some surprising (and some expected) findings. First, we find that the performance on vision tasks increases logarithmically based on volume of training data size. Second, we show that representation learning (or pre-training) still holds a lot of promise. One can improve performance on many vision tasks by just training a better base model. Finally, as expected, we present new state-of-the-art results for different vision tasks including image classification, object detection, semantic segmentation and human pose estimation. Our sincere hope is that this inspires vision community to not undervalue the data and develop collective efforts in building larger datasets.*

1. Introduction

There is unanimous agreement that the current ConvNet revolution is a product of big labeled datasets (specifically, 1M labeled images from ImageNet [35]) and large computational power (thanks to GPUs). Every year we get further increase in computational power (a newer and faster GPU) but our datasets have not been so fortunate. ImageNet, a dataset of 1M labeled images based on 1000 categories, was used to train AlexNet [25] more than five years ago. Curiously, while both GPUs and model capacity have continued to grow, datasets to train these models have remained stagnant. Even a 101-layer ResNet with significantly more capacity and depth is still trained with 1M images from ImageNet circa 2011. Why is that? Have we once again belittled the importance of data in front of deeper models and computational power? What will happen if we scale up the amount of training data $10\times$ or $100\times$, will the performance double?

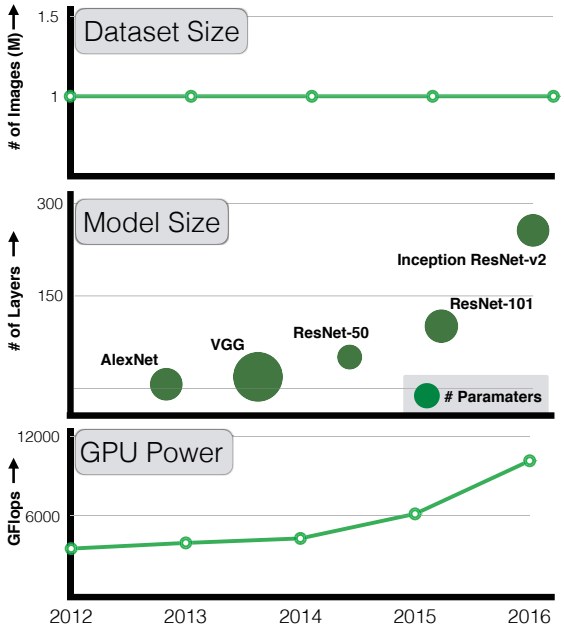

Figure 1. The Curious Case of Vision Datasets: While GPU computation power and model sizes have continued to increase over the last five years, size of the largest training dataset has surprisingly remained constant. Why is that? What would have happened if we have used our resources to increase dataset size as well? This paper provides a sneak-peek into what could be if the dataset sizes are increased dramatically.

This paper takes the first steps towards clearing the clouds of mystery surrounding the relationship between 'enormous data' and deep learning. We exploit the al-

2380-7504/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCV.2017.97

ready existing JFT-image dataset, first introduced by Hinton *et al*. [17] and expanded by [7]. The JFT dataset has more than 300M images that are labeled with 18291 categories. The annotations have been automatically obtained and, therefore, are noisy and not exhaustive. These annotations have been cleaned using complex algorithms to increase the precision of labels; however there is still approximately 20% error in precision. We will use this data to investigate the nature of relationship between amount of data and performance on vision tasks. Specifically, we will look into the power of data for visual representation learning (pre-training). We evaluate our learned representation on a variety of vision tasks: image classification, object detection, semantic segmentation and human pose estimation. Our experiments yield some surprising (and some expected) findings:

- **Better Representation Learning Helps!** Our first observation is that large-scale data helps in representation learning as evidenced by improvement in performance on each and every vision task we study.

  This suggests that collection of a larger-scale dataset to study visual pretraining may greatly benefit the field. Our findings also suggest a bright future for unsupervised or self-supervised [10, 43] representation learning approaches. It seems the scale of data can overpower noise in the label space.

- **Performance increases logarithmically based on volume of training data.** We find there is a logarithmic relationship between performance on vision tasks and the amount of training data used for representation learning. Note that previous papers on large-scale learning [23] have shown diminishing returns even on log-scale.

- **Capacity is Crucial:** We also observe that to fully exploit 300M images, one needs higher capacity models. For example, in case of ResNet-50 the gain on COCO object detection is much smaller (1.87%) compared to (3%) when using ResNet-152.

- **Training with Long-tail:** Our data has quite a long tail and yet the representation learning seems to work. This long-tail does not seem to adversely affect the stochastic training of ConvNets (training still converges).

- **New state of the art results:** Finally, our paper presents new state-of-the-art results on several benchmarks using the models learned from JFT-300M. For example, a single model (without any bells and whistles) can now achieve 37.4 AP as compared to 34.3 AP on the COCO detection benchmark.

2. Related Work

Ever since the seminal work by Krizhevsky *et al*. [25] showcased the power of Convolutional Neural Networks (ConvNets) on large-scale image recognition task, a lot of work has been done to make them more accurate. A common approach is to increase the complexity of these networks by increasing the width or depth of these networks. For example, Simonyan and Zisserman [37] proposed the VGG-19 model which uses smaller convolutional filters and has depth of 19 layers. Since then the representational power and depth of these models have continued to grow every year. GoogleNet [39] was a 22-layer network. In this paper, we perform all our experiments with the ResNet models proposed by He *et al*. [16]. The core idea is to add residual connections between layers which helps in optimization of very-deep models. This results in new state-of-the-art performances on a number of recognition tasks.

Convolutional neural networks learn a hierarchy of visual representations. These visual representations have been shown to be effective on a wide range of computer vision tasks [1, 4, 14, 22, 29, 33, 36]. Learning these visual representations require large-scale training data. However, the biggest detection and segmentation datasets are still on the order of hundreds of thousands of images. Therefore, most of these approaches employ pre-training. The original model is learning using million labeled images in ImageNet and then further trained on target tasks (fine-tuning) to yield better performance [4, 14, 33]. Huang *et al*. [18] thoroughly evaluated the influence of multiple ConvNet architectures on object detection performance, and found that it is closely correlated with the models' capacity and classification performances on ImageNet.

While there has been significant work on increasing the representational capacity of ConvNets, the amount of training data for pre-training has remain kind of fixed over years. The prime reason behind this is the lack of human verified image datasets larger than ImageNet. In order to overcome the bottleneck, there have been recent efforts on visual representation learning using web-supervision [2, 5, 6, 9, 21, 23, 24, 27] or unsupervised [10, 11, 31, 32, 34, 42, 43] paradigms. However, most of these efforts are still are still exploratory in nature and far lower in performance compared to fully-supervised learning.

In this paper, we aim to shift the discussion from models to data. Our paper is inspired from several papers which have time and again paid closer look to impact and properties of data rather than models. In 2009, Pereira *et al*. [30] presented a survey paper to look into impact of data in fields such as natural language processing and computer vision. They argued unlike physics, areas in AI are more likely to see an impact using more data-driven approaches. Another related work is the empirical study by Torralba and Efros [41] that highlighted the dataset biases in current com-

puter vision approaches and how it impacts future research.

Specifically, we focus on understanding the relationship between data and visual deep learning. There have been some efforts to understand this relationship. For example, Oquab *et al*. [28] showed that expanding the training data to cover 1512 labels from ImageNet-14M further improves the object detection performance. Similarly, Huh *et al*. [19] showed that using a smaller subset of images for training from ImageNet hurts performance. Both these studies also show that selection of categories for training is important and random addition of categories tends to hurt the performance. But what happens when the number of categories are increased 10x? Do we still need manual selection of categories? Similarly, neither of these efforts demonstrated data effects at significantly larger scale.

Some recent work [23, 44] have looked at training ConvNets with significantly larger data. While [44] looked at geo-localization, [23] utilized the YFCC-100M dataset [40] for representation learning. However, unlike ours, [23] showed plateauing of detection performance when trained on 100M images. Why is that? We believe there could be two possible reasons: a) YFCC-100M images come only from Flickr. JFT includes images all over the web, and has better visual diversity. The usage of user feedback signals in JFT further reduces label noise. YFCC-100M has a much bigger vocabulary size and noisier annotations. b) But more importantly, they did not see real effect of data due to use of smaller AlexNet of VGG models. In our experiments, we see more gain with larger model sizes.

3. The JFT-300M Dataset

We now introduce the JFT-300M dataset used throughout this paper. JFT-300M is a follow up version of the dataset introduced by [7, 17]. The JFT-300M dataset is closely related and derived from the data which powers the Image Search. In this version, the dataset has 300M images and 375M labels, on average each image has 1.26 labels. These images are labeled with 18291 categories: *e.g*., **1165** type of animals and **5720** types of vehicles are labeled in the dataset. These categories form a rich hierarchy with the maximum depth of hierarchy being **12** and maximum number of child for parent node being **2876**.

The images are labeled using an algorithm that uses complex mixture of raw web signals, connections between webpages and user feedback. The algorithm starts from over one billion image label pairs, and ends up with 375M labels for 300M images with the aim to select labeled images with high precision. However, there is still some noise in the labels: approximately 20% of the labels in this dataset are noisy. Since there is no exhaustive annotation, we have no way to estimate the recall of the labels. Figure 2 shows the kind of noise that exists in the dataset. Because the labels are generated automatically, there is a problem of 'tortoise' being confused with 'tortoise-shell glasses'.

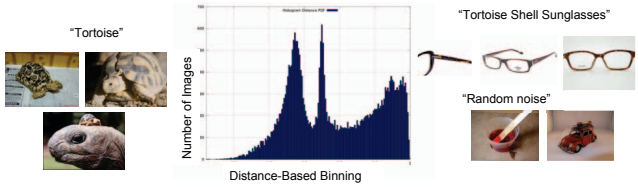

Figure 2. JFT-300M dataset can be noisy in terms of label confusion and incorrect labels. This is because labels are generated via a complex mixture of web signals, and not annotated or cleaned by humans. x-axis corresponds to the quantized distances to K-Means centroids, which are computed based on visual features.

Finally, it is important to discuss the data distribution of JFT-300M. The distribution is heavily long-tailed: *e.g*., there are more than **2M** 'flowers', **3250** 'subarau360' but only **131** images of 'train conductors'. In fact, the tail is so heavy that we have more than 3K categories with less than 100 images each and approximately 2K categories with less than 20 images per category.

4. Training and Evaluation Framework

We now describe our training and evaluation framework for the paper.

4.1. Training on JFT-300M Data

Although there are several novel ConvNet architectures recently proposed, we decide to use a standard Residual Network architecture [16] with 101 layers (ResNet-101) for its state-of-the-art performance and the ease of comparison with previous work. To train a ResNet-101 model on JFT-300M, We add a fully-connected layer with 18291 outputs at the end of the network for classification. As the image labels are not mutually exclusive, we compute per-label logistic loss, and treat all non-present labels as negatives. To alleviate the issue of missing labels, we use a hand-designed label hierarchy and fill in the missing labels accordingly. For example, an image with label 'apple' is also considered as a correct example for 'fruit'.

During training, all input images are resized to 340×340 pixels, and then randomly cropped to 299×299. The image pixels are normalized to the range of $[-1, 1]$ independently per channel, and we use random reflection for data augmentation. We set weight decay to 10^{-4} and use batch normalization [20] after all the convolutional layers. RMSProp optimizer is used with momentum of 0.9, and the batch size is set to 32. The learning rate is 10^{-3} initially and we decay it by 0.9 every 3M steps. We use asynchronous gradient descent training on 50 NVIDIA K80 GPUs. The model is implemented in TensorFlow.

To allow asynchrounous training of models on 50 GPUs, we adopt the Downpour SGD training scheme [8], where we use 17 parameter servers to store and update the model

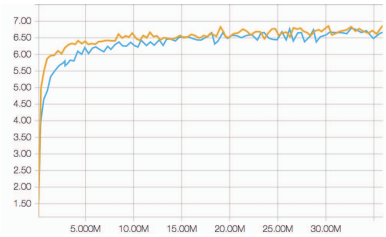

Figure 3. Comparison of training progress with random initialization (blue) and ImageNet initialization (yellow) on JFT-300M data. x-axis is the number of training steps, and y-axis shows the mAP@100 metric computed on FastEval14k.

weights. The final classification fully-connected layer with 2048 input units and over 18K output units has over 36M parameters. To handle this in our parameter servers, we split it vertically into 50 equal sized sub-fc layers, and distribute them around different parameter servers.

ImageNet baseline: As observed by [7], hyperparameters that are selected to train with JFT-300M data yield sub-optimal performance when training on ImageNet (IVSVRC 2012 image classification dataset with 1.2M images). Therefore, for ImageNet, we use a momentum optimizer with the momentum of 0.9, and set the initial learning rate to 5×10^{-2} and batch size to 32. Learning rate is reduced by a factor of 10 every 30 epochs (1.2M steps), and we train the model for a total of 5M steps. Similar to JFT-300M training, we use asynchronous gradient descent training on 50 NVIDIA K80 GPUs and 17 parameter servers.

Our baseline ResNet-101 performs 1% better than the open-sourced ResNet-101 checkpoint from the authors of [16], using the same evaluation protocol.

4.2. Monitoring Training Progress

For monitoring the training progress on JFT-300M, we use the validation set from Chollet [7]: 'FastEval14k'. FastEval14k consists of 14000 images with labels from 6000 classes (subset of 18291 classes from JFT-300M). Unlike labels in JFT-300M, the images in FastEval14k are densely annotated and there are around 37 labels per image on average. We use the same mAP@100 metric as in [7], which is computed as the mean average precision (mAP) for top-100 predictions. Note that the class AP is weighted by how common the class is among social media images.

We tried two strategies to initialize the model weights for training: random initialization and initializing from an ImageNet checkpoint. In both settings, we used the same training schedule (*e.g.*, learning rates). We found that on FastEval14k benchmark, model trained from ImageNet initialization performs better at the first 15M iterations, but then becomes on par with random initialization. Figure 3 shows the training progress for these two settings. On FastEval14k benchmark, model trained from ImageNet initialization performs better at the first 15M iterations, but then becomes on par with random initialization.

Please note that the full training schedule takes 90M iterations or around 10 epochs. However, due to the time constraints, we train the models for 36M iterations or 4 epochs, which takes approximately 2 months. We will study the impact of training iterations in Section 5.

4.3. Evaluating the Visual Representations

We use two approaches to evaluate the quality of visual representations learned from 300M training data. The first approach is to freeze the model weights and use these models as pure feature extractors. The second approach is to use the model weights as initialization and fine-tune the weights for other tasks. For evaluating visual representations, we select three representative computer vision tasks: object detection, semantic segmentation and human pose estimation.

We will perform a more rigorous ablative analysis to observe the effect of dataset size, vocabulary size, *etc*. on the object detection task. For the other tasks, we will just show how JFT-300M provides significant improvement compared to baseline ImageNet ResNet.

De-duplication One concern with using large-scale sets such as JFT-300M is the possible overlap between training and test sets. Such duplication exist in current frameworks as well: *e.g*. 890 out of 50K validation images in ImageNet have near-duplicate images training set. However, to ensure such duplication does not affect our results, we performed all experiments by removing near-duplicate images from test sets. We found the difference in performance to be insignificant for all the experiments. We therefore report de-duplicated test-set results in Appendix A.

Object Detection. We use the Faster RCNN framework [33] for its state-of-the-art performance. Faster RCNN is a two-stage model. The first stage is called region proposal network (RPN), which aims at generating class-agnostic object proposals. The second stage is a box classifier, it takes the boxes predicted by RPN and crops feature maps to generate classification predictions and refined bounding box predictions. These two stages share a common feature map generated by a ConvNet, and box classifier has additional convolutional layers before its final classification and regression layers. To use the ResNet-101 model pre-trained on JFT-300M data, we split the model into two parts: the first part starts from *conv1* block and ends at *conv4* block, it is used for feature extraction and is shared by both RPN and box classifier; the second part consists of the *conv5* block, it is used by box classifier.

Semantic Segmentation. We use the DeepLab framework [4] with ResNet-101 base architecture for the task of semantic segmentation. In particular, we use a variant which adds four branches after the *conv5* block of ResNet-101 architecture. Each branch is an atrous convolutional

Initialization	Top-1 Acc.	Top-5 Acc.
MSRA checkpoint [16]	76.4	92.9
Random initialization	77.5	93.9
Fine-tune from JFT-300M	**79.2**	**94.7**

Table 1. Top-1 and top-5 classification accuracy on the ImageNet 'val' set (single model and single crop inference are used).

layer that predicts a sub-sampled pixel-wise class probabilities. Predictions from all branches are fused together to produce the final segmentation output. Please refer to the DeepLab-ASPP-L model (**A**trous **S**patial **P**yramid **P**ooling, with **L**arge atrous rates) from [4] for details.

Pose Estimation. We follow the framework proposed by Papandreou *et al*. [29]. It uses person bounding boxes detected by Faster RCNN, then applies a ResNet [16] fully convolutionally to produce heatmaps and offsets for all keypoints. A novel scoring and non-maximum suppression (NMS) scheme is used to suppress duplicate detections and improve performance. We simply replace the base models used in their framework by our trained ResNet-101 models.

5. Experiments

We present results of fine-tuning JFT-300M ResNet-101 checkpoints on four tasks: image classification, object detection, semantic segmentation and human pose estimation.

5.1. Image Classification

We fine-tune the JFT-300M pre-trained ResNet101 using ImageNet classification data and compare it with a ResNet101 model trained from scratch. For this experiment, we use the standard ILSVRC 2012 'train' and 'val' sets for training and evaluation. There are 1.2M training images and 50K validation images, over 1000 classes.

We use the same ImageNet training setup as described in Section 4.1 for the ImageNet baseline, but lowered the initial learning rate to 10^{-3} (standard for fine-tuning). We initialize the model weights from the JFT-300M checkpoint trained for 36M iterations and fine-tune on ImageNet for 4M iterations.

Table 1 compares the fine-tuning results with models trained from the scratch. For reference, we show the random initialization performance for the open-sourced checkpoint from the authors of [16]. We report top-1 and top-5 accuracies with a single crop being evaluated. We can see that fine-tuning on JFT-300M gives considerable performance boost for both top-1 and top-5 accuracies.

5.2. Object Detection

We next evaluate the JFT-300M checkpoints on object detection tasks. We evaluate on the two most popular datasets: COCO [26] and PASCAL VOC [13]. Instead of just showing state-of-the-art performance, we will also perform a rigorous ablative analysis to gain insights into the relationship between data and representation learning.

Specifically, we use object detection experiments to answer the following questions:

- How does the performance of trained representations vary with iterations and epochs?
- Does the performance of learned visual representations saturate after certain amount of data? Do we see any plateauing effect with more and more data?
- How important is representational capacity?
- Is the number of classes a key factor in learning visual representation?
- How could clean data (*e.g*., ImageNet) help improve the visual representations?

Experimental Setup

For COCO [26], we use a held-out 8000 images from the standard 'val' set as our validation set, we refer to it as 'minival∗', the same set of images was used by [18]. We use a combination of the standard training set and the remaining validation images for training. Unless otherwise specified, all COCO results are reported on the minival∗ set. In particular, we are interested in mean average precision at 50% IOU threshold (mAP@.5), and the average of mAP at IOU thresholds 50% to 95% (mAP@[.5, .95]). For our best ResNet101 models, we also evaluate on the COCO 'test-dev' split (evaluated by the official result server). For PASCAL VOC, we use the 16551 'trainval' images from PASCAL VOC 2007 and 2012 for training, and report performance on the PASCAL VOC 2007 Test, which has 4952 images using mAP@.5 metric.

We use the TensorFlow Faster RCNN implementation [18] and adopt their default training hyperparameters except for learning rate schedules. We use asynchronous training with 9 GPU workers and 11 parameter servers, momentum optimizer is used with the momentum of 0.9. Each worker takes a single input image per step, the batch size for RPN and box classifier training are 64 and 256 respectively. Input images are resized to have 600 minimum pixels and 1024 maximum pixels while maintaining the aspect ratio. The only data augmentation used is random flipping.

For COCO, we set the initial learning rate to be 4×10^{-4}, and decay the learning rate by a factor of 10 after 2.5M steps, the total number of steps is 3M. For PASCAL VOC, we set the initial learning rate to be 3×10^{-4}, and decay the learning rate by 0.1 after 500K steps, and the model is trained for 700K steps. The training schedules were selected on held-out validation images using the open-source ResNet-101 model (pre-trained on ImageNet). We found the same training schedules work well on other checkpoints, and keep them fixed throughout for fairer comparison. Dur-

Method	mAP@0.5	mAP@[0.5,0.95]
He *et al*. [16]	53.3	32.2
ImageNet	53.6	34.3
300M	56.9	36.7
ImageNet+300M	**58.0**	**37.4**
Inception ResNet [38]	56.3	35.5

Table 2. Object detection performance comparisons with baseline methods on the COCO test-dev split. The first four Faster RCNN detectors are all based on ResNet-101 architecture, the last one is based on the InceptionResNet-v2 architecture. During inference, a single image scale and crop, and a single detection model are used for all experiments. Vanilla Faster RCNN implementations are used for all systems except for He *et al*. [16], which also includes box refinement and context.

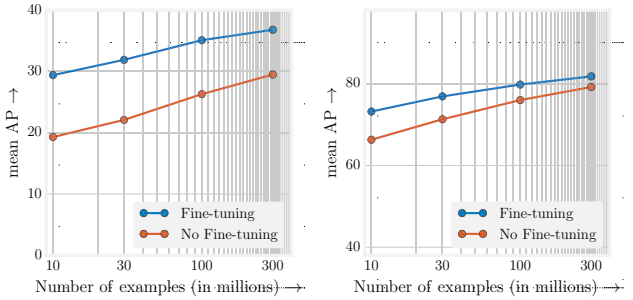

Figure 4. Object detection performance when initial checkpoints are pre-trained on different subsets of JFT-300M from scratch. x-axis is the data size in log-scale, y-axis is the detection performance in mAP@[.5,.95] on COCO minival∗ (left), and in mAP@.5 on PASCAL VOC 2007 test (right).

ing inference, we use 300 RPN proposals. Our vanilla FasterRCNN implementation does not use the multi-scale inference, context or box-refinement as described in [33].

Comparison with ImageNet Models

We first present the performance comparison with ImageNet checkpoints. Table 2 shows the detection performance on COCO 'test-dev' split. To show that our Faster RCNN baseline is competitive, we also report results from the Faster RCNN paper [16], which uses both box refinement and context information. We can see that our ImageNet baseline performs competitively.

We evaluate JFT-300M trained from scratch ('300M') and from ImageNet initialization ('ImageNet+300M'). Both models outperforms the ImageNet baseline by large margins, with 3.3% and 4.4% boost in mAP@.5, 2.4% and 3.1% in mAP@[.5,.95] respectively. As a reference, we also show the performance of ImageNet trained InceptionResNetv2 in Table 2. We would like to point out that the gain is even more significant than recently achieved by doubling the number of layers on Inception ResNet [18]. This clearly indicates that while there are indications of a plateauing effect on model representation capacity; in terms of data there is still a lot that can be easily gained.

Table 3 shows the performance on the PASCAL VOC 2007 'test' set. Again, both JFT-300M checkpoints outperforms the ImageNet baseline significantly, by 5.1% and 5.0% mAP@.5 respectively.

Impact of Epochs

We study how the number of training epochs affects the object detection performance. For this experiment we report results on COCO minival∗ set. Table 4 shows the performance comparison when the JFT-300M model has been trained for 1.3, 2.6 and 4 epochs respectively. We can see that as the number of training steps increases, the performance also improves. As a comparison, in Table 5 we show the ImageNet counterpart when trained for 3, 6, 12 and 150 epochs, we can see that the performance of ImageNet checkpoints improves faster than JFT-300M with respect to the number of epochs.

We would also like to point out that our learning schedules have been developed using the experience from smaller datasets. One can envision better learning schedules which provide more improvement as more epochs are used.

Impact of Data Size

For this experiment, we randomly sample a subset of 10M, 30M and 100M images from the JFT-300M training data. We use the same training schedule as the JFT-300M model training. We pick the checkpoints corresponding to the 4th epoch for each subset. To study the impact of learned visual representations, we also conduct an experiments to freeze the model weights for all layers before the *conv5* block. For this set of experiments we change the learning rate decay to happen at 900K steps, and the total number of training steps to 1.5M, as we find they tend to converge earlier.

In Figure 4, we show the mAP@[.5,.95] with checkpoints trained on different JFT-300M subsets, the blue curve corresponds to the regular faster RCNN training (with fine-tuning), while the red curve corresponds to freezing feature extractors. Not surprisingly, fine-tuning offers significantly better performance on all data sizes. Most interestingly, we can see that the performance grows logarithmically as pre-training data expands, this is particularly true when feature extraction layers are frozen.

Impact of Classes

JFT-300M has 18K labels in total. To understand what the large number of classes brings us, we select a subset of 941 labels which have direct correspondence to the 1000 ImageNet labels, and sample JFT-300M images which contain

method	airplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	TV	mean
ImageNet	79.7	80.6	77.1	65.9	64.2	85.3	81.0	88.4	60.5	83.1	70.8	86.7	86.2	79.7	79.5	49.5	78.3	80.2	79.2	69.7	76.3
300M	87.2	88.8	79.6	75.2	67.9	88.2	89.3	88.6	64.3	86.1	73.6	88.7	89.1	86.5	86.4	57.7	84.2	82.1	86.7	78.6	81.4
ImageNet+300M	86.9	88.0	80.1	74.7	68.8	88.9	89.6	88.0	69.7	86.9	71.9	88.5	89.6	86.9	86.8	53.7	78.2	82.3	87.7	77.9	81.3

Table 3. Average Precision @ IOU threshold of 0.5 on PASCAL VOC 2007 'test' set. The 'trainval' set of PASCAL VOC 2007 and 2012 are used for training.

#Iters on JFT-300M	#Epochs	mAP@[0.5,0.95]
12M	1.3	35.0
24M	2.6	36.1
36M	4	36.8

Table 4. mAP@[.5,.95] on COCO minival∗ with JFT-300M checkpoint trained from scratch for different number of epochs.

#Iters on ImageNet	#Epochs	mAP@[0.5,0.95]
100K	3	22.2
200K	6	25.9
400K	12	27.4
5M	150	34.5

Table 5. mAP@[.5,.95] on COCO minival∗ with ImageNet checkpoint trained for different number of epochs.

Number of classes	mAP@[.5,.95]
1K ImageNet	31.2
18K JFT	31.9

Table 6. Object detection performance in mean AP@[.5,.95] on COCO minival∗ set. We compare checkpoints pre-trained on 30M JFT images where labels are limited to the 1K ImageNet classes, and 30M JFT images covering all 18K JFT classes.

#Layers	ImageNet	300M
50	31.6	33.5
101	34.5	36.8
152	34.7	37.7

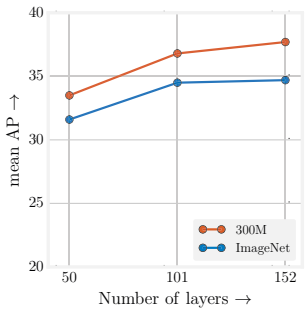

Figure 5. Object detection performance on COCO minival∗ on ResNet models with different number of layers.

at least one of such labels. This results in a subset of 30M images. We then train on this dataset for 4 epochs using the same training scheme.

Table 6 shows the performance comparison on COCO minival∗ set. We see that the two models perform on par with each other. This indicates that the performance benefit comes from more training images instead of more labels.

Impact of Model Capacity

Finally, we study the impact of model capacity when 300M images are available for training. We conduct the experiments on the 50-layer, 101-layer and 152-layer ResNet models. Each model is trained from scratch on the JFT-300M data, with the same hyper parameters used for ResNet-101 experiments. For comparison, we also train the models on ImageNet data till convergence, using the same hyper parameters for ResNet-101.

Figure 5 shows the performance of fine-tuning different pre-trained models on COCO minival∗set. We observe that higher capacity models are better at utilizing 300M data. For example, in case of ResNet-50 the gain is smaller compared to when using ResNet-152.

5.3. Semantic Segmentation

We use the PASCAL VOC 2012 semantic segmentation benchmark [12] which has pixel-wise labels for 20 foreground classes and one background class. As is standard practice, all models are trained on an augmented PASCAL VOC [12] 2012 'trainaug' set with 10582 images (extra annotations from [15]). We report quantitative results on the PASCAL VOC 2012 'val' set (1449 images) using the standard mean intersection-over-union (mIOU) metric.

Implementation details. The DeepLab-ASPP-L model [4] has four parallel branches after *conv5* block of ResNet101 architecture. Each branch is a (3×3) convolutional layer, with a different atrous rate r ($r \in \{6, 12, 8, 24\}$). Different atrous rates enable the model to capture objects and context at different scales. Output of each branch is pixel-wise scores for 21 classes with the same resolution output map (subsampled by factor of 8 compared to the original image). These scores are added together and normalized for the final pixel-wise class probabilities.

For training, we use mini-batch SGD with momentum. Our model is trained for 30k SGD iterations using a mini-batch of 6 images, momentum of 0.9, an initialize learning rate (LR) of 10^{-3} and "polynomial" learning rate policy [4]. All layers are trained with L2-regularization (weight decay of 5×10^{-4}). We do not use any data-augmentation, multi-scale training/testing or post-processing using CRFs for this task. To initialize the DeepLab-ASPP-L model using ImageNet or JFT-300M trained checkpoints, the final classification layer from these checkpoints is replaced with four convolutional branches (initialized using Xavier). All in-

Initialization	mIOU
ImageNet	73.6
300M	75.3
ImageNet+300M	**76.5**

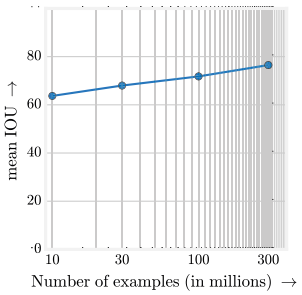

Figure 6. Semantic segmentation performance on Pascal VOC 2012 val set. (left) Quantitative performance of different initializations; (right) Impact of data size on performance.

	AP	AP@.5	AR	AR@.5
CMU Pose [3]	61.8	84.9	66.5	87.2
ImageNet [29]	62.4	84.0	66.7	86.6
300M	**64.8**	**85.8**	**69.4**	**88.4**
ImageNet+300M	64.4	85.7	69.1	88.2

Table 7. Human pose estimation performance on COCO 'test-dev' split. We follow the implementation of G-RMI Pose [29], but change the ResNet-101 initial checkpoints from ImageNet pre-trained to JFT-300M pre-trained.

put images are resized to (513×513), which results in a (65×65) *conv5* block from the ResNet101 network as well as $(65 \times 65 \times 21)$ predictions from the entire model.

Comparison with ImageNet Models. We present quantitative comparison of JFT-300M checkpoints with ImageNet checkpoints in Figure 6 (left). We see that the JFT-300M checkpoint outperforms ImageNet by 1.7% points. We further observe that the JFT-300M model trained from the ImageNet checkpoint provides 2.9% points boost over the vanilla ImageNet checkpoint.

Impact of Data Size. In Figure 6 (right), we further present analysis of impact of training data size by randomly sampling a subset of 10M, 30M and 100M images from the JFT-300M for training base checkpoints (same as Section 5.2). Once again we observe that the performance increases logarithmically as the pre-training dataset increases.

5.4. Human Pose Estimation

We train the fully-convolutional pose detector [29] by initializing the base ResNet model with our checkpoints and fine-tuning. The model is trained with SGD+Momentum for 450K steps. The learning rate was dropped by a factor of 10 after 250K steps, starting with a base learning rate. Best hyper parameter combination for each model was then selected independently and used in further experimentation.

In Table 7, we present the end to end pose estimation results evaluated on COCO 'test-dev' set. G-RMI Pose uses the ImageNet pre-trained checkpoint for fine-tuning, and we can see that our models with JFT-300M initialization perform much better. Note that to have a fair comparison with G-RMI Pose, we show their performance when only COCO images are used for training (fine-tuning) and no ensembling is performed. We use the person detection results provided by the authors and apply our trained pose detectors on the same set of person boxes.

6. Discussions

Is it to be expected that performance of computer vision algorithms would always improve with more and more data? In our personal correspondences with several researchers, the general consensus seems to be that everyone expects some gain in performance numbers if the dataset size is increased dramatically, with decreasing marginal performance as the dataset grows. Yet, while a tremendous amount of time is spent on engineering and parameter sweeps; little to no time has been spent collectively on data.

Our paper is an attempt to put the focus back on the data. The models seem to be plateauing but when it comes to the performance with respect to data – but modest performance improvements are still possible for exponential increases of the data. Another major finding of our paper is that having better models is not leading to substantial gains because ImageNet is no more sufficient to use all the parameters or their representational power.

Representation learning: One of the underlying debates is that should we spend more time collecting data for individual tasks such as detection and segmentation. Our findings show there is still a lot to be gained from representation learning. Improved base models or base features can lead to significant gains in performance.

Disclaimer – Large scale learning: We would like to highlight that the training regime, learning schedules and parameters used in this paper are based on our understanding of training ConvNets with 1M images. Searching the right set of hyper-parameters requires significant more effort: even training a JFT model for 4 epochs needed 2 months on 50 K-80 GPUs. Therefore, in some sense the quantitative performance reported in this paper underestimates the impact of data for all reported image volumes.

Acknowledgements: This work would not have been possible without the heroic efforts of Image Understanding and Expander teams at Google who built the massive JFT dataset. We would specifically like to thank Tom Duerig, Neil Alldrin, Howard Zhou, Lu Chen, David Cai, Gal Chechik, Zheyun Feng, Xiangxin Zhu and Rahul Sukthankar for their help. Also big thanks to the VALE team for APIs and specifically, Jonathan Huang, George Papandreou, Liang-Chieh Chen and Kevin Murphy for helpful discussions.

References

[1] P. Agrawal, R. B. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014. 2

[2] A. Bergamo and L. Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *NIPS*. 2010. 2

[3] Z. Cao, T. Simon, S. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *arXiv:1611.08050*, 2016. 8

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 2, 4, 5, 7

[5] X. Chen and A. Gupta. Webly supervised learning of convolutional networks. In *ICCV*, 2015. 2

[6] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. In *ICCV*, 2013. 2

[7] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv:1610.02357*, 2016. 2, 3, 4

[8] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, K. Yang, and A. Y. Ng. Large scale distributed deep networks. In *NIPS*, 2012. 3

[9] S. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 2

[10] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 2

[11] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv:1605.09782*, 2016. 2

[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 2010. 7

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 5

[14] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524*, 2013. 2

[15] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 7

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 4, 5, 6

[17] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS*, 2014. 2, 3

[18] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 2, 5, 6

[19] M. Huh, P. Agrawal, and A. A. Efros. What makes imagenet good for transfer learning? *arXiv:1608.08614*, 2016. 3

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015. 3

[21] H. Izadinia, B. C. Russell, A. Farhadi, M. D. Hoffman, and A. Hertzmann. Deep classifiers from image tags in the wild. In *ACM MM*, 2015. 2

[22] M. Jain, J. C. van Gemert, and C. G. Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *CVPR*, 2015. 2

[23] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. *arXiv:1511.02251*, 2015. 2, 3

[24] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and F. Li. The unreasonable effectiveness of noisy data for fine-grained recognition. *arXiv:1511.06789*, 2015. 2

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[26] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 5

[27] K. Ni, R. A. Pearce, K. Boakye, B. V. Essen, D. Borth, B. Chen, and E. X. Wang. Large-scale deep learning on the YFCC100M dataset. *arXiv:1502.03409*, 2015. 2

[28] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014. 3

[29] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy. Towards accurate multi-person pose estimation in the wild. *arXiv:1701.01779*, 2017. 2, 5, 8

[30] F. Pereira, P. Norvig, and A. Halev. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 2009. 2

[31] L. Pinto, D. Gandhi, Y. Han, Y. Park, and A. Gupta. The curious robot: Learning visual representations via physical interactions. *arXiv:1604.01360*, 2016. 2

[32] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *arXiv:1509.06825*, 2015. 2

[33] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2, 4, 6

[34] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu. Unsupervised joint object discovery and segmentation in internet images. *CVPR*, 2013. 2

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *arXiv:1409.0575*, 2014. 1

[36] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *arXiv:1406.2199*, 2014. 2

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 2

[38] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv:1602.07261*, 2016. 6

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2

[40] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L. Li. The new data and new challenges in multimedia research. *arXiv:1503.01817*, 2015. 3

[41] A. Torralba and A. Efros. Unbiased look at dataset bias. *CVPR*, 2011. 2

[42] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*, 2016. 2

[43] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. *arXiv:1505.00687*, 2015. 2

[44] T. Weyand, I. Kostrikov, and J. Philbin. Planet - photo geolocation with convolutional neural networks. *arXiv:1602.05314*, 2016. 3