Rethinking ImageNet Pre-training

Kaiming He Ross Girshick Piotr Dollár
Facebook AI Research (FAIR)

Abstract

We report competitive results on object detection and instance segmentation on the COCO dataset using standard models trained ***from random initialization****. The results are* ***no worse*** *than their ImageNet pre-training counterparts even when using the hyper-parameters of the baseline system (Mask R-CNN) that were optimized for fine-tuning pre-trained models, with the sole exception of increasing the number of training iterations so the randomly initialized models may converge. Training from random initialization is surprisingly robust; our results hold even when: (i) using only 10% of the training data, (ii) for deeper and wider models, and (iii) for multiple tasks and metrics. Experiments show that ImageNet pre-training speeds up convergence early in training, but does not necessarily provide regularization or improve final target task accuracy. To push the envelope we demonstrate 50.9 AP on COCO object detection without using any external data—a result on par with the top COCO 2017 competition results that used ImageNet pre-training. These observations challenge the conventional wisdom of ImageNet pre-training for dependent tasks and we expect these discoveries will encourage people to rethink the current de facto paradigm of 'pre-training and fine-tuning' in computer vision.*

1. Introduction

Deep convolutional neural networks [21, 23] revolutionized computer vision arguably due to the discovery that feature representations learned on a *pre-training* task can transfer useful information to target tasks [9, 6, 50]. In recent years, a well-established paradigm has been to pre-train models using large-scale data (*e.g.*, ImageNet [39]) and then to fine-tune the models on target tasks that often have less training data. Pre-training has enabled state-of-the-art results on many tasks, including object detection [9, 8, 36], image segmentation [29, 13], and action recognition [42, 4].

A path to 'solving' computer vision then appears to be paved by pre-training a 'universal' feature representation on ImageNet-like data at *massive* scale [44, 30]. Attempts along this path have pushed the frontier to up to 3000× [30] the size of ImageNet. However, the success of these experiments is mixed: although improvements have been observed, for object detection in particular they are small and scale poorly with the pre-training dataset size. That this path will 'solve' computer vision is open to doubt.

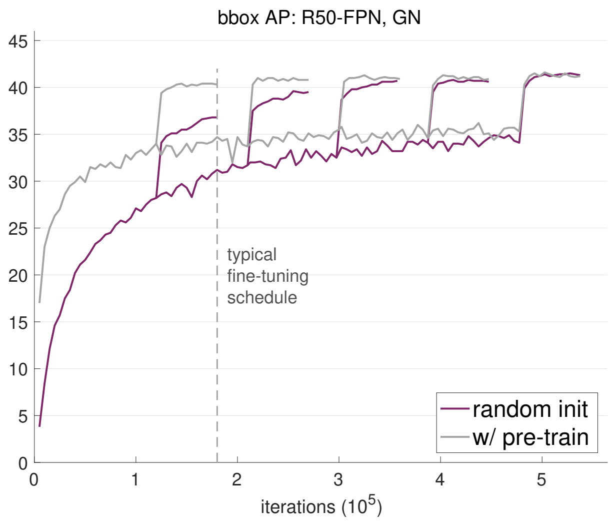

Figure 1. We train Mask R-CNN [13] with a ResNet-50 FPN [26] and GroupNorm [48] backbone on the COCO `train2017` set and evaluate bounding box AP on the `val2017` set, initializing the model by random weights or ImageNet pre-training. We explore different training schedules by varying the iterations at which the learning rate is reduced (where the accuracy leaps). The model trained from random initialization needs more iterations to converge, but converges to a solution that is *no worse than the fine-tuning counterpart*. Table 1 shows the resulting AP numbers.

This paper questions the paradigm of pre-training even further by exploring the opposite regime: we report that competitive object detection and instance segmentation accuracy is achievable when training on COCO *from random initialization ('from scratch'), without any pre-training*. More surprisingly, we can achieve these results by *using baseline systems* [8, 36, 26, 13] *and their hyper-parameters that were optimized for fine-tuning pre-trained models*. We find that there is no fundamental obstacle preventing us from training from scratch, if: (i) we use normalization techniques appropriately for optimization, and (ii) we train the models sufficiently long to compensate for the lack of pre-training (Figure 1).

We show that training from random initialization on COCO can be *on par with* its ImageNet pre-training counterparts for a variety of baselines that cover Average Precision (AP, in percentage) from 40 to over 50. Further, we find that such comparability holds even if we train with as little as 10% COCO training data. We also find that we can train large models from scratch—up to 4× larger than a ResNet-101 [17]—without overfitting. Based on these experiments and others, we observe the following:

(i) ImageNet pre-training *speeds up* convergence, especially early on in training, but training from random initialization can catch up after training for a duration that is roughly comparable to the total ImageNet pre-training plus fine-tuning computation—it has to learn the low-/mid-level features (such as edges, textures) that are otherwise given by pre-training. As the time/resource overhead of ImageNet pre-training is often ignored when studying the target task, 'controlled' comparisons with a *short* training schedule can veil the true behavior of training from random initialization.

(ii) ImageNet pre-training does *not* automatically give better regularization. When training with fewer images (down to 10% of COCO), we find that new hyper-parameters must be selected for fine-tuning (from pre-training) to avoid overfitting. Then, when training from random initialization using these *same* hyper-parameters, the model can match the pre-training accuracy *without* any extra regularization, even with only 10% COCO data.

(iii) ImageNet pre-training shows no benefit when the target tasks/metrics are more sensitive to spatially well-localized predictions. We observe a noticeable AP improvement for high box overlap thresholds when training from scratch; we also find that keypoint AP, which requires fine spatial localization, converges relatively faster from scratch. Intuitively, the task gap between the *classification*-based, ImageNet-like pre-training and *localization*-sensitive target tasks may limit the benefits of pre-training.

Given the current literature, these results are surprising and challenge our understanding of the effects of ImageNet pre-training. These observations hint that ImageNet pre-training is a historical workaround (and will likely be so for some time) for when the community does not have enough target data or computational resources to make training on the target task doable. In addition, ImageNet has been largely thought of as a 'free' resource, thanks to the readily conducted annotation efforts *and* wide availability of pre-trained models. But looking forward, when the community will proceed with more data and faster computation, our study suggests that collecting data and training on the *target* tasks is a solution worth considering, especially when there is a significant gap between the source pre-training task and the target task. This paper provides new experimental evidence and discussions for people to rethink the ImageNet-like pre-training paradigm in computer vision.

2. Related Work

Pre-training and fine-tuning. The initial breakthrough of applying deep learning to object detection (*e.g.*, R-CNN [9] and OverFeat [40]) were achieved by fine-tuning networks that were pre-trained for ImageNet classification. Following these results, most modern object detectors and many other computer vision algorithms employ the 'pre-training and fine-tuning' paradigm. Recent work pushes this paradigm further by pre-training on datasets that are 6× (ImageNet-5k [14]), 300× (JFT [44]), and even 3000× (Instagram [30]) larger than ImageNet. While this body of work demonstrates significant improvements on image classification transfer learning tasks, the improvements on object detection are relatively small (on the scale of +1.5 AP on COCO with 3000× larger pre-training data [30]). The marginal benefit from the kind of large-scale pre-training data used to date diminishes rapidly.

Detection from scratch. Before the prevalence of the 'pre-training and fine-tuning' paradigm, object detectors were trained *with no pre-training* (*e.g.*, [31, 38, 45])—a fact that is somewhat overlooked today. In fact, *it should not be surprising that object detectors can be trained from scratch*.

Given the success of pre-training in the R-CNN paper [9], later analysis [1] found that pre-training plays an important role in detector accuracy when training data is limited, but also illustrated that *training from scratch on more detection data is possible* and can achieve 90% of the fine-tuning accuracy, foreshadowing our results.

As modern object detectors [9, 15, 8, 36, 35, 28, 26, 13] evolved under the pre-training paradigm, the belief that training from scratch is non-trivial became conventional wisdom. Shen *et al.* [41] argued for *a set of new design principles* to obtain a detector that is optimized for the accuracy when trained from scratch. They designed a specialized detector driven by deeply supervised networks [24] and dense connections [18]. DetNet [25] and CornerNet [22] also present results when training detectors from scratch. Similar to [41], these works [25, 22] focus on designing detection-specific architectures. However, in [41, 25, 22] *there is little evidence that these specialized architectures are required for models to be trained from scratch*.

Unlike these papers, our focus is on understanding the role of ImageNet pre-training on *unspecialized* architectures (*i.e.*, models that were originally designed *without* the consideration for training from scratch). Our work demonstrates that it is often possible to *match* fine-tuning accuracy when training from scratch even *without* making any architectural specializations. Our study is on the comparison between 'with *vs.* without pre-training', under controlled settings in which the architectures are not tailored.

3. Methodology

Our goal is to *ablate* the role of ImageNet pre-training via controlled experiments that can be done *without* ImageNet pre-training. Given this goal, architectural improvements are *not* our purpose; actually, to better understand what impact ImageNet pre-training can make, it is desired to enable typical architectures to be trained from scratch under *minimal* modifications. We describe the only two modifications that we find to be necessary, related to model normalization and training length, discussed next.

3.1. Normalization

Image classifier training requires *normalization* to help optimization. Successful forms of normalization include normalized parameter initialization [11, 16] and activation normalization layers [20, 2, 46, 48]. When training object detectors from scratch, they face issues similar to training image classifiers from scratch [11, 16, 20]. Overlooking the role of normalization can give the misperception that detectors are hard to train from scratch.

Batch Normalization (BN) [20], the popular normalization method used to train modern networks, partially makes training detectors from scratch difficult. Object detectors are typically trained with high resolution inputs, unlike image classifiers. This reduces batch sizes as constrained by memory, and small batch sizes severely degrade the accuracy of BN [19, 34, 48]. This issue can be circumvented if pre-training is used, because fine-tuning can adopt the pre-training batch statistics as fixed parameters [17]; however, freezing BN is invalid when training from scratch.

We investigate two normalization strategies in recent works that help relieve the small batch issue:

(i) *Group Normalization* (**GN**) [48]: as a recently proposed alternative to BN, GN performs computation that is independent of the batch dimension. GN's accuracy is insensitive to batch sizes [48].

(ii) *Synchronized Batch Normalization* (**SyncBN**) [34, 27]: this is an implementation of BN [20] with batch statistics computed across multiple devices (GPUs). This increases the effective batch size for BN when using many GPUs, which avoids small batches.

Our experiments show that both GN and SyncBN can enable detection models to train from scratch.

We also report that using appropriately normalized initialization [16], we can train object detectors with VGG nets [43] from random initialization without BN or GN.

3.2. Convergence

It is unrealistic and unfair to expect models trained from random initialization to converge similarly fast as those initialized from ImageNet pre-training. Overlooking this fact one can draw incomplete or incorrect conclusions about the true *capability* of models that are trained from scratch.

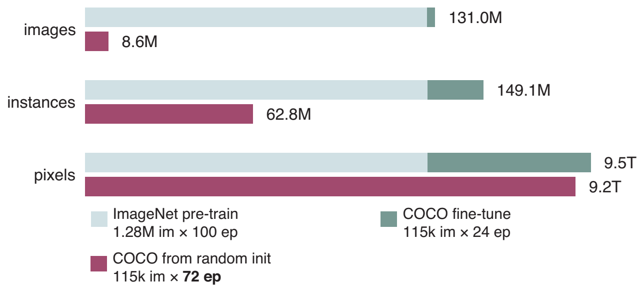

Figure 2. Total numbers of images, instances, and pixels seen during all training iterations, for pre-training + fine-tuning (green bars) *vs.* from random initialization (purple bars). We consider that pre-training takes 100 epochs in ImageNet, and fine-tuning adopts the 2× schedule (~24 epochs over COCO) and random initialization adopts the 6× schedule (~72 epochs over COCO). We count instances in ImageNet as 1 per image (*vs.* ~7 in COCO), and pixels in ImageNet as 224×224 and COCO as 800×1333.

Typical ImageNet pre-training involves over one million images iterated for one hundred epochs. In addition to any semantic information learned from this large-scale data, the pre-training model has also learned low-level features (*e.g.*, edges, textures) that do not need to be re-learned during fine-tuning.[1] On the other hand, when training from scratch the model has to learn low- and high-level semantics, so more iterations may be necessary for it to converge well.

With this motivation, we argue that models trained from scratch must be **trained for longer** than typical fine-tuning schedules. Actually, this is a *fairer* comparison in term of the number of training samples provided. We consider three rough definitions of 'samples'—the number of images, instances, and pixels that have been seen during all training iterations (*e.g.*, one image for 100 epochs is counted as 100 image-level samples). We plot the comparisons on the numbers of samples in Figure 2.

Figure 2 shows a from-scratch case trained for 3 times more iterations than its fine-tuning counterpart on COCO. Despite using more iterations on COCO, if counting image-level samples, the from-scratch case still sees considerably *fewer* samples than its fine-tuning counterpart—the 1.28 million ImageNet images for 100 epochs dominate. Actually, the sample numbers only get closer if we count *pixel-level* samples (Figure 2, bottom)—a consequence of object detectors using higher-resolution images. Our experiments show that under the schedules in Figure 2, the from-scratch detectors can catch up with their fine-tuning counterparts. This suggests that a sufficiently large number of total samples (arguably in terms of pixels) are required for the models trained from random initialization to converge well.

[1]In fact, it is common practice [8, 36] to freeze the convolutional filters in the first few layers when fine-tuning.

4. Experimental Settings

We pursue *minimal* changes made to baseline systems for pinpointing the keys to enabling training from scratch. Overall, our baselines and hyper-parameters follow Mask R-CNN [13] in the publicly available code of Detectron [10], except we use normalization and vary the number of training iterations. The implementation is as follows.

Architecture. We investigate Mask R-CNN [13] with ResNet [17] or ResNeXt [49] plus Feature Pyramid Network (FPN) [26] backbones. We adopt the end-to-end fashion [37] of training Region Proposal Networks (RPN) jointly with Mask R-CNN. GN/SyncBN is used to replace all 'frozen BN' (channel-wise affine) layers. For fair comparisons, in this paper the *fine-tuned* models (with pre-training) are also tuned with GN or SyncBN, rather than freezing them. They have higher accuracy than the frozen ones [34, 27, 48].

Learning rate scheduling. Original Mask R-CNN models in Detectron [10] were fine-tuned with 90k iterations (namely, '1× schedule') or 180k iterations ('2× schedule'). For models in this paper, we investigate longer training and we use similar terminology, *e.g.*, a so-called '6× schedule' has 540k iterations. Following the strategy in the 2× schedule, we always reduce the learning rate by 10× in the last 60k and last 20k iterations respectively, no matter how many total iterations (*i.e.*, the reduced learning rates are always run for the same number of iterations). We find that training longer for the first (large) learning rate is useful, but training for longer on small learning rates often leads to overfitting.

Hyper-parameters. All other hyper-parameters follow those in Detectron [10]. Specially, the initial learning rate is 0.02 (with a linear warm-up [12]). The weight decay is 0.0001 and momentum is 0.9. All models are trained in 8 GPUs using synchronized SGD, with a mini-batch size of 2 images per GPU.

By default Mask R-CNN in Detectron uses *no data augmentation* for testing, and only horizontal flipping augmentation for training. We use the same settings. Also, unless noted, the image scale is 800 pixels for the shorter side.

5. Results and Analysis

5.1. Training from scratch to match accuracy

Our first surprising discovery is that when *only* using the COCO data, models trained from scratch can catch up in accuracy with ones that are fine-tuned.

In this subsection, we train the models on the COCO `train2017` split that has ~118k (118,287) images, and evaluate in the 5k COCO `val2017` split. We evaluate bounding box (bbox) Average Precision (AP) for object detection and mask AP for instance segmentation.

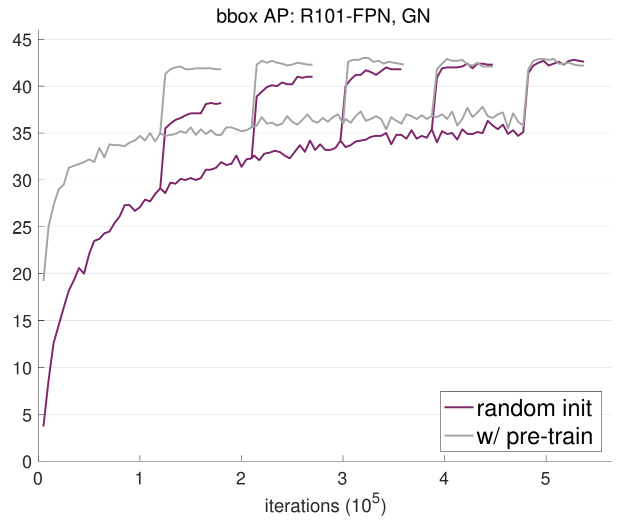

Figure 3. Learning curves of AP$^{\text{bbox}}$ on COCO `val2017` using Mask R-CNN with **R101**-FPN and GN. Table 1 shows the resulting AP numbers.

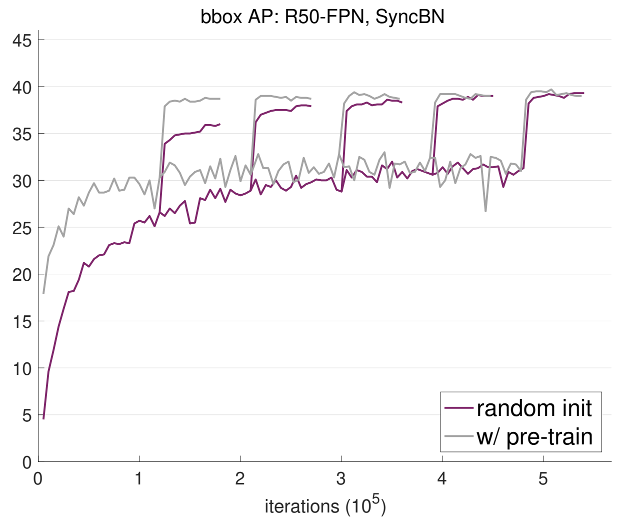

Figure 4. Learning curves of AP$^{\text{bbox}}$ on COCO `val2017` using Mask R-CNN with R50-FPN and **SyncBN** [34, 27] (that synchronizes batch statistics across GPUs). The results of the 6× schedule are 39.3 (random initialization) and 39.0 (pre-training).

Baselines with GN and SyncBN. The validation bbox AP curves are shown in Figures 1 and 3 when using GN for ResNet-50 (R50) and ResNet-101 (R101) backbones and in Figure 4 when using SyncBN for R50. For each figure, we compare the curves between models trained from random initialization *vs.* fine-tuned with ImageNet pre-training.

We study five different schedules for each case, namely, 2× to 6× iterations (Sec. 4). Note that we overlay the five schedules of one model in the same plot. The leaps in the AP curves are a consequence of reducing learning rates, illustrating the results of different schedules.

schedule		2×	3×	4×	5×	6×
R50	random init	36.8	39.5	40.6	40.7	41.3
	w/ pre-train	40.3	40.8	40.9	40.9	41.1
R101	random init	38.2	41.0	41.8	42.2	42.7
	w/ pre-train	41.8	42.3	42.3	41.9	42.2

Table 1. Object detection AP^{bbox} on COCO `val2017` of training schedules from 2× (180k iterations) to 6× (540k iterations). The model is Mask R-CNN with FPN and GN (Figures 1 and 3).

		AP^{bbox}	$\text{AP}^{\text{bbox}}_{50}$	$\text{AP}^{\text{bbox}}_{75}$	AP^{mask}	$\text{AP}^{\text{mask}}_{50}$	$\text{AP}^{\text{mask}}_{75}$
R50	random init	41.3	61.8	45.6	36.6	59.0	38.9
	w/ pre-train	41.1	61.7	44.6	36.4	58.5	38.7
	△	*+0.2*	*+0.1*	*+1.0*	*+0.2*	*+0.5*	*+0.2*
R101	random init	42.7	62.9	47.0	37.6	59.9	39.7
	w/ pre-train	42.3	62.6	46.2	37.2	59.7	39.7
	△	*+0.4*	*+0.3*	*+0.8*	*+0.4*	*+0.2*	*0.0*

Table 2. Training **from random initialization** *vs.* **with ImageNet pre-training** (Mask R-CNN with FPN and GN, Figures 1, 3), evaluated on COCO `val2017`. For each model, we show its results corresponding to the schedule (2 to 6×) that gives the best AP^{bbox}.

Similar phenomena, summarized below, are consistently present in Figures 1, 3, and 4:

(i) Typical fine-tuning schedules (2×) work well for the models with pre-training to converge to near optimum (see also Table 1, 'w/ pre-train'). But these schedules are not enough for models trained from scratch, and they appear to be inferior if they are only trained for a short period.

(ii) Models trained from scratch *can catch up* with their fine-tuning counterparts, if a 5× or 6× schedule is used—actually, when they converge to an optimum, their detection AP is *no worse* than their fine-tuning counterparts.

In the standard COCO training set, ImageNet pre-training mainly helps to *speed up convergence* on the target task early on in training, but shows *little or no evidence* of improving the final detection accuracy.

Multiple detection metrics. In Table 2 we further compare different detection metrics between models trained from scratch and with pre-training, including box-level and segmentation-level AP of Mask R-CNN, under Intersection-over-Union (IoU) thresholds of 0.5 (AP_{50}) and 0.75 (AP_{75}).

Table 2 reveals that models trained from scratch and with pre-training have *similar* AP metrics under various criteria, suggesting that the models trained from scratch catch up not only by chance for a single metric.

Moreover, for the $\text{AP}^{\text{bbox}}_{75}$ metric (using a *high overlap threshold*), training from scratch is better than fine-tuning by noticeable margins (1.0 or 0.8 AP).

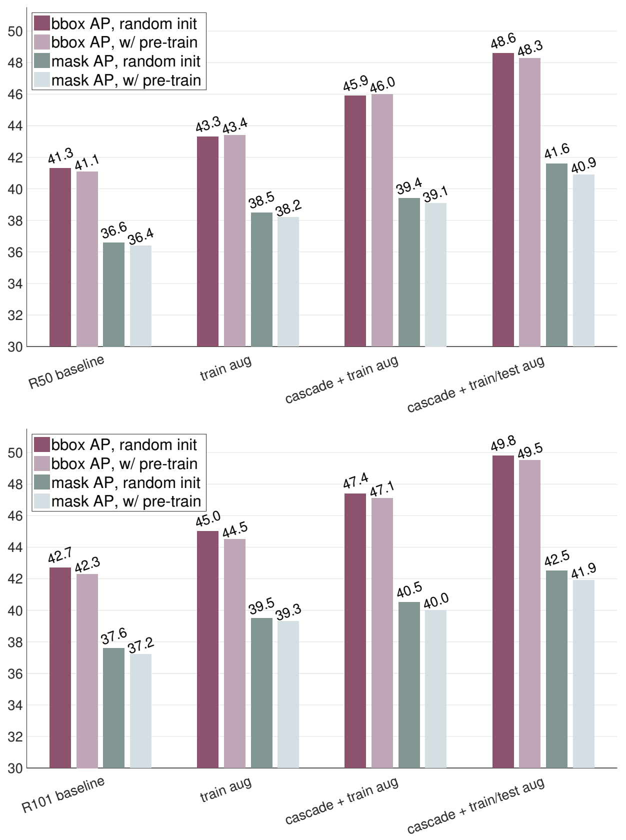

Figure 5. Comparisons between **from random initialization *vs.* with pre-training** on various systems using Mask R-CNN, including: (i) baselines using FPN and GN, (ii) baselines with training-time multi-scale augmentation, (iii) baselines with Cascade R-CNN [3] and training-time augmentation, and (iv) plus test-time multi-scale augmentation. **Top: R50; Bottom: R101**.

Enhanced baselines. The phenomenon that training with and without pre-training can be comparable is also observed in various enhanced baselines, as compared in Figure 5. We ablate the experiments as follows:

– Training-time scale augmentation: Thus far all models are trained with no data augmentation except horizontal flipping. Next we use the simple training-time scale augmentation implemented in Detectron: the shorter side of images is randomly sampled from [640, 800] pixels. Stronger data augmentation requires more iterations to converge, so we increase the schedule to 9× when training from scratch, and to 6× when from ImageNet pre-training.

Figure 5 ('train aug') shows that in this case models trained with and without ImageNet pre-training are still comparable. Actually, stronger data augmentation relieves the problem of insufficient data, so we may expect that models with pre-training have less of an advantage in this case.

– Cascade R-CNN [3]: as a method focusing on improving *localization* accuracy, Cascade R-CNN appends two extra stages to the standard two-stage Faster R-CNN system. We implement its Mask R-CNN version by simply adding a mask head to the last stage. To save running time for the from-scratch models, we train Mask R-CNN from scratch without cascade, and switch to cascade in the final 270k iterations, noting that this does not alter the fact that the final model uses no ImageNet pre-training. We train Cascade R-CNN under the scale augmentation setting.

Figure 5 ('cascade + train aug') again shows that Cascade R-CNN models have similar AP numbers with and without ImageNet pre-training. Supervision about *localization* is mainly provided by the target dataset and is not explicitly available from the classification-based ImageNet pre-training. Thus we do not expect ImageNet pre-training to provide additional benefits in this setting.

– *Test-time augmentation*: thus far we have used no test-time augmentation. Next we further perform test-time augmentation by combining the predictions from multiple scaling transformations, as implemented in Detectron [10].

Again, the models trained from scratch are *no worse* than their pre-training counterparts. Actually, models trained from scratch are even slightly better in this case—for example, mask AP is 41.6 (from scratch) *vs*. 40.9 for R50, and 42.5 *vs*. 41.9 for R101.

Large models trained from scratch. We have also trained a significantly larger Mask R-CNN model from scratch using a ResNeXt-152 8×32d [49] (in short 'X152') backbone with GN. The results are in Table 3.

This backbone has ~4× more FLOPs than R101. Despite being substantially larger, this model shows no noticeable overfitting. It achieves good results of **50.9** bbox AP and **43.2** mask AP in `val2017` when *trained from random initialization*. We submitted this model to COCO 2018 competition, and it has **51.3** bbox AP and **43.6** mask AP in the `test-challenge` set. Our bbox AP is at the level of the COCO 2017 winners (50.5 bbox AP, [34]), and is by far the highest number of its kind (single model, *without ImageNet pre-training*).

We have trained the same model with ImageNet pre-training. It has bbox/mask AP of 50.3/42.5 in `val2017` (*vs*. from-scratch's 50.9/43.2). Interestingly, even for this large model, pre-training does not improve results.

***vs*. previous from-scratch results.** DSOD [41] reported 29.3 bbox AP by using an architecture specially tailored for results of training from scratch. A recent work of CornerNet [22] reported 42.1 bbox AP (w/ multi-scale augmentation) using no ImageNet pre-training. Our results, of various versions, are higher than previous ones. Again, we emphasize that previous works [41, 22] reported *no evidence* that models without ImageNet pre-training can be *comparably* good as their ImageNet pre-training *counterparts*.

	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
R101 w/ train aug	45.0	65.7	49.3	39.5	62.5	42.1
X152 w/ train aug	46.4	67.1	51.1	40.5	63.9	43.4
+ cascade	48.6	66.8	52.9	41.4	64.2	44.6
+ test aug	50.9	68.7	55.4	43.2	66.1	46.8

Table 3. Mask R-CNN with **ResNeXt-152 trained from random initialization** (w/ FPN and GN), evaluated on COCO `val2017`.

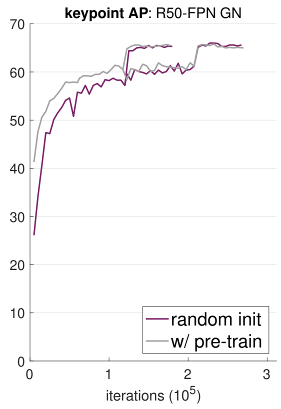

Figure 6. **Keypoint detection** on COCO using Mask R-CNN with R50-FPN and GN. We show keypoint AP on COCO `val2017`. ImageNet pre-training has little benefit, and training from random initialization can quickly catch up *without* increasing training iterations. We only need to use 2× and 3× schedules, unlike the object detection case. The result is 65.6 *vs*. 65.5 (random initialization *vs*. pre-training) with 2× schedules.

Keypoint detection. We also train Mask R-CNN for the COCO human keypoint detection task. The results are in Figure 6. In this case, the model trained from scratch can catch up more quickly, and even when *not* increasing training iterations, it is comparable with its counterpart that uses ImageNet pre-training. Keypoint detection is a task more sensitive to fine spatial localization. Our experiment suggests that ImageNet pre-training, which has little explicit localization information, does not help keypoint detection.

Models without BN/GN — VGG nets. Thus far all of our experiments involve ResNet-based models, which require some form of activation normalization (*e.g*., BN or GN). Shallower models like VGG-16 [43] can be trained from scratch without activation normalization as long as a proper initialization normalization is used [16]. Our next experiment tests the generality of our observations by exploring the behavior of training Faster R-CNN from scratch using VGG-16 as the backbone.

We implement the model following the original Faster R-CNN paper [37] and its VGG-16 architecture; no FPN is used. We adopt standard hyper-parameters with a learning rate of 0.02, learning rate decay factor of 0.1, and weight decay of 0.0001. We use scale augmentation during training. Following previous experiments, we use the exact same hyper-parameters when fine-tuning and training from scratch. When randomly initializing the model, we use the same MSRA initialization [16] for ImageNet pre-training and for COCO from scratch.

The baseline model *with pre-training* is able to reach a maximal bbox AP of 35.6 after an extremely long 9× training schedule (training for longer leads to a slight degradation in AP). Here we note that even *with* pre-training,

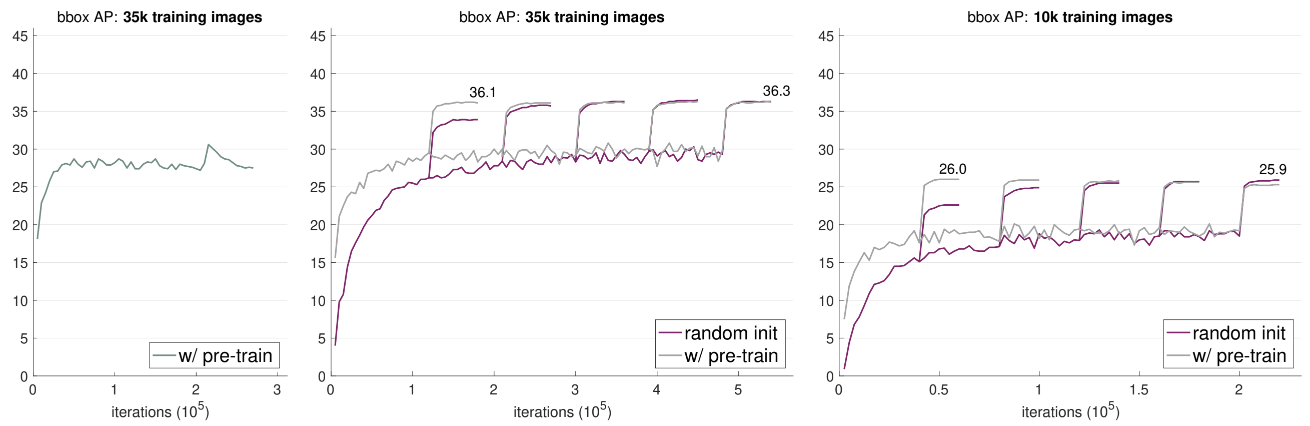

Figure 7. **Training with fewer COCO images** (left/middle: **35k**; right: **10k**). The model is Mask R-CNN with R50-FPN and GN, evaluated by bbox AP in `val2017`. **Left**: training with 35k COCO images, using the default hyper-parameters that were chosen for the 118k `train2017`. It shows overfitting before and after the learning rate changes. **Middle**: training with 35k COCO images, using hyper-parameters optimized for 'w/ pre-train' (the same hyper-parameters are then applied to the model from random initialization). **Right**: training with 10k COCO images, using hyper-parameters optimized for 'w/ pre-training'.

full convergence for VGG-16 is slow. The model trained from scratch reaches a similar level of performance with a maximal bbox AP of 35.2 after an 11× schedule (training for longer resulted in a lower AP, too). These results indicate that our methodology of 'making minimal/no changes' (Sec. 3) but adopting good optimization strategies and training for longer are sufficient for training comparably performant detectors on COCO, compared to the standard 'pre-training and fine-tuning' paradigm.

5.2. Training from scratch with less data

Our second discovery, which is even more surprising, is that with substantially less data (*e.g.*, ∼1/10 of COCO), models trained from scratch are *no worse* than their counterparts that are pre-trained.

35k COCO training images. We start our next investigation with ∼1/3 of COCO training data (35k images from `train2017`, equivalent to the older `val35k`). We train models with or without ImageNet pre-training on this set.

Figure 7 (left) is the result using ImageNet pre-training under the hyper-parameters of Mask R-CNN that were chosen for the 118k COCO set. These hyper-parameters are not optimal, and the model suffers from overfitting even with ImageNet pre-training. It suggests that *ImageNet pre-training does not automatically help reduce overfitting*.

To obtain a healthy baseline, we redo grid search for hyper-parameters on the models that are *with* ImageNet pre-training.[2] The gray curve in Figure 7 (middle) shows the results. It has optimally 36.3 AP with a 6× schedule.

[2]Our new recipe changes are: training-time scale augmentation range of [512, 800] (*vs*. baseline's no scale augmentation), a starting learning rate of 0.04 (*vs*. 0.02), and a learning rate decay factor of 0.02 (*vs*. 0.1).

Then we train our model from scratch using *the exact same* new hyper-parameters that are chosen for the pre-training case. This obviously biases results in favor of the pre-training model. Nevertheless, the model trained from scratch has 36.3 AP and *catches up* with its pre-training counterpart (Figure 7, middle), despite less data.

10k COCO training images. We repeat the same set of experiments on a smaller training set of 10k COCO images (*i.e*., *less than 1/10th of the full COCO set*). Again, we perform grid search for hyper-parameters on the models that use ImageNet pre-training, and apply them to the models trained from scratch. We shorten the training schedules in this small training set (noted by x-axis, Figure 7, right).

The model with pre-training reaches 26.0 AP with 60k iterations, but has a slight degradation when training more. The counterpart model trained from scratch has 25.9 AP at 220k iterations, which is *comparably* accurate.

Breakdown regime: 1k COCO training images. That training from scratch in 10k images is comparably accurate is surprising. But it is not reasonable to expect this trend will last for arbitrarily small target data, as we report next.

In Figure 8 we repeat the same set of experiments using only 1k COCO training images (∼1/100th of full COCO, again optimizing hyper-parameters for the pre-training case) and show the training loss. In terms of *optimization* (*i.e*., reducing training loss), training from scratch is still *no worse* but only converges more slowly, as seen previously. However, in this case, the training loss does not translate into a good validation AP: the model with ImageNet pre-training has 9.9 AP *vs*. the from scratch model's 3.5 AP. For one experiment only we also performed a grid search to optimize the from-scratch case: the

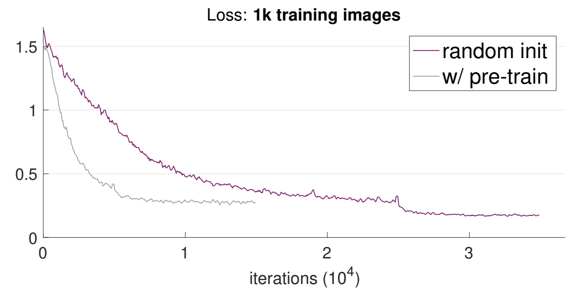

Figure 8. Training with **1k COCO images** (shown as the *loss* in the training set). The model is Mask R-CNN with R50-FPN and GN. As before, we use hyper-parameters optimized for the model with pre-training, and apply the same hyper-parameters to the model from random initialization. The randomly initialized model can catch up for the *training loss*, but has lower *validation accuracy* (3.4 AP) than the pre-training counterpart (9.9 AP).

result improves to 5.4 AP, but does not catch up. This is a sign of strong overfitting due to the severe lack of data.

We also do similar experiments using 3.5k COCO training images. The model that uses pre-training has a peak of 16.0 bbox AP *vs*. the trained from scratch counterpart's 9.3 AP. The breakdown point in the COCO dataset is somewhere between 3.5k to 10k training images.

Breakdown regime: PASCAL VOC. Lastly we report the comparison in PASCAL VOC object detection [7]. We train on the set of `trainval2007`+`train2012`, and evaluate on `val2012`. Using ImageNet pre-training, our Faster R-CNN baseline (with R101-FPN, GN, and only training-time augmentation) has 82.7 mAP at 18k iterations. Its counterpart trained from scratch in VOC has 77.6 mAP at 144k iterations and does not catch up even training longer.

There are 15k VOC images used for training. But these images have on average 2.3 instances per image (*vs*. COCO's ~7) and 20 categories (*vs*. COCO's 80). They are not directly comparable to the same number of COCO images. We suspect that the fewer instances (and categories) has a similar negative impact as insufficient training data, which can explain why training from scratch on VOC is not able to catch up as observed on COCO.

6. Discussions

We summarize the main observations from our experiments as follows:

- *Training from scratch on target tasks is possible without architectural changes.*

- *Training from scratch requires more iterations to sufficiently converge.*

- *Training from scratch can be no worse than its ImageNet pre-training counterparts under many circumstances, down to as few as 10k COCO images.*

- *ImageNet pre-training speeds up convergence on the target task.*

- *ImageNet pre-training does not necessarily help reduce overfitting unless we enter a very small data regime.*

- *ImageNet pre-training helps less if the target task is more sensitive to localization than classification.*

Based on these observations, we provide our answers to a few important questions that may encourage people to rethink ImageNet pre-training:

Is ImageNet pre-training necessary? No—if we have enough target data (and computation). Our experiments show that ImageNet can help speed up convergence, but does not necessarily improve accuracy unless the target dataset is too small (*e.g*., $<$10k COCO images). It can be *sufficient* to directly train on the target data if its dataset scale is large enough. Looking forward, this suggests that *collecting annotations of target data (instead of pretraining data)* can be more useful for improving the target task performance.

Is ImageNet helpful? Yes. ImageNet pre-training has been a *important* auxiliary task for the computer vision community to progress. It enabled people to see significant improvements before larger-scale data was available (*e.g*., in VOC for a long while). It also largely helped to circumvent optimization problems in the target data (*e.g*., under the lack of normalization/initialization methods). Moreover, ImageNet pre-training reduces research cycles, leading to *easier* access to encouraging results—pre-trained models are widely and freely available today, pre-training cost does not need to be wasted repeatedly, and fine-tuning from pre-trained weights converges faster than from scratch. We believe that these advantages will still make ImageNet undoubtedly helpful for computer vision research.

Is big data helpful? Yes. But a generic large-scale, *classification-level* pre-training set is not ideal if we take into account the extra effort of collecting and cleaning data—the resource demand of collecting ImageNet has been largely ignored, but the 'pre-training' step in the 'pre-training + fine-tuning' paradigm is in fact *not free* when we scale out this paradigm. If the gain of large-scale classification-level pre-training becomes exponentially diminishing [44, 30], it would be more effective to collect data in the target domain.

Shall we pursuit universal representations? Yes. We believe learning universal representations is a laudable goal. Our results do not mean deviating from this goal. Actually, our study suggests that the community should be more careful when evaluating pre-trained features (*e.g*., for *self-supervised* learning [5, 47, 33, 32]), as now we learn that even random initialization could produce excellent results.

In closing, ImageNet and its pre-training role have been incredibly influential in computer vision, and we hope that our new experimental evidence about ImageNet and its role will shed light into potential future directions for the community to move forward.

References

[1] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014. 2

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016. 3

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, 2018. 5

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1

[5] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 8

[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 *ICML*, 2014. 1

[7] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 2010. 8

[8] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 2, 3

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[10] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018. 4, 6

[11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 3

[12] 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:1706.02677*, 2017. 4

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2, 4

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *TPAMI*, 2018. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 3, 6

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 4

[18] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 2

[19] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NIPS*, 2017. 3

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3

[21] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[22] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. In *ECCV*, 2018. 2, 6

[23] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989. 1

[24] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015. 2

[25] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. DetNet: A backbone network for object detection. *arXiv:1804.06215*, 2018. 2

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 2, 4

[27] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 3, 4

[28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[30] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018. 1, 2, 8

[31] Ofer Matan, Christopher JC Burges, Yann LeCun, and John S Denker. Multi-digit recognition using a space displacement neural network. In *NIPS*, 1992. 2

[32] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017. 8

[33] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 8

[34] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. MegDet: A large mini-batch object detector. In *CVPR*, 2018. 3, 4, 6

[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 3

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *TPAMI*, 2017. 4, 6

[38] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. In *NIPS*, 1996. 2

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 1

[40] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 2

[41] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. DSOD: Learning deeply supervised object detectors from scratch. In *ICCV*, 2017. 2, 6

[42] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3, 6

[44] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017. 1, 2, 8

[45] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *NIPS*, 2013. 2

[46] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016. 3

[47] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 8

[48] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 1, 3, 4

[49] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 4, 6

[50] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional neural networks. In *ECCV*, 2014. 1