A Random CNN Sees Objects: One Inductive Bias of CNN and Its Applications

Yun-Hao Cao, Jianxin Wu*

State Key Laboratory for Novel Software Technology
Nanjing University, China
caoyh@lamda.nju.edu.cn, wujx2001@nju.edu.cn

Abstract

This paper starts by revealing a surprising finding: without any learning, a randomly initialized CNN can localize objects surprisingly well. That is, a CNN has an inductive bias to naturally focus on objects, named as Tobias ("The object is at sight") in this paper. This empirical inductive bias is further analyzed and successfully applied to self-supervised learning. A CNN is encouraged to learn representations that focus on the foreground object, by transforming every image into various versions with different backgrounds, where the foreground and background separation is guided by Tobias. Experimental results show that the proposed Tobias significantly improves downstream tasks, especially for object detection. This paper also shows that Tobias has consistent improvements on training sets of different sizes, and is more resilient to changes in image augmentations. Code is available at https://github.com/CupidJay/Tobias.

Introduction

Deep convolutional neural networks (CNNs) have achieved great success in various computer vision tasks. However, as of today we still know little about what makes a CNN suitable for analyzing natural images, i.e., what is its *inductive bias*. The inductive bias of a learning algorithm specifies constraints on the hypothesis space, and a model can only be instantiated from the hypothesis space that satisfies these constraints. It is easy to reveal the inductive bias of certain learning algorithms (e.g., a linear classifier specifies a linear relationship between the features and the target variable). But, the inductive bias of complex CNNs is still hidden in the fog (**?**). Successfully identifying CNN's inductive bias will not only deepen our theoretical understanding of this complex model, but also lead to potential important algorithmic progresses.

Objects are the key in most natural images, and CNNs are good at recognizing, detecting and segmenting objects. For instance, weakly supervised object localization (WSOL) (**???**) and unsupervised object localization (USOL) methods (**??**) can even localize objects without training on bounding box annotations. All these methods, however, *rely on ImageNet (**?**) pretrained models* and *non-trivial learning steps*.

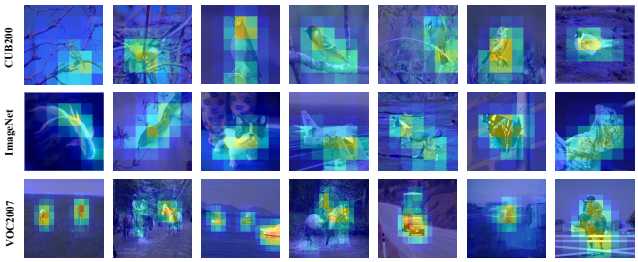

Figure 1: Visualization of localization heatmaps using SCDA (**?**) for a *randomly initialized* ResNet-50. Best viewed in color when zoomed in.

In this paper, we first show that focusing its attention to objects is a born gift of CNNs even *without any training*, i.e., it is CNN's inductive bias (or one inductive bias out of many) from an empirical perspective! A *randomly initialized* CNN has surprisingly good localization ability, as shown in Figure 1. We name this phenomenon "The object is at sight", or "Tobias" for short. The object(s) miraculously pop out ("at sight") without any need for learning. Our conjecture is: the background is relatively texture-less compared to the objects, and texture-less regions have higher chances to be deactivated by activation functions like ReLU.

Tobias then lends us 'free' (free of labels *and* pretrained models) and relatively accurate supervision for where objects are. Hence, a natural application of Tobias is self-supervised learning (SSL), which aims to learn useful representations without requiring labels. After the emerging of the InfoNCE loss (**?**) and the contrastive learning paradigm, many SSL algorithms have been published, such as MoCo (**?**), SimCLR (**?**), BYOL (**?**), and many more. In this paper, we propose to probabilistically change an image's background (selected from other images) while keeping the foreground objects by using Tobias. We thus force the model to learn representations focusing on the objects.

We evaluate the representation learned by Tobias SSL on ImageNet and other vision benchmarks. Our method achieves consistent improvements on various benchmarks, especially on object detection because our method can better capture the foreground objects. Also, we carefully study the influence of

*J. Wu is the corresponding author. This research was supported by the National Natural Science Foundation of China under Grant 61772256 and 61921006.

the number of pretraining images, and our method has consistent improvements on different amounts of training data. Our contributions are: (i) We find the "Tobias" inductive bias of CNN, i.e., a random CNN can localize objects without any learning. (ii) We find that activation functions like ReLU and network depth are essential for a random CNN to localize. (iii) We successfully apply Tobias to SSL and achieve consistent improvements on various benchmarks. (iv) Our method is robust when the amount of data is small or large, and is more resilient to changes in the set of image augmentations.

Related Works

Random networks' potential. ? proposed the Lottery Ticket Hypothesis: A randomly initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations. A lot of works followed this line of research (**???**). The SSL method BYOL (**?**) was also motivated by the random network's potential: the representation obtained by using fixed randomly initialized network to produce the targets can already be much better than the initial fixed representation. DIP (**?**) proposed that a randomly initialized neural network can be used as a handcrafted prior in standard inverse problems. These works show the potential of random networks from the perspective of network pruning, self learning or image denoising. We investigate it from a new perspective: a random CNN sees objects.

Un/Weakly-supervised object localization. Weakly supervised object localization (WSOL) (**??**) learns to localize objects with only image-level labels. CAM (**?**) generated class activation maps with the global average pooling (GAP) layer and the final fully connected (FC) layer (weights of the classifier). Unsupervised localization methods do not even need image-level labels. SCDA (**?**) aggregated information through the channel dimension to get localization masks. DDT (**?**) evaluated the correlation of descriptors. However, they all rely on ImageNet (**?**) pretrained models. Instead, our Tobias does not require any labels or pretrained models.

Self-supervised learning. Self-supervised learning (SSL) has emerged as a powerful method to learn visual representations without the expensive labels. Many recent works follow the contrastive learning paradigm (**?**). SimCLR (**?**) and MoCo (**?**) trained networks to identify a pair of views originating from the same image when contrasted with a large set of views from other images. The most related methods to ours are (**?**) and (**?**), where Mixup (**?**) or CutMix (**?**) was used to combine two images and force the new image to be similar to both. However, they may either generate unnatural images or cut objects out due to the lack of supervision. In contrast, our method provides free foreground vs. background supervision to merge patches, which proves to be useful in subsequent experiments.

Data augmentation. We use Tobias to merge patches from two different images to generate a new image, which keeps the objects and replaces the background. Our method can be viewed as a data augmentation strategy. As aforementioned, Mixup and CutMix do not have the location information as in our method and the random cut in CutMix may cover the foreground area with the background. "Copy and paste" (**??**) is an effective augmentation in object detection and instance segmentation, which cut object instances and paste them on other images. These methods require ground-truth bounding box labels, while ours does not rely on any labels.

Tobias, and SSL with Tobias

Now we first introduce how a randomly initialized CNN localizes objects. Then, we introduce how Tobias is applied to self-supervised learning.

Object localization using a random CNN

Given an input image $\boldsymbol{x}$ of size $H \times W$, the outputs of a CNN (before the GAP layer) are formulated as an order-3 tensor $Q \in \mathbb{R}^{h \times w \times d}$, which include a set of 2-D feature maps $S = \{S_n\}(n = 1, \ldots, d)$. S_n (of size $h \times w$) is the n-th feature map of the corresponding channel (the n-th channel). For instance, by employing the ResNet-50 (**?**) model, Q is the output of 'pool5' (i.e., activations of the last max-pooling layer) and we can get a $7 \times 7 \times 2048$ tensor if the input image is 224×224.

SCDA (**?**) obtains a 2-D aggregation map $A \in \mathbb{R}^{h \times w}$ by adding up Q through the depth direction and then uses the mean value of A as the threshold to localize objects. Formally, $A = \sum_{n=1}^{d} S_n$. Then, a mask map M of the same size as A can be obtained by

$$M_{i,j} = \begin{cases} 1 & \text{if } A_{i,j} > \bar{a} \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where $\bar{a} = \frac{1}{h \times w} \sum_{i,j} A_{i,j}$ and (i, j) denotes any position in these $h \times w$ locations. Those positions (i, j) whose activation responses are higher than $\bar{a}$ (i.e., $M_{i,j} = 1$) indicate the foreground objects.

The original SCDA (**?**) used ImageNet pretrained models for feature extraction and localization, and obtained good localization performance. However, there are many scenarios where pretrained models do not exist. Instead, we follow the same setups as in SCDA but replace the ImageNet pretrained weights by random weights. We find that a pretrained model is not necessary and a randomly initialized CNN can also localize objects surprisingly well. We name this phenomenon "The object is at sight", or "Tobias" for short. Figure 1 visualizes some localization examples, and we defer more results and analyses to the next section.

Tobias self-supervised learning

Based on our finding that an un-trained random network can capture foreground objects surprisingly well (i.e., Tobias), it is natural to wonder if we can take advantage of this property in SSL, where we do not have any pretrained models or annotated labels. In this section, we propose a Tobias augmentation, which keeps the objects and probabilistically changes the background for an image, and can be integrated into any existing SSL method. Moreover, we will demonstrate that our method can be viewed as either a data augmentation or a pseudo supervised contrastive learning method.

The Tobias augmentation. We make two modifications to SCDA in order to better adapt to SSL algorithms. First, we

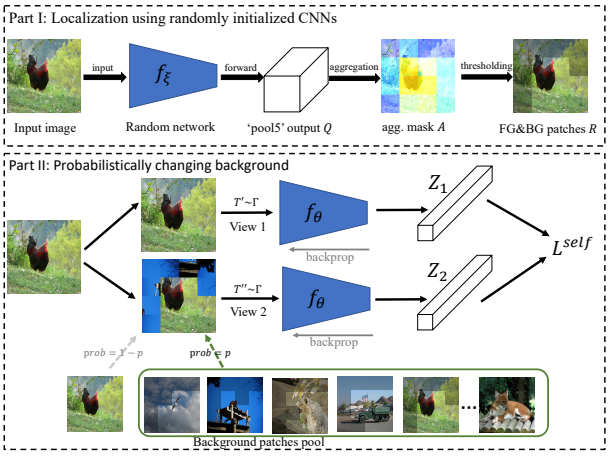

Figure 2: Pipeline of Tobias SSL. Upper part: splitting foreground and background using a *randomly initialized* CNN. Lower part: applying Tobias augmentation into SSL.

add an extra max-pooling layer (with stride=2) after 'pool5' and the mask map M becomes 4×4 instead of 7×7 for a 224×224 input image. The mask M for each image is pre-calculated by a randomly initialized network and do not change during further training. Second, we use *the median* instead of the mean value as the threshold to make sure that we have half the background ($M_{i,j} = 0$) and half the foreground ($M_{i,j} = 1$). Notice that this hard half-half division cannot fit all images exactly, because there exist images where objects cover more than or less than half of the area. However, this choice makes it easier when we combine foreground and background patches from two different images.

Then we split the input image $\boldsymbol{x}$ into $4 \times 4 = 16$ patches $R = \{R_{i,j}\}(i, j = 0, \ldots, 3)$, in which each patch corresponds to one position in M:

$$R_{i,j} = \boldsymbol{x}[i \times r : (i+1) \times r - 1, j \times r : (j+1) \times r - 1], \quad (2)$$

where $[:, :]$ denotes the slice operation, $r \times r$ is the patch size and $r = 224/4 = 56$ in our setting. We call $R_{i,j}$ a foreground patch if $M_{i,j} = 1$ and a background patch otherwise.

Given two image $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we can generate a new image $\boldsymbol{x}_{1,2}$, which contains foreground patches in $\boldsymbol{x}_1$ and background patches in $\boldsymbol{x}_2$. When merging patches from two images, we keep the positions of foreground patches unchanged and fill in other positions with background patches in a random order. Let $R^{(1)}$, $R^{(2)}$ and $R^{(1,2)}$ denote the patches in $\boldsymbol{x}_1, \boldsymbol{x}_2$ and $\boldsymbol{x}_{1,2}$, respectively. Then,

$$R^{(1,2)}_{i,j} = \begin{cases} R^{(1)}_{i,j} & \text{if } M^{(1)}_{i,j} = 1 \\ R^{(2)}_{\sigma(i,j)} & \text{otherwise} \end{cases}, \quad (3)$$

where $\sigma(\cdot, \cdot)$ defines a one-to-one mapping from background positions in $\boldsymbol{x}_1$ to background positions in $\boldsymbol{x}_2$. More specifically, background positions in $\boldsymbol{x}$ means $\{(i, j)|M_{i,j} = 0\}$ and σ defines such a random order to fill in background patches. Notice that all images have the same number of foreground and background patches and we are safe and free to merge these patches.

Applying Tobias to SSL. We now apply Tobias to the contrastive learning paradigm following the notations in SupCon (**?**). Suppose the dataset D has a total of N_t images and we randomly sample N images $\{\boldsymbol{x}_k\}_{k=1\ldots N}$ to form a batch. The corresponding batch used for training consists of $2N$ pairs, $\{\boldsymbol{x}'_k, \boldsymbol{x}''_k\}_{k=1\ldots N}$, where $\boldsymbol{x}'_k$ and $\boldsymbol{x}''_k$ are two random augmentations (i.e., "views") of $\boldsymbol{x}_k$. We denote the transformation as T, which is sampled from the predefined augmentation function space Γ. Hence we have $\boldsymbol{x}'_k = T'(\boldsymbol{x}_k)$ and $\boldsymbol{x}''_k = T''(\boldsymbol{x}_k)$, where $T', T'' \sim \Gamma$. In self-supervised contrastive learning, e.g., MoCo (**?**), the loss takes the following form:

$$L^{self} = -\sum_i \log \frac{e^{\boldsymbol{z}'_i \cdot \boldsymbol{z}''_i / \tau}}{\sum_{j \neq i} e^{\boldsymbol{z}'_i \cdot \boldsymbol{z}'_j / \tau} + \sum_j e^{\boldsymbol{z}'_i \cdot \boldsymbol{z}''_j / \tau}}, \quad (4)$$

where $\boldsymbol{z}'_i = f(\boldsymbol{x}'_i)$, $\boldsymbol{z}''_i = f(\boldsymbol{x}''_i)$, the $\cdot$ symbol denotes the inner product and τ is the temperature parameter. Here $f(\cdot) \equiv \text{Proj}(\text{Enc}(\cdot))$ denotes the composition of a encoder and a projection network.

Then we introduce Tobias into SSL (illustrated in Figure 2). Given an image $\boldsymbol{x}_k$, we generate the first view as before, i.e., $\boldsymbol{x}'_k = T'(\boldsymbol{x}_k)$. However, for another view $\boldsymbol{x}''_k$, we transform $\boldsymbol{x}_k$ into $\boldsymbol{x}_{k,m}$ by changing its background patches with another randomly selected image $\boldsymbol{x}_m$ with probability p, where p is a hyper-parameter:

$$\begin{cases} Pr\left(\boldsymbol{x}''_k = T''(\boldsymbol{x}_k)\right) = 1 - p \\ Pr\left(\boldsymbol{x}''_k = T''(\boldsymbol{x}_{k,m})\right) = \frac{p}{N_t} \quad , m = 1, \ldots, N_t \end{cases}. \quad (5)$$

Hence, the loss function becomes

$$L^{Tobias} = -\sum_i \log \frac{e^{\boldsymbol{z}'_i \cdot \boldsymbol{z}^p_i / \tau}}{\sum_{j \neq i} e^{\boldsymbol{z}'_i \cdot \boldsymbol{z}'_j / \tau} + \sum_j e^{\boldsymbol{z}'_i \cdot \boldsymbol{z}^p_j / \tau}}, \quad (6)$$

where $\boldsymbol{z}^p_i = f(\boldsymbol{x}^p_i)$ and $\boldsymbol{x}^p_i$ is one of the augmented samples in $P(i) \equiv \{\boldsymbol{x}_i, \boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,N_t}\}$, which follows the distribution in Equation 5. Notice that when $p = 0$, L^{Tobias} degenerates into L^{self}. Furthermore, Equation 6 can be seen as a pseudo supervised contrastive loss, where $P(i)$ contains images with the same foreground object.

Experimental Results

We use CUB-200 (**?**) and ImageNet (**?**) for our experiments. First, we show the localization results of randomly initialized CNNs and make further analyses. Then, we apply our Tobias method into SSL and demonstrate its effectiveness across various pretraining datasets, downstream tasks, backbone architectures and SSL algorithms. Finally, we study the effects of different components and hyper-parameters and sensitivity to data augmentations in our algorithm. All our experiments were conducted using PyTorch (**?**) and we used 8 Titan Xp GPUs for our experiments.

Localization ability of random CNNs

In this section, we study the localization ability of randomly initialized CNNs. We use SCDA (**?**) for localization and conduct experiments on two popular datasets for object localization, i.e., ImageNet and CUB-200. Notice that **?** used

Table 1: Comparisons of localization accuracy between ImageNet pretrained and randomly initialized CNNs on ImageNet and CUB-200. '#ReLU' and '#stages' represent the number of ReLU units and stages, respectively. 'IN super.' stands for 'ImageNet supervised'. We report the average accuracy and standard deviation of 3 trials for randomly initialized models.

Method	Backbone	#ReLU / #stages	ImageNet		CUB-200	
			IN super.	random init.	IN super.	random init.
SCDA (**?**)	R-50	33 / 5	51.9	48.2±0.6	44.8	41.8±0.6
	R-50 (sigmoid)	0 / 5	46.9	45.5±1.9	32.6	22.6±3.3
	R-50 (arctan)	0 / 5	34.4	36.6±0.7	19.1	18.1±0.3
	R-50 (conv1)	1 / 1	44.1	41.3±1.5	33.8	30.5±1.0
	R-50 (conv1-2)	7 / 2	38.4	39.7±1.5	22.1	29.6±0.9
	R-50 (conv1-3)	15 / 3	45.0	42.2±0.9	31.0	31.8±0.2
	R-50 (conv1-4)	27 / 4	49.9	47.2±1.3	39.2	40.1±0.4
	Vit-Base	- / -	50.9	40.5±0.5	48.6	31.9±1.3
CAM (**?**)	R-50	33 / 5	52.9	33.8±0.1	50.0	26.0±0.3

ImageNet pretrained models for evaluation while we study the potential of randomly initialized models here. The localization is correct when the intersection over union (IoU) between the ground truth bounding box and the predicted box is 50% or more. In Table 1, we report the average localization accuracy and standard deviation of 3 trials for randomly initialized models and we adopt Kaiming initialization (**?**) used in the PyTorch official code. We use the PyTorch official models for ImageNet pretrained models. We show some visualization results on CUB-200, ImageNet as well as one complex multi-object dataset Pascal VOC2007 (**?**) in Figure 1. The heatmap in Figure 1 is calculated by the 2-D aggregation mask A, as noted in the previous section.

As shown in Table 1, a randomly initialized ResNet-50 (R-50) (**?**) achieves comparable localization accuracies with its ImageNet supervised counterpart on both ImageNet and CUB-200. We also present one popular WSOL method CAM (**?**) for comparison and it further shows that our results for random CNNs are accurate. Notice that SCDA relies only on convolution feature maps while CAM also relies on the trained FC weights, hence we can see a significant drop for CAM with randomly initialized models. Also, from Figure 1 we can observe more intuitively that randomly initialized CNNs can not only locate a single object, but multiple objects as well. Furthermore, we can observe that the standard deviation of multiple trials is small for randomly initialized models (there is also only small difference between the visualization results of different trials). The results show that a randomly initialized CNN can achieve surprisingly good localization results and the localization results are robust with different random weights. Moreover, as the core component of CNNs is convolution, we also investigate what the localization effect has to do with convolution. We compare with the non-CNN architecture ViT-Base (**?**) and there is a large gap between the pretrained and randomly initialized ViT models. Hence, we can conclude that it is one inductive bias for CNNs, not for MLP-based architectures, e.g., ViT.

But, why can a random CNN see objects without any learning? Given the empirical results and in particular its stability under different random initializations, we believe it is the inductive bias of modern CNNs. There are a lot of ReLU and convolution layers inside ResNet-50 (and most other modern CNNs). Remember that SCDA simply adds feature maps across the channel dimension. Hence, if one spatial location has many zeros (i.e., deactivated after ReLU), we expect it to have a low SCDA score and thus being predicted as belonging to the background.

Our conjecture is then: *the background is relatively textureless when compared to the objects, and texture-less regions have higher chances to be deactivated by ReLU when the network depth increases.* We design two experiments to verify it. One is to replace all ReLU activations with other activation functions (e.g., sigmoid). The other one is to gradually reduce the number of ReLU units and we directly remove whole stages for R-50. For instance, 'conv1-4' means that we remove the last stage in R-50 (i.e., 'conv5'). From Table 1 we can have the following two conclusions. First, ReLU plays an important role because when we replace ReLU with sigmoid or arctan, a significant decrease in localization accuracy was observed. Second, network depth is also important and we can observe a significant performance degradation as the network depth decreases (i.e., fewer stages).

To make our conclusions more convincing, we add more baselines and further investigate different components in ResNet-50 as well as other CNN architectures in Table 2.

More baselines. We provide some more upper and lower bounds to help understanding. The lower bound is the accuracy of predicting the entire image as the bounding box (not trivial given that many images have a single prominent object in CUB and ImageNet). The upper bound is the accuracy of pretrained Faster R-CNN R-50 (**?**), which is directly supervised on the detection task COCO (**?**). We also compare with object proposal method EdgeBox (**?**). These results further prove that the localization results of random CNNs are good.

Different components in ResNet-50.

(1) Skip connection and batch normalization (BN) (**?**) are not crucial. We remove all skip connections in R-50 (row 2) and we even achieve slightly better performance than the original R-50 (row 1). Also, when we remove all BN (row 3), we achieve comparable performance.

(2) Network depth is important. We reduced the number of stages in Table 1 before and now we keep the number of stages unchanged but change the number of bottlenecks in each stage. The number of bottlenecks in each stage for R-50 is 3, 4, 6 and 3, respectively (denoted as [3,4,6,3]). When reduced to [1,2,3,1] (row 4), we have 4.3 and 5.1 points

Table 2: Localization accuracy of various CNNs on ImageNet and CUB-200. We report the average accuracy and standard deviation of 3 trials for randomly initialized models.

id	Backbone	ImageNet	CUB-200
1	R-50	48.2±0.6	41.8±0.6
2	R-50 (w/o skip connection)	50.8±1.0	42.3±1.5
3	R-50 (w/o batch normalization)	49.1±0.5	41.0±1.4
4	R-50 (shallow [1,2,3,1])	43.9±1.5	36.7±0.7
5	R-50 (shallow [1,1,1,1])	42.1±1.7	31.8±2.1
6	R-50 (deep [6,8,12,6])	50.0±0.8	45.0±0.9
7	R-50 (ELU)	49.3±0.9	46.6±2.7
8	R-50 (SELU)	50.4±0.6	45.4±3.5
9	R-50 (softplus)	51.0±2.7	51.6±3.1
10	R-50 (init: Normal(0,0.1))	50.6±0.3	43.4±0.5
11	R-50 (init: Uniform(-0.1,0.1))	50.0±0.4	43.4±0.4
12	R-50 (init: Xavier)	42.2±1.1	32.6±0.9
13	VGG-11	40.0±0.5	30.6±1.4
14	VGG-16	40.8±0.5	33.5±1.9
15	VGG-16 (sigmoid)	39.8±0.6	18.2±1.3
16	VGG-16 (arctan)	34.6±0.5	20.1±1.0
17	VGG-16 (ELU)	40.4±1.0	32.5±1.0
18	VGG-19	41.4±1.8	34.2±0.3
19	AlexNet	34.6±1.5	24.8±0.3
20	Inception v3	52.2±0.6	49.6±0.9
21	Hourglass	52.6±0.2	46.9±0.4
22	EdgeBox	31.8	32.7
23	lower bound (whole image)	38.8	19.1
24	upper bound (faster R-CNN)	58.9	96.2

decrease compared with original R-50 on ImageNet and CUB, respectively. When further reduced to [1,1,1,1] (row 5), we have 6.1 and 10.0 points decrease on ImageNet and CUB, respectively. Conversely, when we increase the number of bottlenecks to [6,8,12,6], we can get 1.8 and 3.2 points gains on ImageNet and CUB, respectively. It indicates that deeper architectures can better capture high-level information and localize objects better, even when randomly initialized.

(3) Other ReLU-like unbounded activations also help. When we use other activations, e.g., ELU, SELU and softplus (row 7∼9), we can get comparable or even better results than ReLU. All these activations have one thing in common: deactivate negative values and unbounded for positive values.

Other CNN architectures. Other randomly initialized CNN architectures can also localize objects well. We study with AlexNet (**?**), VGG-style networks (**?**), Hourglass network (**?**) and Inception v3 (**?**). We can observe that other architectures (e.g., VGG-19, Hourglass and Inception v3) can also achieve non-trivial localization ability.

When comparing among VGG-style networks, we can also observe that the localization accuracy increases with the increase of network depth (row 13∼18). Also, activations like ReLU perform better than sigmoid and arctan activations.

When comparing among different CNN architectures, we can see that deeper networks (e.g., Inception v3) perform better than shallow networks (e.g., AlexNet). As shown in Fig. 3, we rank the following according to the localization accuracy on ImageNet: AlexNet<VGG-11<VGG-16<VGG-19<ResNet-50<Inception v3. Note that the ranking of the localization accuracy of these random networks is surprisingly

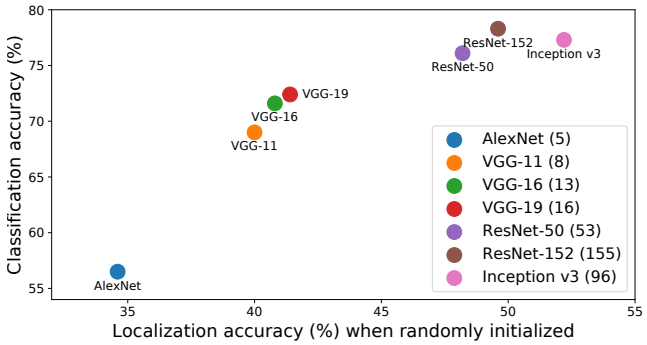

Figure 3: Classification accuracy after training (PyTorch model zoo) versus localization accuracy (when randomly initialized) on ImageNet. The number in brackets represents the number of convolutions in the model (i.e., depth).

consistent with their classification accuracy on ImageNet. We can conclude that deeper networks perform better in terms of localization, even when randomly initialized.

Initialization scheme. We observe that other initialization methods can also get good and robust localization results (row 10 and row 11) and Tobias is actually a general phenomenon. However, as discussed by **?**, the linear assumption in Xavier initialization (**?**) is invalid for ReLU and we can observe degraded localization accuracy (row 12) here.

In short, we find that randomly initialized CNNs can localize objects surprisingly well, which is even comparable to their supervised counterparts. Also, we analyze the effect of different components in modern CNNs. The results reveal the potential of a random CNN in localizing objects and provide a new perspective to explain why modern CNNs achieve such good performance in visual analysis.

Tobias self-supervised learning

Now we apply Tobias to SSL (Equation 6) and evaluate its effectiveness on CUB200 and ImageNet. Then, we will analyze the effects of different components and hyper-parameters and the sensitivity to data augmentations.

Results on CUB-200 We carefully study our Tobias using 2 typical SSL methods, namely MoCov2 (**?**) and SimCLR (**?**) under both ResNet-18 and ResNet-50. We follow the training and evaluation protocols in (**?**) and conduct experiments on CUB-200. The full learning process contains two stages: pretraining and fine-tuning. We use the pretrained weights obtained by SSL for initialization and then fine-tune networks for classification. Note that *SSL pretraining and fine-tuning are both performed only on the target dataset CUB-200* in this subsection.

For the fine-tuning stage, we fine-tune all methods for 120 epochs using SGD with a batch size of 64, a momentum of 0.9 and a weight decay of 5e-4 for fair comparison. The learning rate starts from 0.1 with cosine learning rate decay. We also list the results using the Mixup strategy, where the alpha is set to 1.0. For the SSL pretraining stage, we follow the same settings in the original papers. '-Tobias' denotes our method and we only change the data loading process and other training settings remain the same as baseline methods.

Table 3: Comparisons of pretraining details and accuracies (%) on CUB-200. 'N/A' means that pretraining are conducted on ImageNet instead of CUB-200 for ImageNet supervised models. 'FT' is short for 'fine-tuning'.

Backbone	SSL pretraining		Fine-tuning accuracy (%)	
	method	epochs	Normal FT	Mixup FT
ResNet-18	ImageNet super.	N/A	76.2	75.0
	random init.	0	62.0	63.4
	MoCov2	200	63.7	65.8
	MoCov2-Tobias		64.4 (+0.7)	66.3 (+0.5)
	MoCov2	800	65.0	66.3
	MoCov2-Tobias		66.2 (+1.2)	67.7 (+1.4)
	SimCLR	200	63.6	64.5
	SimCLR-Tobias		65.4 (+1.8)	68.6 (+4.1)
	SimCLR	800	66.0	67.3
	SimCLR-Tobias		67.4 (+1.4)	69.3 (+2.0)
ResNet-50	ImageNet super.	N/A	81.3	82.1
	random init.	0	58.6	56.3
	MoCov2	200	56.2	53.0
	MoCov2-Tobias		63.6 (+7.4)	62.0 (+9.0)
	MoCov2	800	66.5	62.0
	MoCov2-Tobias		67.2 (+0.7)	71.5 (+9.5)
	SimCLR	200	68.0	66.5
	SimCLR-Tobias		68.4 (+0.4)	71.7 (+5.2)
	SimCLR	800	69.2	73.0
	SimCLR-Tobias		70.0 (+0.8)	73.6 (+0.6)

Table 4: Object detection on PASCAL VOC trainval07+12 (default VOC metric AP_{50}, COCO-style AP, and AP_{75}).

pretraining method	R-50-FPN (24k)			R-50 C4 (24k)		
	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}
random init.	63.0	36.7	36.9	60.2	33.8	33.1
IN supervised	80.8	53.5	58.4	81.3	53.5	58.8
MoCov2 200ep	81.8	55.0	60.5	82.2	57.1	64.5
MoCov2-Tobias 200ep	**82.0**	**55.5**	**61.1**	**82.6**	**57.7**	**64.9**
MoCov2 800ep	81.5	55.0	61.0	82.6	57.7	64.5

The results are shown in Table 3. Tobias has consistent improvements under various backbones, pretraining epochs and SSL algorithms. Taking ResNet-50 as an example, our Tobias achieves 13.2% relative higher accuracies than the baseline MoCov2 with normal fine-tuning when both pretrained for 200 epochs. Also, the relative improvement has risen to 17.0% if we use MixUp. It is because that we also merge image patches (in an informative way) during pretraining and it is more friendly to subsequent fine-tuning with MixUp. Moreover, we can observe that the improvement is larger when pretrained for fewer epochs (200 vs. 800). It is because that our method can better capture foreground objects, which leads to faster convergence during pretraining. We will further demonstrate the effectiveness of such foreground vs. background information.

Results on ImageNet Now we move on to the large-scale dataset ImageNet. We use MoCv2 for illustration following the official training protocols in (**?**). We adopt ResNet-50 as backbone and set the batch size to 256, learning rate to 0.03 and number of epochs to 200. We study the downstream object detection performance on Pascal VOC 07&12 (**?**) in Table 4. The detector is Faster R-CNN with a backbone of R-50-FPN (**?**) or R-50-C4 (**?**), implemented in (**?**).

Table 5: Downstream object detection performance on VOC 07&12 and linear evaluation accuracy on Tiny-IN-200 when pretrained on ImageNet subsets using ResNet-50. '#imgs' ('#eps') represent the number of images (epochs).

pretraining			VOC 07&12		Tiny-IN-200
method	#imgs	#eps	AP_{50}	AP_{75}	
random init.	0	0	63.0	36.9	0.5
MoCov2	10k	200	71.1	45.8	0.5
MoCov2-Tobias			71.4 (+0.3)	47.0 (+1.2)	9.9 (+9.4)
MoCov2	10k	800	71.6	45.9	23.6
MoCov2-Tobias			73.2 (+1.6)	48.5 (+2.6)	23.9 (+0.3)
MoCov2-RM			72.0 ↓1.2	47.4 ↓1.1	23.5 ↓0.4
MoCov2-Mixup			70.9 ↓2.3	43.3 ↓5.2	19.3 ↓4.6
MoCov2	50k	200	72.2	46.8	26.3
MoCov2-Tobias			73.7 (+1.5)	49.2 (+2.4)	26.0 (-0.3)
MoCov2	50k	800	77.5	53.3	37.9
MoCov2-Tobias			77.9 (+0.4)	54.9 (+1.6)	40.7 (+2.8)
MoCov2-RM			77.4 ↓0.5	53.3 ↓1.6	40.1 ↓0.6
MoCov2-Mixup			76.7 ↓1.2	52.4 ↓2.5	38.7 ↓2.0
MoCov2	100k	200	76.2	51.6	35.3
MoCov2-Tobias			77.5 (+1.3)	53.9 (+2.3)	36.5 (+1.2)
MoCov2	100k	800	78.7	56.3	43.7
MoCov2-Tobias			79.4 (+0.7)	57.3 (+1.0)	44.3 (+0.6)

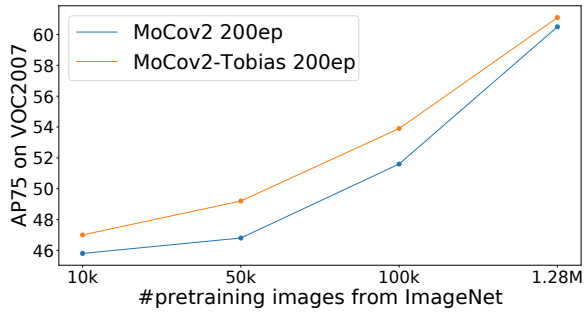

Figure 4: Performance of Tobias on Pascal VOC (AP_{75}) with respect to different training data size.

As shown in Table 4, Tobias achieves better performance than baseline MoCov2 on Pascal VOC. Also notice that our Tobias 200ep even achieves slightly better performance than MoCov2 800ep (pretrained much longer).

Apart from the full large-scale ImageNet dataset, we also study the performance under different data volumes by sampling the original ImageNet to smaller subsets, motivated by **?**. We randomly sample (*without using any image label*) 10 thousand (10k), 50 thousand (50k) and 100 thousand (100k) images to construct IN-10k, IN-50k and IN-100k, respectively. We only change the amount of data here and other training settings remain the same as before. The results are shown in Table 5 and we adopt Pascal VOC 07&12 for object detection and Tiny-ImageNet-200 (100,000 training and 10,000 validation images from 200 classes at 64×64 resolution) for linear evaluation.

As can be seen in Table 5 and Figure 4, our Tobias achieves significant improvements on both downstream tasks, especially on VOC 07&12 object detection. For instance, when both trained for 200 epochs on IN-100k, our Tobias is significantly better than baseline counterpart: up to **+1.3** AP_{50} and **+2.3** AP_{75}. Also notice that when both trained for 200

epochs on IN-10k, MoCov2 performs the same as random guess (0.5%) while our method learns much better representations (9.9%) in terms of Tiny-IN-200 linear evaluation. In general, our method improves the most on AP_{75}, which is a more stringent metric for detection accuracy. It indicates that our model can better capture foreground objects across changing backgrounds during pretraining, hence improving performance for object detection as well as image classification. Moreover, our method is especially effective (i.e., has greater improvements) when the amount of data is small.

Ablation studies

In this section, we will first study the effectiveness of the foreground vs. background (Tobias) information (generated by random networks). Then, we will study the effect of the hyper-parameter p in our method. Finally, we study the sensitivity to data augmentations.

Effect of Tobias information. Notice that we use the foreground vs. background information when merging patches from two images. To demonstrate its effectiveness, we design a random merging strategy for comparison (MoCov2-RM in Table 5). More specifically, we do not use such information and randomly select patches from two images for merging (also half-half division) and it can also be viewed as one kind of patch-level CutMix. We also compare with MoCov2-Mixup where we use Mixup when merging images. We keep all other settings the same and conduct pretraining on both IN-10k and IN-50k. As can be seen in Table 5, we will see a significant drop, especially in object detection performance if we discard the foreground vs. background information provided by our Tobias: up to **-1.2** AP_{50} for RM and **-2.3** AP_{50} for Mixup when trained on IN-10k for 800 epochs. It demonstrates the Tobias information provided by a randomly initialized network is vital. Another interesting thing is that RM achieves better performance than the baseline MoCov2, which indicates that this kind of data augmentation is somehow useful for SSL, as shown in (**?**).

Effect of hyper-parameter. Now we study the effect of the hyper-parameter p, i.e., the probability of changing backgrounds in another view. We study p = 0, 0.3, 0.5, 0.7 and 1.0. Notice that when p=0, our Tobias degenerates into the baseline MoCov2. We train on IN-10k for 800 epochs for all settings in Table 6. For object detection, we can see that when p grows, the result becomes better and will not continue to improve when it grows beyond 0.5. For Tiny ImageNet, $p = 0.7$ achieves the highest accuracy. Notice that we *directly set p to 0.3 for all our experiments throughout this paper* and did not tune it under different settings. It also indicates that we can get better results with more carefully tuned p.

Sensitivity to image augmentations. Now we study the sensitivity to image augmentations of our Tobias by progressively removing transformations in the transformation set following **?**. The results in Table 7 show that the performance of Tobias is much less affected than the performance of MoCov2 when removing the color distortion from the set of image augmentations, especially on Tiny-IN-200. Also we can observe that color distortion (e.g., grayscale and color-jitter) has greater impact on downstream image classification and less impact on object detection. When image augmentations are reduced to a mere random crop, the gap between our Tobias and baseline MoCov2 has increased to 2.9 and 3.2 points for Tiny-IN-200 and VOC detection (AP_{75}), respectively. It indicates that our Tobias is itself an effective data augmentation and less sensitive to other augmentations.

Table 6: Effect of hyper-parameter p. All settings are pretrained on IN-10k for 800 epochs using ResNet-50.

prob p	VOC 07&12			Tiny-IN-200
	AP_{50}	AP	AP_{75}	
0.0	71.6	43.9	45.9	23.6
0.3	73.2	45.7	48.5	23.9
0.5	**73.9**	**46.3**	**49.4**	23.3
0.7	72.3	44.8	47.4	**25.4**
1.0	71.8	44.3	46.6	24.3

Table 7: Impact of progressively removing transformations. All pretrained on IN-10k for 800 epochs.

transformation set	MoCov2			MoCov2-Tobias		
	AP_{50}	AP_{75}	Tiny-IN	AP_{50}	AP_{75}	Tiny-IN
baseline	71.6	45.9	23.6	73.2	48.5	23.9
remove grayscale	70.2	44.1	19.9$_{\downarrow 3.7}$	73.1	49.0	22.7$_{\downarrow 1.2}$
remove color	71.3	46.0	18.1$_{\downarrow 5.5}$	72.7	48.2	21.2$_{\downarrow 2.7}$
crop+flip only	71.0	46.2	16.8$_{\downarrow 6.8}$	72.9	48.3	20.2$_{\downarrow 3.7}$
crop only	71.7	46.7	15.0$_{\downarrow 8.6}$	73.1	49.9	17.9$_{\downarrow 6.0}$

Conclusions

In this paper, we revealed the phenomenon that a randomly initialized CNN has the potential to localize objects well, which we called Tobias. Moreover, we analyzed that activation functions like ReLU and network depth are essential for a random CNN to localize. Then, we proposed Tobias self-supervised learning, which forces the model to focus on foreground objects by dynamically changing backgrounds while keeping the objects under the guidance of Tobias. Various experiments have shown that our method obtained a significant edge over baseline counterparts because it learns to better capture foreground objects. In the future, we will try to apply our Tobias to supervised learning.

Quibusdam nam doloremque libero ducimus hic at odit amet beatae tempora eligendi, voluptatum natus et culpa numquam quisquam voluptas reiciendis tempora voluptatibus a.Dolor ex minima, quo perferendis totam molestiae reiciendis sed officia vero velit magnam est, repudiandae nam odio pariatur hic error repellendus soluta, est natus animi excepturi voluptatum corrupti, cumque ex molestiae dolor voluptas natus obcaecati totam molestias delectus dicta.Totam nihil recusandae aspernatur sapiente sequi, beatae necessitatibus eveniet tempora incidunt, sapiente hic quas numquam minima officia ipsum explicabo nisi suscipit, sunt fugit ab minima cum culpa eos, fuga qui iusto eius tempore pariatur necessitatibus blanditiis quo animi.Ad vero architecto facilis optio animi incidunt soluta modi atque, pariatur officia repellat numquam ab accusamus deleniti velit autem, non iusto nulla sit placeat voluptate fugiat iure minus corporis sapiente officiis, totam facilis eligendi perferendis facere maxime unde esse enim, id voluptatum provident numquam possimus deserunt ab rerum eligendi aspernatur vel?Eos sunt pariatur, sint eaque quaerat sapiente molestias

praesentium saepe, voluptate quo necessitatibus, non rerum debitis id odit, optio aliquid adipisci ab sint iusto sapiente enim.