Tobias: A Random CNN Sees Objects

Yun-Hao Cao, and Jianxin Wu, *Member, IEEE*

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 theoretically analyzed and empirically verified, and successfully applied to self-supervised learning as well as supervised learning. For 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. Furthermore, we apply Tobias to supervised image classification by letting the average pooling layer focus on foreground regions, which achieves improved performance on various benchmarks.

Index Terms—Convolutional Neural Networks, Randomly Initialized Networks, Object Localization, Self-supervised Learning.

1 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 [1]. 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) [2], [3], [4] and unsupervised object localization (USOL) methods [5], [6] can even localize objects without training on bounding box annotations. All these methods, however, rely on ImageNet [7] pretrained models and non-trivial learning steps.

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. We will then'' provide theoretical analyses to verify our conjecture.

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 [8] and the contrastive learning paradigm, many SSL algorithms have been published, such as MoCo [9], SimCLR [10], BYOL [11], 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 the number of pretraining images, and our method has consistent improvements on different amounts of training data.

Furthermore, we apply Tobias to supervised image classification and we let the average pooling layer in typical CNNs to focus only on foreground locations determined by Tobias. Our method achieves consistent improvements on various classification benchmarks. Our contributions are:

- We find the "Tobias" inductive bias of CNN, i.e., a random CNN can localize objects without any learning.
- We give theoretical analyses on why a random CNN can localize objects. We find that activations like ReLU and network depth are essential for a random CNN to localize.
- We successfully apply Tobias to SSL and achieve consistent improvements on various benchmarks. 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.
- We successfully apply Tobias to supervised image classification and achieve both a faster convergence rate and higher accuracy on various benchmarks.

The rest of this paper is organized as follows. First, we

- *All authors are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. J. Wu is the corresponding author.*
E-mail: {caoyh, wujx}@lamda.nju.edu.cn.

- *This research was partly supported by the National Natural Science Foundation of China under Grant 61772256 and Grant 61921006.*

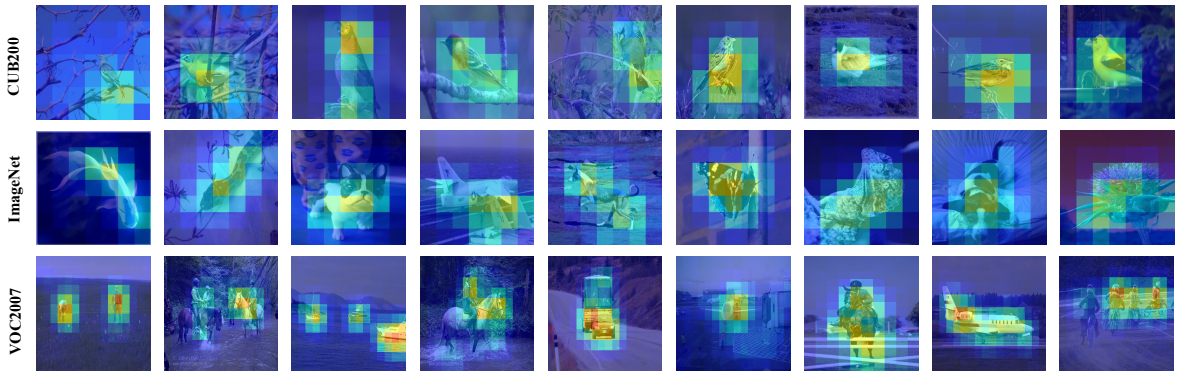


Figure 1: Visualization of localization heatmaps using SCDA [5] for a *randomly initialized* ResNet-50. Best viewed in color when zoomed in.

review the related work in Section 2. Then, we introduce our method in Section 3, and mathematically analyze Tobias in Section 4. Experimental results are reported and analyzed in Section 5. Finally, Section 6 concludes this paper.[1]

2 Related Works

Random networks' potential. [13] 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 [14], [15], [16]. The SSL method BYOL [11] 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 [17] 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 already sees objects well.

Un-/Weakly-supervised object localization. Weakly supervised object localization (WSOL) [3], [4] learns to localize objects with only image-level labels. CAM [2] generated class activation maps with the global average pooling (GAP) layer and the final fully connected (FC) layer. Unsupervised localization methods do not even need image-level labels. SCDA [5] aggregated information through the channel dimension to get localization masks. DDT [6] evaluated the correlation of descriptors. However, they all rely on ImageNet [7] 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 [18]. Many recent works follow the contrastive learning paradigm [8]. SimCLR [10] and MoCo [9] 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 [19] and [20], where Mixup [21] or CutMix [22] 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" [23], [24] 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.

3 Tobias, and SSL with Tobias

We first propose how a randomly initialized CNN localizes objects, but leave the theoretical justification of this surprising phenomenon to the next section. Then, in this section we also propose how to apply Tobias to both self-supervised and supervised learning.

3.1 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 [25] 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 [5] 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,

1. This paper is extended based on our preliminary work [12]. Now we provide theoretical proofs on our conjecture about why Tobias works, add discussions about various initialization schemes and more visualizations, supply more baselines, ablation studies and transfer learning experiments for Tobias SSL, and extend Tobias on the supervised learning task.

$A = \sum_{n=1}^{d} S_n$. Then, a mask map M of the same size as A is obtained by

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

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 [5] 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 following sections.

3.2 Tobias self-supervised learning

Based on our finding that an un-trained random network can capture foreground objects 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 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 to merge these patches.

Applying Tobias to SSL. We now apply Tobias to the contrastive learning paradigm following the notations in SupCon [26]. 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 N 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 [9], 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 an 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:

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

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.

3.3 Tobias supervised learning

Now we further apply our Tobias finding into supervised learning (image classification) and we modify the GAP layer in typical CNNs. As noted in Sec. 3.1, the outputs before the GAP layer are formulated as an order-3 tensor $Q \in \mathbb{R}^{h \times w \times d}$. Then, after GAP we can get $G = \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} Q_{i,j} \in \mathbb{R}^d$. We propose to average over only foreground locations,

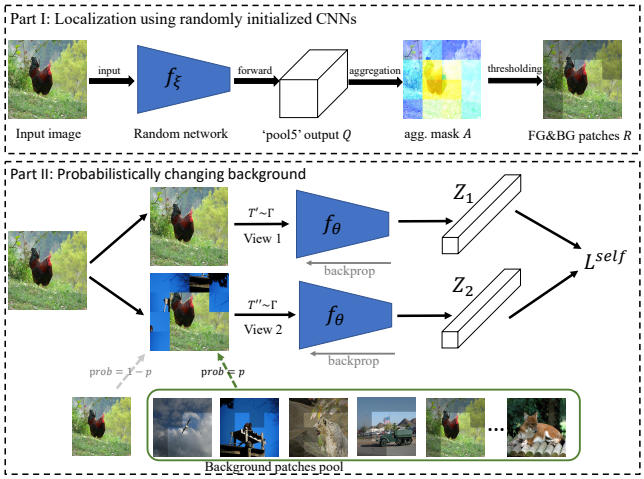


Figure 2: Pipeline of Tobias SSL. Upper: splitting foreground and background using a *randomly initialized* CNN. Lower: applying Tobias augmentation into SSL.

i.e., those with $M_{i,j} = 1$. Under this formulation, the CNN is encouraged to utilize only foreground regions for classification. The mask map M is dynamically generated by Q during training while n Sec. 3.2 M was pre-calculated using a random network. Also notice that M is generated by using a threshold λ, where λ equals the mean value in Sec. 3.1 and median in Sec. 3.2. We will study different strategies of the threshold in Sec. 5.3.

We name this pooling strategy as Tobias average pooling (TAP) and it can be easily implemented by adding an extra element-wise multiplication operation before the GAP layer:

$$Q^n_{i,j} = Q^n_{i,j} \times M_{i,j}\,, \tag{7}$$

where n denotes the channel index ($n = 1, \ldots, d$).

4 Theoretical and Empirical Analyses

Randomly initialized networks are more convenient to analyze than trained ones. In this case, all the weights are i.i.d. variables and we assume the input image is constant and only focus on the behavior of random networks.

Now we will mathematically analyze why a randomly initialized CNN can localize objects. In order to facilitate the description, we first give some assumptions and notations used in this section. Assume the pixels are indexed by (x, y) and the layers are indexed by p. Let $f(x, y)$ and $f^p(x, y)$ denote the value at location (x, y) of the input image and the output of the p-th layer, respectively. We start by the simple the case of n convolutional layers using $k \times k$ kernels with stride one, and only one single channel in each layer, i.e.,

$$f^p(x,y) = h\Big(\sum_{i=0}^{k-1}\sum_{j=0}^{k-1} w^p_{i,j} f^{p-1}(x+i, y+j)\Big)\,, \tag{8}$$

where w^p_{ij} denotes the weight of the convolution filter at the p-th layer and $h(\cdot)$ denotes the activation function. w^p_{ij} are i.i.d. variables and we assume $w^p_{ij} \sim \mathcal{N}(0, \sigma^2)$ unless otherwise specified. When referring to $E[\cdot]$ and $\mathrm{Var}[\cdot]$, we mean the expectation and variance w.r.t. many different sampled random networks (i.e., w) for the same input.

First, the following Claim 1 says that the first convolution layer acts as an edge detector and regions with higher image gradients are expected to have larger activations after ReLU.

Claim 1. Assume $k = 2$, $h(x) = \max(0, x)$ (ReLU activation) and $\sum_{i,j} w^1_{i,j} = 0$, then $E[f^1(x, y)] = \frac{\sqrt{|g_x|+|g_y|+|g_{xy}|}}{\sqrt{2\pi}}\sigma$, where g_x, g_y and $g_{x,y}$ are the image gradient along the vertical, horizontal and diagonal direction, respectively.

Next, the following Claim 2 states that the receptive filed [27] expands along with the increase of depth, and the influence (variance of values at different spatial locations) decays quickly from the center. Hence, a deep linear convolutional network can capture features from local to global along with the increase of depth, and can find larger regions containing richer features (e.g., edges in Claim 1).

Claim 2. Assume $k = 2$, $h(x) = x$ (linear activation) and the depth p is large, then

$$f^p(x,y) = \sum_{s=0}^{p}\sum_{t=0}^{p} W^p_{s,t} f(x+s, y+t)\,,$$

in which $W^p_{s,t} \sim N(0, \frac{C_{s,t}}{p}\sigma^2)$ and the constant $C_{s,t} \approx 2^{2p}\frac{1}{2\pi p\times\frac{1}{4}} e^{-\frac{(s-\frac{p}{2})^2+(t-\frac{p}{2})^2}{2p\times\frac{1}{4}}}$.

Then, the following Claim 3 says that the expectation of the output equals 0 without non-linearity. Finally, Claim 4 and Claim 5 explain that regions with large values are expected to remain a large value for unbounded activation functions (e.g., ReLU) but not for bounded activations (e.g., sigmoid). In other words, regions with larger values (textures, edges, and potential objects) are likely to stay prominent after ReLU but will be suppressed for sigmoid.

Claim 3. Suppose $h(x) = x$ (no activation), then

$$E_w[f^p(x, y)] = 0\,, \tag{9}$$

$$\mathrm{Var}[f^p(x, y)] = T\sigma^2\,, \tag{10}$$

where $T = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} |f^{p-1}(x+i, y+j)|$.

Claim 4. Suppose $h(x) = \max(0, x)$ (ReLU activation), then

$$E_w[f^p(x, y)] = \sqrt{\frac{T}{2\pi}}\sigma\,, \tag{11}$$

$$\mathrm{Var}[f^p(x, y)] = (\frac{1}{2} - \frac{1}{2\pi})T\sigma^2 \approx 0.34T\sigma^2\,, \tag{12}$$

where $T = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} f^{p-1}(x+i, y+j)$.

Claim 5. Suppose $h(x) = \frac{1}{1+e^{-x}}$ (sigmoid activation), then $\frac{1}{4} < E_w[f^p(x, y)] < \frac{3}{4}$.

In conclusion, Claim 1 and Claim 2 show that a random CNN can capture local edges (with larger activations values) as well as global features (with increasing receptive field sizes). Then, Claim 3∼Claim 5 show that unbounded activations like ReLU can help regions with large activations stay prominent. Hence, we can conclude that regions with higher activations in the final feature map are more likely to contain objects. More results and analyses will be shown in Section 5.1.

All proofs of Claim 1∼Claim 5 are provided in the appendix to this paper.

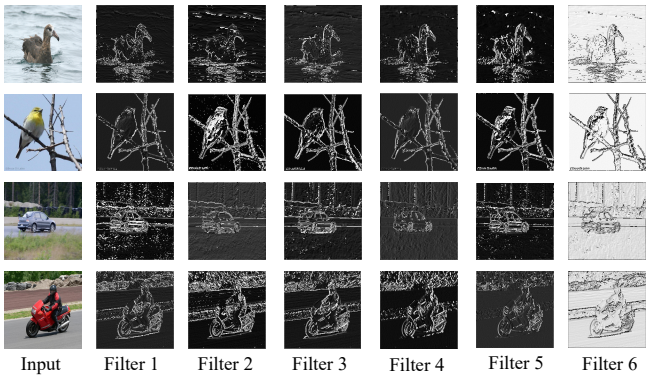


Figure 3: Visualization of the output of one 3×3 convolution filter when $C = 0$. In this case, *random* convolution filters act as edge detectors.

4.1 Empirical Analysis

We then conduct empirical experiments as a supplement to the above theoretical analyses.

Empirical results for Claim 1. We visualize the outputs of one 3×3 convolution filter, as shown in Figure 3. The weights $w_{ij} \sim \mathcal{N}(0, 0.1)$ and after initialization we set $w_{ij} \leftarrow w_{ij} - \mu$ to let $C = 0$, where $\mu = \frac{1}{9} \sum w_{i,j}$. As can be seen, in this case convolution filters act as edge detectors in different directions. Notice that sometimes the brightness will be reversed (the last column) because the sign of the output may be negative and be deactivated after ReLU. It is worth noting that we remove the constraint of $C = 0$ in all subsequent experiments in Section 5 and we can also get similar good results.

Empirical results for Claim 3∼Claim 5. Now we fix our input image to better understand the behavior of randomly initialized networks. We generate L ($L = 100$ in our experiments) randomly initialized networks and analyze the mean and standard deviation of the output at different depth. Let us denote the output at the (x, y) position of the p-th layer of the l-th model as $f_l^p(x, y)$. We calculate the mean and standard deviation at different locations of different depth as follows:

$$U^p(x,y) = \frac{1}{L} \sum_{l=1}^{L} f_l^p(x,y) \,,$$

$$S^p(x,y) = \sqrt{\frac{1}{L} \sum_{l=1}^{L} [f_l^p(x,y) - U^p(x,y)]^2} \,.$$

We visualize the heatmap of $U^p(x, y)$ (normalized to $[0, 1]$) using one example image in Figure 4. We can see that ReLU can preserve regions with large activations (potential objects) but it is not the case for linear and sigmoid.

Moreover, we average over spatial locations for $S^p(x, y)$ to get S^p and plot it in Figure 5:

$$S^p = \frac{1}{H \times W} \sum_{x,y} S^p(x,y) \,.$$

When comparing Equation (10) and (12), we can see that ReLU has an extra $(\frac{1}{2} - \frac{1}{2\pi})^p$ constant factor multiplier on the variance. As can be seen from Figure 5, ReLU activation effectively reduces the variance. This can partly explain the robustness of the localization results in Sec. 5.1.

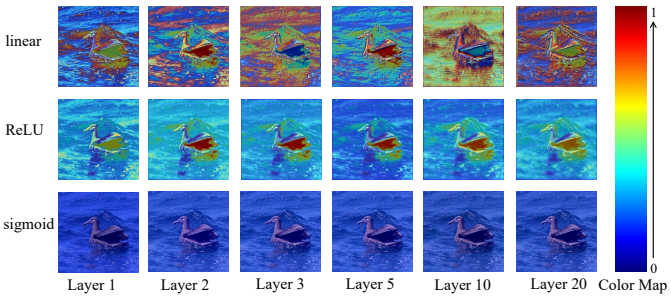
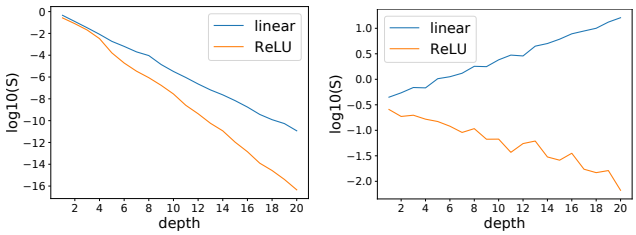


Figure 4: Visualization of the heatmap of $U^p(x, y)$.



(a) 1 channel (b) 16 channels

Figure 5: Average standard deviation S^p (in $\log_{10}$) w.r.t. depth with different number of channels and activations.

5 Experimental Results

We use CUB-200 [28] and ImageNet [7] 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. Then, we study the effects of different components and hyper-parameters and sensitivity to data augmentations in our algorithm. Finally, we apply our Tobias into supervised image classification tasks. All our experiments were conducted using PyTorch [29] and we used 8 Titan Xp GPUs for our experiments.

5.1 Localization ability of random CNNs

We start by studying the localization ability of randomly initialized CNNs. We use SCDA [5] for localization and conduct experiments on two popular datasets for object localization, i.e., ImageNet and CUB-200. Notice that [5] used 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 [30] 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 [31] in Figure 1. The heatmap in Figure 1 is calculated by the 2-D aggregation mask A, as noted in Sec. 3.1.

As shown in Table 1, a randomly initialized ResNet-50 (R-50) [25] achieves comparable localization accuracies with its ImageNet supervised counterpart on both ImageNet

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 [5]	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 [2]	R-50	33 / 5	52.9	33.8±0.1	50.0	26.0±0.3

and CUB-200. We also present one popular WSOL method, CAM [2], 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 (also see the appendix—there is 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 [32] and there is a large gap between pretrained and randomly initialized ViT models. Hence, we can conclude that it is one inductive bias for CNNs, but not for MLP-based architectures such as ViT.

But, why can a random CNN see objects without any learning? Given the theoretical and 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 texture-less when compared to the objects, and texture-less regions have higher chances to be deactivated by ReLU as the increase of network depth.* In addition to the theoretical analyzes in Sec. 4, we also 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 (cf. appendix for visualization). Second, network depth is also important and we can observe a significant performance

Figure 6: Ablation of network depth for randomly initialized ResNet-50 using SCDA on ImageNet.

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	VGG-11	40.0±0.5	30.6±1.4
11	VGG-16	40.8±0.5	33.5±1.9
12	VGG-16 (sigmoid)	39.8±0.6	18.2±1.3
13	VGG-16 (arctan)	34.6±0.5	20.1±1.0
14	VGG-16 (ELU)	40.4±1.0	32.5±1.0
15	VGG-19	41.4±1.8	34.2±0.3
16	AlexNet	34.6±1.5	24.8±0.3
17	Inception v3	52.2±0.6	49.6±0.9
18	Hourglass	52.6±0.2	46.9±0.4
19	EdgeBox	31.8	32.7
20	lower bound (whole image)	38.8	19.1
21	upper bound (faster R-CNN)	58.9	96.2

degradation as the network depth decreases (i.e., fewer stages). Visualizations in Figure 6 more intuitively show that shallow layers (e.g., 'conv1' and 'conv1-2') are confused by edges in backgrounds (such as grass or floor) but the full network can successfully see the whole object.

To make our conclusions more convincing, we add more baselines and further investigate different components in

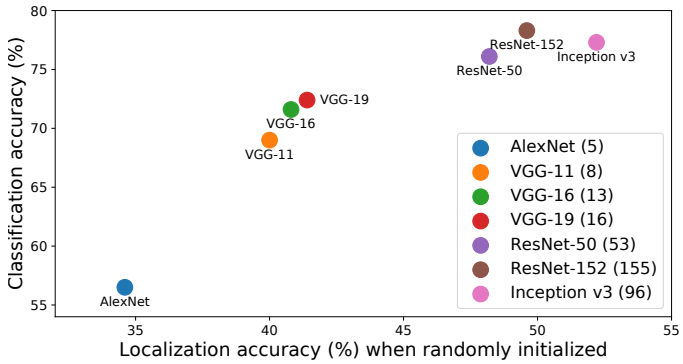


Figure 7: Classification accuracy after training (PyTorch model zoo) versus localization accuracy (when randomly initialized) on ImageNet. The numbers in brackets represent the number of convolutions in the models (i.e., depth).

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, which is 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 [33], which is directly supervised on the detection task COCO [34]. We also compare with the object proposal method EdgeBox [35]. 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) [36] 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 losses 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 losses 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 unbounded activations, e.g., ELU, SELU and softplus (row 7∼9), we can get comparable or even better results than ReLU. All these activation functions have one thing in common: deactivate negative values and unbounded for positive values. This is consistent with our previous theoretical analyses Claim 3 and Claim 4.

Other CNN architectures. Other randomly initialized CNN architectures can also localize objects well. We study AlexNet [37], VGG-style networks [38], Hourglass [17] and Inception v3 [39]. 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 10∼15). 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. 7, we rank the networks 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 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. Notice that we used PyTorch default initialization method Kaiming normal initialization before. Now we investigate the effect of other initialization methods, i.e., Kaiming uniform, Xavier normal, Xavier uniform, Normal(0,0.1), Normal(0,1), Uniform(-0.1,0.1), Uniform(-1,1) and zero init for experiments, and the localization results are shown in Table 3. We can observe that other initialization methods, e.g., Kaiming uniform, Normal(0,0.1) and Uniform(-0.1,0.1), can also get good and robust localization results and Tobias is actually a general phenomenon. However, we can not expect a good localization result with inappropriate initialization methods. For instance, when we set all the weights to zero, we will get all zero activation map and zero localization accuracy (it will also destroy supervised training with this bad initialization). Also, as discussed in [30], the linear assumption in Xavier init. [40] is invalid for ReLU and we can observe degraded localization accuracy here.

Also, we design an experiment to investigate the relationship between the localization accuracy of a random network before any training and its classification accuracy after training. We train on CIFAR-10 for 60 epochs using ResNet-50 with different initialization methods as aforementioned. We can observe that the localization accuracy of a random network can somehow indicate its classification accuracy after training (i.e., has high correlation). For instance, Normal(0,1) has much worse localization performance than Normal(0,0.1) (without any training using SCDA) and it also gets lower accuracy after training on CIFAR-10. Also,

Table 3: Comparisons of localization accuracy (without training) using SCDA on ImageNet and CUB-200, and classification accuracy (after training) on CIFAR-10 using different initialization methods. We report the average accuracy and standard deviation of 3 trials.

Initialization method	Loc acc.		Cls acc.
	ImageNet	CUB-200	CIFAR-10
Kaiming normal (default)	48.2±0.6	41.8±0.6	83.4±0.3
Kaiming uniform	49.2±0.6	43.3±0.6	82.7±0.7
Xavier normal	42.2±1.1	32.6±0.9	80.9±0.6
Xavier uniform	41.3±1.0	33.3±0.6	80.4±0.4
Normal(0,0.1)	50.6±0.3	43.4±0.5	82.8±0.8
Normal(0,1)	0.0*	0.0*	77.4±0.8
Uniform(-0.1,0.1)	50.0±0.4	43.4±0.4	82.3±0.3
Uniform(-1,1)	0.0*	0.0*	79.5±0.5
Zero init	0.0*	0.0*	10.0±0.0

* When using Normal(0,1) or Uniform(-1,1) for initialization, the network fails to localize objects (obtain nearly all zero activation map and the localization accuracy is actually 0).

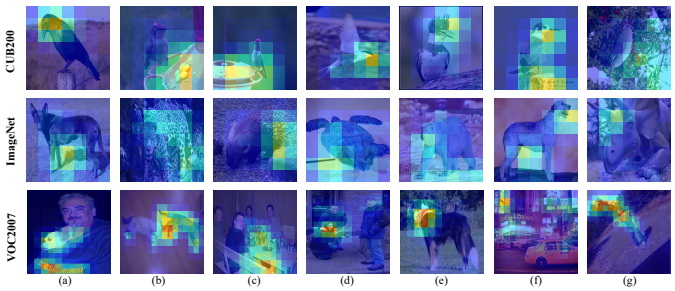


Figure 8: Visualization of some failure cases using SCDA for a *randomly initialized* ResNet-50. Best viewed in color.

Xavier initialization performs worse on localization than Kaiming initialization and lower classification accuracy after training is observed. In other words, we observe that a random network that sees objects better is more likely to perform better for classification after training.

Failure case analysis. Here we present and analyze some failure cases on these 3 datasets in Figure 8. For CUB-200, the random network focuses only on the head of birds in column (a), (d) and (e) while gets confused by complex backgrounds in column (b) and (g). For VOC2007, the random network does not localize all objects when there are multiple objects in column (a) (miss people), column (b) (miss the other cat), column (c) (miss people) and column (d) (miss other people).

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.

5.2 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.

5.2.1 Results on CUB-200

We carefully study our Tobias using 2 typical SSL methods, namely MoCov2 [41] and SimCLR [10] under both ResNet-18 and ResNet-50. We follow the training and evaluation protocols in [42] 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 α is set to 1.0. For the SSL pretraining stage, we follow the same settings in the original papers and more details are included in the appendix. '-Tobias' denotes our method and we only change the data loading process and other training settings remain the same as baseline methods for fair comparison.

Table 4: 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 method	epochs	Fine-tuning accuracy (%) 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	200	64.4 (+0.7)	66.3 (+0.5)
	MoCov2	800	65.0	66.3
	MoCov2-Tobias	800	66.2 (+1.2)	67.7 (+1.4)
	SimCLR	200	63.6	64.5
	SimCLR-Tobias	200	65.4 (+1.8)	68.6 (+4.1)
	SimCLR	800	66.0	67.3
	SimCLR-Tobias	800	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	200	63.6 (+7.4)	62.0 (+9.0)
	MoCov2	800	66.5	62.0
	MoCov2-Tobias	800	67.2 (+0.7)	71.5 (+9.5)
	SimCLR	200	68.0	66.5
	SimCLR-Tobias	200	68.4 (+0.4)	71.7 (+5.2)
	SimCLR	800	69.2	73.0
	SimCLR-Tobias	800	70.0 (+0.8)	73.6 (+0.6)

The results are shown in Table 4. 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 v.s. 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 later.

5.2.2 Results on ImageNet

Now we move on to the large-scale dataset ImageNet. We use MoCv2 for illustration following the official training protocols in [41]. 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 investigate the downstream object detection performance on Pascal VOC 07&12 [31] in Table 5. We carefully investigate the downstream object detection performance on COCO2017 [34] and Pascal VOC 07&12 in Table 5. The detector is Faster R-CNN [33] with a backbone of R50-FPN [43] or R50-C4 [44] for Pascal VOC, and Mask R-CNN [44] with R50-FPN backbone for COCO, implemented in [45]. Also, we experimented on 8 downstream classification benchmarks following [42], with results in Table 7.

As shown in Table 5, Tobias achieves better performance than baseline MoCov2 on both COCO and Pascal VOC. Also notice that our Tobias even achieves slightly better performance than MoCov2 800ep (pretrained much longer) on Pascal VOC. Also, Table 7 shows that although our Tobias achieves slightly lower accuracy in ImageNet linear evaluation, it outperforms MoCov2 baseline on 6 out of 8

Table 5: Left: Object detection and instance segmentation on COCO, showing bounding-box AP (AP^{bb}) and mask AP (AP^{mk}) evaluated on val2017; Right: Object detection on PASCAL VOC trainval07+12 (default VOC metric AP_{50}, COCO-style AP, and AP_{75} evaluated on test2007).

pretraining method	R50-FPN (1x) AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	31.0	49.5	33.2	28.5	46.8	30.4
IN supervised	38.4	59.2	41.6	35.0	55.9	37.1
MoCov2 200ep	39.0	59.5	42.4	35.6	56.6	38.0
MoCov2-Tobias 200ep	**39.3**	**59.9**	**42.7**	**35.8**	**56.9**	**38.4**
MoCov2 800ep	39.5	59.8	43.2	36.0	56.9	38.6

(a) COCO2017

R50-FPN (24k) AP_{50}	AP	AP_{75}	R-50 C4 (24k) AP_{50}	AP	AP_{75}
63.0	36.7	36.9	60.2	33.8	33.1
80.8	53.5	58.4	81.3	53.5	58.8
81.8	55.0	60.5	82.2	57.1	64.5
82.0	**55.5**	**61.1**	**82.6**	**57.7**	**64.9**
81.5	55.0	61.0	82.6	57.7	64.5

(b) Pascal VOC 07&12

Table 6: 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 method	#imgs	#eps	VOC 07&12 AP_{50}	AP_{75}	Tiny-IN-200
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-CutMix			71.8 ↓1.4	47.1 ↓1.4	23.2 ↓0.7
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-CutMix			77.0 ↓0.9	53.0 ↓1.9	39.7 ↓1.0
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)

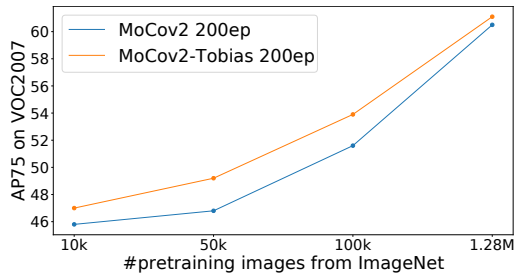


Figure 9: Performance of Tobias on Pascal VOC (AP_{75}) with respect to different training data size.

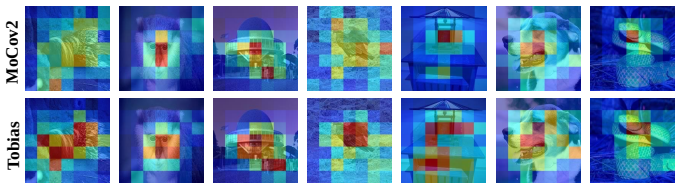


Figure 10: Visualizations of heatmaps using SCDA for MoCov2 and MoCov2-Tobias pretrained models.

downstream classification benchmarks in linear evaluation and 7 out of 8 in fine-tuning.

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 [42]. 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 6 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 6 and Figure 9, 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. We also visualize the heatmaps of MoCov2 and MoCov2-Tobias pretrained models in Figure 10. We compare the checkpoints pretrained on IN-100k for 800 epochs. We can see that Tobias will give higher scores (more red color) on objects and it further demonstrates that our Tobias can help the network focus on foreground objects.

5.2.3 *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 6). 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. As can be seen, CutMix achieves slightly worse performance when compared with the 'RM'

Table 7: Transfer learning results from ImageNet with the standard ResNet-50 architecture.

Method	ImageNet	VOC2007	CUB200	Cars	Aircrafts	Caltech-101	Flowers	Dogs	DTD
Linear evaluation:									
MoCov2 200ep	**67.7**	80.6	17.8	14.1	12.3	80.8	68.5	42.1	64.9
MoCov2-Tobias 200ep	67.4↓	**82.0**↑	18.2↑	12.3↓	12.9↑	81.7↑	70.1↑	42.9↑	63.9↓
IN supervised	-	73.9	**61.7**	**47.1**	**23.7**	**89.1**	**86.9**	**82.2**	**68.2**
Fine-tuned:									
MoCov2 200ep	73.9	85.6	75.5	89.2	86.5	89.2	95.7	76.6	68.6
MoCov2-Tobias 200ep	74.0↑	86.3↑	76.1↑	89.5↑	**87.8**↑	90.7↑	96.2↑	77.5↑	68.2↓
IN supervised	**76.1**	**89.0**	**81.3**	**90.6**	86.7	**93.0**	**96.7**	**80.1**	**74.7**

Table 8: Effect of the 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 9: 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	$17.9_{\downarrow 6.0}$

baseline (also our Tobias). We also compare with MoCov2-Mixup where we use mixup in 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 6, 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, -1.4 AP_{50} for CutMix 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 [19].

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, as shown in Table 8. 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 hyper-parameter p (see appendix).

Sensitivity to image augmentations. Now we study the sensitivity to image augmentations of our Tobias by progressively removing transformations in the transformation set following [11]. The results in Table 9 show that the performance of Tobias is much less affected than the performance of MoCov2 when removing the color distortion, 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 10: Effect of the number of patches. All settings are pretrained on IN-50k for 200 epochs using ResNet-50.

Method	VOC 07&12			Tiny-IN-200
	AP_{50}	AP	AP_{75}	
MoCov2	72.2	44.7	46.8	26.3
MoCov2-Tobias (4×4)	73.7	46.0	49.2	26.0
MoCov2-Tobias (8×8)	74.3	46.7	50.0	26.3
MoCov2-Tobias (16×16)	**74.9**	**47.4**	**50.6**	**28.5**

Effect of the number of patches. Notice that all the experiments were done using a 4 x 4 patch grid in our Tobias SSL, as noted in Sec. 3.2. To better leverage localization, we conduct experiments by increasing the size of grids, e.g., to 8x8 and 16x16. As can be seen in Table 10, we can achieve better results when increasing the size of grids, which further proves the localization ability of random CNNs.

5.3 Tobias supervised learning

Now we apply Tobias to supervised learning (Equation 7) and evaluate its effectiveness on 5 classification benchmarks.

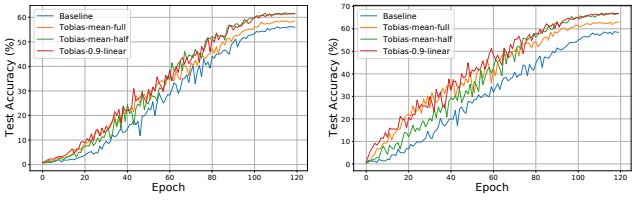
We conduct experiments for both randomly initialized models (i.e., train from scratch) and ImageNet pretrained models (i.e., fine-tune from IN supervised). When randomly initialized, we train the network for 120 epochs with a batch size of 64 and a weight decay of 5e-4. The learning rate starts from 0.1 with cosine learning rate decay. For ImageNet pretrained fine-tuning, we train for 60 epochs with lr initialized to 0.01, which is divided by 10 every 20 epochs.

We design different strategies on the threshold λ for TAP:

(1) Tobias-mean-full: this means that λ is set to the mean value $\bar{a}$ throughout the whole training process.
(2) Tobias-mean-half: this means that λ is set to the mean value in the first half of the training epochs and then normal GAP is used in the second half (i.e., $\lambda = -\infty$).
(3) Tobias-β-linear: this means λ equals the βth percentile at the beginning, where $\beta \in [0, 1]$ is a hyper-parameter. Then, β is linearly reduced to 0 at the last epoch following linear decay. Take 'Tobias-0.9-linear' as an example, the top 10% locations are selected for average pooling at the beginning and more and more areas are engaged as

Table 11: Supervised fine-tuning results with the standard ResNet-50 architecture.

Method	CUB200	Caltech-101	Flowers	Dogs	Pets
Random init.:					
Baseline	55.9	62.5	58.4	62.8	51.6
Tobias-mean-full	58.6	60.2	62.9	67.2	62.4
Tobias-mean-half	**61.6**	67.0	66.7	67.6	65.4
Tobias-0.9-linear	61.5	**67.8**	**66.8**	**69.8**	**71.6**
IN supervised:					
Baseline	81.3	93.0	**96.7**	80.1	90.0
Tobias-0.9-linear	**82.5**	**94.2**	96.5	**81.8**	**91.4**



(a) CUB200 (b) Flowers

(c) Dogs (d) Pets

Figure 11: Test accuracy curve with different methods on 4 datasets (random init.). This figure is best viewed in color.

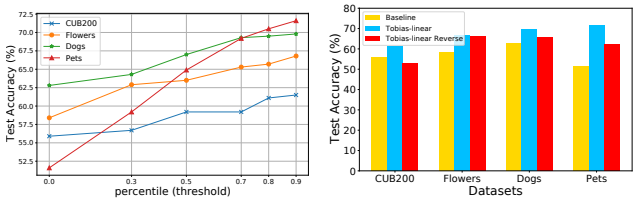
the increase of epochs (from local to global). Finally, the network can utilize all locations during average pooling. (i.e., TAP becomes GAP when β becomes 0).

The results are shown in Table 11 and the test accuracy curves are in Figure 11. All variants of our TAP method achieves faster convergence and higher accuracy than the baseline GAP. It indicates that this kind of local average pooling is effective (especially for training from scratch) and the localization mask provided by a randomly initialized network is vital. Moreover, we can see that 'Tobias-mean-half' achieves higher accuracy than 'Tobias-mean-full'. It indicates that it is beneficial to include more regions in the later training process. In summary, our 'Tobias-0.9-linear' achieves the best results and it can even improve the baseline GAP when using ImageNet pretrained models.

Now we investigate the effect of different components in our Tobias-β-linear strategy. First, we vary β from 0 to 0.9, as shown in Figure 12a. Notice that when $\beta = 0$, TAP becomes GAP and we can observe that the accuracy increases as β increase. Then, we further design a 'Tobias-β-linear-reverse' strategy for comparison, i.e., M is generated from $-Q$. In this case, when $\beta = 0.9$, 10% locations with the *lowest* scores will be included at the beginning. The results are shown in Figure 12b and we can see that our 'Tobias-β-linear' achieves higher accuracy than the reversed counterpart consistently, which further verifies the effectiveness of Tobias.

6 Conclusions

In this paper, we revealed the phenomenon that a randomly initialized CNN has the potential to localize objects well,



(a) Effect of percentile β (b) Effect of order ($\beta = 0.9$)

Figure 12: Ablation study on our Tobias-β-linear fine-tuning strategy on different datasets (random init.). Left: effect of the percentile hyper-parameter. Right: effect of the order.

which we called Tobias. Moreover, we analyzed that activation functions like ReLU and network depth are essential for a random CNN to localize from both empirical and theoretical perspectives. 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. Furthermore, we applied Tobias to supervised image classification by letting the average pooling layer focus on foreground regions, which also achieves superior performance. In the future, we will try to explore other applications of Tobias and further explore the potential of random networks.

References

[1] N. Cohen and A. Shashua, "Inductive bias of deep convolutional networks through pooling geometry," in *The International Conference on Learning Representations*, 2017, pp. 1–28. 1

[2] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929. 1, 2, 6

[3] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *The IEEE International Conference on Computer Vision*, 2017, pp. 618–626. 1, 2

[4] C.-L. Zhang, Y.-H. Cao, and J. Wu, "Rethinking the route towards weakly supervised object localization," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 460–13 469. 1, 2

[5] X.-S. Wei, J.-H. Luo, J. Wu, and Z.-H. Zhou, "Selective convolutional descriptor aggregation for fine-grained image retrieval," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2868–2881, 2017. 1, 2, 3, 5, 6

[6] X.-S. Wei, C.-L. Zhang, J. Wu, C. Shen, and Z.-H. Zhou, "Unsupervised object discovery and co-localization by deep descriptor transformation," *Pattern Recognition*, vol. 88, pp. 113–126, 2019. 1, 2

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. 1, 2, 5

[8] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018. 1, 2

[9] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738. 1, 2, 3

[10] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *The International Conference on Machine Learning*, 2020, pp. 1597–1607. 1, 2, 8

[11] J.-B. Grill, F. Strub, F. Altche, C. Tallec, P. H.Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Boostrap your own latent: A new approach to self-supervised learning," in *Advances in Neural Information Processing Systems*, 2020, pp. 21 271–21 284. 1, 2, 10

[12] Y.-H. Cao and J. Wu, "A random CNN sees objects: One inductive bias of CNN and its applications," in *The 36th AAAI Conference on Artificial Intelligence*, 2022, p. in press. 2

[13] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *The International Conference on Learning Representations*, 2019, pp. 1–13. 2

[14] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, M. Carbin, and Z. Wang, "The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 306–16 316. 2

[15] S. Girish, S. R. Maiya, K. Gupta, H. Chen, L. Davis, and A. Shrivastava, "The lottery ticket hypothesis for object recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 762–771. 2

[16] E. Malach, G. Yehudai, S. Shalev-Schwartz, and O. Shamir, "Proving the lottery ticket hypothesis: Pruning is all you need," in *The International Conference on Machine Learning*, 2020, pp. 6682–6691. 2

[17] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9446–9454. 2, 7

[18] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 4037–4058, 2021. 2

[19] Z. Shen, Z. Liu, Z. Liu, M. Savvides, T. Darrell, and E. Xing, "Un-mix: Rethinking image mixtures for unsupervised visual representation learning," *arXiv preprint arXiv:2003.05438*, 2020. 2, 10

[20] X. Chu, X. Zhan, and X. Wei, "Beyond single instance multi-view unsupervised representation learning," *arXiv preprint arXiv:2011.13356*, 2020. 2

[21] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *The International Conference on Learning Representations*, 2018, pp. 1–13. 2

[22] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *The IEEE International Conference on Computer Vision*, 2019, pp. 6023–6032. 2

[23] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *The IEEE International Conference on Computer Vision*, 2019, pp. 1301–1310. 2

[24] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D.Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," *arXiv preprint arXiv:2012.07177v1*, 2012. 2

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. 2, 5

[26] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems*, 2020, pp. 18 661–18 673. 3

[27] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4905—-4913. 4, 15

[28] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011. 5

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035. 5

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *The IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034. 5, 7

[31] M. Everingham, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010. 5, 8

[32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *ICLR*, pp. 1–9, 2021. 6

[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017. 7, 8

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *The European Conference on Computer Vision*, ser. LNCS. Springer, 2014, vol. 8693, pp. 740–755. 7, 8

[35] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *The European Conference on Computer Vision*, ser. LNCS. Springer, 2014, vol. 8693, pp. 391–405. 7

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *The International Conference on Machine Learning*, 2015, pp. 448–456. 7

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105. 7

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *The International Conference on Learning Representations*, 2015, pp. 1–14. 7

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826. 7

[40] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, 2010, pp. 249–256. 7

[41] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020. 8, 16

[42] Y.-H. Cao and J. Wu, "Rethinking self-supervised learning: Small is beautiful," *arXiv preprint arXiv:2103.13559*, 2021. 8, 9

[43] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2177–2125. 8

[44] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *The IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969. 8

[45] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019. 8

Yun-Hao Cao received his BS degree in Computer Science and Technology from Nanjing University. He is currently a Ph.D. student in the Department of Computer Science and Technology in Nanjing University, China. His research interests are computer vision and machine learning.

Jianxin Wu received his BS and MS degrees from Nanjing University, and his PhD degree from the Georgia Institute of Technology, all in computer science. He is currently a professor in the School of Artificial Intelligence at Nanjing University, China, and is associated with the State Key Laboratory for Novel Software Technology, China. He has served as an (senior) area chair for CVPR, ICCV, ECCV, AAAI and IJCAI, and as an associate editor for the IEEE Transactions on Pattern Analysis and Machine Intelligence. His research interests are computer vision and machine learning.

APPENDIX A
PROOF OF CLAIM 1

Before the formal proof of claims, we prove Lemma A.1 that will be used later.

Lemma A.1. Suppose $X \sim N(0, \sigma^2)$, i.e., the probability density function $p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$, and let $Y = h(x) = \max(0, x)$. Then, $E(Y) = \frac{\sigma}{\sqrt{2\pi}}$, $\mathrm{Var}(Y) = (\frac{1}{2} - \frac{1}{2\pi})\sigma^2$.

Proof. Using the fact that $\int_0^{+\infty} e^{-ax^2} x \, \mathrm{d}x = \frac{1}{2a}$, we have:

$$\begin{aligned} E(Y) =& \int_{-\infty}^{+\infty} h(x)p(x) \, \mathrm{d}x \\ =& \int_{-\infty}^{0} 0 \times p(x) \, \mathrm{d}x + \int_{0}^{+\infty} xp(x) \, \mathrm{d}x \\ =& \frac{1}{\sqrt{2\pi}\sigma} \int_0^{\infty} x e^{-\frac{x^2}{2\sigma^2}} \, \mathrm{d}x \\ =& \frac{\sigma}{\sqrt{2\pi}} , \end{aligned}$$

and

$$\begin{aligned} \mathrm{Var}(Y) =& E(Y^2) - E^2(Y) \\ =& \int_{-\infty}^{+\infty} [h(x)]^2 p(x) \, \mathrm{d}x - \frac{\sigma^2}{2\pi} \\ =& \int_{-\infty}^{0} 0 \times p(x) \, \mathrm{d}x + \int_{0}^{+\infty} x^2 p(x) \, \mathrm{d}x - \frac{\sigma^2}{2\pi} \\ =& \frac{1}{2} \int_{-\infty}^{+\infty} x^2 p(x) \, \mathrm{d}x - \frac{\sigma^2}{2\pi} \\ =& (\frac{1}{2} - \frac{1}{2\pi})\sigma^2 . \end{aligned}$$

□

Then, we will give formal proofs of the claims and we assume that we use Gaussian distribution $\mathcal{N}(0, \sigma^2)$ for random initialization of the weights.

We start from Claim 1.

Proof. Without loss of generality, we first consider 2×2 convolution filter of depth 1 and we omit the superscript '1' for $w_{i,j}^1$ for simplicity:

$$W = \begin{bmatrix} w_{00} & w_{01} \\ w_{10} & w_{11} \end{bmatrix} \in \mathbb{R}^{2\times 2} , \tag{13}$$

where w_{ij} are i.i.d. variables and we assume $w_{ij} \sim \mathcal{N}(0, \sigma^2)$. Suppose the input image is f, and we use $f(x, y)$ to denote the pixel value at position (x, y), where $x \in [0, H-1], y \in [0, W-1]$ and H, W are the height and width of f, respectively. The image gradient at (x, y) can be written as:

$$\nabla f(x, y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} f(x+1, y) - f(x, y) \\ f(x, y+1) - f(x, y) \end{bmatrix} , \tag{14}$$

where g_x and g_y represent the vertical and horizontal gradients, respectively. Also, we denote the gradient along the diagonal as:

$$\begin{aligned} g_{xy} &= f(x+1, y+1) - f(x, y) && (15) \\ &= f(x+1, y+1) - f(x+1, y) + f(x+1, y) - f(x, y) && (16) \\ &= g_{y+1} + g_x . && (17) \end{aligned}$$

If we convolve W with the 2×2 square starting at (x, y), we use f_w^1 to denote the output after the first convolution:

$$\begin{aligned} f_w^1(x, y) =& w_{00} f(x, y) + w_{01} f(x, y+1) \\ &+ w_{10} f(x+1, y) + w_{11} f(x+1, y+1) && (18) \\ =& w_{01} \Big(f(x, y+1) - f(x, y) \Big) \\ &+ w_{10} \Big(f(x+1, y) - f(x, y) \Big) \\ &+ w_{11} \Big(f(x+1, y+1) - f(x, y) \Big) \\ &+ (w_{00} + w_{01} + w_{10} + w_{11}) f(x, y) && (19) \\ =& w_{01} g_y + w_{10} g_x + w_{11} g_{xy} + C f(x, y) , && (20) \end{aligned}$$

where $C = \sum_{i,j} w_{ij}$. We can see that the resulted output $f_w^1(x, y)$ can be decomposed into four terms, three of which are related to image gradients, and one represents pixel value.

If we constrain $C = 0$, i.e., the elements in W sum to 0, then $f_w^1(x, y) = w_{01} g_y + w_{10} g_x + w_{11} g_{xy}$. Hence,

$$\mathrm{Var}[f_w^1(x, y)] = (|g_x| + |g_y| + |g_{xy}|)\sigma^2 .$$

Notice that the input f is given and only w are random variables. It is thus obvious that $f_w^1(x, y) \sim \mathcal{N}(0, (|g_x| + |g_y| + |g_{xy}|)\sigma^2)$. Hence, from Lemma A.1, the expectation after ReLU activation (h) is:

$$E[f^1(x, y)] = E[h(f_w^1(x, y))] = \frac{\sqrt{|g_x| + |g_y| + |g_{xy}|}}{\sqrt{2\pi}} \sigma . \tag{21}$$

Therefore, we can conclude that regions with larger image gradients are expected to have higher activations after the first convolution layer. If (x, y) is located in the texture-less background, then the image gradients at (x, y) are relatively small and these background regions will be most likely deactivated. Conversely, if (x, y) is located in textured edges, then the image gradients will become big.

Notice that with different w_{01} and w_{10}, we can capture image gradients w.r.t different angles θ, e.g., $\theta = \arctan(\frac{w_{10}}{w_{01}})$ for $w_{01}, w_{10} > 0$. In practice we use multiple convolution filters to output multiple channels and hence we can detect various angles with various parameters. □

APPENDIX B
PROOF OF CLAIM 2

Proof. Assume there is no non-linearity in Equation (8), i.e., for a deep linear CNN and $k = 2$:

$$f^p(x, y) = \sum_{i=0}^{1} \sum_{j=0}^{1} w_{i,j}^p f^{p-1}(x+i, y+j) , \tag{22}$$

where $w_{i,j}^p$ denotes the weight of convolution filter at the p-th layer. By induction we know that

$$\begin{aligned} f^p(x, y) =& \sum_{i_1=0}^{1} \sum_{j_1=0}^{1} \cdots \sum_{i_p=0}^{1} \sum_{j_p=0}^{1} \Big(w_{i_1,j_1}^p \cdots w_{i_p,j_p}^p \times \\ & f(x + i_1 + \cdots + i_p, y + j_1 + \cdots + j_p) \Big) && (23) \\ \triangleq& \sum_{s=0}^{p} \sum_{t=0}^{p} W_{s,t}^p f(x+s, y+t) , && (24) \end{aligned}$$

where $s = \sum_{u=1}^{p} i_u$ and $t = \sum_{u=1}^{p} j_u$. If we set all w to 1, then $W_{s,t}^p$ becomes the number of combinations $C_{s,t}$ where the sum of subscript i equals s and the sum of subscript j equals t:

$$C_{s,t} = \sum_{i_1,j_1,\ldots,i_p,j_p} \mathbb{I}(i_1 + \cdots + i_p - s) \times \mathbb{I}(j_1 + \cdots + i_p - t)\,,$$

where

$$\mathbb{I}(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise,} \end{cases}$$

is the indicator function. Now we let $C_s = \sum_{i_1,\ldots,i_p} \mathbb{I}(i_1 + \cdots + i_p - s)$ and each $i_t \in \{0, 1\}$. Suppose there are k elements equal 1 (hence $p - k$ elements equal 0), then we have $C_{k+0\times(p-k)} = C_k = \binom{p}{k}$. As can be seen, C_s follows the binomial distribution, which distributes w.r.t. s like a Gaussian when p becomes large.

Notice that the indexes of i and j are independent and we have $C_{s,t} = C_s C_t$. Hence, Eq. (23) can be reformulated as:

$$f^p(x,y) = \sum_{s=0}^{p}\sum_{t=0}^{p} C_{s,t} f(x+s, y+t) \tag{25}$$

$$= \sum_{s=0}^{p}\sum_{t=0}^{p} C_s C_t f(x+s, y+t) \tag{26}$$

$$= \sum_{s=0}^{p}\sum_{t=0}^{p} \binom{p}{s}\binom{p}{t} f(x+s, y+t)\,. \tag{27}$$

Notice that as p becomes large,

$$C_s = \binom{p}{s} \approx 2^p \frac{1}{\sqrt{2\pi p \times \frac{1}{4}}} e^{\frac{-(s-\frac{p}{2})^2}{2p\times\frac{1}{4}}}\,,$$

$$C_{s,t} = \binom{p}{s}\binom{p}{t} \approx 2^{2p} \frac{1}{2\pi p \times \frac{1}{4}} e^{-\frac{(s-\frac{p}{2})^2+(t-\frac{p}{2})^2}{2p\times\frac{1}{4}}}\,.$$

Let $g(s,t) \triangleq \frac{1}{2\pi p\times\frac{1}{4}} e^{-\frac{(s-\frac{p}{2})^2+(t-\frac{p}{2})^2}{2p\times\frac{1}{4}}}$ and we can see that $g(s,t)$ becomes a Gaussian kernel with $\boldsymbol{\mu} = (\frac{p}{2}, \frac{p}{2})$ and $\boldsymbol{\Sigma} = \begin{bmatrix} \frac{p}{4} & 0 \\ 0 & \frac{p}{4} \end{bmatrix}$. In other words, convolutional layers of depth p in a deep linear CNN corresponds to Gaussian kernels with width p.

Extension to $k > 2$. In this case the coefficients are known as "extended binomial coefficients" or "polynomial coefficients", and they too distribute like Gaussian [27].

Extension to random weights. Notice that we set all w to 1 in Equation (25) and now we consider $w \sim N(0, \sigma^2)$. Notice that $\{w_{i,j}^k | k = 1, \ldots, p; i = 0, \ldots, k-1; j = 0, \ldots, k-1\}$ are mutually independent. Suppose w_1 and w_2 are independent random variables that follows $N(0, \sigma^2)$. It is well-known that the product of two independent Gaussian distributions is still Gaussian: $w_1 w_2 \sim N(0, \frac{\sigma^2}{2})$. It is obvious that $w_1 w_2 \ldots w_n \sim N(0, \frac{\sigma^2}{n})$. Also, we know that $w_1 + w_2 + \cdots + w_n \sim N(0, n\sigma^2)$. Hence, from Equation (23) we have $W_{s,t}^p \sim N(0, \frac{C_{s,t}}{p}\sigma^2)$. In this case, the weight of the kernel is sampled from a Gaussian, whose variance distributes like a Gaussian kernel. In other words, the receptive field increases along with the increase of the depth and the impact of each pixel (i.e., variance of weight) decays quickly from the center. □

APPENDIX C
PROOF OF CLAIM 3

Proof. Now we analyze the expectation and variance of the output layers and we first consider the case without non-linearity. Then if we take expectations over both sides of Equation (22) we have

$$E_w[f^p(x,y)] = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} E_w[w_{i,j}^p f^{p-1}(x+i, y+j)] \tag{28}$$

$$= \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} E_w[w_{i,j}^p] f^{p-1}(x+i, y+j) \tag{29}$$

$$= 0\,. \tag{30}$$

Notice that we assume $f^{p-1}(\cdot,\cdot)$ are given as constant value when analyzing $f^p(,)$ and hence for the variance we have

$$\mathrm{Var}[f^p(x,y)] = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} \mathrm{Var}[w_{i,j}^p f^{p-1}(x+i, y+j)] \tag{31}$$

$$= \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} \mathrm{Var}[w_{i,j}^p] |f^{p-1}(x+i, y+j)| \tag{32}$$

$$= \sigma^2 \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} |f^{p-1}(x+i, y+j)|\,. \tag{33}$$

□

APPENDIX D
PROOF OF CLAIM 4

Proof. Now we introduce ReLU non-linearity back into our analyses. Let us denote $S = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} w_{i,j}^p f^{p-1}(x+i, y+j)$. Notice that $w_{i,j}^p$ are independent and identically distributed from $N(0, \sigma^2)$. Hence we have $S \sim \mathcal{N}(0, T\sigma^2)$, where $T = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} f^{p-1}(x+i, y+j)$. Hence,

$$f^p(x,y) = h\Big(\sum_{i=0}^{k-1}\sum_{j=0}^{k-1} w_{i,j}^p f^{p-1}(x+i, y+j)\Big) \tag{34}$$

$$= \max(S, 0)\,. \tag{35}$$

Fom Lemma A.1 we have

$$E_w[f^p(x,y)] = E_w[h(S)] = \sqrt{\frac{T}{2\pi}}\sigma\,, \tag{36}$$

and

$$\mathrm{Var}[f^p(x,y)] = (\frac{1}{2} - \frac{1}{2\pi})T\sigma^2\,. \tag{37}$$

Hence, from Equation (36) and (37) we know that under ReLU activation, both the expectation and variance of $f^p(x,y)$ is positively correlated with T, which is the sum of the neighborhood region in the previous layer. In other words, a region with relatively large values will remain a relatively large value in the expectation ($O(\sqrt{T})$) and will be deactivated for $T = 0$. □

APPENDIX E
PROOF OF CLAIM 5

We first prove Lemma E.1 that will be used soon.

Lemma E.1. Suppose $X \sim N(0, \sigma^2)$, i.e., the probability density function $p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$, and let $Y = h(x) = \frac{1}{1+e^{-x}}$. Then, $\frac{1}{4} < E(Y) < \frac{3}{4}$.

Proof. It is obvious that $0 < h(x) < \frac{1}{2}$ for $x < 0$ and $\frac{1}{2} < h(x) < 1$ for $x > 0$. Hence,

$$E(Y) = \int_{-\infty}^{+\infty} h(x)p(x)\,\mathrm{d}x \tag{38}$$

$$= \int_{-\infty}^{0} h(x)p(x)\,\mathrm{d}x + \int_{0}^{\infty} h(x)p(x)\,\mathrm{d}x \tag{39}$$

$$< \frac{1}{2}\int_{-\infty}^{0} p(x)\,\mathrm{d}x + \int_{0}^{\infty} p(x)\,\mathrm{d}x \tag{40}$$

$$= \frac{3}{4} \tag{41}$$

For the lower bound, we have

$$E(Y) = \int_{-\infty}^{0} h(x)p(x)\,\mathrm{d}x + \int_{0}^{\infty} h(x)p(x)\,\mathrm{d}x \tag{42}$$

$$> \int_{-\infty}^{0} 0 \times p(x)\,\mathrm{d}x + \frac{1}{2}\int_{0}^{\infty} p(x)\,\mathrm{d}x \tag{43}$$

$$= \frac{1}{4} \tag{44}$$

□

Then the proof of Claim 5 follows.

Proof. If we instead use a bounded activation function, e.g., sigmoid and now $h(x) = \frac{1}{1+e^{-x}}$.

From Lemma E.1, if we use sigmoid activation, then Equation (36) becomes:

$$E_w[f^p(x, y)] = E_w[h(S)] \in (\frac{1}{4}, \frac{3}{4}).$$

In this case the expectation of output is bounded by constant value and we can not expect a large T to get a large $E_w[f^p(x, y)]$. In other words, regions with relatively high activation values (possibly objects) will not continue to be maintained at large values when the number of layers increases if we use sigmoid (or other bounded activations). □

The following sections contain more visualization results, experimental details and experimental results not included in the main paper.

APPENDIX F
VISUALIZATION AND LOCALIZATION RESULTS

In Table 1 of the main paper, we analyzed the localization ability of randomly initialized ResNet-50 and we present more visualization results of Table 1 here.

Visualization of multiple randomly initialized ResNet-50. From Table 1 of the main paper we have seen that randomly initialized networks can get robust localization results because it achieves a low standard deviation of 0.6 on ImageNet and CUB-200, respectively. We show some visualization results of 3 randomly initialized ResNet-50 using SCDA on ImageNet and CUB-200 in Figure 13 and Figure 14, respectively. We color foreground regions, i.e., those patches with $(i, j)|M_{i,j} = 1$ with the red color. We can intuitively see that the localization results of different randomly initialized ResNet-50 are very similar and it further suggests that Tobias is one inductive bias of CNN. Also, we can observe that a randomly initialized CNN can localize objects surprisingly well, which is comparable to the ImageNet supervised counterpart.

Visualization of using different activation functions. We change the default activation function ReLU to sigmoid and arctan for a randomly initialized ResNet-50 and show the results on ImageNet in Figure 15. As can be seen, ReLU achieves significantly better localization performance than sigmoid and arctan. It demonstrates that ReLU plays an important role in modern CNNs to deactivate texture-less backgrounds and activate objects.

Table 12: Training details for MoCov2, SimCLR and BYOL on CUB200 for experiments presented in Table 2. τ denotes the temperature parameter and k denotes the size of the memory bank in MoCov2.

Method	backbone	Settings				
		bs	lr	lr schedule	τ	k
MoCov2	ResNet-18	128	0.03	cosine	0.2	4096
	ResNet-50	128	0.03	cosine	0.2	4096
SimCLR	ResNet-18	512	0.5	cosine	0.1	-
	ResNet-50	128	0.125	cosine	0.1	-

APPENDIX G
TRAINING DETAILS

We present training details for self-supervised learning and downstream evaluation here.

Training details for SSL: The training details for MoCov2 and SimCLR on CUB200 for those experimental results presented in Table 2 in the main paper are shown in Table 12. We adopt several common data augmentations and compose them stochastically: (a) random scaling and cropping with a scaling factor chosen between [0.2, 1.0]; (b) random horizontal flipping with a probability of 0.5; (c) color distortion with a probability of 0.8; (d) color dropping (i.e., randomly convert images to grayscale with 20% probability for each image); (e) random gaussian blur.

Training details for downstream classification benchmarks: For ImageNet linear evaluation, we follow the same settings in [41]. For ImageNet fine-tuning, we train for 30 epochs with the learning rate initialized to 0.01, which is divided by 10 every 10 epochs. For other classification benchmarks, we train the network for 120 epochs with a batch size of 64 and a weight decay of 5e-4. The learning rate starts from 10.0 for linear evaluation and 0.01 for fine-tuning and is decreased every 40 epochs.

APPENDIX H
MORE RESULTS

In this section, we show experiments that were not included in the main paper due to limited space.

We set the hyper-parameter p to 0.3 for all experiments in the main paper and we have seen that we can get better results with more carefully tuned p in Section 5.2.3. Here we

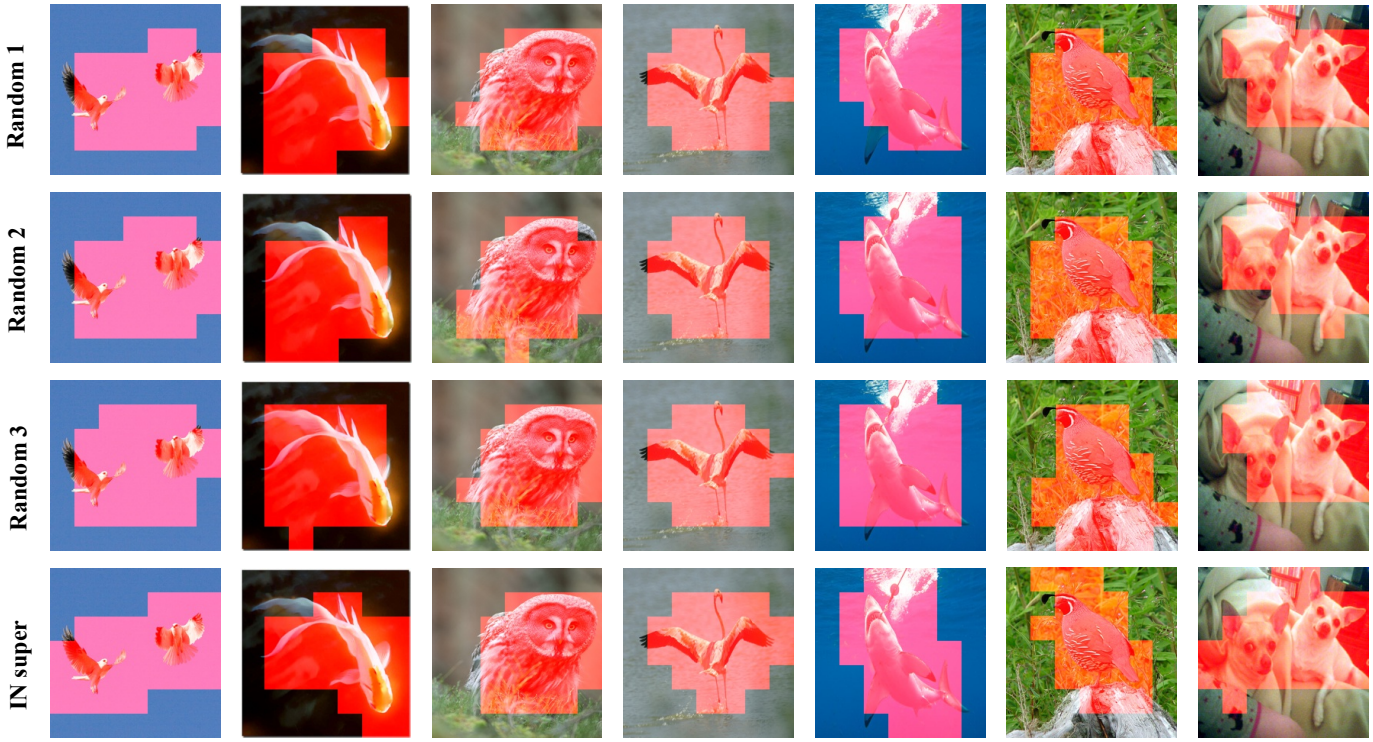


Figure 13: Visualization of multiple *randomly initialized* ResNet-50 using SCDA on ImageNet. 'Random 1' represents the first trial and the same for 'Random 2' and 'Random 3'. We color foreground regions, i.e., those patches with $(i, j)|M_{i,j} = 1$ with the red color.

Figure 14: Visualization of multiple *randomly initialized* ResNet-50 using SCDA on CUB-200.

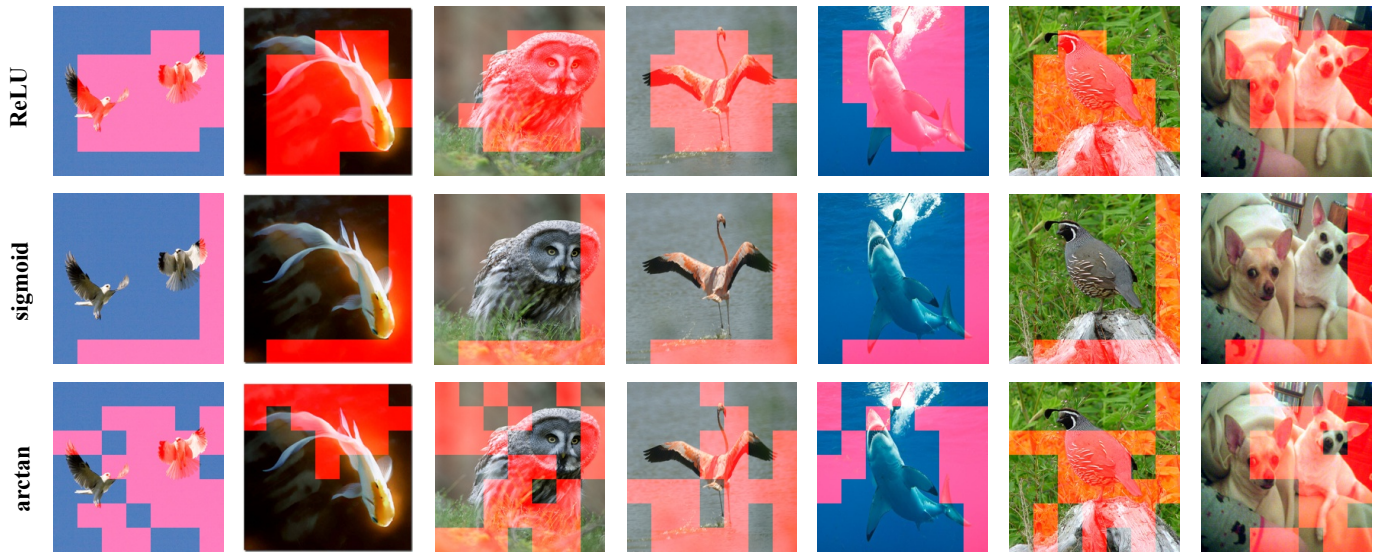


Figure 15: Ablation of activation functions for randomly initialized ResNet-50 on ImageNet.

investigate the influence of data volume on hyper-parameter selection and present more results by setting different p values on ImageNet subsets in Table 13.

As shown in Table 13, we can get much better performance with more carefully tuned p. For instance, when both trained for 200 epochs on IN-50k (50,000 images), our Tobias (with $p = 0.5$) achieves better results than those reported in the main paper (with $p = 0.3$), and is significantly better than baseline counterpart: up to +2.5 AP_{50}, +2.7 AP and +3.7 AP_{75}. It confirms that our method has the potential to perform better with more carefully tuned p. Also, we can observe that when trained for 200 epochs on IN-100k, $p = 0.5$ achieves worse results than $p = 0.3$, which indicates that a higher p value is more suitable for small data while a lower p value is more suitable for large data.

APPENDIX I
INTRODUCTION OF ACTIVATION FUNCTIONS

Now we introduce the activation functions used in the main paper.

The ReLU activation takes the following form:

$$\mathrm{ReLU}(x) = \max(0, x)\,.$$

The softplus activation takes the following form:

$$\mathrm{softplus}(x) = \frac{1}{\beta}\log(1 + e^{\beta x})\,,$$

where the default value for β is 1 in PyTorch.

The ELU activation takes the following form:

$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases},$$

where the default value for α is 1 in PyTorch.

The SELU activation takes the following form:

$$\mathrm{SELU}(x) = s * (\max(0, x) + \min(0, \alpha(e^x - 1)))\,,$$

where $s = 1.0507$ and $\alpha = 1.6733$ in PyTorch.

The sigmoid activation takes the following form:

$$\mathrm{sigmoid}(x) = \frac{1}{1 + e^{-x}}\,.$$

The arctan activation takes the following form:

$$\arctan(x) = \tan^{-1}(x)\,.$$

We also plot these activation functions in Figure 16 for better illustration.

Table 13: Downstream object detection performance on VOC 07&12 and linear evaluation accuracy on Tiny-ImageNet-200 when pretrained on ImageNet subsets using ResNet-50 (with different hyper-parameter p).

pretraining			VOC 07&12			Tiny-IN-200
method	#images	epochs	AP_{50}	AP	AP_{75}	
random init.	0	0	63.0	36.7	36.9	0.5
MoCov2	10,000	200	71.1	43.6	45.8	0.5
MoCov2-Tobias (p=0.3)			71.4 (+0.3)	44.3 (+0.7)	47.0 (+1.2)	9.9 (+9.4)
MoCov2-Tobias (p=0.5)			71.5 (+0.4)	44.3 (+0.7)	47.3 (+1.5)	10.2 (+9.7)
MoCov2	10,000	800	71.6	43.9	45.9	23.6
MoCov2-Tobias (p=0.3)			73.2 (+1.6)	45.7 (+1.8)	48.5 (+2.6)	23.9 (+0.3)
MoCov2-Tobias (p=0.5)			73.9 (+2.3)	46.3 (+2.4)	49.4 (+3.5)	23.3 (-0.3)
MoCov2	50,000	200	72.2	44.7	46.8	26.3
MoCov2-Tobias (p=0.3)			73.7 (+1.5)	46.0 (+1.3)	49.2 (+2.4)	26.0 (-0.3)
MoCov2-Tobias (p=0.5)			74.7 (+2.5)	47.4 (+2.7)	50.5 (+3.7)	26.3
MoCov2	50,000	800	77.5	49.3	53.3	37.9
MoCov2-Tobias (p=0.3)			77.9 (+0.4)	50.3 (+1.0)	54.9 (+1.6)	40.7 (+2.8)
MoCov2-Tobias (p=0.5)			78.4 (+0.9)	50.9 (+1.6)	55.1 (+1.8)	39.4 (+1.5)
MoCov2	100,000	200	76.2	48.0	51.6	35.3
MoCov2-Tobias (p=0.3)			77.5 (+1.3)	49.8 (+1.8)	53.9 (+2.3)	36.5 (+1.2)
MoCov2-Tobias (p=0.5)			76.8 (+0.6)	49.2 (+1.2)	53.3 (+1.7)	36.4 (+1.1)

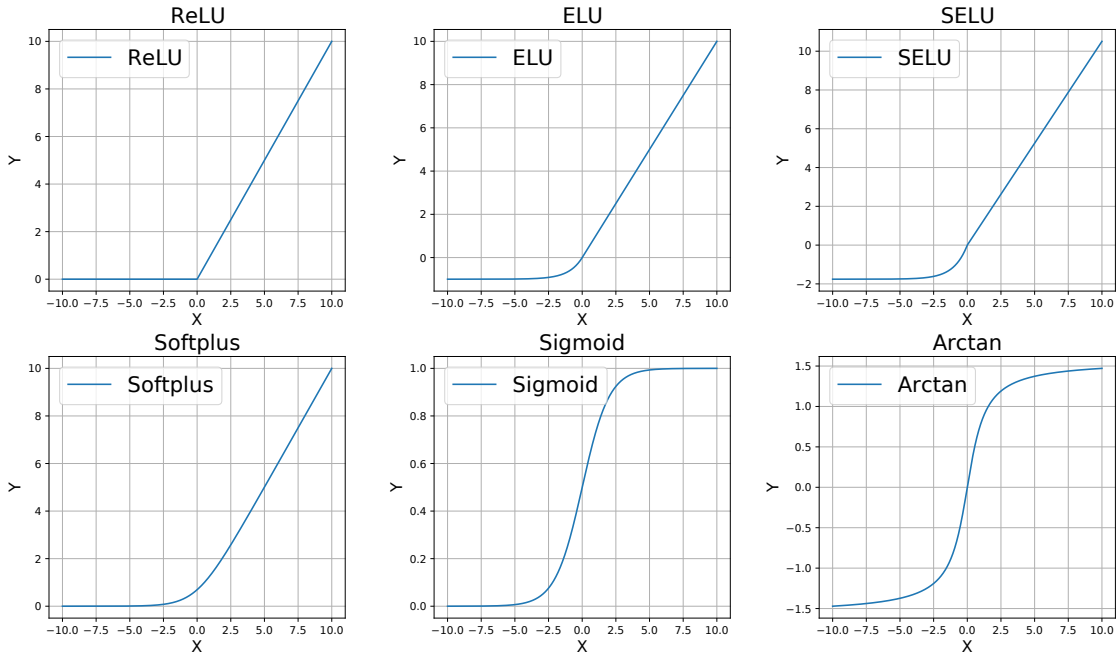


Figure 16: Visualization of different activation functions used in the main paper.