Momentum Contrast for Unsupervised Visual Representation Learning

Kaiming He Haoqi Fan Yuxin Wu Saining Xie Ross Girshick

Facebook AI Research (FAIR)

Code: https://github.com/facebookresearch/moco

Abstract

We present Momentum Contrast (MoCo) for unsupervised visual representation learning. From a perspective on contrastive learning [29] as dictionary look-up, we build a dynamic dictionary with a queue and a moving-averaged encoder. This enables building a large and consistent dictionary on-the-fly that facilitates contrastive unsupervised learning. MoCo provides competitive results under the common linear protocol on ImageNet classification. More importantly, the representations learned by MoCo transfer well to downstream tasks. MoCo can outperform its supervised pre-training counterpart in 7 detection/segmentation tasks on PASCAL VOC, COCO, and other datasets, sometimes surpassing it by large margins. This suggests that the gap between unsupervised and supervised representation learning has been largely closed in many vision tasks.

【摘要】

我们提出了无监督视觉表征学习的动量对比方法（MoCo）。从对比学习的角度看[29]，作为字典查找，我们建立了一个动态字典队列和一个移动平均编码器。这使得能够快速构建一个大型且一致的词典，从而促进对比无监督学习。MoCo 在 ImageNet 分类的共同线性协议下提供了有竞争力的结果。更重要的是，MoCo 学习到的表示可以很好地迁移到下游任务。在 PASCAL-VOC、COCO 和其他数据集上，MoCo 可以在 7 个检测/分割任务上优于其相对应的监督式学习方法，有时甚至会大大超过它。这表明，在许多视觉任务中，无监督和有监督的表征学习之间的差距已经很大程度上缩小了。

1. Introduction

Unsupervised representation learning is highly successful in natural language processing, e.g., as shown by GPT[50, 51] and BERT [12]. But supervised pre-training is stilldominant in computer vision, where unsupervised methods generally lag behind. The reason may stem from differences in their respective signal spaces. Language tasks have discrete signal spaces (words, sub-word units, etc.) for building tokenized dictionaries, on which unsupervised learning can be based. Computer vision, in contrast, further concerns dictionary building [54, 9, 5], as the raw signal is in a continuous, high-dimensional space and is not structured for human communication (e.g., unlike words).

【1.介绍】

无监督表示学习在自然语言处理中已经非常成功，如GPT[50,51]和BERT[12]所示。但在计算机视觉领域，有监督的预训练仍然占主导地位，而无监督的方法通常落后于计算机视觉。原因可能是它们各自信号空间的差异。语言任务具有离散的信号空间（单词、子单词单元等），可以用于构建标记化词典，在此基础上可以进行无监督学习。相比之下，计算机视觉更关注词典的建立[54、9、5]，因为原始信号处于连续的高维空间中，并且不是为人类交流而构建的（例如，不同于文字）。

Several recent studies [61, 46, 36, 66, 35, 56, 2] present promising results on unsupervised visual representation learning using approaches related to the contrastive loss[29]. Though driven by various motivations, these methods can be thought of as building dynamic dictionaries. The “keys” (tokens) in the dictionary are sampled from data (e.g., images or patches) and are represented by an encoder network. Unsupervised learning trains encoders to perform dictionary look-up: an encoded “query” should be similar to its matching key and dissimilar to others. Learning is formulated as minimizing a contrastive loss [29].

最近的一些研究[61，46，36，66，35，56，2]提出了使用与对比损失相关的方法进行无监督视觉表征学习的有希望的结果[27]。尽管这些方法受到各种动机的驱动，但可以看作是构建动态词典。字典中的“密钥”（令牌）是从数据（如图像或补丁）中采样的，并由编码器网络表示。无监督学习训练编码器进行字典查找：一个编码的“查询”应该与其匹配的密钥相似，而与其他查询不同。学习被定义为最小化对比损失[29]。

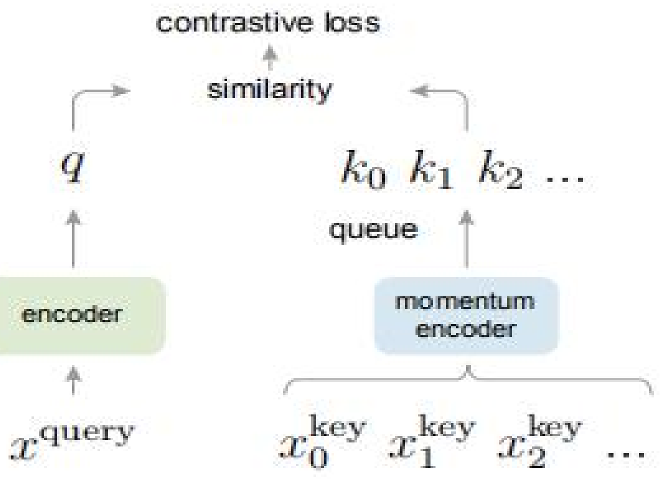


Figure 1. Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys {k0, k1, k2, ...} are defined on-the-fly by a set of data samples.

The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from the mini-batch size. The keys are encoded by a slowly progressing encoder, driven by a momentum update with the query encoder.This method enables a large and consistent dictionary for learning visual representations.

图 1：动量对比度（MoCo）通过使用对比损失将编码查询 q 与编码密钥字典相匹配来训练视觉表示编码器。字典密钥{k0，k1，k2，…}由一组数据样本动态定义。字典作为队列构建，当前的小批量入队，最旧的小批量出队，将其与小批量大小分离。密钥由一个缓慢进行的编码器进行编码，并由查询编码器的动量更新驱动。这种方法为学习视觉表征提供一个大而一致的字典。

From this perspective, we hypothesize that it is desirable to build dictionaries that are: (i) large and (ii) consistent as they evolve during training. Intuitively, a larger dictionary may better sample the underlying continuous, high-dimensional visual space, while the keys in the dictionary should be represented by the same or similar encoder so that their comparisons to the query are consistent. However, existing methods that use contrastive losses can be limited in one of these two aspects (discussed later in context).

从这个角度出发，我们假设我们希望建立的词典是：（i）大的和（ii）一致的, 同时它们在训练过程中发展。直观地说，较大的字典可以更好地采样底层的连续、高维视觉空间，而字典中的密钥应该由相同或相似的编码器表示，以便它们与查询的比较是一致的。然而，使用对比损失的现有方法可能局限于这两个方面中的一个（稍后在上下文中讨论）。

We present Momentum Contrast (MoCo) as a way of building large and consistent dictionaries for unsupervised learning with a contrastive loss (Figure 1). We maintain the dictionary as a queue of data samples: the encoded representations of the current mini-batch are enqueued, and the oldest are dequeued. The queue decouples the dictionary size from the mini-batch size, allowing it to be large. Moreover, as the dictionary keys come from the preceding several mini-batches, a slowly progressing key encoder, implemented as a momentum-based moving average of the query encoder, is proposed to maintain consistency.

我们提出动量对比（MoCo）作为一种建立大型且一致性词典的方法，用于对比损失的无监督学习（图 1）。我们将字典维持为一个数据样本队列：当前小批量的编码表示是排队的，最早的表示是出队的。队列将字典大小与小批量大小分离，允许其变大。此外，由于字典密钥存在来自前面的几个小批量，为了保持一致性，提出了一种缓慢进行的密钥编码器，该编码器由基于动量的移动平均的查询编码器实现。

MoCo is a mechanism for building dynamic dictionaries for contrastive learning, and can be used with various pretext tasks. In this paper, we follow a simple instance discrimination task [61, 63, 2]: a query matches a key if they are encoded views (e.g., different crops) of the same image. Using this pretext task, MoCo shows competitive results under the common protocol of linear classification in the ImageNet dataset [11].

MoCo 是一种为对比学习建立动态词典的机制，可用于各种代理任务（pretext tasks）。在本文中，我们遵循一个简单的实例判别任务[61，63，2]：如果一个查询是同一图像的编码视图（例如，不同的裁剪），则它会匹配一个密钥。利用这个代理任务，MoCo 在 ImageNet 数据集中显示了在共同的线性分类协议下的有竞争力的结果[11]。

A main purpose of unsupervised learning is to pre-train representations (i.e., features) that can be transferred to downstream tasks by fine-tuning. We show that in 7 downstream tasks related to detection or segmentation, MoCo unsupervised pre-training can surpass its ImageNet supervised counterpart, in some cases by nontrivial margins. In these experiments, we explore MoCo pre-trained on ImageNet or on a one-billion Instagram image set, demonstrating that MoCo can work well in a more real-world, billion-image scale, and relatively uncurated scenario. These results show that MoCo largely closes the gap between unsupervised and supervised representation learning in many computer vision tasks, and can serve as an alternative to ImageNet supervised pre-training in several applications.

无监督学习的一个主要目的是预先训练可通过微调迁移到下游任务的表示（即特征）。我们发现，在与检测或分割相关的 7 个下游任务中，MoCo 无监督预训练可以超过 ImageNet 上有监督的预训练，在某些情况下可以超过 ImageNet 上有监督的预训练。在这些实验中，我们探索了 MoCo 在 ImageNet 或 10 亿 Instagram 图像集上的预训练，证明了 MoCo 可以在更真实的、十亿图像规模和相对未定级的场景中很好地工作。这些结果表明，MoCo 在很大程度上弥补了在许多计算机视觉任务中无监督和有监督表示学习之间的差距，并且可以作为 ImageNet 上有监督预训练的替代方案。

2. Related Work

Unsupervised/self-supervised learning methods generally involve two aspects: pretext tasks and loss functions. The term “pretext” implies that the task being solved is not of genuine interest, but is solved only for the true purpose of learning a good data representation. Loss functions can often be investigated independently of pretext tasks. MoCo focuses on the loss function aspect. Next we discuss related studies with respect to these two aspects.

【2. 相关工作】

无监督/自监督学习方法通常包括两个方面：代理任务和损失函数。“借口”一词意味着正在解决的任务不是真正感兴趣的，而是为了学习良好的数据表示的真正目的而解决的。损失函数通常可以独立于代理任务进行研究。MoCo 着重于损失函数方面。接下来我们介绍下这两个方面的相关研究。

Loss functions. A common way of defining a loss function is to measure the difference between a model's prediction and a fixed target, such as reconstructing the input pixels (e.g., auto-encoders) by L1 or L2 losses, or classifying the input into pre-defined categories (e.g., eight positions [13], color bins [64]) by cross-entropy or margin-based losses. Other alternatives, as described next, are also possible.

Contrastive losses [29] measure the similarities of sample pairs in a representation space. Instead of matching an input to a fixed target, in contrastive loss formulations the target can vary on-the-fly during training and can be defined in terms of the data representation computed by a network[29]. Contrastive learning is at the core of several recent works on unsupervised learning [61, 46, 36, 66, 35, 56, 2],which we elaborate on later in context (Sec. 3.1).

Adversarial losses [24] measure the difference between probability distributions. It is a widely successful technique for unsupervised data generation. Adversarial methods for representation learning are explored in [15, 16]. There are relations (see [24]) between generative adversarial networks and noise-contrastive estimation (NCE) [28].

损失函数 定义损失函数的一种常见方法是测量模型的预测与固定目标之间的差异，例如通过 L1 或 L2 损失重建输入像素（例如，自动编码器），或通过交叉熵或基于边距的损失将输入分类为预定义的类别(例如,八个位置[13]、色箱[64])。其他替代方案，如下文所述，也是可能的。

对比损失[29]衡量代表空间中样本对的相似性。在对比损失公式中，目标可以在训练期间动态变化，而不是将输入与固定目标匹配，并且可以根据网络计算的数据表示来定义[29]。对比学习是最近几部关于无监督学习的著作的核心[61,46,36,66,35,56,2]，我们在后面的上下文中对此进行了详细阐述。（3.1部分）。

对抗性损失[24]衡量概率分布之间的差异。它是一种广泛成功的无监督数据生成技术。在[15，16]中探讨了对抗性方法的表征学习。生成对抗网络与噪声对比估计（NCE）[28]之间存在联系（见[24]）。

Pretext tasks A wide range of pretext tasks have been proposed. Examples include recovering the input under some corruption, e.g., denoising auto-encoders [58], context auto-encoders [48], or cross-channel auto-encoders (colorization) [64, 65]. Some pretext tasks form pseudo-labels by, e.g., transformations of a single ("exemplar") image [17], patch orderings [13, 45], tracking [59] or segmenting objects [47] in videos, or clustering features [3, 4].

代理任务 人们提出了各种各样的代理任务。示例包括在某些损坏情况下恢复输入，例如去噪自动编码器[58]、上下文自动编码器[48]或跨通道自动编码器（着色）[64，65]。一些代理任务通过变换单个（“示例”）图像[17]、补丁排序[13、45]、跟踪[59]或分割视频中的对象[47]或聚类特征[3，4]来形成伪标签。

Contrastive learning vs. pretext tasks Various pretext tasks can be based on some form of contrastive loss functions. The instance discrimination method [61] is related to the exemplar-based task [17] and NCE [28]. The pretext task in contrastive predictive coding (CPC) [46] is a form of context auto-encoding [48], and in contrastive multiview coding (CMC) [56] it is related to colorization [64].

对比学习与代理任务 各种代理任务都可以建立在基于某种形式的对比损失函数。实例判别方法[61]与基于实例的任务[17]和 NCE[28]有关。对比预测编码（CPC）[46]中的代理任务是上下文自动编码的一种形式[48]，在对比多视图编码（CMC）[56]中，代理任务与颜色化有关[64]。

3. Method

3.1. Contrastive Learning as Dictionary Look-up

Contrastive learning [29], and its recent developments,can be thought of as training an encoder for a dictionary look-up task, as described next.

Consider an encoded query q and a set of encoded samples {k0, k1, k2, ...} that are the keys of a dictionary. Assume that there is a single key (denoted as k+) in the dictionary that q matches. A contrastive loss [29] is a function whose value is low when q is similar to its positive key k+ and dissimilar to all other keys (considered negative keys for q). With similarity measured by dot product, a form of a contrastive loss function, called InfoNCE [46], is considered in this paper:

$$\mathcal{L}_q = -\log \frac{\exp(q{\cdot}k_+/\tau)}{\sum_{i=0}^{K} \exp(q{\cdot}k_i/\tau)} \qquad (1)$$

where τ is a temperature hyper-parameterper [61]. The sum is over one positive and K negative samples. Intuitively, this loss is the log loss of a (K+1)-way softmax-based classifier that tries to classify q as k+. Contrastive loss functions can also be based on other forms [29, 59, 61, 36], such as margin-based losses and variants of NCE losses.

The contrastive loss serves as an unsupervised objective function for training the encoder networks that represent the queries and keys [29]. In general, the query representation is q = fq(xq) where fq is an encoder network and xq is a query sample (likewise, k = fk(xk)). Their instantiations depend on the specific pretext task. The input xq and xk can be images [29, 61, 63], patches [46], or context consisting a set of patches [46]. The networks fq and fk can be identical [29, 59, 63], partially shared [46, 36, 2], or different [56]

注：公式不太友好，请查看原论文，这里只做翻译。

注：下述键与密钥，含义一致。

【3. 方法】

3. 1. 对比学习与查字典

对比学习[29]及其最新发展，可以被认为是训练编码器进行字典查找任务，如下所述。

考虑一个编码查询 q 和一组编码样本{k0，k1，k2，…}它们是字典的键。假设字典中有一个 q 匹配的键（表示为 k+）。对比损失[29]是一个函数，当 q 与 q 的正键 k+相似而与其他所有键（被认为是 q 的负键）不同时，该函数的值很低。

利用点积度量相似度，本文考虑了一种称为 InfoNCE[46]的对比损失函数形式：

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)} \qquad (1)$$

其中 τ 是[59]中的温度超参数。其和大于一个正样本和 K 个负样本。直观地说，这种损失是基于（K+1）类 softmax 的分类器的 log 损失，该分类器试图将其 q 分类为 k+。对比损失函数也可以基于其他形式[29，59，61，36]，例如基于 margin-based losses 和 NCE 损失的变种。

对比损失作为一个无监督的目标函数，用于训练表示查询和键（key）的编码器网络[29]。一般来说，查询表示是 q = fq(xq)，其中 fq 是编码器网络，xq 是查询样本（同理，k = fk(xk)）。它们的实例化依赖于特定的代理任务。输入 xq 和 xk，可以是图像[29，61，63]、patches[46]或由一组 patches 组成的上下文[46]。网络 fq 和 fk 可以是相同的[29，59，63]，部分共享[46，36，2]或不同[56]。

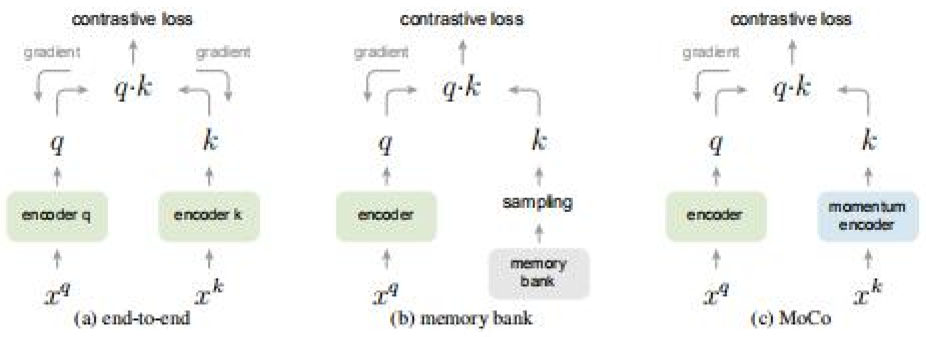


Figure 2. Conceptual comparison of three contrastive loss mechanisms (empirical comparisons are in Figure 3 and Table 3). Here we illustrate one pair of query and key. The three mechanisms differ in how the keys are maintained and how the key encoder is updated. (a): The encoders for computing the query and key representations are updated end-to-end by back-propagation (the two encoders can be different). (b): The key representations are sampled from a memory bank [61]. (c): MoCo encodes the new keys on-the-fly by a momentum-updated encoder, and maintains a queue (not illustrated in this figure) of keys.

图一. 三种对比损失机制的概念比较（实验对比见图 3 和表 3）。这三种机制在键的维护方式和键编码器的更新方式上有所不同。（a）：用于计算查询和键表示的编码器通过反向传播进行端到端更新（两个编码器可以不同）。（b）：键表示是从存储库中取样[61]。（c）：MoCo 通过动量更新的编码器对新键进行动态编码，并维护键队列（本图中未说明）。

3.2. Momentum Contrast

From the above perspective, contrastive learning is a way of building a discrete dictionary on high-dimensional continuous inputs such as images. The dictionary is dynamic in the sense that the keys are randomly sampled, and that the key encoder evolves during training. Our hypothesis is that good features can be learned by a large dictionary that covers a rich set of negative samples, while the encoder for the dictionary keys is kept as consistent as possible despite its evolution. Based on this motivation, we present Momentum Contrast as described next.

3.2 动量对比

从以上的角度来看，对比学习是一种基于高维连续输入（如图像）构建离散字典的方法。字典是动态的，因为键是随机抽样的，键编码器在训练过程中进化。我们的假设是，好的特征可以通过一个包含大量负样本的大字典来学习，而字典键的编码器在进化过程中尽可能保持一致。基于这一动机，我们提出动量对比，如下所述。

Dictionary as a queue. At the core of our approach is maintaining the dictionary as a queue of data samples. This allows us to reuse the encoded keys from the immediate preceding mini-batches. The introduction of a queue decouples the dictionary size from the mini-batch size. Our dictionary size can be much larger than a typical mini-batch size, and can be flexibly and independently set as a hyper-parameter.

The samples in the dictionary are progressively replaced. The current mini-batch is enqueued to the dictionary, and the oldest mini-batch in the queue is removed. The dictionary always represents a sampled subset of all data, while the extra computation of maintaining this dictionary is manageable. Moreover, removing the oldest mini-batch can be beneficial, because its encoded keys are the most outdated and thus the least consistent with the newest ones.

字典作为队列。我们方法的核心是将字典维护为一个数据样本队列。这使我们能够重用前一个小批量中的已编码键。队列的引入将字典大小与小批量大小解耦。我们的字典大小可以比典型的小批量大小大得多，并且可以灵活独立地设置为超参数。

词典中的样本逐渐被替换。当前的小批量将排入字典，队列中最早的小批量将被删除。字典总是代表所有数据的抽样子集，而维护这个字典的额外计算是可管理的（可操纵的）。此外，删除最早的小批量是有益的，因为它编码的键是最过时的，因此与最新的键最不一致。

Momentum update. Using a queue can make the dictionary large, but it also makes it intractable to update the key encoder by back-propagation (the gradient should propagate to all samples in the queue). Ana¨ıve solution is to copy the key encoder fk from the query encoder fq, ignoring this gradient. But this solution yields poor results in experiments (Sec. 4.1). We hypothesize that such failure is caused by the rapidly changing encoder that reduces the key representations' consistency. We propose a momentum update to address this issue.

Formally, denoting the parameters of fk as θk and those of fq as θq, we update θk by:

$$\theta_k \leftarrow m\theta_k + (1 - m)\theta_q. \qquad (2)$$

Here m∈[0,1) is a momentum coefficient. Only the parameters θq are updated by back-propagation. The momentum update in Eqn.(2) makes θk evolve more smoothly than θq. As a result, though the keys in the queue are encoded by different encoders (in different mini-batches), the difference among these encoders can be made small. In experiments, a relatively large momentum (e.g., m = 0.999, our default) works much better than a smaller value (e.g., m = 0.9), suggesting that a slowly evolving key encoder is a core to making use of a queue.

动量更新 使用队列可以使字典变大，但它也使通过反向传播更新键编码器变得困难（梯度应该传播到队列中的所有样本）。一个简单的解决方案是从查询编码器复制到键编码器，忽略这个梯度。但是这种解决方案在实验中产生的结果很差（第。4.1 条）。我们假设这种故障是由快速变化的编码器导致的，这种编码器降低了键表示的一致性。我们建议对这一问题进行动态更新。

形式上，编码器 fk 的参数表示为θk，编码器 fq 的参数表示为θq，我们更新参数θk：

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q. \qquad (2)$$

其中 m∈[0,1）是一个动量系数。只有 fq 参数通过反向传播更新。方程（2）中的动量更新使得θk 比θq 更平滑的演化。因此，虽然队列中的键由不同的编码器（在不同的小批量中）编码，但是这些编码器之间的差异可以变得很小。在实验中，一个相对较大的动量（例如，m=0.999，我们的默认值）比一个较小的值（例如，m=0.9）好得多，这表明缓慢演化的键编码器的核心是利用队列。

Relations to previous mechanisms. MoCo is a general mechanism for using contrastive losses. We compare it with two existing general mechanisms in Figure 2. They exhibit different properties on the dictionary size and consistency.

The end-to-end update by back-propagation is a natural mechanism (e.g., [29, 46, 36, 63, 2, 35], Figure 2a). It uses samples in the current mini-batch as the dictionary, so the keys are consistently encoded (by the same set of encoder parameters). But the dictionary size is coupled with the mini-batch size, limited by the GPU memory size. It is also challenged by large mini-batch optimization [25]. Some recent methods [46, 36, 2] are based on pretext tasks driven by local positions, where the dictionary size can be made larger by multiple positions. But these pretext tasks may require special network designs such as patchifying the input [46] or customizing the receptive field size [2], which may complicate the transfer of these networks to downstream tasks.

Another mechanism is the memory bank approach proposed by [61] (Figure 2b). A memory bank consists of the representations of all samples in the dataset. The dictionary for each mini-batch is randomly sampled from the memory bank with no back-propagation, so it can support a large dictionary size. However, the representation of a sample in the memory bank was updated when it was last seen, so the sampled keys are essentially about the encoders at multiple different steps all over the past epoch and thus are less consistent. A momentum update is adopted on the memory bank in [61]. Its momentum update is on the representations of the same sample, not the encoder. This momentum update is irrelevant to our method, because MoCo does not keep track of every sample. Moreover, our method is more memory-efficient and can be trained on billion-scale data, which can be intractable for a memory bank.

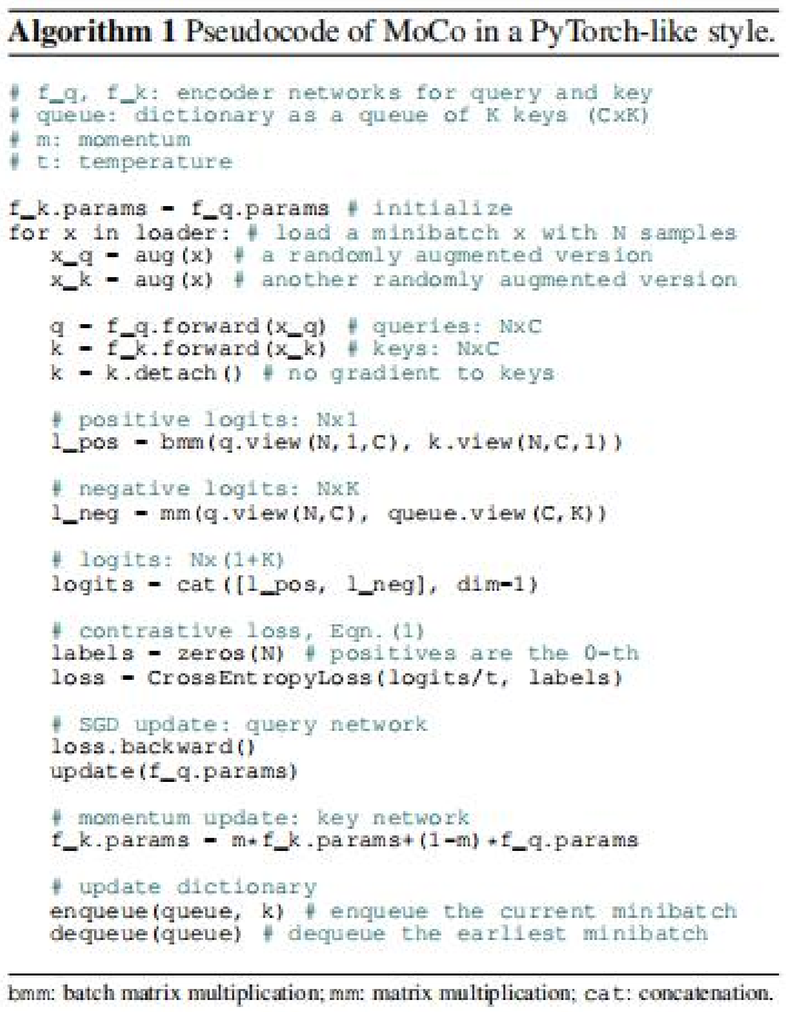
Sec. 4 empirically compares these three mechanisms.

与先前机制的关系 MoCo 是一种使用对比损失的通用机制。我们将其与图 2 中的两个现有通用机制进行比较。它们在词典大小和一致性上表现出不同的特性。

反向传播的端到端更新是一种自然机制（例如，[29,46,36,63,2,35]，图 2a）。它使用当前小批量中的样本作为字典，因此键被一致地编码（由同一组编码器参数组成）。但是字典大小与小批量大小相结合，受到 GPU 内存大小的限制。它也受到了大规模（大的批次）小批量优化的挑战[25]。最近的一些方法[46，36，2]是基于局部位置驱动的代理任务，其中字典的大小可以通过多个位置来扩大。但是这些代理任务可能需要特殊的网络设计，例如修补输入[46]或定制感受野大小[2]，这可能会使这些网络向下游任务的迁移复杂化。

另一种机制是[61]提出的存储库方法（图 2b）。存储库由数据集中所有样本的表示形式组成。每一个小批量的字典都是从内存库中随机抽取的，没有反向传播，因此可以支持较大的字典大小。但是，在存储库中的一个样本表示在最后一次看到时才被更新，所以采样的键基本上是关于编码器在过去 epoch 中多个不同步骤的，因此不太一致。在[61]中对存储库采用了动量更新。它的动量更新是基于同一个样本的表示，而不是编码器。这个动量更新与我们的方法无关，因为 MoCo 并不跟踪每个样本。此外，我们的方法具有更高的存储效率，并且可以在十亿级的数据上进行训练，这对于存储库（前面提到的 memory bank.方法）来说是很困难的。

第 4 部分将实验对比这三种机制。



Algorithm 1 Pseudocode of MoCo in a PyTorch-like style.

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
    x_q = aug(x) # a randomly augmented version
    x_k = aug(x) # another randomly augmented version

    q = f_q.forward(x_q) # queries: NxC
    k = f_k.forward(x_k) # keys: NxC
    k = k.detach() # no gradient to keys

    # positive logits: Nx1
    l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

    # negative logits: NxK
    l_neg = mm(q.view(N,C), queue.view(C,K))

    # logits: Nx(1+K)
    logits = cat([l_pos, l_neg], dim=1)

    # contrastive loss, Eqn.(1)
    labels = zeros(N) # positives are the 0-th
    loss = CrossEntropyLoss(logits/t, labels)

    # SGD update: query network
    loss.backward()
    update(f_q.params)

    # momentum update: key network
    f_k.params = m*f_k.params+(1-m)*f_q.params

    # update dictionary
    enqueue(queue, k) # enqueue the current minibatch
    dequeue(queue) # dequeue the earliest minibatch
```

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

3.3. Pretext Task

Contrastive learning can drive a variety of pretext tasks. As the focus of this paper is not on designing a new pretext task, we use a simple one mainly following the instance discrimination task in [61], to which some recent works [63, 2] are related.

Following [61], we consider a query and a key as a positive pair if they originate from the same image, and otherwise as a negative sample pair. Following [63, 2], we take two random “views” of the same image under random data augmentation to form a positive pair. The queries and keys are respectively encoded by their encoders, fq and fk. The encoder can be any convolutional neural network [39].

Algorithm 1 provides the pseudo-code of MoCo for this pretext task. For the current mini-batch, we encode the queries and their corresponding keys, which form the positive sample pairs. The negative samples are from the queue.

3.3 代理任务

对比学习可以驱动各种各样的借口任务。由于本文的重点不在于设计一个新的代理任务，因此我们使用了一个简单的借口任务，主要遵循文献[61]中的实例判别任务，并与一些近期的工作[63，2]相关。

在[61]之后，如果查询和键来自同一个图像，我们将它们视为正对，否则将

它们视为负样本对。在文献[63，2]的基础上，我们在随机数据增广的情况下，对同一幅图像取两个随机“视图”，形成一个正对。查询和键分别由其编码器 fq 和 fk 进行编码。编码器可以是任何卷积神经网络[39]。

算法 1 为这个代理任务提供了 MoCo 的伪代码。对于当前的小批量，我们对查询及其对应的键进行编码，它们构成了正样本对。负样本来自于队列。

Technical details. We adopt a ResNet [33] as the encoder, whose last fully-connected layer (after global average pooling) has a fixed-dimensional output (128-D [61]). This output vector is normalized by its L2-norm [61]. This is the representation of the query or key. The temperature τ in Eqn.(1) is set as 0.07 [61]. The data augmentation setting follows [61]: a 224×224-pixel crop is taken from a randomly resized image, and then undergoes random color jittering, random horizontal flip, and random grayscale conversion, all available in PyTorch's torchvision package.

技术细节 我们采用 ResNet[33]作为编码器，其最后一个全连接层（全局平均池化层之后）具有固定维度的输出（128-D[61]）。这个输出向量被其 L2 范数规范化[61]。这是查询或键的表示形式。等式（1）中的τ设置为 0.07[61]。数据增强设置如下[61]：从随机调整大小的图像中截取 224×224 像素的裁剪，然后进行随机颜色抖动、随机水平翻转和随机灰度转换，这些都在 Pythorch 的 torchvision 软件包中提供。

Shuffling BN. Our encoders fq and fk both have Batch Normalization (BN) [37] as in the standard ResNet [33]. In experiments, we found that using BN prevents the model from learning good representations, as similarly reported in [35] (which avoids using BN). The model appears to “cheat” the pretext task and easily finds a low-loss solution. This is possibly because the intra-batch communication among samples (caused by BN) leaks information.

We resolve this problem by shuffling BN. We train with multiple GPUs and perform BN on the samples independently for each GPU (as done in common practice). For the key encoder fk, we shuffle the sample order in the current mini-batch before distributing it among GPUs (and shuffle back after encoding); the sample order of the mini-batch for the query encoder fq is not altered. This ensures the batch statistics used to compute a query and its positive key come from two different subsets. This effectively tackles the cheating issue and allows training to benefit from BN.

We use shuffled BN in both our method and its end-to-end ablation counterpart (Figure 2a). It is irrelevant to the memory bank counterpart (Figure 2b), which does not suffer from this issue because the positive keys are from different mini-batches in the past.

洗牌 BN 我们的编码器 fq 和 fk 两者都有批归一化（BN）[37]，如标准 ResNet[33]所示。在实验中，我们发现使用 BN 会阻止模型学习良好的表示，如[35]中所述（应避免使用 BN）。该模型似乎“作弊（欺骗）”了代理任务，很容易找到一个低损失的解决方案。这可能是因为样本之间的批内通信（由 BN 引起）泄漏了

信息。

我们通过洗牌 BN 来解决这个问题。我们使用多个 GPU 进行训练，并对每个 GPU 独立地对样本执行 BN（就像在一般实践中所做的那样）。对于键编码器 fk，我们在将当前小批量中的样本顺序在 gpu 之间分发之前进行洗牌（在编码后重新洗牌）；查询编码器 fq 的小批量样本顺序没有改变。这可以确保用于计算查询的批处理统计信息及其正键来自两个不同的子集。这有效地解决了“作弊（欺骗）”问题，并使训练受益于 BN

我们在我们的方法和相对应的端到端方法（图 2a 方法）中使用洗牌 BN。它与相对应的存储库方法（图 2b）无关，后者不受此问题的影响，因为正密钥来自过去不同的小批量。

4. Experiments

We study unsupervised training performed in:

ImageNet-1M (IN-1M): This is the ImageNet [11] training set that has ~ 1.28 million images in 1000 classes (often called ImageNet-1K; we count the image number instead, as classes are not exploited by unsupervised learning). This dataset is well-balanced in its class distribution, and its images generally contain iconic view of objects.

Instagram-1B (IG-1B): Following [44], this is a dataset of ~ 1 billion (940M) public images from Instagram. The images are from ~ 1500 hashtags [44] that are related to the ImageNet categories. This dataset is relatively uncurated comparing to IN-1M, and has a long-tailed, unbalanced distribution of real-world data. This dataset contains both iconic objects and scene-level images.

Training. We use SGD as our optimizer. The SGD weight decay is 0.0001 and the SGD momentum is 0.9. For IN-1M, we use a mini-batch size of 256 (N in Algorithm 1) in 8 GPUs, and an initial learning rate of 0.03. We train for 200 epochs with the learning rate multiplied by 0.1 at 120 and 160 epochs [61], taking ~ 53 hours training ResNet-50. For IG-1B, we use a mini-batch size of 1024 in 64 GPUs, and a learning rate of 0.12 which is exponentially decayed by 0.9× after every 62.5k iterations (64M images). We train for 1.25M iterations (~ 1.4 epochs of IG-1B), taking ~ 6 days for ResNet-50.

【4.实验】

我们研究在以下领域进行的无监督训练：

ImageNet-1M (IN-1M)：这是 ImageNet[11]训练集，在 1000 个类中有 128 万张图片（通常称为 ImageNet-1K；我们计算图像数量，因为无监督学习不会利用类别）。这个数据集的类分布非常均衡，它的图像通常含有对象的图像视图。

Instagram-1B（IG-1B）：以下是[44]，这是在 Instagram 中的~ 10 亿（940M）公共图像的数据集。这些图像来自~ 1500 个与 ImageNet 类别相关的标签[44]。与 IN-1M 相比，该数据集相对未评级，并且具有长尾、不平衡的真实世界数据分布。此数据集包含图标对象和场景级图像。10 亿

训练 我们使用 SGD 作为我们的优化器。SGD 重量衰减为 0.0001，SGD 动量为 0.9。对于 IN-1M，我们在 8 个 gpu 中使用 256 个小批量（在算法 1 中），

初始学习率为0.03。我们训练200个时期，120和160个时期的学习率乘以0.1[61]，用~ 53 小时的训练 ResNet-50。对于 IG-1B，我们在 64 个 gpu 中使用 1024 个小批量，学习率为 0.12，在每 62.5k 次迭代（64M 图像）后以指数衰减 0.9 倍。我们为 125 万次迭代（IG-1B 的 1.4 个周期）进行训练，ResNet-50 需要 6 天的时间。

4.1. Linear Classification Protocol

We first verify our method by linear classification on frozen features, following a common protocol. In this subsection we perform unsupervised pre-training on IN-1M. Then we freeze the features and train a supervised linear classifier (a fully-connected layer followed by softmax). We train this classifier on the global average pooling features of a ResNet, for 100 epochs. We report 1-crop, top-1 classification accuracy on the ImageNet validation set.

For this classifier, we perform a grid search and find the optimal initial learning rate is 30 and weight decay is 0 (similarly reported in [56]). These hyper-parameters perform consistently well for all ablation entries presented in this subsection. These hyper-parameter values imply that the feature distributions (e.g., magnitudes) can be substantially different from those of ImageNet supervised training, an issue we will revisit in Sec. 4.2.

4.1. 线性分类协议

我们首先通过冻结特征的线性分类来验证我们的方法，并且遵循一个共同的协议。在本小节中，我们在 In-1M 上进行无监督的预训练，然后冻结特征并训练一个有监督的线性分类器（一个带有 softmax 的全连接层）。我们对这个分类器在 ResNet 全局平均池化特征下进行 100 个 epoch 的训练。我们在 ImageNet 验证集上报告了 1-crop，top-1 分类精度。

对于该分类器，我们进行网格搜索，发现最佳初始学习率为 30，权重衰减为 0（类似于文献[56]中所述）。这些超参数在本小节介绍的所有消融实验中都表现良好。这些超参数值意味着特征分布（例如幅度 magnitudes）可能与 ImageNet 监督训练的特征分布有很大不同，我们将在 4.2 部分重新讨论这个问题。

Ablation: contrastive loss mechanisms. We compare the three mechanisms that are illustrated in Figure 2. To focus on the effect of contrastive loss mechanisms, we implement all of them in the same pretext task as described in Sec. 3.3. We also use the same form of InfoNCE as the contrastive loss function, Eqn.(1). As such, the comparison is solely on the three mechanisms.

The results are in Figure 3. Overall, all three mechanisms benefit from a larger K. A similar trend has been observed in [61, 56] under the memory bank mechanism, while here we show that this trend is more general and can be seen in all mechanisms. These results support our motivation of building a large dictionary.

The end-to-end mechanism performs similarly to MoCo when K is small. However, the dictionary size is limited by the mini-batch size due to the end-to-end requirement. Here the largest mini-batch a high-end machine (8 Volta 32GB GPUs) can afford is 1024. More essentially, large mini-batch training is an open problem [25]: we found it necessary to use the linear learning rate scaling rule [25] here, without which the accuracy drops (by ~ 2% with a 1024 mini-batch). But optimizing with a larger mini-batch is harder [25], and it is questionable whether the trend can be extrapolated into a larger K even if memory is sufficient.

The memory bank [61] mechanism can support a larger dictionary size. But it is 2.6% worse than MoCo. This is inline with our hypothesis: the keys in the memory bank are from very different encoders all over the past epoch and they are not consistent. Note the memory bank result of 58.0% reflects our improved implementation of [61].

消融：对比损失机制。我们比较图 2 中所示的三种机制。为了研究对比损失机制的效果，我们在 3.3 节所述的同一个代理任务中实现了所有这些机制。我们还使用了与对比损失函数相同的 InfoNCE 形式，Eqn.（1）。因此，比较仅限于这三种机制。

结果如图 3 所示。总的来说，这三种机制都受益于更大的 K。在[61，56]中，在存储库机制下也观察到了类似的趋势，而这里我们表明这种趋势更普遍，并且可以在所有机制中可以看到。这些结果支持了我们建立大型词典的动机

类似地，当 K 较小时，端到端机制的性能与 MoCo 相似。但是，由于端到端的要求，字典大小受小批量大小的限制。在这里，高端机器（8 volta32 gb gpu）所能承受的最大的小批量是 1024。更重要的是，大规模的小批量培训是一个开放的问题[25]：我们发现了使用线性学习率缩放规则[25]是必要的，没有这个规则，准确度会下降（1024 个小批量减少~ 2%）。但是，用一个更大的小批量进行优化比较困难[25]，而且即使内存足够，这种趋势是否可以外推到更大的 K 中也是值得怀疑的。

存储库[61]机制可以支持更大的字典大小。但比 MoCo 还差 2.6%。这与我们的假设是一致的：存储库中的键来自于不同的编码器，并且它们并不一致。注意，存储库结果为 58.0%，这反映了我们改进了对[61]的实现。

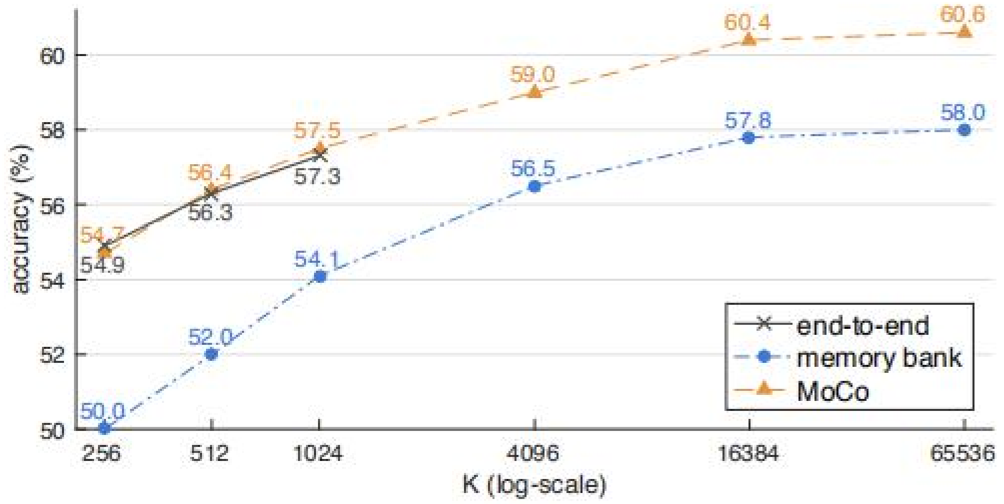
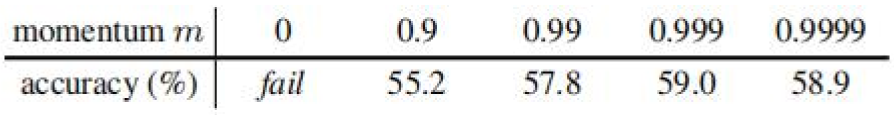


Figure 3. **Comparison of three contrastive loss mechanisms** under the ImageNet linear classification protocol. We adopt the same pretext task (Sec. 3.3) and only vary the contrastive loss mechanism (Figure 2). The number of negatives is K in memory

bank and MoCo, and is KK1 in end-to-end (offset by one because the positive key is in the same mini-batch). The network is ResNet-50.
图 3 ImageNet 线性分类协议下三种对比损失机制的比较。我们采取同样的代理任务（第 3.3 部分所述）并且只改变对比损失机制（如图 2 中）。负类的数量在存储库机制和 MoCo 机制中是 K，在端到端机制是 K−1（因为正键在同一个小批量中，所以偏移一个）。网络使用 ResNet-50。

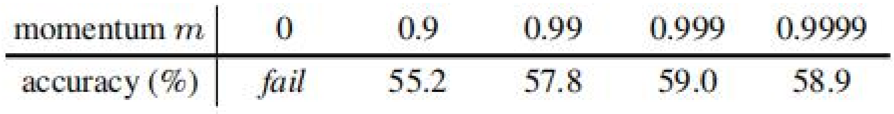
Ablation: momentum. The table below shows ResNet-50 accuracy with different MoCo momentum values (m in Eqn.(2)) used in pre-training (K = 4096 here) :



momentum m	0	0.9	0.99	0.999	0.9999
accuracy (%)	*fail*	55.2	57.8	59.0	58.9

It performs reasonably well when m is in 0.99 ~ 0.9999, showing that a slowly progressing (i.e., relatively large momentum) key encoder is beneficial. When m is too small (e.g., 0.9), the accuracy drops considerably; at the extreme of no momentum (m is 0), the training loss oscillates and fails to converge. These results support our motivation of building a consistent dictionary.

消融：动量。下表显示了在 ResNet-50 预训练（K=4096）中使用的不同 MoCo 动量值（m 在等式（2））的精度：



momentum m	0	0.9	0.99	0.999	0.9999
accuracy (%)	*fail*	55.2	57.8	59.0	58.9

m 在 0.99~0.9999 的情况下表现相当好，表明一个缓慢改进的（即，相对较大的动量）键编码器是有益的。当 m 太小（例如 0.9）时，精度会显著下降；在无动量的极端（m 为 0），训练损失振荡，无法收敛。这些结果支持我们建立一个一致的词典的动机。

Comparison with previous results. Previous unsupervised learning methods can differ substantially in model sizes. For a fair and comprehensive comparison, we report accuracy vs. #parameters trade-offs. Besides ResNet-50 (R50) [33], we also report its variants that are 2× and 4× wider (more channels), following [38]. We set K = 65536 and m = 0.999. Table 1 is the comparison.

MoCo with R50 performs competitively and achieves 60.6% accuracy, better than all competitors of similar model sizes (~ 24M). MoCo benefits from larger models and achieves 68.6% accuracy with R50w4×.

Notably, we achieve competitive results using a standard ResNet-50 and require no specific architecture designs, e.g., patchified inputs [46, 35], carefully tailored receptive fields [2], or combining two networks [56]. By using an architecture that is not customized for the pretext task, it is easier to transfer features to a variety of visual tasks and make comparisons, studied in the next subsection.

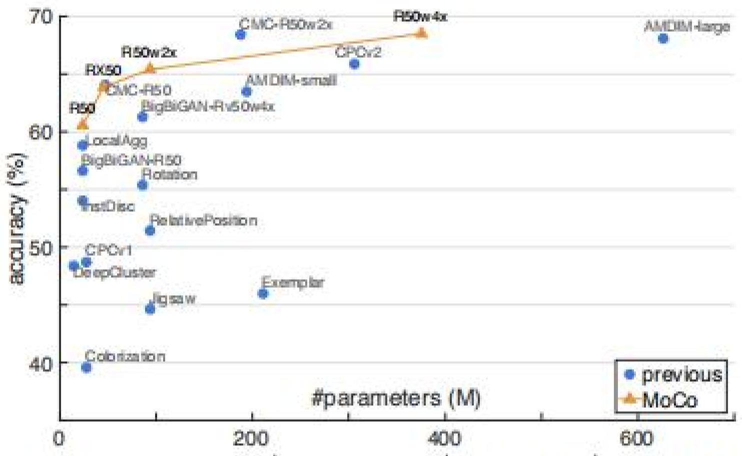
This paper’s focus is on a mechanism for general contrastive learning; we do not explore orthogonal factors (such as specific pretext tasks) that may further improve accuracy. As an example, “MoCo v2” [8], an extension of a preliminary version of this manuscript, achieves 71.1% accuracy with R50 (up from 60.6%), given small changes on the data augmentation and output projection head [7]. We believe that this additional result shows the generality and robustness of the MoCo framework.

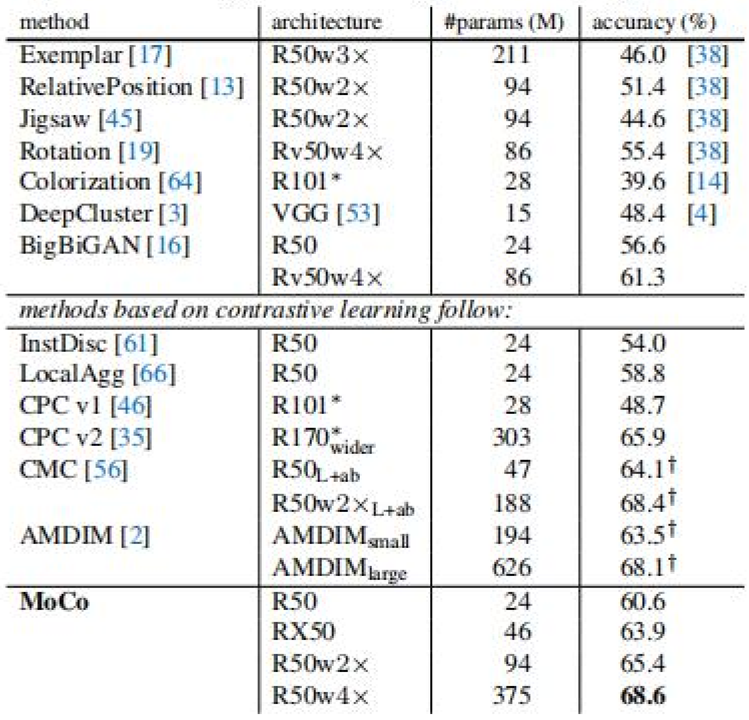
与之前的结果进行比较 以前的无监督学习方法在模型大小上可能有很大的不同。为了进行公正和全面的比较，我们报告了准确性与参数偏差。除了 ResNet-50（R50）[33]，我们还报告了其 2×和 4×宽（更多通道）的变体，如文献[38]所示。我们设置 K = 65536 和 m = 0.999。表 1 是比较。

具有 R50 的 MoCo 性能优异，精度达到 60.6%，优于所有模型同样大小(24M)的竞争对手。MoCo 得益于更大的模型和实现 68.6%的精度在 R50w4+上。

值得注意的是，我们使用统一标准的 ResNet-50 实现了有竞争力的结果，并且不需要特定的架构设计，例如，拼接输入[46，35]，精心定制的感受野[2]，或组合两个网络[56]。通过使用一个不是为代理任务定制的架构，可以更容易地将特征迁移到各种视觉任务中并进行比较，下一小节将对此进行研究。

本文的重点是一般性对比学习的一种机制；我们不探讨可能进一步提高准确性的正交因素（如特定的代理任务）。例如，“MoCo v2”[8]是本手稿初稿的一个扩展，在数据扩充和输出映射头上做了微小的改动，R50（从 60.6%）提高到 71.1%的准确率[7]。我们相信这个额外的结果显示了 MoCo 框架的通用性和健壮性。





method	architecture	#params (M)	accuracy (%)
Exemplar [17]	R50w3×	211	46.0 [38]
RelativePosition [13]	R50w2×	94	51.4 [38]
Jigsaw [45]	R50w2×	94	44.6 [38]
Rotation [19]	Rv50w4×	86	55.4 [38]
Colorization [64]	R101*	28	39.6 [14]
DeepCluster [3]	VGG [53]	15	48.4 [4]
BigBiGAN [16]	R50	24	56.6
	Rv50w4×	86	61.3
methods based on contrastive learning follow:			
InstDisc [61]	R50	24	54.0
LocalAgg [66]	R50	24	58.8
CPC v1 [46]	R101*	28	48.7
CPC v2 [35]	$\text{R170}^{*}_{\text{wider}}$	303	65.9
CMC [56]	$\text{R50}_{\text{L+ab}}$	47	64.1†
	$\text{R50w2×}_{\text{L+ab}}$	188	68.4†
AMDIM [2]	$\text{AMDIM}_{\text{small}}$	194	63.5†
	$\text{AMDIM}_{\text{large}}$	626	68.1†
MoCo	R50	24	60.6
	RX50	46	63.9
	R50w2×	94	65.4
	R50w4×	375	**68.6**

Table 1. **Comparison under the linear classification protocol on ImageNet.** The figure visualizes the table. All are reported as unsupervised pre-training on the ImageNet-1M training set, followed by supervised linear classification trained on frozen features, evaluated on the validation set. The parameter counts are those of the feature extractors. We compare with improved reimplementations if available (referenced after the numbers).
Notations: R101*/R170* is ResNet-101/170 with the last residual stage removed [14, 46, 35], and R170 is made wider [35]; Rv50 is a reversible net [23], RX50 is ResNeXt-50-32×8d [62].
†: Pre-training uses FastAutoAugment [40] that is supervised by ImageNet labels.
表 1。ImageNet 上线性分类协议下的比较。图中显示了表格。在 ImageNet-1M 训练集上，所有这些都被报告为无监督的预训练，然后在验证集上对冻结特征进行有监督的线性分类训练。参数计数是特征提取层的参数计数。我们将与改进后的重新实现（如有）进行比较。
注：R101/R170 为 ResNet-101/170，去掉了最后一个残留级[14,46,35]，R170 变

宽[35]；Rv50 为可逆网[23]，RX50 为 ResNeXt-50-32×8d[62]。
†：训练前使用 FastAutoAugment[40]这是由 ImageNet 标签监督的。

4.2. Transferring Features

A main goal of unsupervised learning is to learn features that are transferrable. ImageNet supervised pre-training is most influential when serving as the initialization for fine-tuning in downstream tasks (e.g., [21, 20, 43, 52]). Next we compare MoCo with ImageNet supervised pre-training, transferred to various tasks including PASCAL VOC [18], COCO [42], etc. As prerequisites, we discuss two important issues involved [31]: normalization and schedules.

4.2 迁移特征

无监督学习的主要目标是学习可迁移的特征。ImageNet 上有监督的预训练在作为下游任务微调的初始化时最具影响力（例如[21,20,43,52]）。接下来，我们将 MoCo 与 ImageNet 上有监督的预训练进行比较，迁移到包括 PASCAL VOC[18]、COCO[42]等各种任务。作为先决条件，我们讨论了涉及到的两个重要问题：规范化和调度表（schedules）。

Normalization. As noted in Sec. 4.1, features produced by unsupervised pre-training can have different distributions compared with ImageNet supervised pre-training. But a system for a downstream task often has hyper-parameters (e.g., learning rates) selected for supervised pre-training. To relieve this problem, we adopt feature normalization during fine-tuning: we fine-tune with BN that is trained (and synchronized across GPUs [49]), instead of freezing it by an affine layer [33]. We also use BN in the newly initialized layers (e.g., FPN [41]), which helps calibrate magnitudes.

We perform normalization when fine-tuning supervised and unsupervised pre-training models. MoCo uses the same hyper-parameters as the ImageNet supervised counterpart.

规范化。如 4.1 部分，.与 ImageNet 有监督的预训练相比，无监督预训练产生的特征具有不同的分布。但是，一个用于下游任务的系统通常会选择超参数（如学习率）来进行有监督预训练。为了解决这个问题，我们在微调过程中采用了特征规范化：我们使用经过训练的 BN 进行微调（并在 gpu 上同步[49]），而不是通过仿射层冻结它[33]。我们还在新初始化的层中使用 BN（例如 FPN[41]），这有助于校准震级。

当对有监督和无监督的预训练模型进行微调时，我们执行规范化。MoCo 也用同样的超参数来作为 ImageNet 有监督的对比方。

Schedules. If the fine-tuning schedule is long enough, training detectors from random initialization can be strong baselines, and can match the ImageNet supervised counterpart on COCO [31]. Our goal is to investigate transferability of features, so our experiments are on controlled schedules, e.g., the 1× (~ 12 epochs) or 2× schedules [22] for COCO, in contrast to 6×~ 9× in [31]. On smaller datasets like

VOC, training longer may not catch up [31].

Nonetheless, in our fine-tuning, MoCo uses the same schedule as the ImageNet supervised counterpart, and random initialization results are provided as references.

Put together, our fine-tuning uses the same setting as the supervised pre-training counterpart. This may place MoCo at a disadvantage. Even so, MoCo is competitive. Doing so also makes it feasible to present comparisons on multiple datasets/tasks, without extra hyper-parameter search.

调度表 如果微调时间足够长，随机初始化的训练检测器可以是强基线，并且可以匹配 COCO 上 ImageNet 有监督的检测器[31]。我们的目标是研究特征的可转移性。我们的实验是在受控的调度表上进行的，例如 COCO 的 1×（~ 12 个时代）或 2×时间表[22]，而不是[31]中的 6×~ 9×时间表。在 VOC 这样的小数据集上，训练时间过长可能赶不上[31]。

然而，在我们的微调中，MoCo 使用相同的调度表作为 ImageNet 有监督的对比，同时随机初始化结果也作为参考提供。

总而言之，我们微调使用的设置与有监督的预训练对应的设置相同。这可能会使 MoCo 处于劣势。即便如此，MoCo 还是有竞争力的。这样做也使得在多个数据集/任务上进行比较变得可行，而无需额外的超参数搜索。

4.2.1 PASCAL VOC Object Detection

Setup. The detector is Faster R-CNN [52] with a backbone of R50-dilated-C5 or R50-C4 [32] (details in appendix), with BN tuned, implemented in [60]. We fine-tune all layers end-to-end. The image scale is [480, 800] pixels during training and 800 at inference. The same setup is used for all entries, including the supervised pre-training baseline. We evaluate the default VOC metric of AP50 (i.e., IoU threshold is 50%) and the more stringent metrics of COCO-style AP and AP75. Evaluation is on the VOC test2007 set.

Ablation: backbones. Table 2 shows the results fine-tuned on trainval07+12 (~16.5k images). For R50-dilatedC5 (Table 2a), MoCo pre-trained on IN-1M is comparable to the supervised pre-training counterpart, and MoCo pre-trained on IG-1B surpasses it. For R50-C4 (Table 2b), MoCo with IN-1M or IG-1B is better than the supervised counterpart: up to +0.9 AP50, +3.7 AP, and +4.9 AP75.

Interestingly, the transferring accuracy depends on the detector structure. For the C4 backbone, by default used in existing ResNet-based results [14, 61, 26, 66], the advantage of unsupervised pre-training is larger. The relation between pre-training vs. detector structures has been veiled in the past, and should be a factor under consideration.

Ablation: contrastive loss mechanisms. We point out that these results are partially because we establish solid detection baselines for contrastive learning. To pin-point the gain that is solely contributed by using the MoCo mechanism in contrastive learning, we fine-tune the models pre-trained with the end-to-end or memory bank mechanism, both implemented by us (i.e., the best ones in Figure 3), using the same fine-tuning setting as MoCo.

These competitors perform decently (Table 3). Their AP and AP75 with the C4 backbone are also higher than the ImageNet supervised counterpart's, c.f . Table 2b, but other metrics are lower. They are worse than MoCo in all metrics. This shows the benefits of MoCo. In addition, how to train these competitors in larger-scale data is an open question, and they may not benefit from IG-1B.

Comparison with previous results. Following the competitors, we fine-tune on trainval2007 (~ 5k images) using the C4 backbone. The comparison is in Table 4.

For the AP50 metric, no previous method can catch up with its respective supervised pre-training counterpart. MoCo pre-trained on any of IN-1M, IN-14M (full ImageNet), YFCC-100M [55], and IG-1B can outperform the supervised baseline. Large gains are seen in the more stringent metrics: up to +5.2 AP and +9.0 AP75. These gains are larger than the gains seen in trainval07+12 (Table 2b).

4.2.1 PASCAL VOC 目标检测

设置 检测器 Faseter R-CNN[52]，主干为 R50-Expanded-C5 或 R50-C4[32]（详见附录），BN 调谐，在[60]中实现。我们对所有层进行端到端微调。训练时图像比例尺为[480,800]像素，推断时为 800 像素。所有条目都使用相同的设置，包括有监督的预训练基线。我们评估 AP50 的默认 VOC 指标（即 IoU 阈值为 50%），以及更严格的 COCO 式 AP 和 AP75。评估在 VOC test2007 集合上。

消融：主干。表 2 显示了在 trainval07+12（16.5k 图像）上进行微调的结果。对于 R50-dilatedC5 (表 2a），在 IN-1M 上进行预训练的 MoCo 可与有监督的预训练对手相媲美，而在 IG-1B 上进行预训练的 MoCo 更为出色。对于 R50-C4（表 2b），带有 IN-1M 或 IG-1B 的 MoCo 比有监督的对照更好：提高+0.9AP50、+3.7AP 和+4.9AP75。

有趣的是，迁移的精度取决于检测器的结构。对于 C4 主干，在现有的基于 ResNet 的结果中默认使用[14,61,26,66]，无监督预训练的优势更大。训练前与检测器结构之间的关系在过去一直被掩盖，应该是一个正在考虑的因素。

消融：对比损失机制。我们指出这些结果部分是因为我们为对比学习建立了坚实的检测基线。为了确定在对比学习中使用 MoCo 机制所带来的收益，我们使用与 MoCo 相同的微调设置，对使用端到端或存储库机制预训练的模型进行微调，这两种机制都是由我们实现的（即图 3 中最好的）。

这些竞争对手表现不错（表 3）。他们 C4 主干上的 AP 和 AP75 也优于 ImageNet 上的有监督，c.f.表 2b，但其他指标更低。他们在所有指标上都比 MoCo 差。这说明了 MoCo 的好处。此外，如何在更大规模的数据方面培训这些竞争对手是一个开放的问题，他们可能不会从 IG-1B 中获益。

与之前的结果进行比较。类比竞争对手，我们使用 C4 主干对 trainval2007（~5k 图像）进行微调。比较见表 4。

对于 AP50 指标，之前没有任何一种方法可以赶上其各自的有监督预训练方

法。MoCo 在 IN-1M、IN-14M（全图像网）、YFCC-100M[55]和 IG-1B 中的任何一个进行了预训练，其性能可以超过监督基线。在更严格的指标中可以看到大幅增长：最高+5.2 AP 和+9.0 AP75。这些收益大于 trainval07+12（表 2b）中的收益。

pre-train	AP_{50}	AP	AP_{75}
random init.	64.4	37.9	38.6
super. IN-1M	81.4	54.0	59.1
MoCo IN-1M	81.1 (−0.3)	54.6 (+0.6)	59.9 (+0.8)
MoCo IG-1B	81.6 (+0.2)	55.5 (+1.5)	61.2 (+2.1)

(a) Faster R-CNN, R50-**dilated-C5**

pre-train	AP_{50}	AP	AP_{75}
random init.	60.2	33.8	33.1
super. IN-1M	81.3	53.5	58.8
MoCo IN-1M	81.5 (+0.2)	55.9 (+2.4)	62.6 (+3.8)
MoCo IG-1B	82.2 (+0.9)	57.2 (+3.7)	63.7 (+4.9)

(b) Faster R-CNN, R50-**C4**

Table 2. **Object detection fine-tuned on PASCAL VOC** trainval07+12. Evaluation is on test2007: AP50 (default VOC metric), AP (COCO-style), and AP75, averaged over 5 trials. All are fine-tuned for 24k iterations (~ 23 epochs). In the brackets are the gaps to the ImageNet supervised pre-training counterpart. In green are the gaps of at least +0.5 point.
表 2 在 PASCAL VOC trainval07+12 上微调目标检测。评估是在 test2007，AP50（默认 VOC 指标）、AP（COCO 风格）和 AP75 上进行的，平均 5 次试验。所有这些都针对 24k 次迭代（23 个时期）进行了微调。括号中是与 ImageNet 监督的预训练的差距。绿色表示间隙至少为+0.5 点。

	R50-dilated-C5			R50-C4		
pre-train	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}
end-to-end	79.2	52.0	56.6	80.4	54.6	60.3
memory bank	79.8	52.9	57.9	80.6	54.9	60.6
MoCo	**81.1**	**54.6**	**59.9**	**81.5**	**55.9**	**62.6**

Table 3. **Comparison of three contrastive loss mechanisms** on PASCAL VOC object detection, fine-tuned on trainval07+12 and evaluated on test2007 (averages over 5 trials). All models are implemented by us (Figure 3), pre-trained on IN-1M, and fine-tuned using the same settings as in Table 2.
表 3 比较 PASCAL VOC 目标检测的三种对比损失机制，在 trainval07+12 上进行微调，并在 test2007 上进行评估（平均 5 次试验）。所有模型都由我们实现（图 3），在 IN-1M 上进行了预先培训，并使用与表 2 中相同的设置进行了微调。

	AP_{50}					AP	AP_{75}	
pre-train	RelPos, by [14]	Multi-task [14]	Jigsaw, by [26]	LocalAgg [66]	**MoCo**	**MoCo**	Multi-task [14]	**MoCo**
super. IN-1M	74.2	74.2	70.5	74.6	74.4	42.4	44.3	42.7
unsup. IN-1M	66.8 (−7.4)	70.5 (−3.7)	61.4 (−9.1)	69.1 (−5.5)	74.9 (+0.5)	46.6 (+4.2)	43.9 (−0.4)	50.1 (+7.4)
unsup. IN-14M	-	-	69.2 (−1.3)	-	75.2 (+0.8)	46.9 (+4.5)	-	50.2 (+7.5)
unsup. YFCC-100M	-	-	66.6 (−3.9)	-	74.7 (+0.3)	45.9 (+3.5)	-	49.0 (+6.3)
unsup. IG-1B	-	-	-	-	75.6 (+1.2)	47.6 (+5.2)	-	51.7 (+9.0)

Table 4. **Comparison with previous methods on object detection fine-tuned on PASCAL VOC** trainval2007. Evaluation is on test2007. The ImageNet supervised counterparts are from the respective papers, and are reported as having the same

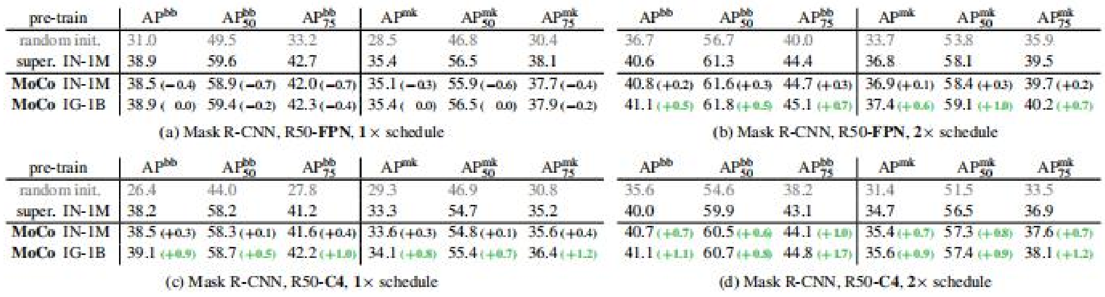
structure as the respective unsupervised pre-training counterparts. All entries are based on the C4 backbone. The models in [14] are R101 v2 [34], and others are R50. The RelPos (relative position) [13] result is the best single-task case in the Multi-task paper [14]. The Jigsaw [45] result is from the ResNet-based implementation in [26]. Our results are with 9k-iteration fine-tuning, averaged over 5 trials. In the brackets are the gaps to the ImageNet supervised pre-training counterpart. In green are the gaps of at least +0.5 point.
表 4 与之前在 PASCAL VOC trainval2007 上微调的目标检测方法进行比较。评估在 test2007 上。ImageNet 有监督方法来自各自的论文，并且被设置为具有与相应的无监督的预训练相同的结构。所有模型都基于 C4 主干。[14]中的方法为 R101 v2[34]，其他的方法为 R50。RelPos（相对位置）[13]的结果是多任务论文[14]中最好的单任务案例。Jigsaw[45]的结果来自于[26]中基于 ResNet 的实现。我们的结果是 9k 迭代微调，平均 5 次试验。括号中是与 ImageNet 监督预训练的差距。绿色表示间隙至少为+0.5 点。

4.2.2 COCO Object Detection and Segmentation

Setup. The model is Mask R-CNN [32] with the FPN [41] or C4 backbone, with BN tuned, implemented in [60]. The image scale is in [640, 800] pixels during training and is 800 at inference. We fine-tune all layers end-to-end. We fine-tune on the train2017 set (~ 118k images) and evaluate on val2017. The schedule is the default 1× or 2× in [22].
Results. Table 5 shows the results on COCO with the FPN (Table 5a, b) and C4 (Table 5c, d) backbones. With the 1× schedule, all models (including the ImageNet supervised counterparts) are heavily under-trained, as indicated by the ~ 2 points gaps to the 2 × schedule cases. With the 2 × schedule, MoCo is better than its ImageNet supervised counterpart in all metrics in both backbones.

4.2.2 COCO 目标检测与分割

设置。该模型是 Mask R-CNN[32]，具有 FPN[41]或 C4 主干，BN 调谐，在[60]中实现。在训练期间，图像的比例是[640,800]像素，推断时是 800 像素。我们对所有层进行端到端微调。我们对 train2017 系列（118k 图像）进行了微调，并对 val2017 进行了评估。调度表是[22]中默认的 1×或 2×。
结果。表 5 显示了使用 FPN（表 5a，b）和使用 C4（表 5c，d）主干的 COCO 的结果。对于 1×调度，所有模型（包括 ImageNet 监督的对应模型）都严重训练不足，对于 2×调度有~ 2 点差距所示。在 2×调度下，MoCo 在两个主干网的所有指标上都优于 ImageNet 上的有监督。



pre-train	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
super. IN-1M	38.9	59.6	42.7	35.4	56.5	38.1
MoCo IN-1M	38.5 (−0.4)	58.9 (−0.7)	42.0 (−0.7)	35.1 (−0.3)	55.9 (−0.6)	37.7 (−0.4)
MoCo IG-1B	38.9 (0.0)	59.4 (−0.2)	42.3 (−0.4)	35.4 (0.0)	56.5 (0.0)	37.9 (−0.2)

(a) Mask R-CNN, R50-**FPN**, 1× schedule

pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	36.7	56.7	40.0	33.7	53.8	35.9
super. IN-1M	40.6	61.3	44.4	36.8	58.1	39.5
MoCo IN-1M	40.8 (+0.2)	61.6 (+0.3)	44.7 (+0.3)	36.9 (+0.1)	58.4 (+0.3)	39.7 (+0.2)
MoCo IG-1B	41.1 (+0.5)	61.8 (+0.5)	45.1 (+0.7)	37.4 (+0.6)	59.1 (+1.0)	40.2 (+0.7)

(b) Mask R-CNN, R50-**FPN**, 2× schedule

pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	26.4	44.0	27.8	29.3	46.9	30.8
super. IN-1M	38.2	58.2	41.2	33.3	54.7	35.2
MoCo IN-1M	38.5 (+0.3)	58.3 (+0.1)	41.6 (+0.4)	33.6 (+0.3)	54.8 (+0.1)	35.6 (+0.4)
MoCo IG-1B	39.1 (+0.9)	58.7 (+0.5)	42.2 (+1.0)	34.1 (+0.8)	55.4 (+0.7)	36.4 (+1.2)

(c) Mask R-CNN, R50-**C4**, 1× schedule

pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	35.6	54.6	38.2	31.4	51.5	33.5
super. IN-1M	40.0	59.9	43.1	34.7	56.5	36.9
MoCo IN-1M	40.7 (+0.7)	60.5 (+0.6)	44.1 (+1.0)	35.4 (+0.7)	57.3 (+0.8)	37.6 (+0.7)
MoCo IG-1B	41.1 (+1.1)	60.7 (+0.8)	44.8 (+1.7)	35.6 (+0.9)	57.4 (+0.9)	38.1 (+1.2)

(d) Mask R-CNN, R50-**C4**, 2× schedule

Table 5. **Object detection and instance segmentation fine-tuned on COCO:** bounding-box AP (APbb) and mask AP (APmk) evaluated on val2017. In the brackets are the gaps to the ImageNet supervised pre-training counterpart. In green are the gaps of at least +0.5 point.
表 5。目标检测和实例分割在 COCO 上进行了微调：在 val2017 上进行了评估边界框 AP（APbb）和 mask AP（APmk）。括号中是与 ImageNet 监督预训练的差距。绿色表示间隙至少为+0.5 点。

4.2.3 More Downstream Tasks

Table 6 shows more downstream tasks (implementation details in appendix). Overall, MoCo performs competitively with ImageNet supervised pre-training:

COCO keypoint detection: supervised pre-training has no clear advantage over random initialization, whereas MoCo outperforms in all metrics.

COCO dense pose estimation [1]: MoCo substantially outperforms supervised pre-training, e.g., by 3.7 points in APdp 75, in this highly localization-sensitive task.

LVIS v0.5 instance segmentation [27]: this task has ~1000 long-tailed distributed categories. Specifically in LVIS for the ImageNet supervised baseline, we find fine-tuning with frozen BN (24.4 APmk) is better than tunable BN (details in appendix). So we compare MoCo with the better supervised pre-training variant in this task. MoCo with IG-1B surpasses it in all metric.

Cityscapes instance segmentation [10]: MoCo with IG-1B is on par with its supervised pre-training counterpart in APmk, and is higher in APmk 50 .

Semantic segmentation: On Cityscapes [10], MoCo outperforms its supervised pre-training counterpart by up to 0.9 point. But on VOC semantic segmentation, MoCo is worse by at least 0.8 point, a negative case we have observed.

4.2.3 更多下游任务

表 6 显示了更多的下游任务（实现细节见附录）。总的来说，MoCo 的表现很有竞争力比 ImageNet 上的有监督的预训练。

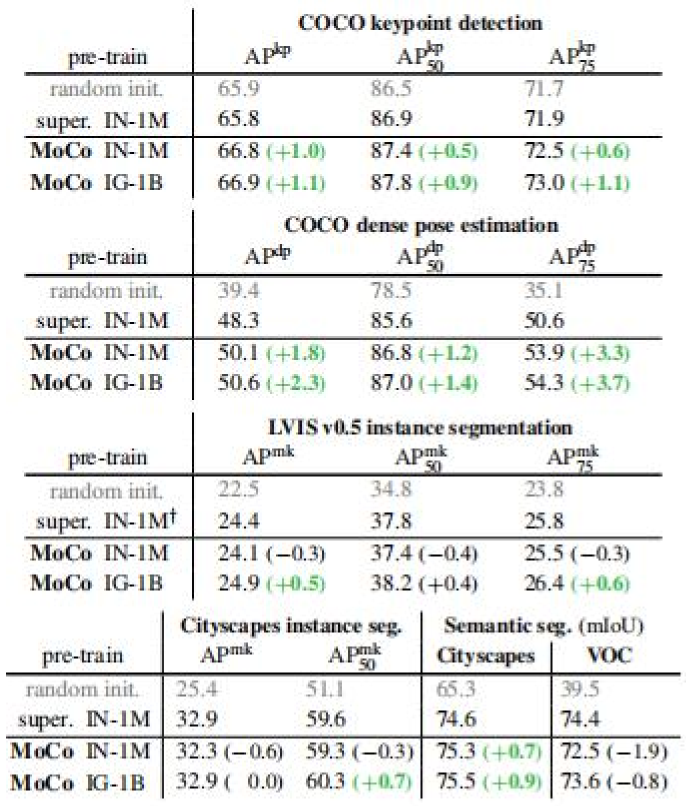
COCO 关键点检测：监督预训练与随机初始化相比没有明显优势，而 MoCo 在所有指标上都优于随机初始化。

COCO 稠密姿态估计[1] ：在这项高度本地化敏感的任务中，MoCo 在 APdp75 中的表现明显优于监督预训练，提高了 3.7 分。

LVIS v0.5 实例分割[27]：本任务有~1000 个长尾分布类别。特别是在 ImageNet 监督基线的 LVIS 中，我们发现使用冻结 BN（24.4apmk）进行微调优于可调整 BN（详见附录）。因此，我们在这个任务中比较了 MoCo 和更好的监督预训练变体。采用 IG-1B 的 MoCo 在所有指标上都优于它。

城市景观实例分割[10] ：带有 IG-1B 的 MoCo 与其监督预训练在 APmk 中的相当，在 APmk50 中更高。

语义分割：在城市景观[10]，MoCo 比其监督预训练高出 0.9 分。但在 VOC 语义分割上，MoCo 至少差 0.8 个百分点，这是我们观察到的一个负面情况。



pre-train	COCO keypoint detection		
	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}
random init.	65.9	86.5	71.7
super. IN-1M	65.8	86.9	71.9
MoCo IN-1M	66.8 (+1.0)	87.4 (+0.5)	72.5 (+0.6)
MoCo IG-1B	66.9 (+1.1)	87.8 (+0.9)	73.0 (+1.1)

pre-train	COCO dense pose estimation		
	AP^{dp}	AP^{dp}_{50}	AP^{dp}_{75}
random init.	39.4	78.5	35.1
super. IN-1M	48.3	85.6	50.6
MoCo IN-1M	50.1 (+1.8)	86.8 (+1.2)	53.9 (+3.3)
MoCo IG-1B	50.6 (+2.3)	87.0 (+1.4)	54.3 (+3.7)

pre-train	LVIS v0.5 instance segmentation		
	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	22.5	34.8	23.8
super. IN-1M†	24.4	37.8	25.8
MoCo IN-1M	24.1 (−0.3)	37.4 (−0.4)	25.5 (−0.3)
MoCo IG-1B	24.9 (+0.5)	38.2 (+0.4)	26.4 (+0.6)

pre-train	Cityscapes instance seg.		Semantic seg. (mIoU)	
	AP^{mk}	AP^{mk}_{50}	Cityscapes	VOC
random init.	25.4	51.1	65.3	39.5
super. IN-1M	32.9	59.6	74.6	74.4
MoCo IN-1M	32.3 (−0.6)	59.3 (−0.3)	75.3 (+0.7)	72.5 (−1.9)
MoCo IG-1B	32.9 (0.0)	60.3 (+0.7)	75.5 (+0.9)	73.6 (−0.8)

Table 6. **MoCo vs. ImageNet supervised pre-training, fine-tuned on various tasks.** For each task, the same architecture and schedule are used for all entries (see appendix). In the brackets are the gaps to the ImageNet supervised pre-training counterpart. In green are the gaps of at least +0.5 point.
†: this entry is with BN frozen, which improves results; see main text.
表 6。MoCo vs.ImageNet 监督预训练，对各种任务进行了微调。对于每个任务，所有方法都使用相同的体系结构和时间表（见附录）。括号中是与 ImageNet 监督预训练的差距。绿色表示间隙至少为+0.5 点。
†：此条目已冻结 BN，这提高了结果；请参阅正文。

Summary. In sum, MoCo can outperform its ImageNet supervised pre-training counterpart in 7 detection or segmentation tasks.5 Besides, MoCo is on par on Cityscapes instance segmentation, and lags behind on VOC semantic segmentation; we show another comparable case on iNaturalist [57] in appendix. Overall, MoCo has largely closed the gap between unsupervised and supervised representation learning in multiple vision tasks.

Remarkably, in all these tasks, MoCo pre-trained on IG-1B is consistently better than MoCo pre-trained on IN-1M. This shows that MoCo can perform well on this large-scale, relatively uncurated dataset. This represents a scenario towards real-world unsupervised learning.

总结。总的来说，MoCo 可以在 7 个检测或分割方面优于 ImageNet 监督预训练。还有，MoCo 在城市景观实例分割上不相上下，而在 VOC 语义分割方面则相对落后；附录中给出了另一个关于 iNaturalist[57]的可比案例。总体而言，MoCo 在

多个视觉任务中基本上弥补了无监督和有监督表征学习之间的差距。[5]

值得注意的是，在所有这些任务中，MoCo 在 IG-1B 的预训练始终优于在 IN-1M 上的预训练，这表明 MoCo 可以在这个大规模的、相对未评级的数据集上表现良好。这代表了一种面向现实世界的无监督学习的场景。

5. Discussion and Conclusion

Our method has shown positive results of unsupervised learning in a variety of computer vision tasks and datasets. A few open questions are worth discussing. MoCo's improvement from IN-1M to IG-1B is consistently noticeable but relatively small, suggesting that the larger-scale data may not be fully exploited. We hope an advanced pretext task will improve this. Beyond the simple instance discrimination task [61], it is possible to adopt MoCo for pretext tasks like masked auto-encoding, e.g., in language [12] and in vision [46]. We hope MoCo will be useful with other pretext tasks that involve contrastive learning.

【5. 讨论与结论】

我们的方法在各种计算机视觉任务和数据集中显示了无监督学习的积极效果。有几个开放性问题值得讨论。MoCo 从 IN-1M 到 IG-1B 的改进一直是显著的，但相对较小，这表明更大规模的数据可能没有得到充分利用。我们希望一个先进的代理任务将改善这一点。除了简单的实体识别任务[61]，还可以将 MoCo 用于 masked auto-encoding 等代理任务，例如在语言[12]和视觉[46]中。我们希望 MoCo 对其他涉及对比学习的代理任务也有帮助。

A. Appendix

A.1. Implementation: Object detection backbones

The R50-dilated-C5 and R50-C4 backbones are similar to those available in Detectron2 [60]: (i) R50-dilated-C5: the backbone includes the ResNet conv5 stage with a dilation of 2 and stride 1, followed by a 3×3 convolution (with BN) that reduces dimension to 512. The box prediction head consists of two hidden fully-connected layers. (ii) R50-C4: the backbone ends with the conv4 stage, and the box prediction head consists of the conv5 stage (including global pooling) followed by a BN layer.

A.2. Implementation: COCO keypoint detection

We use Mask R-CNN (keypoint version) with R50-FPN, implemented in [60], fine-tuned on COCO train2017 and evaluated on val2017. The schedule is 2×.

A.3. Implementation: COCO dense pose estimation

We use DensePose R-CNN [1] with R50-FPN, implemented in [60], fine-tuned on COCO train2017 and evaluated on val2017. The schedule is "s1×".

A.附录

A 1 实验：目标检测主干

R50-dilated-C5 和 R50-C4 主干与 Detectron2[60]中提供的主干相似：（i）R50-dilated-C5：主干包括 ResNet conv5 阶段，扩张 2 和步长 1，然后是 3×3 卷积（带 BN），将维度减小到 512。盒预测头由两个完全连接的隐藏层组成。（ii）

R50-C4：主干以 conv4 阶段结束，盒预测头由 conv5 阶段（包括全局池）和 BN 层组成。

A 2 实验：COCO 关键点检测

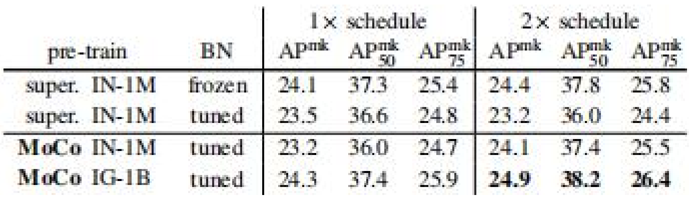
我们使用 Mask R-CNN（关键点版本）和 R50-FPN，在[60]中实现，在 COCO train2017 上进行微调，并在 val2017 上进行评估。调度表是 2×。

A3. 实验：COCO 密集姿态估计

我们使用 DensePose R-CNN[1]和 R50-FPN，在[60]中实现，在 COCO train2017 上进行微调，并在 val2017 进行评估。时间表为“s1×”。

A.4. Implementation: LVIS instance segmentation

We use Mask R-CNN with R50-FPN, fine-tuned in LVIS[27] train v0.5 and evaluated in val v0.5. We follow the baseline in [27] (arXiv v3 Appendix B). LVIS is a new dataset and model designs on it are to be explored. The following table includes the relevant ablations (all are averages of 5 trials):



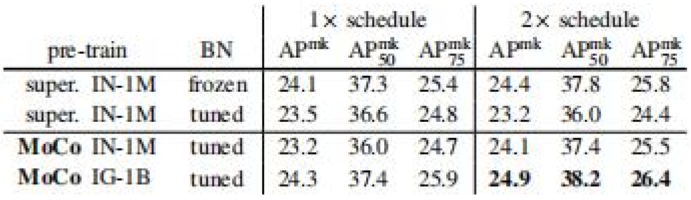
		1× schedule			2× schedule		
pre-train	BN	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
super. IN-1M	frozen	24.1	37.3	25.4	24.4	37.8	25.8
super. IN-1M	tuned	23.5	36.6	24.8	23.2	36.0	24.4
MoCo IN-1M	tuned	23.2	36.0	24.7	24.1	37.4	25.5
MoCo IG-1B	tuned	24.3	37.4	25.9	**24.9**	**38.2**	**26.4**

A supervised pre-training baseline, end-to-end tuned but with BN frozen, has 24.4 APmk. But tuning BN in this baseline leads to worse results and overfitting (this is unlike on COCO/VOC where tuning BN gives better or comparable accuracy). MoCo has 24.1 APmk with IN-1M and 24.9 APmk with IG-1B, both outperforming the supervised pretraining counterpart under the same tunable BN setting. Under the best individual settings, MoCo can still outperform the supervised pre-training case (24.9 vs. 24.4, as reported in Table 6 in Sec 4.2).

A4 实验：LVIS 实例分割

我们使用带有 R50-FPN 的 Mask R-CNN，在 LVIS[27]trainv0.5 中进行微调，并在 valv0.5 中进行评估。我们遵循[27]中的基线（arXiv v3 附录 B）。

LVIS 是一个新的数据集，它的模型设计有待探索。下表包括相关消融（均为 5 次试验的平均值）：



		1× schedule			2× schedule		
pre-train	BN	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
super. IN-1M	frozen	24.1	37.3	25.4	24.4	37.8	25.8
super. IN-1M	tuned	23.5	36.6	24.8	23.2	36.0	24.4
MoCo IN-1M	tuned	23.2	36.0	24.7	24.1	37.4	25.5
MoCo IG-1B	tuned	24.3	37.4	25.9	**24.9**	**38.2**	**26.4**

一个有监督的预训练基线，端到端调整，但 BN 冻结，APmk 为 24.4。但是，在这个基线中调整 BN 会导致更糟糕的结果和过度拟合（这与 COCO/VOC 不同，在 COCO/VOC 中，调整 BN 可以获得更好或相当的精度）。MoCo 的 IN-1M 和 IG-1B 的 APmk 分别为 24.1APMK 和 24.9APMK，在相同的可调 BN 设置下均优于监督预训练。在最佳的个体设置下，MoCo 仍然可以优于有监督的预训练案例（24.9 比 24.4，如第 4.2 节表 6 所示）。

A.5. Implementation: Semantic segmentation

We use an FCN-based [43] structure. The backbone consists of the convolutional layers in R50, and the 3×3 convolutions in conv5 blocks have dilation 2 and stride 1. This is followed by two extra 3×3 convolutions of 256 channels,with BN and ReLU, and then a 1×1 convolution for perpixel classification. The total stride is 16 (FCN-16s [43]). We set dilation = 6 in the two extra 3×3 convolutions, following the large field-of-view design in [6].

Training is with random scaling (by a ratio in [0.5, 2.0]), cropping, and horizontal flipping. The crop size is 513 on VOC and 769 on Cityscapes [6]. Inference is performed on the original image size. We train with mini-batch size 16 and weight decay 0.0001. Learning rate is 0.003 on VOC and is 0.01 on Cityscapes (multiplied by 0.1 at 70-th and 90-th percentile of training). For VOC, we train on the train aug2012 set (augmented by [30], 10582 images) for 30k iterations, and evaluate on val2012. For Cityscapes, we train on the train fine set (2975 images) for 90k iterations, and evaluate on the val set. Results are reported as averages over 5 trials.

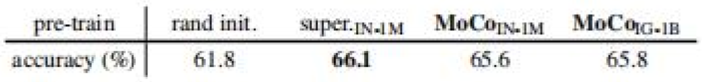
A5 实现：语义分割

我们使用基于 FCN 的[43]结构。主干由 R50 的卷积层组成，convblocks 中的 3×3 卷积具有扩张 2 和步长 1。接下来是两个额外的 3×3 卷积，256 个通道，使用 BN 和 ReLU，然后是 1×1 卷积用于每像素分类。总步幅为 16（FCN-16s[43]）。我们在两个额外的 3×3 卷积中设置了扩张=6，遵循[6]中的大视场设计。

训练包括随机缩放（按[0.5,2.0]的比率）、裁剪和水平翻转。VOC 的作物大小为 513，城市景观为 769[6]。对原始图像大小执行推断。我们训练小批量 16 和重量衰减 0.0001。VOC 的学习率为 0.003，城市景观的学习率为 0.01（在 70%和 90%的培训中乘以 0.1）。对于 VOC，我们在 trainaug2012 数据集（增加了[30]，10582 个图像）上进行了 30k 迭代，并在 val2012 上进行了评估。对于城市景观，我们在 trainfine 数据集（2975 个图像）上训练 90k 次迭代，并在 val 集合上求值。结果报告为 5 次试验的平均值。

A.6. iNaturalist fine-grained classification

In addition to the detection/segmentation experiments in the main paper, we study fine-grained classification on the iNaturalist 2018 dataset [57]. We fine-tune the pre-trained models end-to-end on the train set (~ 437k images, 8142 classes) and evaluate on the val set. Training follows the typical ResNet implementation in PyTorch with 100 epochs. Fine-tuning has a learning rate of 0.025 (vs. 0.1 from scratch) decreased by 10 at the 70-th and 90-th percentile of training. The following is the R50 result:



pre-train	rand init.	super.$_{IN-1M}$	**MoCo$_{IN-1M}$**	**MoCo$_{IG-1B}$**
accuracy (%)	61.8	**66.1**	65.6	65.8

MoCo is ~ 4% better than training from random initialization, and is closely comparable with its ImageNet supervised counterpart. This again shows that MoCo unsupervised pre-training is competitive.

A 6 iNaturalist 细粒度分类

除了主论文中的检测/分割实验外，我们还研究了 iNaturalist 2018 数据集的细粒度分类[57]。我们在训练集（437k 个图像，8142 个类）上对预训练模型进行端到端微调，并对 val 集进行评估。培训遵循 Pythorch 中典型的 ResNet 实现，有 100 个时期。微调的学习率为 0.025（与 0.1 从头开始）相比，在训练的第 70 和第 90 个百分位降低了 10。以下是 R50 结果：

pre-train	rand init.	super.$_{\text{IN-1M}}$	**MoCo**$_{\text{IN-1M}}$	**MoCo**$_{\text{IG-1B}}$
accuracy (%)	61.8	**66.1**	65.6	65.8

MoCo 比随机初始化的训练效果好~ 4%，并且与 ImageNet 监督的训练结果非常接近。这再次表明，MoCo 无监督的预训练是有竞争力的。

A.7. Fine-tuning in ImageNet

Linear classification on frozen features (Sec. 4.1) is a common protocol of evaluating unsupervised pre-training methods. However, in practice, it is more common to fine- tune the features end-to-end in a downstream task. For completeness, the following table reports end-to-end fine-tuning results for the 1000-class ImageNet classification, compared with training from scratch (fine-tuning uses an initial learning rate of 0.03, vs. 0.1 from scratch):

pre-train	random init.	**MoCo**$_{\text{IG-1B}}$
accuracy (%)	76.5	**77.3**

As here ImageNet is the downstream task, the case of MoCo pre-trained on IN-1M does not represent a real scenario (for reference, we report that its accuracy is 77.0% after fine-tuning). But unsupervised pre-training in the separate, unlabeled dataset of IG-1B represents a typical scenario: in this case, MoCo improves by 0.8%.

A7 ImageNet 中的微调

冻结特征的线性分类（第 4.1 部分）是评估无监督预培训方法的通用协议。然而，在实践中，更常见的是在下游任务中端到端地微调特性。为了完整起见，下表报告了 1000 类 ImageNet 分类的端到端微调结果，与从头开始的培训相比（微调使用 0.03 的初始学习率，而从零开始为 0.1）：

pre-train	random init.	**MoCo**$_{\text{IG-1B}}$
accuracy (%)	76.5	**77.3**

由于 ImageNet 是下游任务，因此在 IN-1M 上预先训练的 MoCo 的情况并不代表真实的场景（作为参考，我们报告其精确性在微调后为 77.0%）。但是，在独立的、未标记的 IG-1B 数据集中进行无监督的预训练是一个典型的情况：在这种情况下，MoCo 提高了 0.8%。

A.8. COCO longer fine-tuning

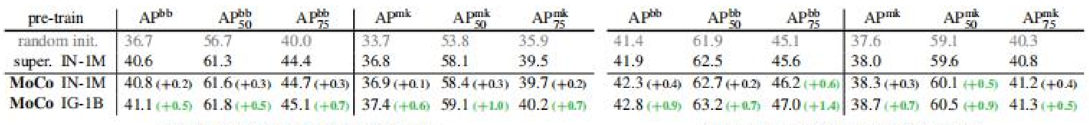
In Table 5 we reported results of the 1× (~ 12 epochs) and 2× schedules on COCO. These schedules were inherited from the original Mask R-CNN paper [32], which could be suboptimal given later advance in the field. In Table A.1, we supplement the results of a 6× schedule (~ 72 epochs)[31] and compare with those of the 2× schedule.

We observe: (i) fine-tuning with ImageNet-supervised pre-training still has improvements (41.9 APbb); (ii) training from scratch largely catches up (41.4 APbb); (iii) the MoCo counterparts improve further (e.g., to 42.8 APbb) and have larger gaps (e.g., +0.9 APbb with 6×, vs. +0.5 APbb with 2×). Table A.1 and Table 5 suggest that the MoCo pre-trained features can have larger advantages than the ImageNet-supervised features when fine-tuning longer.

A 8 COCO 更长时间微调

在表 5 中，我们报告了 1×（~ 12 个时代）和 2×时间表的结果。这些时间表是从原始的 Mask R-CNN 论文[32]继承而来的，考虑到后来在该领域的进展，这可能是次优的。在表 A.1 中，我们补充了 6×时间表（~ 72 个时期）[31]的结果，并与 2×进度表的结果进行了比较。

我们观察到：（i）使用 ImageNet 监督的预训练进行微调仍有改进（41.9 个 APbb）；（ii）从头开始的训练基本上赶上了（41.4 个 APbb）；（iii）MoCo 对应的进一步改进（例如，达到 42.8 APbb），并且差距更大（例如+0.9 APbb 为 6×，而+0.5 APbb 为 2×）。表 A.1 和表 5 表明，当微调时间较长时，MoCo 预训练特征比 ImageNet 监督特征具有更大的优势。



pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	36.7	56.7	40.0	33.7	53.8	35.9
super. IN-1M	40.6	61.3	44.4	36.8	58.1	39.5
MoCo IN-1M	40.8 (+0.2)	61.6 (+0.3)	44.7 (+0.3)	36.9 (+0.1)	58.4 (+0.3)	39.7 (+0.2)
MoCo IG-1B	41.1 (+0.5)	61.8 (+0.5)	45.1 (+0.7)	37.4 (+0.6)	59.1 (+1.0)	40.2 (+0.7)

(a) Mask R-CNN, R50-FPN, 2× schedule

pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	41.4	61.9	45.1	37.6	59.1	40.3
super. IN-1M	41.9	62.5	45.6	38.0	59.6	40.8
MoCo IN-1M	42.3 (+0.4)	62.7 (+0.2)	46.2 (+0.6)	38.3 (+0.3)	60.1 (+0.5)	41.2 (+0.4)
MoCo IG-1B	42.8 (+0.9)	63.2 (+0.7)	47.0 (+1.4)	38.7 (+0.7)	60.5 (+0.9)	41.3 (+0.5)

(b) Mask R-CNN, R50-FPN, 6× schedule

Table A.1. Object detection and instance segmentation fine-tuned on COCO: 2× vs. 6× schedule. In the brackets are the gaps to the ImageNet supervised pre-training counterpart. In green are the gaps of at least +0.5 point.

表 A.1。目标检测和实例分割在 COCO:2××上微调 2× 与 6×时间表。括号中是与 ImageNet 监督的预训练的差距。绿色表示间隙至少为+0.5 点。

A.9. Ablation on Shuffling BN

Figure A.1 provides the training curves of MoCo with or without shuffling BN: removing shuffling BN shows obvious overfitting to the pretext task: training accuracy of the pretext task (dash curve) quickly increases to >99.9%, and the kNN-based validation classification accuracy (solid curve) drops soon. This is observed for both the MoCo and end-to-end variants; the memory bank variant implicitly has different statistics for q and k, so avoids this issue.

These experiments suggest that without shuffling BN, the sub-batch statistics can serve as a “signature” to tell which sub-batch the positive key is in. Shuffling BN can remove this signature and avoid such cheating.

A9 洗牌 BN 的消融实验

图 A.1 给出了 MoCo 有无洗牌 BN 的训练曲线：去除洗牌 BN 对代理任务有明显的过度拟合：代理任务（短划线曲线）的训练精度迅速提高到 99.9%，基于 kNN 的验证分类准确率（实曲线）很快下降。在 MoCo 和端到端变体中都可以观察到这一点；存储库变体隐式地对 q 和 k 有不同的统计信息，因此避免了这个

问题。

这些实验表明，在不改变 BN 的情况下，子批统计信息可以作为一个“签名”来判断正密钥在哪个子批中。洗牌 BN 可以删除该签名并避免这种欺骗。

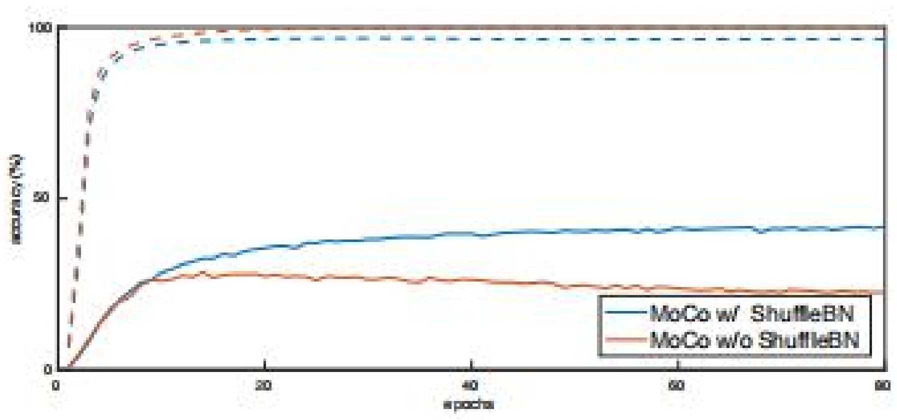


Figure A.1. Ablation of Shuffling BN. Dash: training curve of the pretext task, plotted as the accuracy of (K+1)-way dictionary lookup. Solid: validation curve of a kNN-based monitor [61] (not a linear classifier) on ImageNet classification accuracy. This plot shows the first 80 epochs of training: training longer without shuffling BN overfits more.

图 A.1 洗牌 BN 的消融实验。破折号：代理任务的训练曲线，绘制为（k+1）w 维字典查找的准确度。实线：基于 kNN 的判别器[61]（不是线性分类器）对 ImageNet 分类精度的验证曲线。这个情节展示了训练的前 80 个阶段：训练时间长而不洗牌 BN 过拟合的会更多。

代码解析：

MoCo 论文的本身的代码量不算大，代码的核心文件为 builder.py，该代码实现了 MoCo 论文中的核心算法。

主目录

main_moco.py，用于训练代理任务（train 函数等），即编码器 q，k。

main_lincls.py，用于对预训练好的 MoCo 进行迁移学习最后的线性分类器。

moco 目录

builder.py，包含 moco 模型及其核心算法。

loader.py，用于图像增强。

detection 目录

包含用于将 moco 迁移到目标检测任务的说明使用及代码。