Momentum **Co**ntrast for Unsupervised Visual Representation Learning (CVPR, 2020) - Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, Ross Girshick

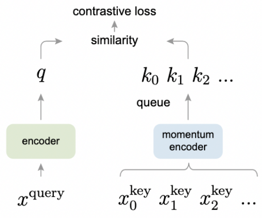
Presentation Author
Mathieu Ravaut

Nanyang Technological University

Table of content

Introduction

- **Contrastive learning** is a paradigm in machine learning which consists in matching together data points which are similar, and pushing apart the ones which are not.
- A **query** is being compared (with a **similarity**) to a set of **k** different **keys**, among which there is a **positive one** and many **negative** ones.



Introduction

- Contrastive learning does not necesarily need labels.
- Typically : use the rest of the batch as (easy) negatives.
- Here, the authors propose **MoCo**, a new contrastive learning objective to pre-train vision models in an unsupervised fashion.
- It breaks the **supervised pre-training on ImageNet** paradigm in vision.
- MoCo produces image representations which transfer very well on many downstream tasks : ImageNet linear classification protocol, PASCAL VOC, COCO, etc.

Introduction

Multiple questions and design choices arise :

1. How to build the (query, positive key) pair ?
   This is referred to as the **pretext task**.
2. How many k negative samples to use ? **MAJOR CHOICE**
3. How to store all these negative samples ? **MAJOR ISSUE**
4. Which similarity function to use ?
5. Which constrastive loss function to use ?

Approach

[1] MoCo's pretext task is ***instance discrimination*** :

- Apply data augmentation on the images : 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. This gives us **multiple views of each image**.
- To build the queries and keys : final layer of a **ResNet-50 encoder**, add a few fully-connected layers with **non-linearities** (**MoCo v2**), and apply **L2-norm**.
- **Same** ResNet-50 architecture for both queries and keys encoders.
- A (query, key) pair is positive if they are encoded views of *the same image*.

Approach

[2] & [3] MoCo's main novelty resides in the way it treats keys :

- Store all keys in a **dictionary**.
- This dictionary is a **queue** of data points from the recent batches : enqueue the current batch, dequeue the last one.
- This ***decouples the number of negatives k from the batch size***.

Concretely :
If the batch size is n, and we use $k = p * n$, the keys dictionary is the current batch + the previous $(p - 1)$ batches :

$$keys = \{x_0, ..., x_{n-1}, x_n, ...x_{2*n-1}, ..., x_{(p-1)*n}, ...x_{p*n-1}\}$$

Approach

There's a new issue : since the keys
$\{x_n, ...x_{2*n-1}, ..., x_{(p-1)*n}, ...x_{p*n-1}\}$
were encoded at the previous iterations, the **encoder weights were different**, thus the keys representations are **inconsistent** with the representation of the queries and the current batch keys.

Approach

To alleviate this issue, the authors propose **momentum update** :

- The key encoder θ_k has the same architecture as the query encoder θ_q, but different weights.
- They key encoder is not updated by gradient descent.
- Instead, it is updated by **momentum updates** : $\theta_k \leftarrow m * \theta_k + (1 - m) * \theta_q$
- m must be really high (0.999) : slow updates.
- We thus have another neural network than the base model, in parallel with it and with the same architecture, but not updated via gradient descent : the **momentum model**.

Approach

The base model is updated with the contrastive loss :

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

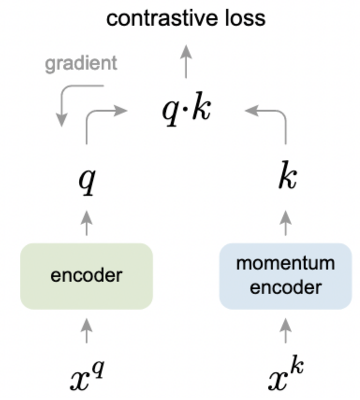
Approach

One last issue : ResNet-50 uses Batch-Norm, but the model appears to learn the pretext task too easily with it.

Thus, MoCo uses **shuffling BatchNorm** : train with multiple GPUs and perform BN on the samples independently for each GPU. They shuffle the sample order in the current mini-batch for the key encoder before distributing it among GPUs, but NOT for the query encoder.

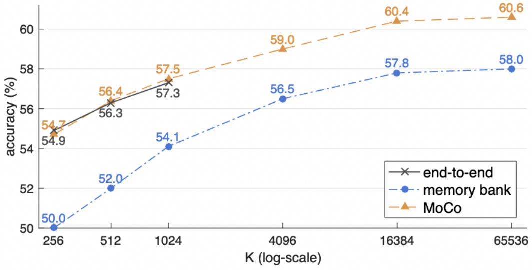
Approach

Model recap :



Results

Linear protocol on ImageNet : train a linear classifier on the image representations obtained with the pre-training.



MoCo enables to use a very large k (65,536 !), and accuracy keeps increasing : +2.9 point when k goes from 1024 to 16384.

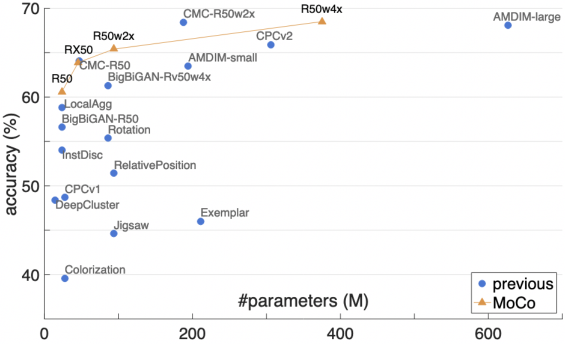
Results

Baselines :

- *end-to-end* : use negatives (keys) from the current mini batch only. **Constrained by the batch size, which means GPU RAM. And optimization in large batch sizes is tricky.**
- *memory bank* : store all data points in a dictionary (memory bank), sample them and update the representations of the sampled ones. No back-propagation. **The representation of a sample is updated when it was last seen.**

Results

Comparison with other pre-training methods on the linear protocol on ImageNet :



Results

Comparison with other pre-training methods on the linear protocol on ImageNet :

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	$64.1^\dagger$
	$\text{R50w2}\times_{\text{L+ab}}$	188	$68.4^\dagger$
AMDIM [2]	$\text{AMDIM}_{\text{small}}$	194	$63.5^\dagger$
	$\text{AMDIM}_{\text{large}}$	626	$68.1^\dagger$
MoCo	R50	24	60.6
	RX50	46	63.9
	R50w2×	94	65.4
	R50w4×	375	**68.6**

Results

On Pascal VOC, MoCo representations are better than supervised pre-training ones :

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**

Results

MoCo representations are great on multiple vision tasks (object detection on COCO, pose estimation, segmentation on LVIS) :

	COCO keypoint detection		
pre-train	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**)
	COCO dense pose estimation		
pre-train	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**)
	LVIS v0.5 instance segmentation		
pre-train	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**)

Conclusion

Takeaways :

- New method for unsupervised pretraining leveraging contrastive learning.
- Enables to scale the number of negatives very high thanks to the queue and the decoupling with the batch size, which drastically improves performance.
- Great performance on multiple vision tasks (classification, detection, etc). Even better than some supervised pre-training methods.
- Consistent gain when going from ImageNet (1M images) to InstaGram (1B images) as pre-training dataset. But still quite a small gain given the data size.

Conclusion

How can this be applied to NLP :

- I am currently pre-training a new unsupervised summarization method based on MoCo. There are gains from scaling k from 1024 to 16384.
- In generation tasks in NLP, we have an **encoder-decoder** model, not just a plain **encoder**. Thus, should we use representations from the encoder, decoder, or both ?
- Exploring how to leverage the momentum encoder : use it for data augmentation ?