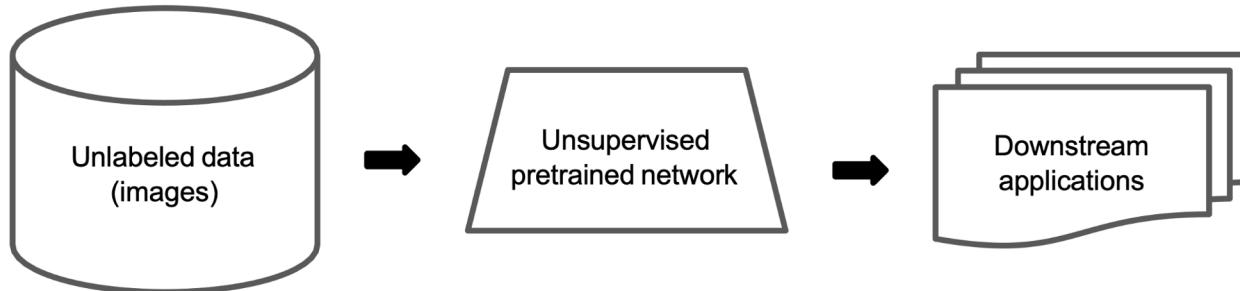


# Self-Supervised Learning論文介紹： MoCo

# MoCo

MoCo全名為“Momentum Contrast for Unsupervised Visual Representation Learning”，是Facebook AI Research團隊於CVPR2020發表的論文。

MoCo目標與SimCLR一樣，都要利用**無標註的資料**訓練模型成為好的特徵抽取器。而這特徵抽取器可以很好的遷移到其他的電腦視覺任務。



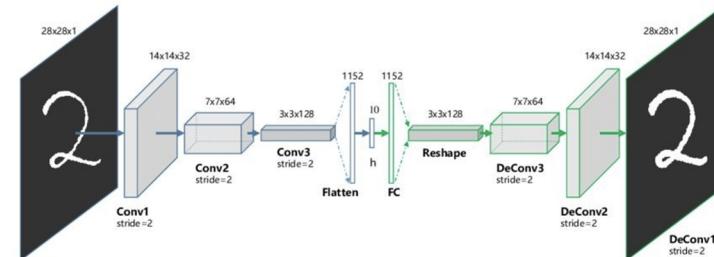
# MoCo並非傳統的無監督式學習方式

與SimCLR一樣，MoCo既不是生成類別（ generative modeling ）也不是利用間接任務（ pretext task ）的訓練方式，而是利用正負樣本的對比學習。

生成類別的缺點：生成高度還原的圖片對於學習特徵並不必要。

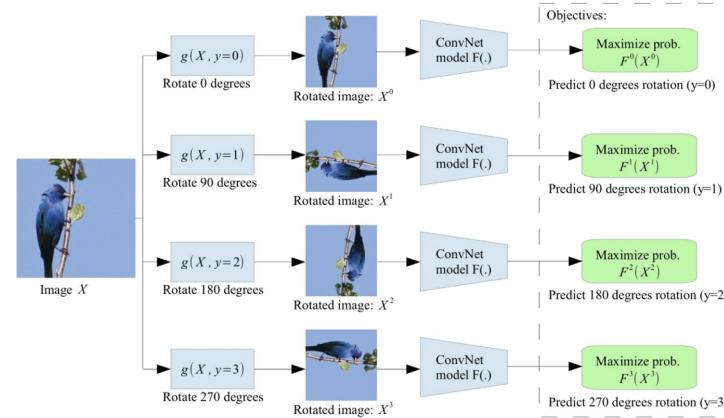
間接任務的缺點：相對來說需要更多人為的知識與前處理。

generative modeling



Autoencoder

pretext task

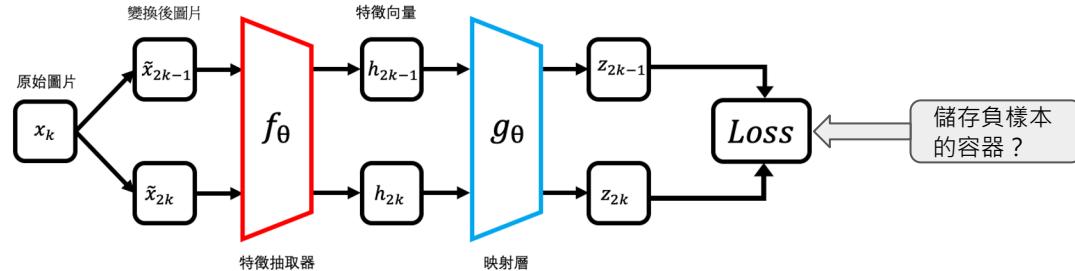


# MoCo貢獻

負樣本數量是對比學習中關鍵的部分，SimCLR使用超大batch size來提供足夠的負樣本，而MoCo則嘗試利用**額外的儲存機制**來解決負樣本數量不足的問題，為了讓儲存的負樣本可以達到數量多且穩定的情況，MoCo又提出一個**非對稱的模型架構**來達成。

大家可以想一下MoCo會是怎樣的架構？怎樣的儲存機制？怎樣的非對稱模型？

SimCLR Framework + 儲存機制=MoCo?

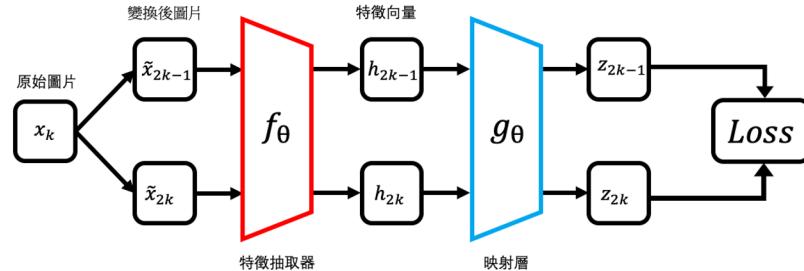


# MoCo模型架構

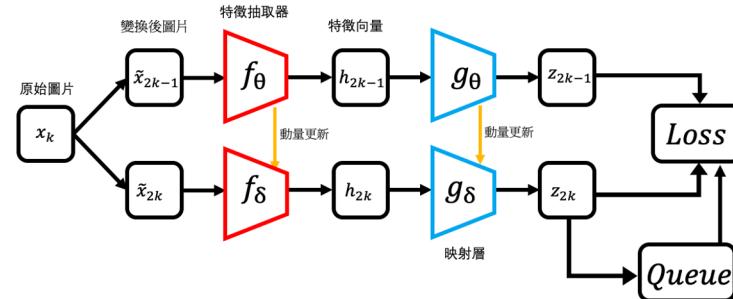
與SimCLR不一樣的地方在於，MoCo設計一個queue (first in first out)來儲存/提供負樣本，而為了達到負樣本數量夠多且穩定的條件，MoCo增加額外的緩慢更新的模型 ( $f_\delta$ 和 $g_\delta$ ) 。

SimCLR負樣本來源：batch，越多顯卡記憶體，就可以有越多的負樣本。  
MoCo負樣本來源：queue，顯卡記憶體不用很多就可以讓queue很大。

SimCLR Framework



MoCo Framework

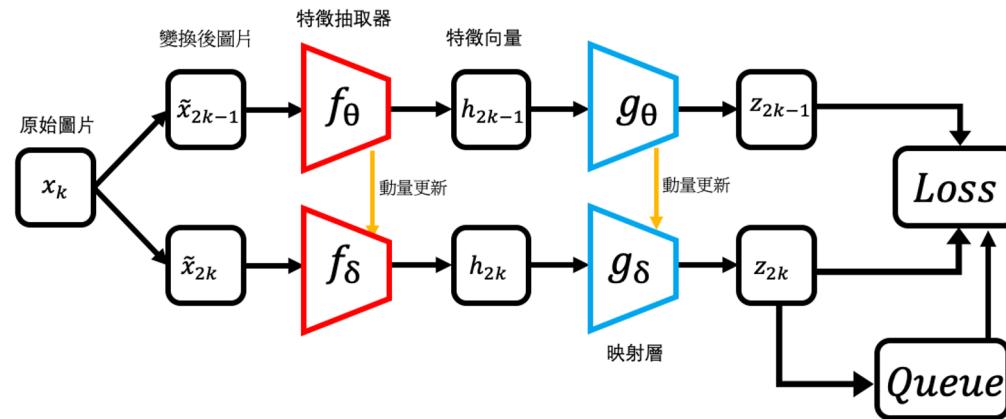


# 為什麼Queue可行

大家可以想一下為什麼queue可以不需要很多的顯卡記憶體，就可以儲存很多的負樣本呢？

提示：queue裡面裝的不是圖片！

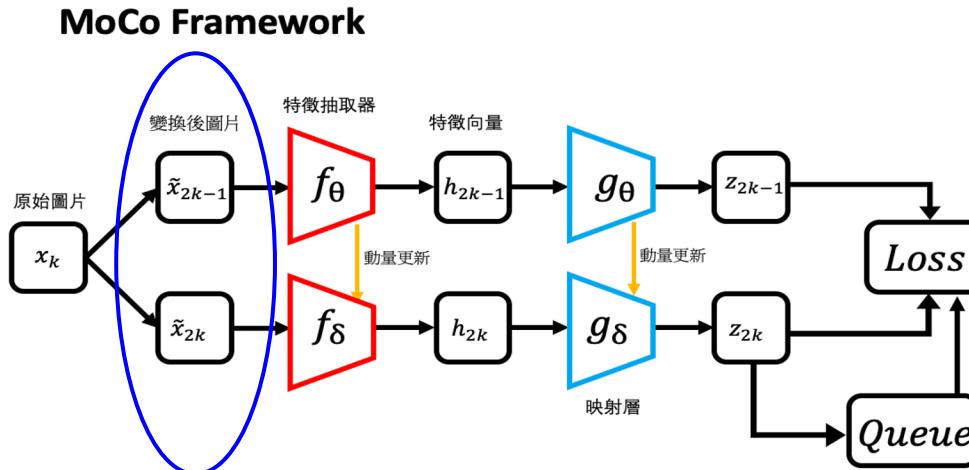
MoCo Framework



# MoCo第一階段的圖片變換

對比學習的第一階段都是圖片變換，要在圖片變換夠多的情況下又保持重要的特徵，讓模型學習圖片不變性。

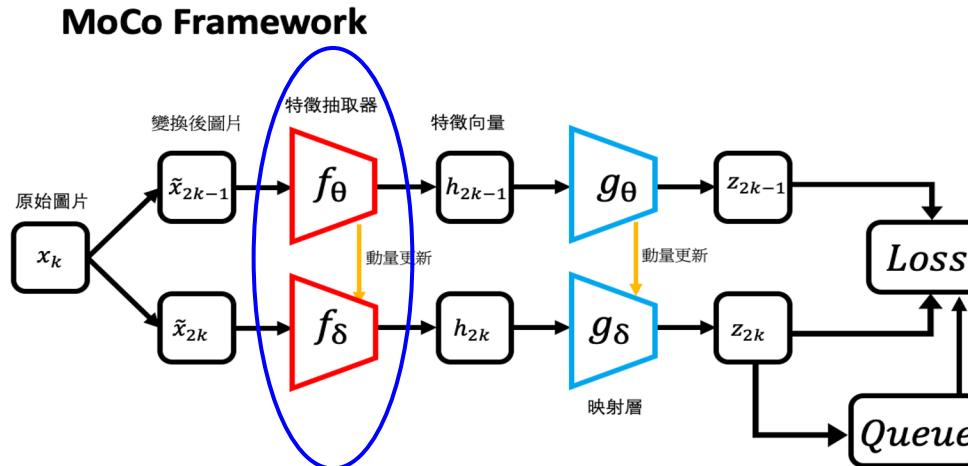
MoCo採取的主要圖片變換也是裁切與顏色抖動，原論文沒有深入研究，但在實作上我們會設跟SimCLR一樣的圖片變換與強度。



# MoCo第二階段的特徵抽取

MoCo使用的特徵抽取器也是採取CNN理念的ResNet架構，需要注意的是上面的 $f_\theta$ 是最後要保留的特徵抽取器，是使用gradients更新，而下面的 $f_\delta$ 是動量更新，不使用gradients。

動量更新： $f_\delta = m * f_\delta + (1-m) * f_\theta$ ，在MoCo中m設為0.99來模擬緩慢更新的 $f_\delta$ 。

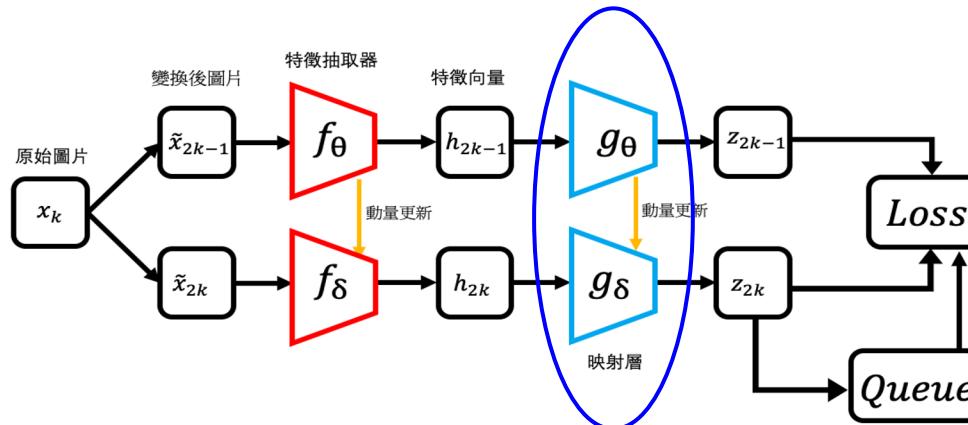


# MoCo第三階段的特徵映射

映射層是SimCLR論文提出的架構，MoCo原論文並沒有映射層，但在SimCLR發布不久後MoCo v2也採取了映射層，而MoCo v2實驗結果顯示映射層在此種模型架構下也可以大幅提升結果。下圖畫的MoCo是有映射層版本。

映射層 $g_{\theta}$ 是用gradients更新，而映射層 $g_{\delta}$ 是採取動量更新， $g_{\delta} = m^*g_{\delta} + (1-m)^*g_{\theta}$ 。

**MoCo Framework**

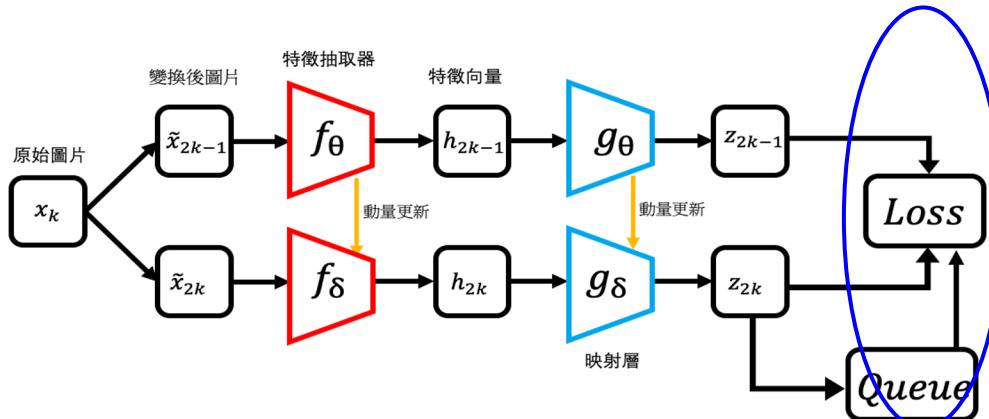


# MoCo第四階段的Loss計算

MoCo用的loss function基本上跟SimCLR一樣，只是負樣本來源是用儲存在queue裡面過去樣本的 $z$ 。

假設queue裡面存有K個負樣本，那MoCo loss想法就是 $z_{\{2k-1\}}$ 在 $1+K$ 個類別中，要分類在 $z_{\{2k\}}$ 這個類別（分在正樣本的類別中）。

## MoCo Framework



$k+$ 是 $q$ 的正樣本，分母的  
 $k_1 \sim k_K$ 是queue裡面儲存的  
負樣本( $k_0=k+$ )。

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

where  $\tau$  is a temperature hyper-parameter per [61]. The sum is over one positive and  $K$  negative samples.

# MoCo的演算法流程

**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
```

f\_q是gradients更新的模型，f\_k是動量更新的模型。

queue: 含有K個負樣本的queue。

輸入圖片經過aug做圖片變換

變換後的圖片經過模型抽取特徵，得到特徵向量

l\_pos: N個樣本特徵向量與其對應的正樣本做內積

l\_neg: N個樣本特徵向量與queue裡面的K個負樣本特徵向量做內積

loss: CrossEntropyLoss，labels=0因為對每個樣本，都是第0類別（正樣本）

用gradients更新完f\_q之後用動量更新f\_k，最後把經過f\_k得到的圖片特徵向量存進去queue當作下一輪的負樣本，如果queue滿了就把最舊的清空。

# 評估模型效果的方式(same as SimCLR)

在Self-Supervised Learning訓練中，因為不使用標註資料，所以訓練過程無法知道準確率如何，只能觀察loss是否下降，那要怎樣評估模型的好壞呢？

**Linear Evaluation Protocol**：在SSL訓練結束後，我們會把**模型f的參數凍結**，並在模型f後面接上全連接層做分類任務，這個階段我們會用有標註的資料訓練全連接層，這樣的方式可以評估模型f的特徵抽取能力好不好。

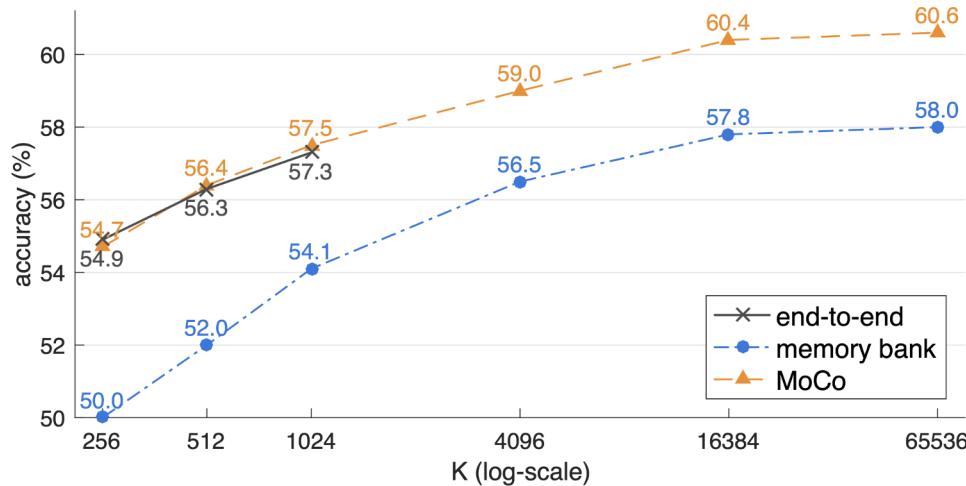
**Fine-tune**：模型f的參數可以一起更新，但通常這樣的評估方式會只使用小部分的標註資料，可能10%或1%訓練資料。

**Transfer Pretrained Model**：把在A資料集訓練好的模型f當作起始參數用在其他的資料集B上。

# MoCo實驗：負樣本數量K

在對比學習中，負樣本的數量決定模型的效果，我們也在SimCLR看到越大的batch size準確率越高，那MoCo呢？更大的queue size會有更好的效果嗎？

MoCo實驗在ImageNet上面的結果顯示，確實K越大準確率越高！符合負樣本越多越好的假設。



# MoCo實驗：動量更新參數m

為什麼MoCo需要額外的緩慢更新的模型？如果更新很快，或是跟SimCLR一樣只有一個模型會如何？透過實驗來觀察更新參數m對實驗結果的影響。

動量更新： $f_{\delta} = m * f_{\delta} + (1-m) * f_{\theta}$

可以看到，模型更新幅度太大（ $m=0.9$ ）會導致效果不好，那使用跟SimCLR一樣的單一模型（ $m=0$ ）呢？

**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 (%)	猜猜看會如何？	55.2	57.8	59.0	58.9

## MoCo實驗：動量更新參數 $m$

MoCo使用queue來存過去的負樣本，這樣的機制會需要這些負樣本變化不能太大，所以其抽取特徵的模型也不能更新太快！

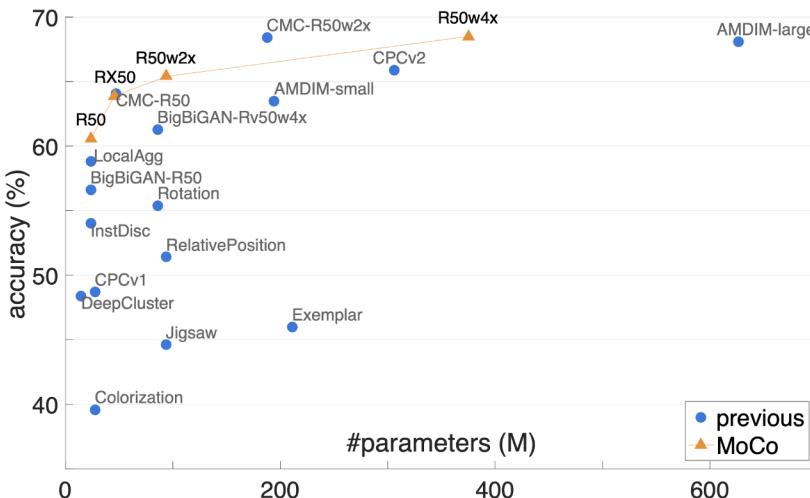
在ImageNet大資料集上，如果像SimCLR這樣的單一模型（ $m=0$ ）會訓練失敗！

**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 (%)	<i>fail</i>	55.2	57.8	59.0	58.9

# MoCo實驗：與其他無監督演算法比較

MoCo不需要使用特殊設計的模型，使用預設ResNet50、 $m=0.999$ 、 $K=65536$ 的實驗設置，在相同模型參數量下可以比大多數的方法都還好，使用更大的模型可以進一步提升效果。



method	architecture	#params (M)	accuracy (%)
Exemplar [17]	R50w3x	211	46.0 [38]
RelativePosition [13]	R50w2x	94	51.4 [38]
Jigsaw [45]	R50w2x	94	44.6 [38]
Rotation [19]	Rv50w4x	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
MoCo	Rv50w4x	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]	R170* <sub>wider</sub>	303	65.9
CMC [56]	R50 <sub>L+ab</sub>	47	64.1 <sup>†</sup>
MoCo	R50w2x <sub>L+ab</sub>	188	68.4 <sup>†</sup>
AMDIM [2]	AMDIM <sub>small</sub>	194	63.5 <sup>†</sup>
MoCo	AMDIM <sub>large</sub>	626	68.1 <sup>†</sup>
MoCo	R50	24	60.6
MoCo	RX50	46	63.9
MoCo	R50w2x	94	65.4
MoCo	R50w4x	375	<b>68.6</b>

監督式大約  
75%

MoCo v2可以從60.6%提  
升到71.1% !

# MoCo實驗：遷移到PASCAL VOC物件偵測任務

MoCo在ImageNet/IG無監督學習特徵 vs Supervised Learning在ImageNet有監督學習特徵。

把訓練完成的pretrained model遷移到PASCAL VOC資料集上面 ( object detection 任務 )，綜合來說MoCo的pretrained model可以表現的比隨機初始參數和 supervised learning的pretrained model還要好。顯示無監督學習的特徵更能遷移到其他資料集或其他任務上面。

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

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

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

(b) Faster R-CNN, R50-C4

# MoCo實驗：遷移到COCO資料集上

延續上一張投影片，換成更加複雜的COCO資料集，MoCo仍舊能在多數情況下贏過監督式學習的pretrained model（只有在ResNet50-FPN, 1x schedule情況下輸給監督式學習，原因是訓練時間不夠長）。

pre-train	AP <sup>bb</sup>	AP <sub>50</sub> <sup>bb</sup>	AP <sub>75</sub> <sup>bb</sup>	AP <sup>mk</sup>	AP <sub>50</sub> <sup>mk</sup>	AP <sub>75</sub> <sup>mk</sup>
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
<b>MoCo</b> 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)
<b>MoCo</b> 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 <sup>bb</sup>	AP <sub>50</sub> <sup>bb</sup>	AP <sub>75</sub> <sup>bb</sup>	AP <sup>mk</sup>	AP <sub>50</sub> <sup>mk</sup>	AP <sub>75</sub> <sup>mk</sup>
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
<b>MoCo</b> 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)
<b>MoCo</b> 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 <sup>bb</sup>	AP <sub>50</sub> <sup>bb</sup>	AP <sub>75</sub> <sup>bb</sup>	AP <sup>mk</sup>	AP <sub>50</sub> <sup>mk</sup>	AP <sub>75</sub> <sup>mk</sup>
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
<b>MoCo</b> 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)
<b>MoCo</b> 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 <sup>bb</sup>	AP <sub>50</sub> <sup>bb</sup>	AP <sub>75</sub> <sup>bb</sup>	AP <sup>mk</sup>	AP <sub>50</sub> <sup>mk</sup>	AP <sub>75</sub> <sup>mk</sup>
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
<b>MoCo</b> 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)
<b>MoCo</b> 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 (AP<sup>bb</sup>) and mask AP (AP<sup>mk</sup>) 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.

# MoCo實驗：一些比較不利的實驗結果

實際上pretrained model遷移到其他的資料集常常會發生進步幅度過小，或是沒有進步的情形，這一部分MoCo實際上也有發生，並且列在原論文的附錄中。遷移學習中使用pretrained model是最常見的方式，但可能不是最好的方式。

在fine tune iNaturalist資料集下，MoCo不管pretrain在ImageNet或是IG上面都比supervised learning pretrained在ImageNet上面還要差。

pre-train	rand init.	super. <sub>IN-1M</sub>	<b>MoCo<sub>IN-1M</sub></b>	MoCo <sub>IG-1B</sub>
accuracy (%)	61.8	<b>66.1</b>	65.6	65.8

在fine tune ImageNet資料集下，MoCo pretrain在IG上的效果比起隨機初始參數進步幅度不大。

pre-train	random init.	<b>MoCo<sub>IG-1B</sub></b>
accuracy (%)	76.5	<b>77.3</b>

## 結論

MoCo作為最成功的對比學習框架之一，提出了以queue儲存負樣本的機制，有效解決對比樣本缺少的情況，並且提出以緩慢更新的額外模型使queue中的負樣本更穩定，避免訓練失敗，如果結合SimCLR中的映射層可以媲美監督式學習的準確率。

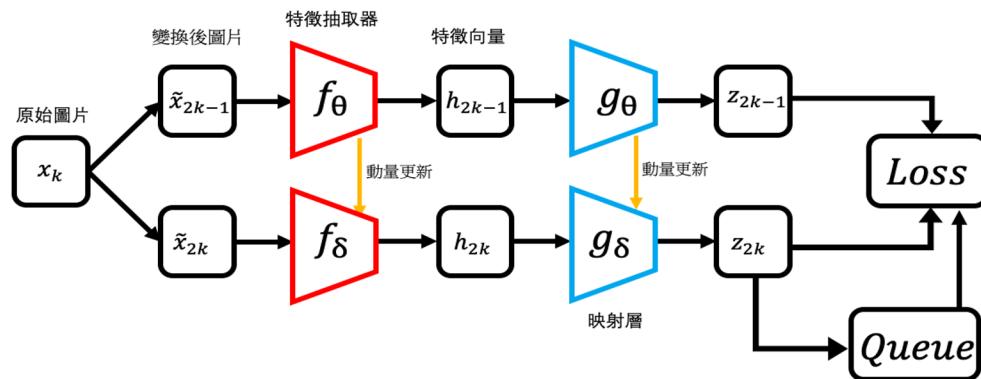
MoCo也做一系列的遷移實驗，證明此方法所訓練出來的模型有好的遷移能力。

# 補充：如何簡單改善MoCo loss

大家可以思考一下，MoCo計算loss的時候，可以怎樣進步？

為什麼只去分類圖片 $\tilde{x}_{2k-1}$ 所產生的特徵向量呢？是不是把變換後的圖片上下對調去計算loss能更加的平均？

## MoCo Framework



## 補充：MoCo v2

前面投影片的實驗結果為MoCo v1（沒有採取高強度圖片變換和映射層），在SimCLR發布後，MoCo採用了SimCLR的圖片變換與映射層設置，並且做實驗與SimCLR比較，可以看到MoCo v2效果大幅進步且可以比SimCLR好（尤其在batch size小的情況下）。

使用ResNet50模型：

case	MLP	aug+	cos	unsup. pre-train epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
<b>MoCo v2</b>	✓	✓	✓	200	256	<b>67.5</b>

*results of longer unsupervised training follow:*

SimCLR [2]	✓	✓	✓	1000	4096	69.3
<b>MoCo v2</b>	✓	✓	✓	800	256	<b>71.1</b>

可以看到SimCLR需要大batch size，相對之下MoCo可以使用小batch size達到一樣的效果。

# 補充：MoCo v2

MoCo v2對各個部分（高強度圖片變換、映射層、lr調整）做單獨的比較實驗，從實驗結果可以看到重要性為，映射層(MLP)>高強度圖片變換(aug+)>lr調整(cos)。

另外有趣的是，在ImageNet上表現好的模型不代表遷移到VOC物件偵測任務上就會比較好，加上遷移的進步幅度較低，顯示遷移學習還有很多改善的空間。

case	unsup. pre-train				ImageNet acc.	VOC detection		
	MLP	aug+	cos	epochs		AP <sub>50</sub>	AP	AP <sub>75</sub>
supervised					76.5	81.3	53.5	58.8
MoCo v1				200	60.6	81.5	55.9	62.6
(a)	✓			200	66.2	82.0	56.4	62.6
(b)		✓		200	63.4	82.2	56.8	63.2
(c)	✓	✓		200	67.3	82.5	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	800	71.1	82.5	57.4	64.0

Table 1. **Ablation of MoCo baselines**, evaluated by ResNet-50 for (i) ImageNet linear classification, and (ii) fine-tuning VOC object detection (mean of 5 trials). “MLP”: with an MLP head; “aug+”: with extra blur augmentation; “cos”: cosine learning rate schedule.

## 補充：MoCo v2

超參數 $\tau$ 也會因為增加了映射層而改變，在MoCo v1中是設置0.07，但是在MoCo v2實驗中發現0.2比0.07還好，這也顯示在深度學習中，更動模型架構或是更換資料集都很有可能需要再次尋找適合的超參數。

	$\tau$	0.07	0.1	0.2	0.3	0.4	0.5
MoCo v1	w/o MLP	60.6	<b>60.7</b>	59.0	58.2	57.2	56.4
MoCo v2	w/ MLP	62.9	64.9	<b>66.2</b>	65.7	65.0	64.3