Unsupervised Learning of Visual Features by Contrasting Cluster Assignments

[Caron et al., NeurIPS 2020]

-Summary-

Introduction (SwAV)

- Current contrastive learning strategy :
  - Compares pairs (positive, true) – (negative, true) to pull (positive, true) and push (negative, true) .
  - Problem : computing all the pairs on a large dataset (ex : ImageNet) is not practical
  - Alternatives:
    - ① : compute contrastive loss based on random batch (widely adopted)
    - ② : **clustering-based methods** discriminating between groups of images with similar features instead of individual images (Caron et al., 2018)
- This paper tries to improve their previous work (②) by changing clustering-based methods.

Introduction (SwAV)

- SwAV (**Sw**apping **A**ssignments between multiple **V**iews of the same image) :
  - **Goal : To learn a useful prototype $\{\boldsymbol{c_1}, \dots, \boldsymbol{c_k}\}$ that discriminate feature vectors $\{\boldsymbol{z_1}, \dots \boldsymbol{z_B}\}$, which goes along with the concept of CL by using swapping assignments.**
  - Method :

1. Image x_n is transformed into an augmented view x_{nt} by transformation $t \sim \mathcal{T}$.
2. x_t is mapped into <u>normalized</u> feature vector $z_{nt} = \frac{f_\theta(x_{nt})}{\|f_\theta(x_{nt})\|_2} \in \mathbb{R}^D$ by model f_θ.
3. Then, we compute a '**code**' q_{nt} by exploiting 'optimal transport' between z_{nt} and prototype vectors $\{c_1, \dots, c_K\} \subset \mathbb{R}^D$

Note : prototype matrix $C = [c_1, \dots c_K] \in \mathbb{R}^{D \times K}$, code matrix $Q = [q_1, \dots q_B] \in \mathbb{R}^{K \times B}$

(where D = dimension of feature z_{nt}, B = batch size, K = the number of polytopes)

Introduction (SwAV)

©: https://www.youtube.com/watch?v=7QmsTleiRLs

- Recall that SwAV clusters multiple samples to the prototypes.

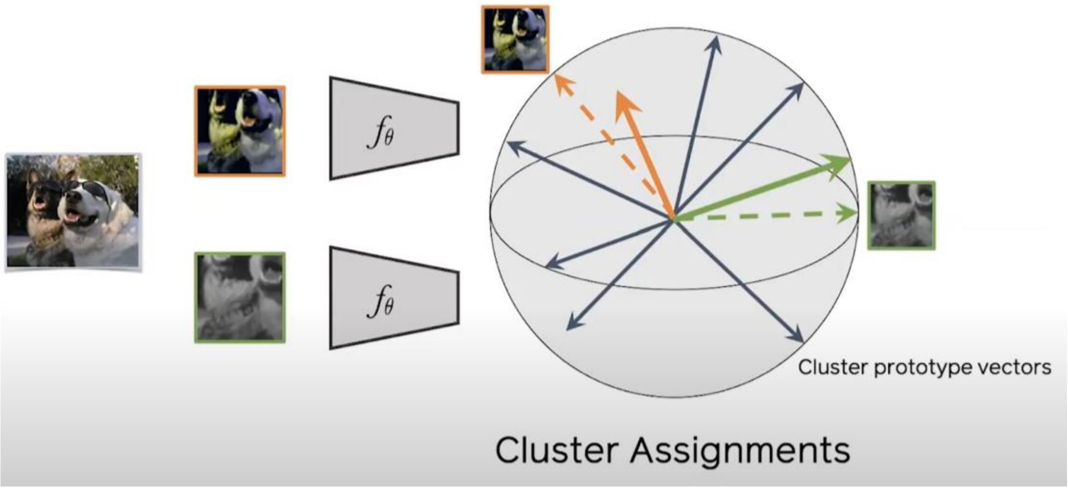


Illustration of clustering on SwAV

Note : why we use optimal transportation to get code q_{nt}?

1. ***q_{nt} should be probability vector (used as loss)***
   (achieved by constraint in optimal transportation)
2. ***Higher coupling value for positive pair / Smaller coupling value for negative pair***

 # Above objectives can be achieved by optimal transportation by setting $-C^T Z$ as the cost matrix.

- But how to exploit optimal transportation to compute the codes q_{nt}?
  - Set code matrix Q as the coupling matrix in optimal transportation.
  - Set $-C^T Z$ as the cost matrix in optimal transportation. (where $Z = [z_1, \dots, z_B] \in \mathbb{R}^{D \times B}$)
  - Now, we want to minimize the coupling × cost entirely (= solving optimal transportation)

Introduction (SwAV)

- Why we choose $-C^T Z = \begin{bmatrix} -c_1^T z_1 & \cdots & -c_B^T z_1 \\ \vdots & \ddots & \vdots \\ -c_K^T z_1 & \cdots & -c_K^T z_B \end{bmatrix} \in \mathbb{R}^{K \times B}$ as the cost matrix ?
  - This enforces the optimal transportation to assign high coupling value $(= q_{ij})$ for positive pairs.

    # Logic : Positive pair -> high similarity (=$c^T z$) -> low cost (=$-c^T z$) -> high coupling value assigned by optimal transportation algorithm
  - Also, this method assign low coupling value $(= q_{ij})$ for negative pair in the similar logic above.
- This optimization maximizes the similarity between the features and the prototypes.

Introduction (SwAV)

- To prevent trivial solution such as giving all coupling values to smallest cost-valued item, we add entropy term $H(Q) \coloneqq -\sum_{i,j} Q_{ij} \log Q_{ij}$.
- To sum up, our optimization problem is following :

$$\max_{Q \in \mathcal{Q}} < Q, C^T Z >_F + \epsilon H(Q)$$

Note (Frobenius inner product) :

$< \boldsymbol{A}, \boldsymbol{B} >_{\boldsymbol{F}} = \sum_{i,j} \boldsymbol{A}_{ij} \boldsymbol{B}_{ij} = \boldsymbol{Tr}(\boldsymbol{A}^{\boldsymbol{T}} \boldsymbol{B})$

where $\mathcal{Q} = \left\{ Q \in \mathbb{R}_+^{K \times B} \middle| Q \cdot 1_B = \frac{1}{k} 1_K, Q^T \cdot 1_K = \frac{1}{B} 1_B \right\}$ (finite resource constraint in optimal transport.)

- Luckily, there is a good algorithm to solve this problem (**Sinkhorn-Knopp algorithm**) :
    - Solution : $Q^* = diag(u) \cdot K \cdot diag(v)$

    where $K = \exp(\frac{C^T Z}{\epsilon})$, $u^{(k+1)} = \frac{r}{KV^{(k)}}$, $v^{(k+1)} = \frac{c}{K^T u^{(k+1)}}$ (here, $r = \frac{1}{k} 1_K$, $c = \frac{1}{B} 1_B$)

Introduction (SwAV)

Note : Overall loss adopting $L(z_t, z_S)$ can be rewritten as follows : (N : # of data)

$$-\frac{1}{N}\sum_{n=1}^{N}\sum_{s,t\sim\mathcal{T}}\left[\frac{1}{\tau}\mathbf{z}_{nt}^{\top}\mathbf{C}\mathbf{q}_{ns}+\frac{1}{\tau}\mathbf{z}_{ns}^{\top}\mathbf{C}\mathbf{q}_{nt}-\log\sum_{k=1}^{K}\exp\left(\frac{\mathbf{z}_{nt}^{\top}\mathbf{c}_k}{\tau}\right)-\log\sum_{k=1}^{K}\exp\left(\frac{\mathbf{z}_{ns}^{\top}\mathbf{c}_k}{\tau}\right)\right]$$

- (Back to) Method :

4. Calculate '**estimated similarity vector $\boldsymbol{p_{nt}}$**' of z_{nt} based on the previous projection:

$$p_{nt}^{(k)} = \frac{\exp\left(\frac{1}{\tau}z_{nt}^{T}c_k\right)}{\sum_{k'}\exp\left(\frac{1}{\tau}z_{nt}^{T}c_{k'}\right)}$$

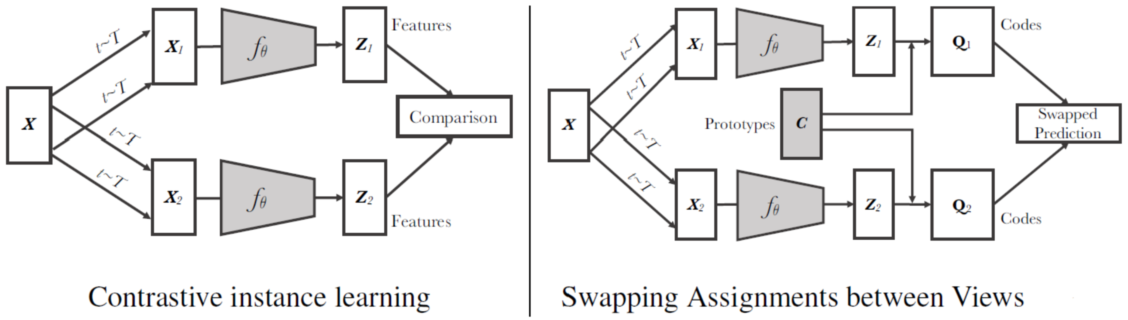
Softmax of columns of $z_{nt}^T C$ with temperature τ

Why the loss is based on swapped prediction?

Idea : z_t, z_s should be assigned into same prototype.

5. Calculate loss based on **swapped prediction** : (Note : z_t, z_s are two different augmentations of x)

$$L(z_{nt}, z_{ns}) = l(z_{nt}, q_{ns}) + l(z_{ns}, q_{nt}),\quad l(z_{nt}, q_{ns}) = -\sum_k q_{ns}^{(k)}\log p_{nt}^{(k)}$$



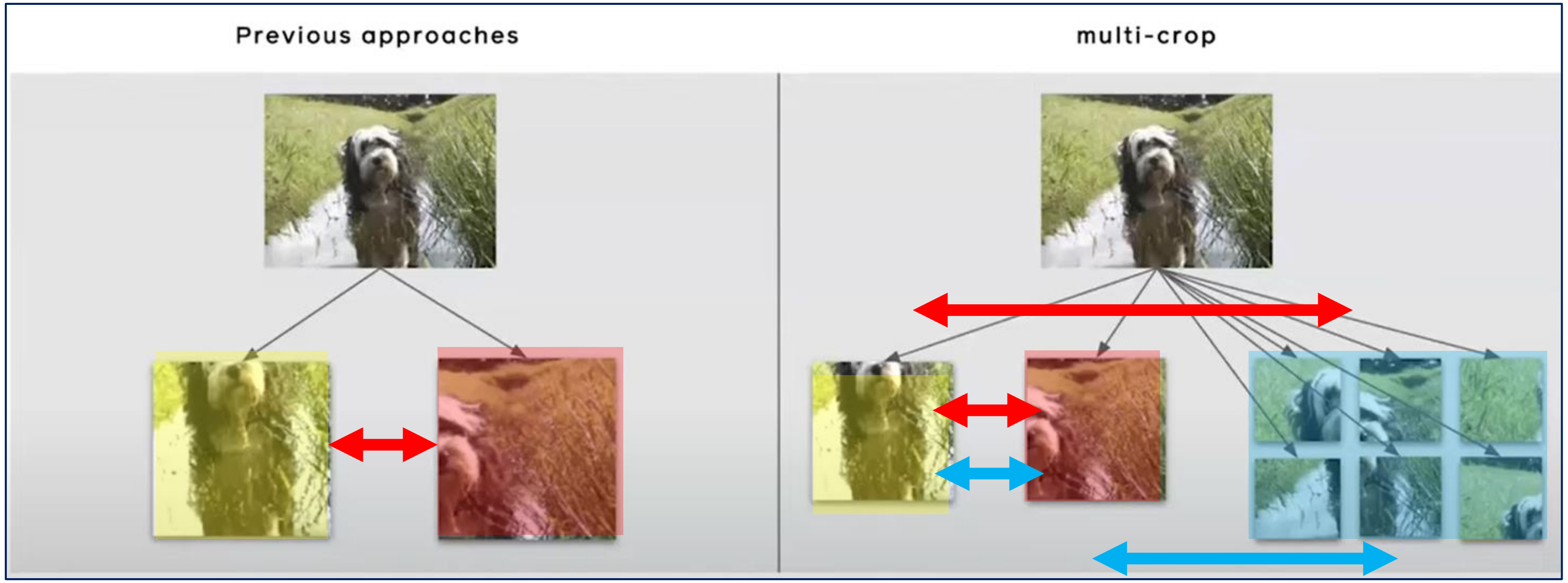
Comparison between CL instance learning vs. SwAV

(Note : Gray color implies trainable parameters)

Introduction (Multi-crop)

- It is widely known that comparing more views during CL training improves the resulting model. (Misra et al., 2019)
- However, most CL methods compare 'one' pair of transformations per image for saving comp. cost.
  - To exploit the advantage of multi view training, they propose to use V low resolution crops that cover only small parts of the image in addition to two standard resolution crops.

Illustration of multi-crop concept



Introduction (Multi-crop)

- Detailed description of multi-crop :
  - Recall the loss based on **swapped prediction** appeared in SwAV.

$$L\left(z_{t_1}, z_{t_2}\right) = l\left(z_{t_1}, q_{t_2}\right) + l\left(z_{t_2}, q_{t_1}\right)$$

  which adopts only two standard resolution crops z_{t_1}, z_{t_2}.

  - We now sample V additional low resolution crops $z_{t_3}, \ldots z_{t_{V+2}}$ and generalize the loss :

$$L\left(z_{t_1}, z_{t_2}, \ldots z_{t_{V+2}}\right) = \sum_{i \in \{1,2\}} \sum_{v=1}^{V+2} 1_{v \neq i} l\left(z_{t_v}, q_{t_i}\right)$$

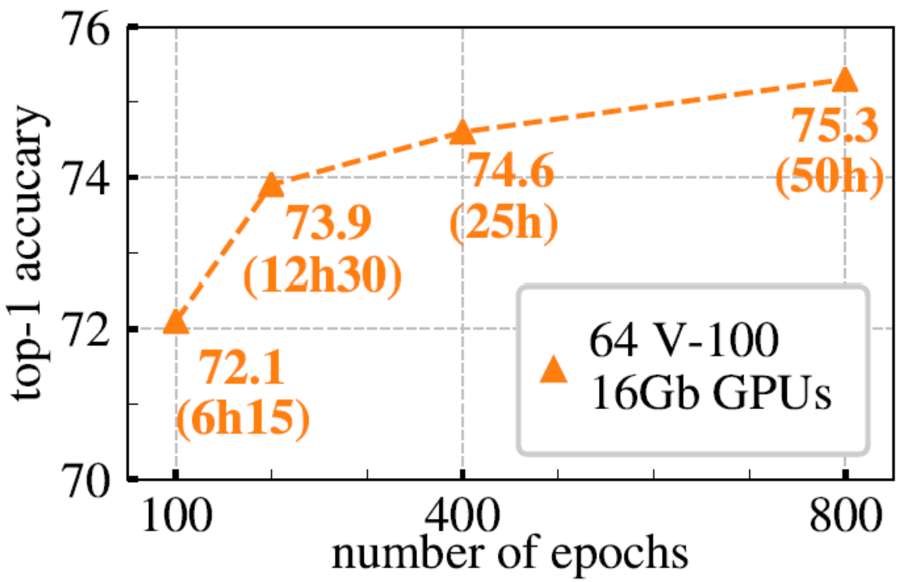
  - Note : Using low resolution images ensures only a small increase in computational cost.

Experiments

Multi-crop (x) *Multi-crop (o)*

- To verify downstream performance, they used linear evaluation protocol.

Method	Top-1 2x224	Top-1 2x160+4x96	Δ
Supervised	76.5	76.0	−0.5
Contrastive-instance approaches			
SimCLR	68.2	70.6	+2.4
Clustering-based approaches			
SeLa-v2	67.2	71.8	+4.6
DeepCluster-v2	70.2	74.3	+4.1
SwAV	70.1	74.1	+4.0



Left : Comparision between clustering-based and contrastive instance methods and impact of multi-crop. (400 /200 epochs for unsupervised / supervised)

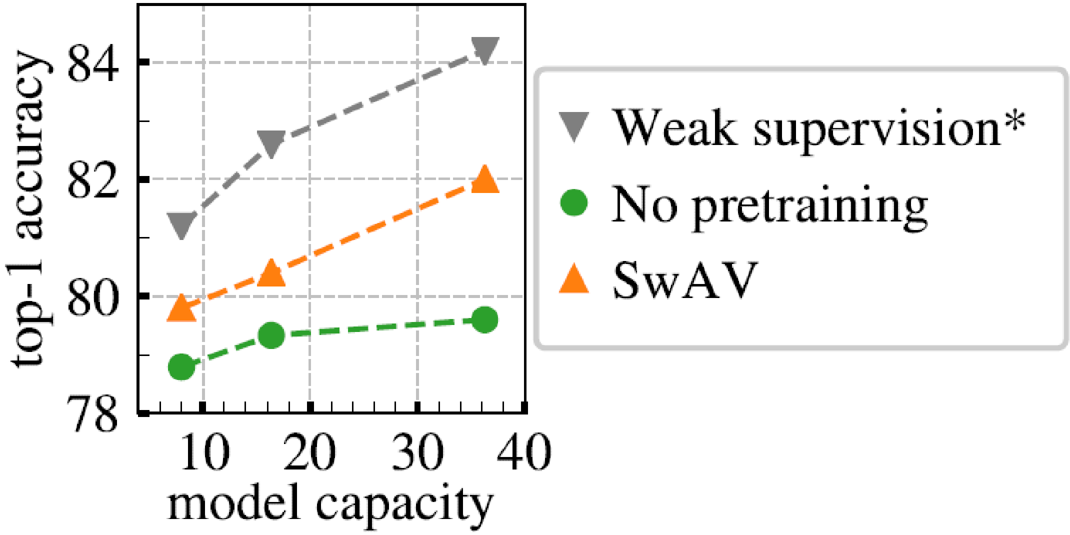
Right : SwAV (with multi-crop) downstream performance as a function of epochs

Note : DeepCluster-v2 is not online which makes it impractical for extremely large datasets (= requires much longer training compared to SwAV)

Experiments

- Further experiments demonstrate the SwAV's higher performance on representation learning.

Method	Frozen	Finetuned
Random	15.0	76.5
MoCo	-	77.3*
SimCLR	60.4	77.2
SwAV	**66.5**	**77.8**



Left : Top-1 acc on ImageNet for pretrained models on uncurated 1B Instagram images (linear evaluation on frozen features or finetuned features)

Right : Performance of finetuned models as we increase the capacity of ResNext

Note (weak supervision) : pretrained on a curated 1B Instragram images filtered with 1.5k hashtags similar to ImageNet classes.