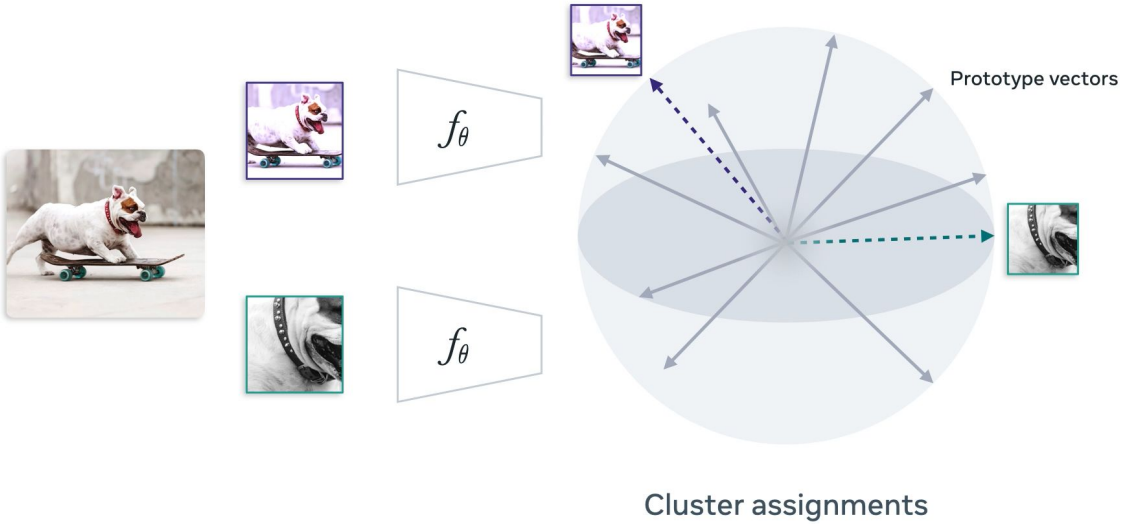
Unsupervised Learning of Visual Features by Contrasting Cluster Assignments



Facebook AI Research, Oct 15, presented at NeurIPS

Main Subjects of the paper

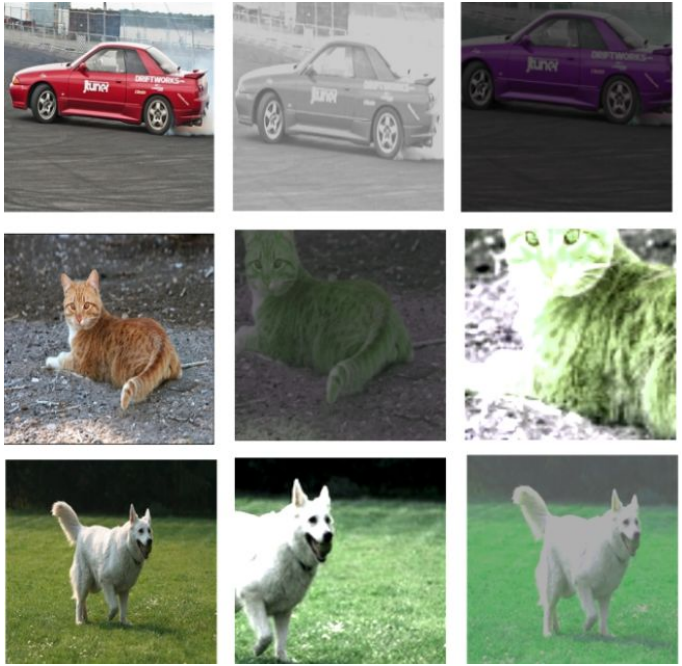
- SSRL: self supervised representation learning of images
- Using clustering for deep representation learning
- New methods for saving computational and memory resources when training model that way

Intro: contrasting learning

- Contrastive learning approaches, learn representations **by contrasting positive pairs against negative pairs.**
- **How?** By applying augmentations on images (random crop and resize, color distortion etc)
- Then define **positive pairs** as those pair of **(augmented)images** which we got from the **same original image.**
- All other pairs that are not coming from the same image are considered negative pairs

Common methods of SSRL: simCLR

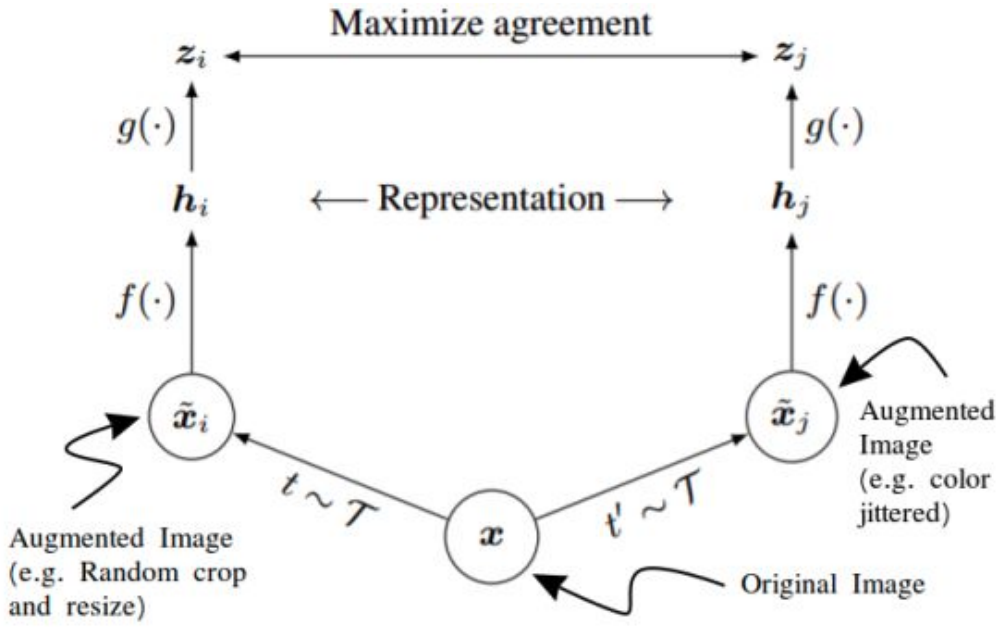
- For a batch of N images, 2N augmented images are created.
- Given a particular positive pair (i,j) from these 2N images, we consider the other 2(N-1) images as negative examples for i and j.



source

Rachel Shalom

Common methods of SSRL: simCLR



We want different augmentations of the same image to have similar vector representations.

paper

Rachel Shalom

Common methods of SSRL: simCLR loss function

$$l_{i,j} = -\log \frac{\exp(\mathrm{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)}$$

By maximize the similarity between positive pairs , we minimize the loss

$$\mathrm{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^T \mathbf{v}}{\|\mathbf{u}\| \, \|\mathbf{v}\|} = \hat{\mathbf{u}}^T \hat{\mathbf{v}}$$

Cosine similarity

source

Rachel Shalom

Common methods of SSRL: simCLR loss function

Make numerator smaller

$$\downarrow l_{i,j} = \log\left(1 + \frac{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i,j]} \exp(\downarrow \operatorname{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)}{\exp(\uparrow \operatorname{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}\right)$$

Make denominator bigger

By maximize the similarity between positive pairs , we minimize the loss

we not only make the vector z_i more similar to z_j but also make it dissimilar to all the other vectors.

source

Rachel Shalom

Common methods of SSRL: simCLR loss function

$$\downarrow l_{i,j} = \log\left(1 + \frac{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i,j]} \exp(\downarrow \operatorname{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)}{\exp(\uparrow \operatorname{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}\right)$$

$$\begin{aligned}
l_{i,j} &= -\log \frac{\exp(\operatorname{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(\operatorname{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)} \\
&= \log \frac{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(\operatorname{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)}{\exp(\operatorname{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)} \\
&= \log\left(\frac{\exp(\operatorname{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau) + \sum_{k=1}^{2N} \mathbb{1}_{[k\neq i,j]} \exp(\operatorname{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)}{\exp(\operatorname{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}\right) \\
&= \log\left(1 + \frac{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i,j]} \exp(\operatorname{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)}{\exp(\operatorname{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}\right) \\
\downarrow l_{i,j} &= \log\left(1 + \frac{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i,j]} \exp(\downarrow \operatorname{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)}{\exp(\uparrow \operatorname{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}\right)
\end{aligned}$$

By maximize the similarity between positive paris , we minimize the loss

we not only make the vector z_i more similar to z_j but also make it dissimilar to all the other vectors.

Rachel Shalom

Common methods of SSRL: bank of negative examples (MoCo)

- Increasing the number of negative examples for each positive pair helps optimize the representation of each example
- As a result the training is done on large batches which requires large compute resources (need to compute thousands of representation of batch examples)
- In order to reduce the computational resources, a bank of “negative examples” is offered where representations of negative examples from previous batches are saved for use in the next batches(less computations) but this requires more memory

Intro: Contrasting loss

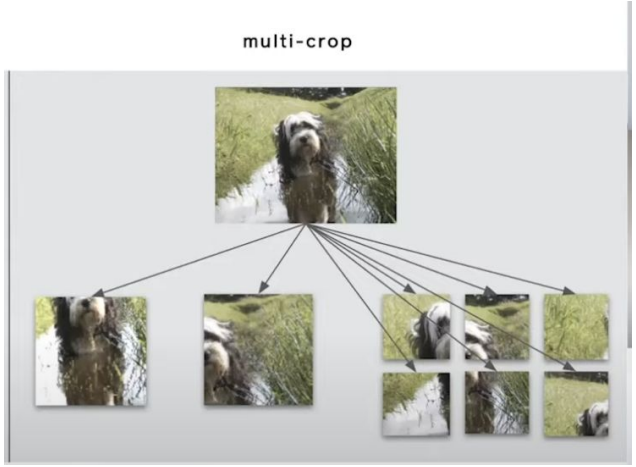
- **CL: contrasting loss**: the loss function of contrasting learning is based on the assumption that vector representations of close examples are also close
- These methods count on large number of pairs of vector representations (features) comparisons and this **Requires a lot of memory and computational resources**

Motivation for new method of training

- Computing all pairwise comparisons is not practical on a large dataset (computations and memory resources)
- Few options to solve this:
  - Approximation of loss by by reducing the number of images for comparison (fixed num of random images
  - “Instead of approximate the loss, we can approximate the task” .e.g. Instead of discriminating between each pair discriminate between groups of images with similar features

New method: Multi Crop Augmentations strategy

- Start with “standard” crops: crop x_c1 and x_c2 of image X
- then take smaller crops from x_c1 and x_c2 in various smaller resolutions
- Build positive examples from these crops
- They argue that using low resolution images ensures only a small increase in the compute cost



source

Rachel Shalom

New method: SwAV

- Main idea:train an embedding with consistent cluster assignments between views of the same image
- Redefine the task of similarity of image representations with clustering
- Enforcing close examples pairs to be part of the same cluster in the representations space instead of directly compare between different example representations
- This method should require less computational and memory resources

The Loss function of SwAV

For Every Example X:

- Create number of augmentations (with multi crop or other methods)
- Create different positive pairs from these augmentations
- Map the augmentation to a vector of representations, z
- Define K clusters C1,...Ck (C1,..Ck are considered centroids of the clusters)
- For every pair of vector representations Zt and Zs create a “code” vectors qt and qs
- The code vectors are vectors that describe probability of “belonging” to each cluster (prototypes) {c1,...,ck} -the level of “closeness” to each cluster
- There are no negative examples

Rachel Shalom

The Loss function of SwAV

$$L(\mathbf{z}_t, \mathbf{z}_s) = \ell(\mathbf{z}_t, \mathbf{q}_s) + \ell(\mathbf{z}_s, \mathbf{q}_t)$$

- The l functions are log loss that measures the **fit** between feature z and a code q
- The loss is the sum of the “swapped” similarity(swap of codes s and t)
- Intuition: this “enforce” every representation to learn the code of it’s similar representation
- If two features capture the same information, it should be possible to predict the code of one vector from the other
- The name SwAV is coming from swapping the codes of representations of similar examples

Zt and Zs vector representations

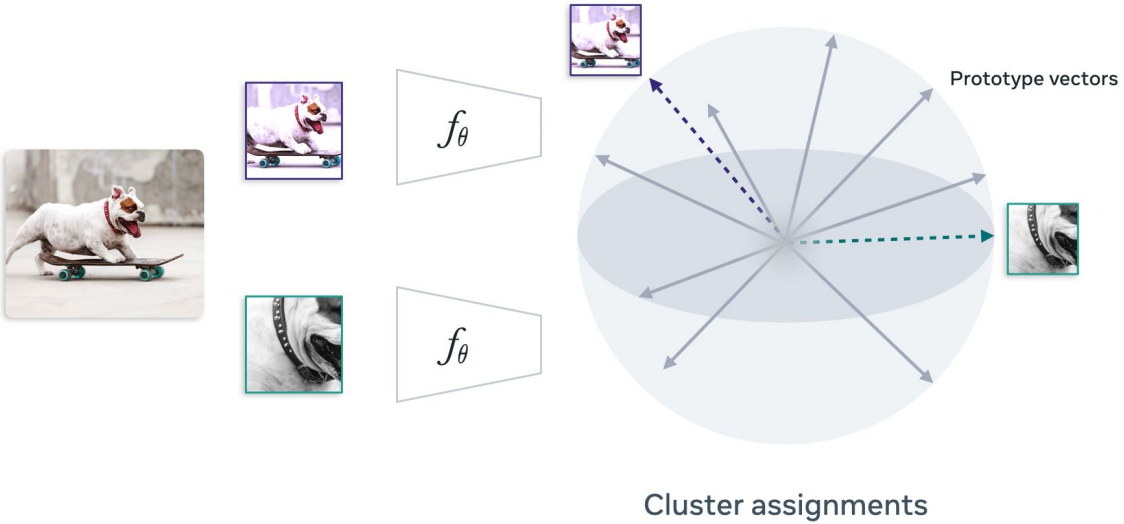
K clusters C1,...Ck

qt and qs are vectors that describes the level of “closeness” of zt and zs to each class Ci

Rachel Shalom

Animation of the swav process

https://ai.facebook.com/blog/high-performance-self-supervised-image-classification-with-contrastive-clustering/



Rachel Shalom

Swav vs. contrastive instance learning

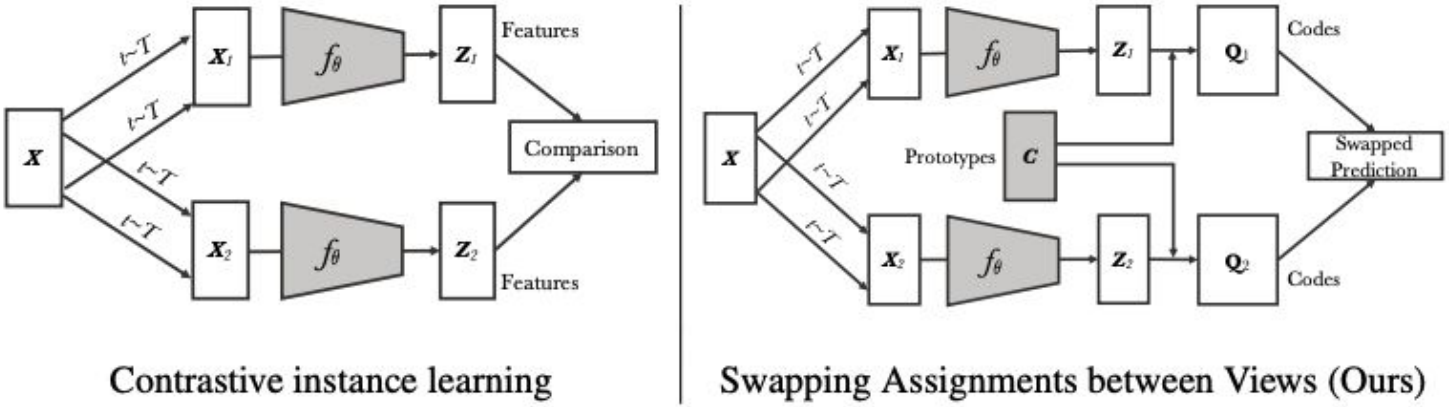


Figure 1: **Contrastive instance learning (left) *vs*. SwAV (right).** In contrastive learning methods applied to instance classification, the features from different transformations of the same images are compared directly to each other. In SwAV, we first obtain "codes" by assigning features to prototype vectors. We then solve a "swapped" prediction problem wherein the codes obtained from one data augmented view are predicted using the other view. Thus, SwAV does not directly compare image features. Prototype vectors are learned along with the ConvNet parameters by backpropagation.

Why does Swav save resources vs other methods mentioned?

-SimCLR: batch of 3 (car, cat, dog), create 6 augmentations, and 6X6 comparisons (this is O(n^2))

- SwaV:batch of 3 (car, cat dog), create 6 augmentations and make 6X2 comparisons (linear time)

Rachel Shalom

How is the loss function defined?

$$L(\mathbf{z}_t, \mathbf{z}_s) = \ell(\mathbf{z}_t, \mathbf{q}_s) + \ell(\mathbf{z}_s, \mathbf{q}_t)$$

Swapped similarity of representation vector z and code vector q. Vector q refelcet the distances or level of closeness to clusters c_i (centroid of cluster of vector representations)

$$\ell(\mathbf{z}_t, \mathbf{q}_s) = -\sum_k \mathbf{q}_s^{(k)} \log \mathbf{p}_t^{(k)}$$

Cross entropy loss between q and distance vectors p_k which represent the similarity to cluster k or the probabilty of rep. vector z_t to belong to cluster k

$$\mathbf{p}_t^{(k)} = \frac{\exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_k\right)}{\sum_{k'} \exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_{k'}\right)}.$$

This is a normalized exponent of dot product between z_t and C_k. It is basically softmax of similarity vector

Rachel Shalom

How is the loss function defined?

$$L(\mathbf{z}_t, \mathbf{z}_s) = \ell(\mathbf{z}_t, \mathbf{q}_s) + \ell(\mathbf{z}_s, \mathbf{q}_t)$$

$$\ell(\mathbf{z}_t, \mathbf{q}_s) = -\sum_k \mathbf{q}_s^{(k)} \log \mathbf{p}_t^{(k)}, \quad \text{where} \quad \mathbf{p}_t^{(k)} = \frac{\exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_k\right)}{\sum_{k'} \exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_{k'}\right)}.$$

Pt represent the normalized distance vector of z from each cluster ci

Summing the loss over all images pairs augmentations leads to

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

The loss function is <u>jointly minimized with respect to the prototypes C and the parameters of the encoder used to produce the features z</u>

Rachel Shalom

How is the loss function defined? intuition

$$\ell(\mathbf{z}_t, \mathbf{q}_s) = -\sum_k \mathbf{q}_s^{(k)} \log \mathbf{p}_t^{(k)}, \quad \text{where} \quad \mathbf{p}_t^{(k)} = \frac{\exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_k\right)}{\sum_{k'} \exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_{k'}\right)}. \quad (2$$

If we pay attention the the “softmax” part of the loss equation is very similar to “classic” contrastive loss function definition.

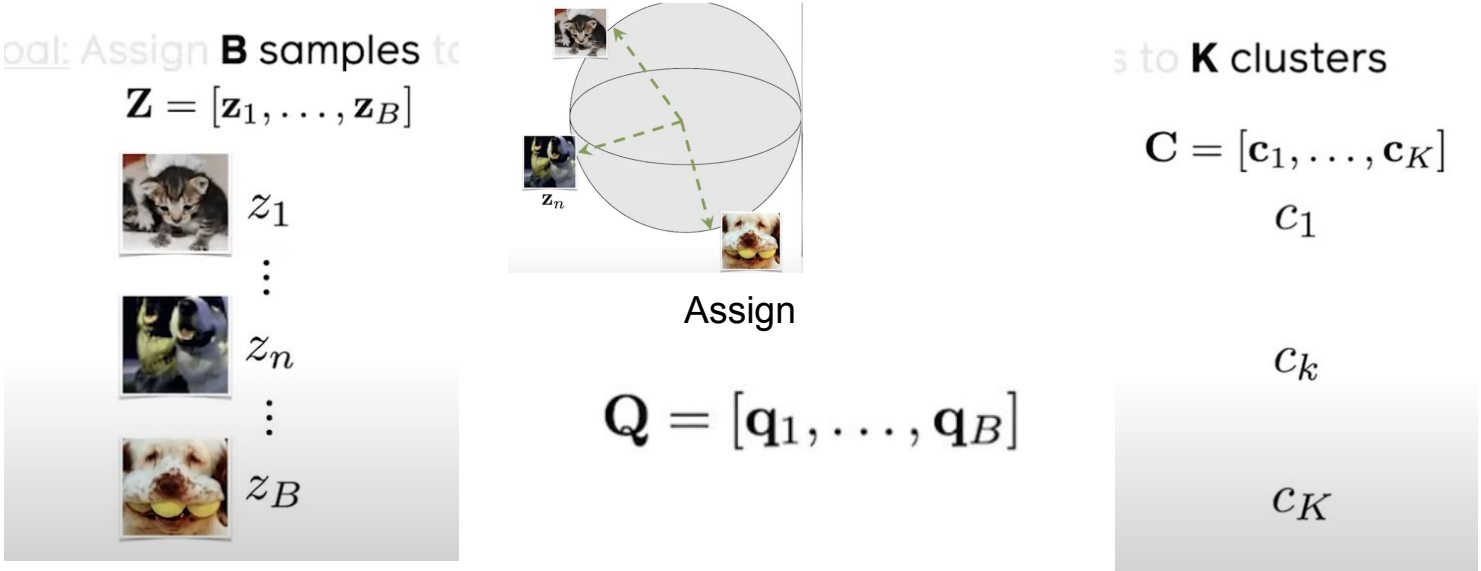
Here we don’t have “negative examples” for comparisons, but the centroids Ci that are far from z are playing the part of “negative examples”.

This means that vectors of positive images are forced to be far from negative clusters and close to positive clusters at the same way

Rachel Shalom

Cluster assignments

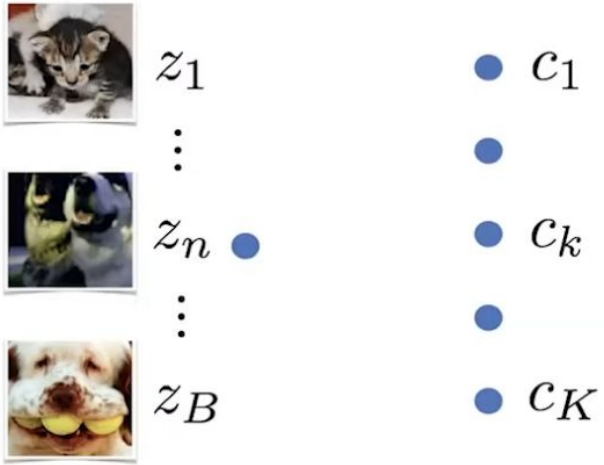
Goal: assign **B** samples to **K** clusters



source

Rachel Shalom

Computing the Codes Q online- once per batch

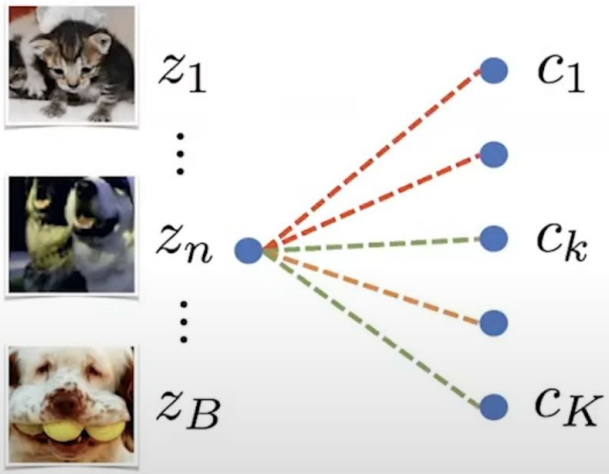


Cost matrix $-\mathbf{C}^\top \mathbf{Z}$

source

Rachel Shalom

Computing the Codes Q online- once per batch



Cost matrix $-\mathbf{C}^\top \mathbf{Z}$

	Cluster 1	...	Cluster k	...	Cluster K
z_1	...	...	...	...	...
z_n	0.91	0.15	- 0.73	- 0.48	- 0.74
...	...	...	...	...	...
z_B	...	...	...	...	...

source

Rachel Shalom

Cluster assignments -recap

- Goal: get mapping of **B** samples to **K** clusters with equipartition constraint and following a cost matrix
- <u>Optimization problem:</u> $\max_{\mathbf{Q}\in\mathcal{Q}} \mathrm{Tr}\left(\mathbf{Q}^\top \mathbf{C}^\top \mathbf{Z}\right) + \varepsilon H(\mathbf{Q}), \quad H(\mathbf{Q}) = -\sum_{ij} \mathbf{Q}_{ij} \log \mathbf{Q}_{ij}$

- We denote this mapping (or codes) by the Matrix Q=[q1,....qb]

- **We want to optimize Q to maximize the similarity between the features Z and the prototypes C**
- Similar to optimal transport problem, solution is given by sinkhorn-Knopp algorithm

Rachel Shalom

Results: Evaluating the unsupervised features on ImageNet

- Freezing the features extractor layers and add on top linear classifier
- representation of images built by Swav outperforms other representations from pretrained networks on imageMet on 3 different datasets

Both for classification tasks (resnet 50) and for object detection tasks (Faster R-CNN)

	Linear Classification			Object Detection	
	Places205	VOC07	iNat18	VOC07+12 (Faster R-CNN)	COCO (DETR)
Supervised	53.2	87.5	46.7	81.3	40.8
SwAV	**56.7**	**88.9**	**48.6**	**82.6**	**42.1**

Rachel Shalom

Results: Linear classification on ImageNet

Method	Arch.	Param.	Top1
Supervised	R50	24	76.5
Colorization [64]	R50	24	39.6
Jigsaw [45]	R50	24	45.7
NPID [57]	R50	24	54.0
BigBiGAN [15]	R50	24	56.6
LA [67]	R50	24	58.8
NPID++ [43]	R50	24	59.0
MoCo [24]	R50	24	60.6
SeLa [2]	R50	24	61.5
PIRL [43]	R50	24	63.6
CPC v2 [27]	R50	24	63.8
PCL [36]	R50	24	65.9
SimCLR [10]	R50	24	70.0
MoCov2 [11]	R50	24	71.1
SwAV	R50	24	**75.3**

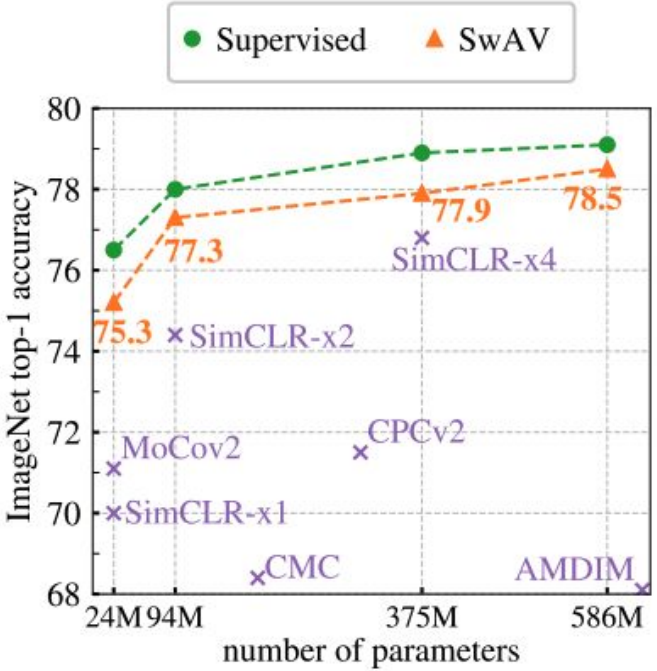


Figure 2: **Linear classification on ImageNet.** Top-1 accuracy for linear models trained on frozen features from different self-supervised methods. **(left)** Performance with a standard ResNet-50. **(right)** Performance as we multiply the width of a ResNet-50 by a factor $\times 2$, $\times 4$, and $\times 5$.

Rachel Shalom

Practical use?

- Saves label costs especially when labels are noisy or scarce (like medical images)
- Can be used not only with images(sensors samples with augmentations)
- Any use case at your company you can think of?

Useful resources

1. Video of the author author Mathilde Caron explaining the main idea
2. Understanding simCLR
3. Momentum Contrast V2
4. Optimal Transport and the Sinkhorn Transformer
5. Calculating transport plans with Sinkhorn-Knopp
6. https://github.com/PythonOT/POT

Thank you!

Rachel Shalom, Data Scientist

Keep in touch:

https://www.linkedin.com/in/rachelshalom/

appendix

Reminder: cross entropy loss

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:

$$-(y\log(p) + (1-y)\log(1-p))$$

If M>2 we calculate the loss separately for each class per each observation and summarize the result

$$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$$

M- number of classes
P- predicted probability that observation o is of class c
Y- binary indicator (0 or 1) if class c is is the correct one for observation o

Rachel Shalom