Contrastive Learning

Eduardo Morales, Hugo Jair Escalante

INAOE

Self-Supervised Learning

- Supervised learning requires a large number of labeled examples
- Semi-supervised learning can use some labeled examples with a larger set of unlabeled examples
- Usupervised learning tries to find structures in data without labels
- Self-supervised learning (SSL) receives unlabeled data which is annotated by the model
- It employs a surrogate objective to learn useful representations from unlabeled data

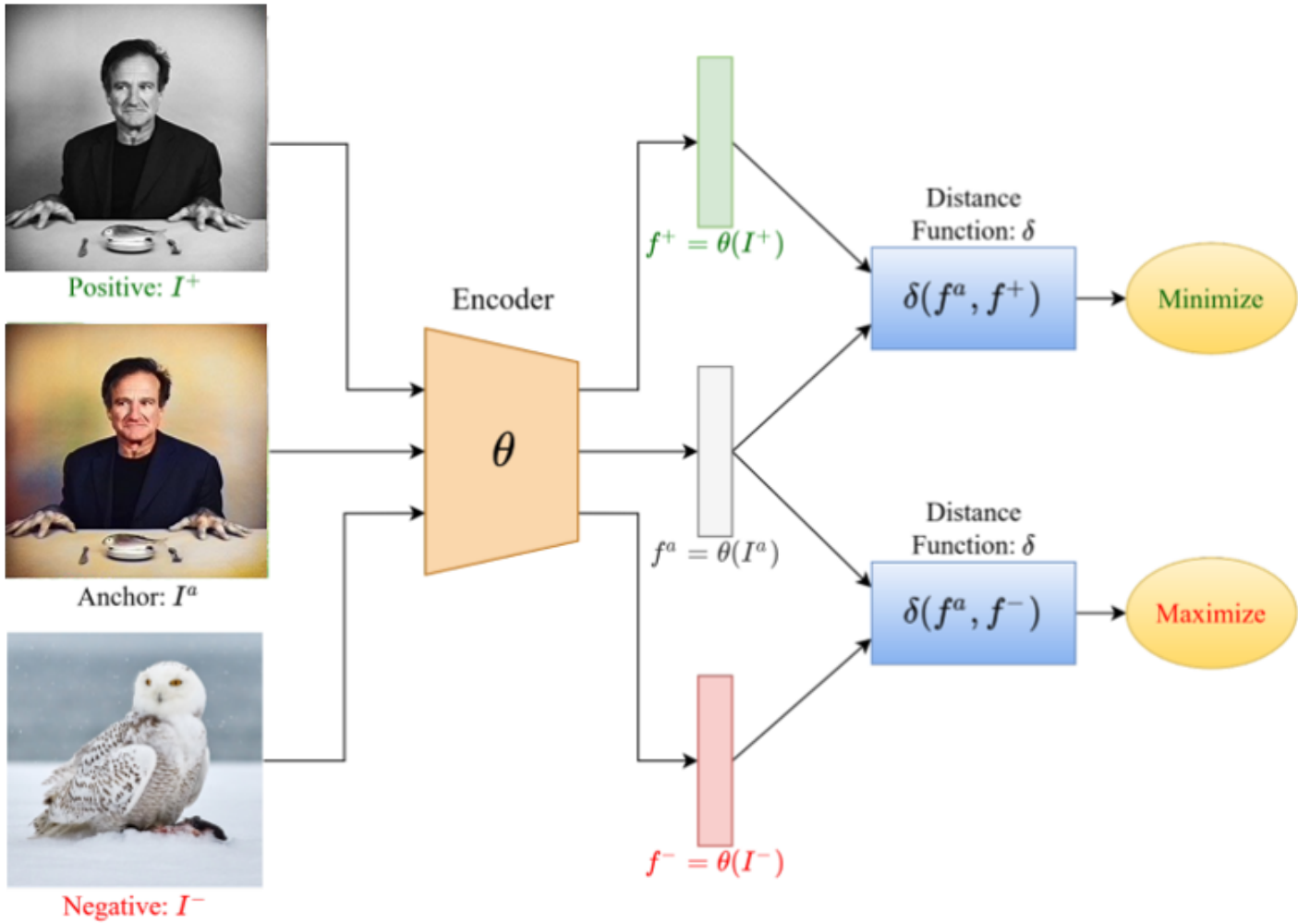
Contrastive Learning

- Constrastive learning (CL) is a sub-family of SSL that enhances the performance on classification tasks by contrasting examples
- Learn an embedding space where similar sample pairs (same class) are close while dissimilar sample pairs (different classes) are far away
- The mechanism used is to move closer points of the same distribution (class) and pull apart points belonging to different distributions in an embedding space

Contrastive Learning

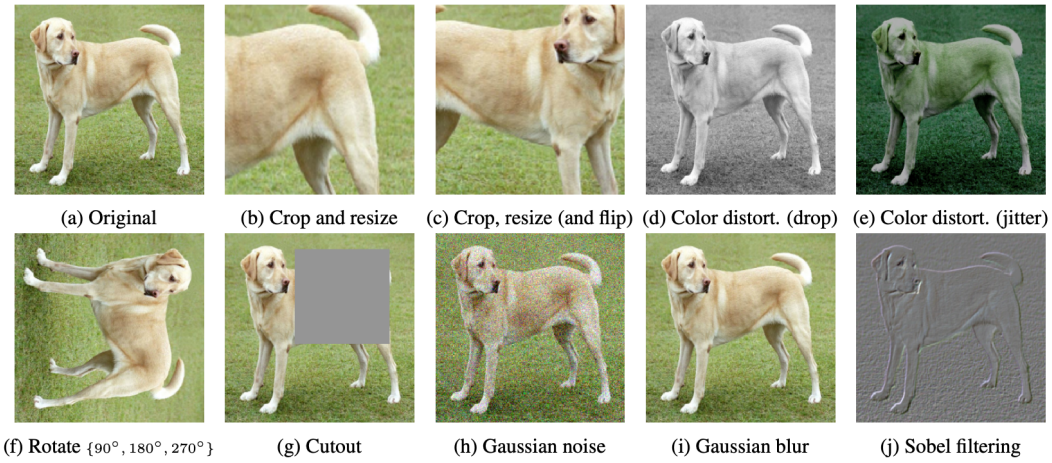
- The general idea is that although we may not know information of a class, we can still distinguish between classes (e.g., elephants and tigers)
- Procedure: Select a data sampe, called *anchor*, select another data point belonging to the same distribution, called *positive* and select another data point belonging to a different distribution, called *negative*
- Minimize the distance between the anchor and the positive example and maximize the distance between the anchor and the positive in an embedding space

Contrastive Learning



Contrastive Learning

- It is common to use one image as anchor and employ data augmentation, such as jittering, rotation, flipping, croping, add noise, and other random affine transformations as positives



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

- The distance function used in an embedding space can be Eucliden, Mahalanobis, Cosine, ...
- Most of the research work differs in terms of the loss function and number of used examples

Loss functions

- Max margin constrastive loss:

$$\mathcal{L}_{contrastive}(x_i, x_j, \theta) = \mathbb{1}[y_i = y_j] \cdot \| f_\theta(x_i) - f_\theta(x_j) \|_2^2 + \mathbb{1}[y_i \neq y_j] \cdot \max(0, \epsilon - \| f_\theta(x_i) - f_\theta(x_j) \|_2^2)$$

  where x_i and x_j are the samples with labels y_i and y_j, θ are the parameters of the embedding network, and ϵ defines the lowerbound distance between samples of different classes

- Triplet Loss: Use positive and negative samples to compute the loss

$$\mathcal{L}_{triplet}(x_a, x_+, x_-, \theta) = \sum_{\forall x} \max(0, \| f_\theta(x_a) - f_\theta(x_+) \|_2^2 - \| f_\theta(x_a) - f_\theta(x_-) \|_2^2 + \epsilon)$$

Loss functions

- N-pair Loss: Is an extension of the triplet loss to multiple negative samples

$$\mathcal{L}_{N-pair}(x^a, x^+, \{x_i^-\}_{i=1}^{N-1}, \theta) =$$
$$= \log\left(1 + \sum_{i=1}^{N-1} e^{f_\theta^T(x^a)\cdot f_\theta(x_i^-) - f_\theta^T(x^a)\cdot f_\theta(x^+)}\right)$$
$$= -\log\left(\frac{e^{f_\theta^T(x^a)\cdot f_\theta(x^+)}}{e^{f_\theta^T(x^a)\cdot f_\theta(x^+)} + \sum_{i=1}^{N-1} e^{f_\theta^T(x^a)\cdot f_\theta(x_i^-)}}\right)$$

  which translates into a softmax loss function

Loss functions

- InfoNCE (Noise-Contrastive Estimation): Identifies a positive sample among a set of unrelated noise samples
- Given a context vector (c), the positive sample should be drawn from the conditional distribution ($p(x|c)$) while the $N-1$ negative samples drawn from ($p(s)$)
- The probability of detecting x^+ correctly is:

$$P(x^+|X,c) = \frac{p(x^+)\prod_{i=1,\ldots,N;i\neq+} p(x_i)}{\sum_{j=1}^{N}[p(x_j|c)\prod_{i=1,\ldots,N;i\neq+} p(x_i)]}$$
$$= \frac{\frac{p(x^+|c)}{p(x^+)}}{\sum_{j=i}^{N}\frac{p(x_j|c)}{p(x_j)}} = \frac{f(x^+,c)}{\sum_{j=i}^{N} f(x_j,c)}$$

Loss functions

- InfoNCE optimizes the negative log probability of classifying the positive sample correctly:

$$\mathcal{L}_{InfoNCE} = -\mathbb{E}_X \cdot \log \left[\frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)} \right]$$

$$f_k(x_{t+k}, c_t) \propto \frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$$

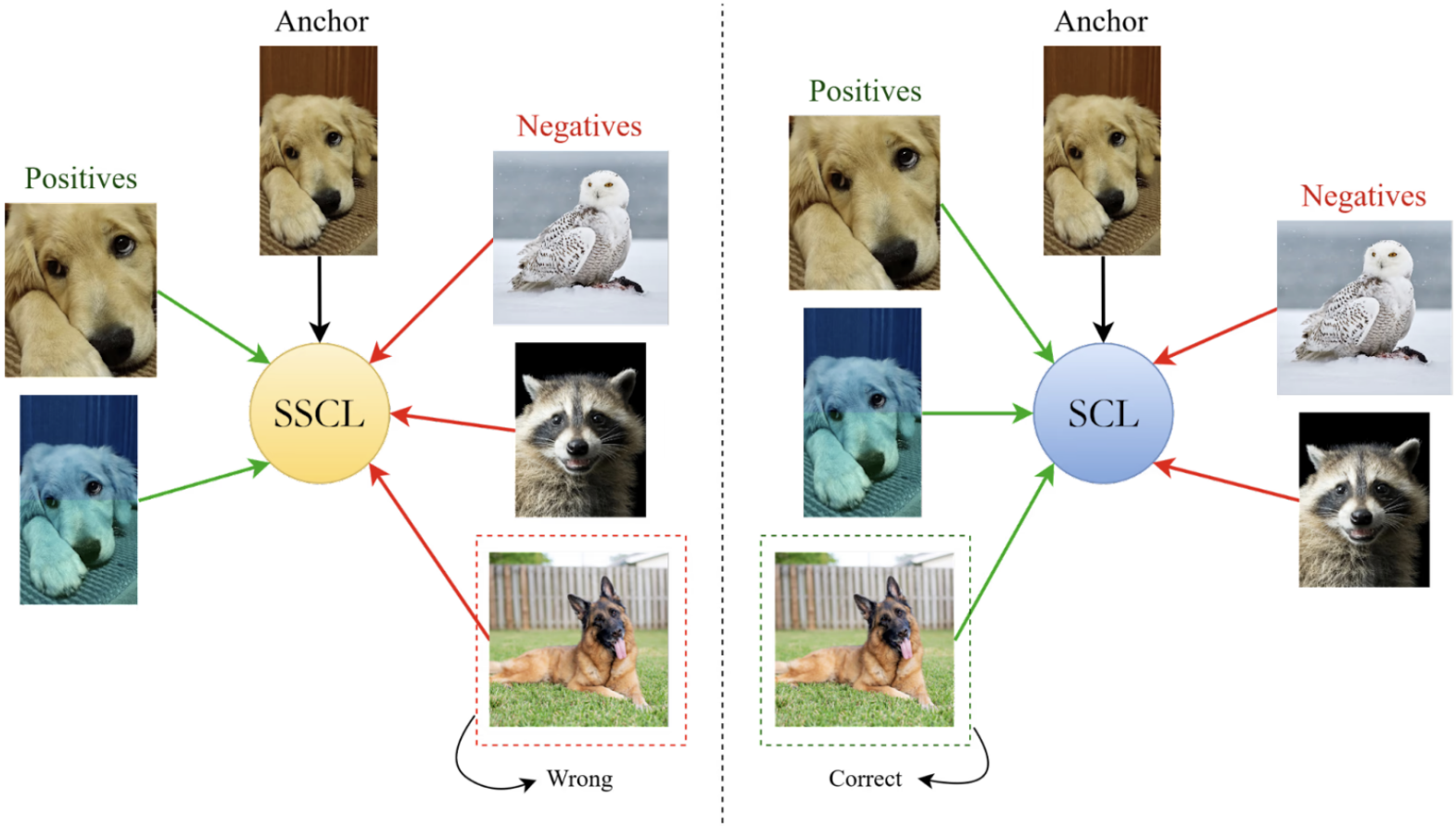
- In practice, most CL methods use:

$$\mathcal{L}(f) = -\mathbb{E}_{x,x^+,\{x_i^-\}} \left[-\ln \frac{e^{f^T(x)f(x^+)/\tau}}{e^{f^T(x)f(x^+)/\tau} + \sum_{i=1}^{N-1} e^{f^T(x)f(x_i^-)/\tau}} \right]$$

Contrastive Learning

- Self-supervised constractive learning has no labels and generates positive examples using data augmentation techniques
- A clear problem is that if we have two different instances of the same class, it will generate positive examples for one instance (the anchor) and will consider the other instance as a negative example (two different dog breeds)
- Supervised contrastive learning has information of the instances of the same class

Supervised Contrastive Learning



SimCLR

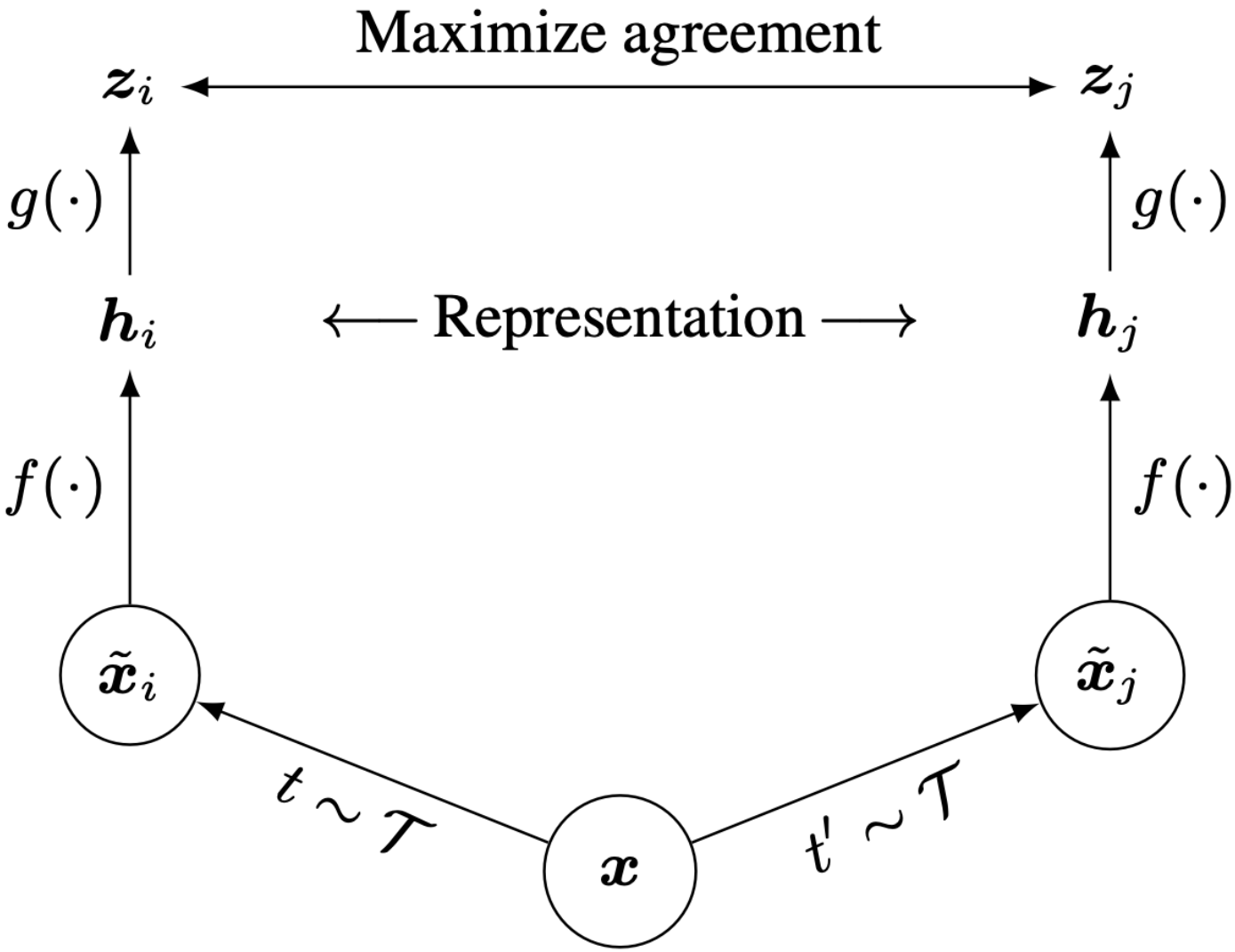
Components:

- Several data augmentation techniques to transform any given example (image) into two correlated views of the same example
- An encoder that uses ResNet50 to obtain a vector representation ($f(x)$)
- A small neural network ($g(\cdot)$) that maps the vector representations into a common latent space where the contranstive loss is applied
- Loss function:

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

- They found that constrastive learning benefits from larger batch sizes and longer training times

SimCLR


Maximize agreement
z_i
z_j
$g(\cdot)$
$g(\cdot)$
h_i
Representation
h_j
$f(\cdot)$
$f(\cdot)$
$\tilde{x}_i$
$\tilde{x}_j$
$t \sim \mathcal{T}$
x
$t' \sim \mathcal{T}$

Algorithm SimCLR

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of f, g, $\mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^{\top} \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

NNCLR

- Use positive examples (nearest neighbors) from other instances in the dataset as positives rather than augmenting the same image
- The NNCLR loss function:

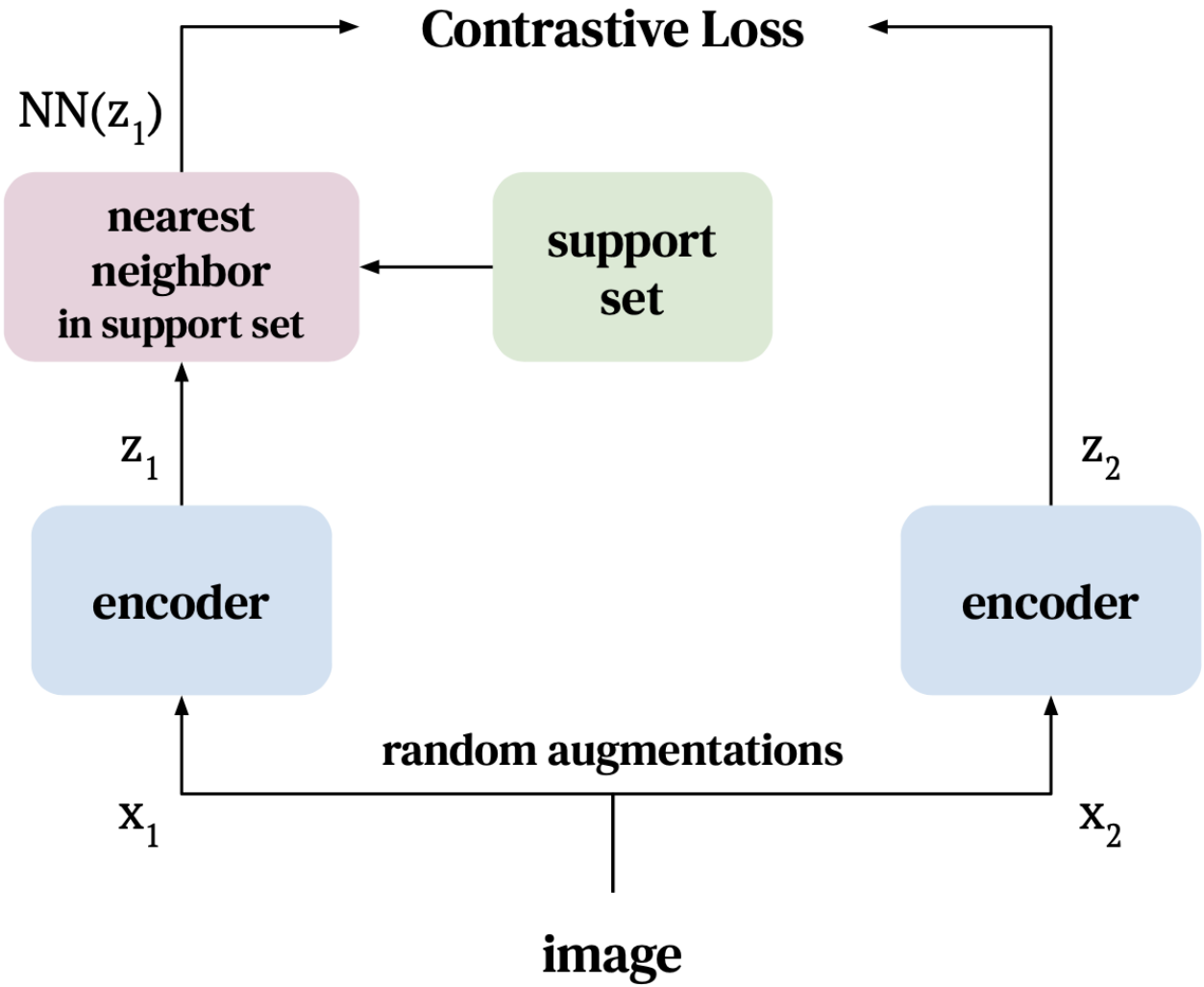
$$\mathcal{L}_i^{NNCLR} = -\log \frac{\exp(NN(z_i, Q) \cdot z_i^+ / \tau)}{\sum_{k=1}^{N} \exp(NN(z_i, Q) \cdot z_k^+ / \tau)}$$

where $NN(z, Q)$ is the nearest neighbor operator defined as:

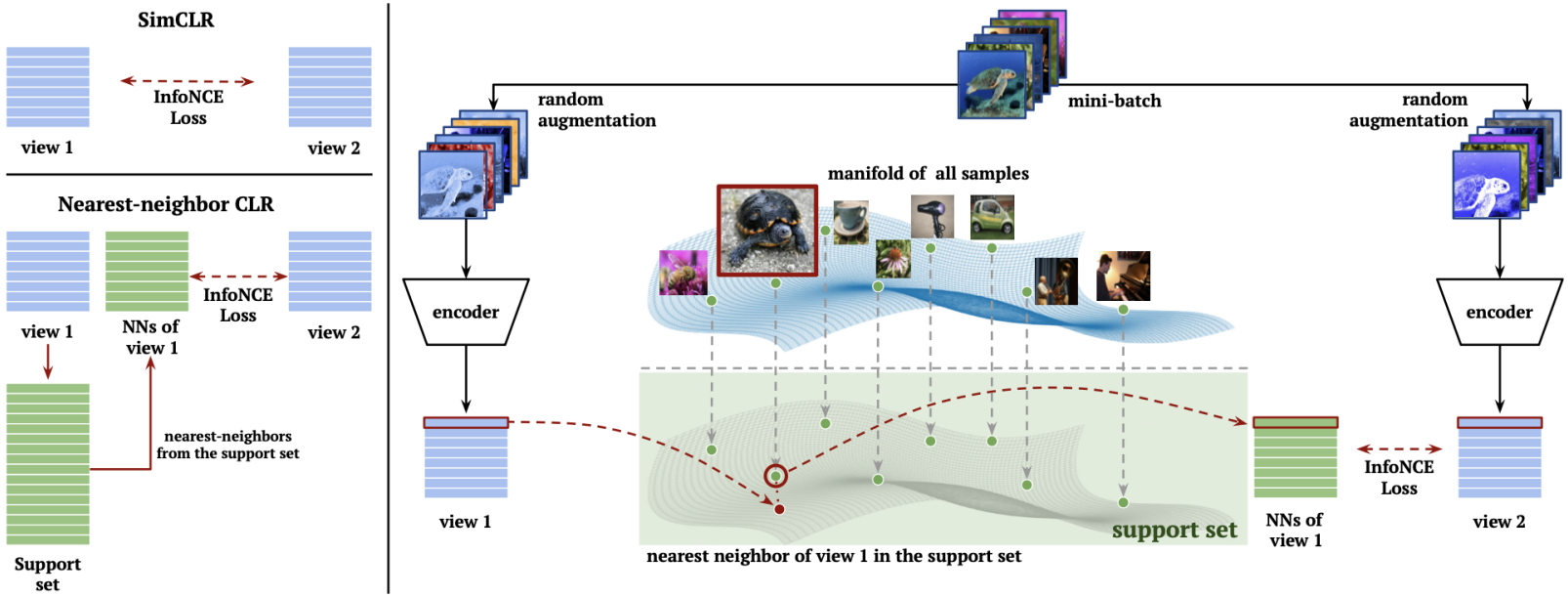
$$NN(z, Q) = \arg\min_{q \in Q} \| z - q \|_2$$

- It minimizes the average loss over all the elements in the mini-batch: $\mathcal{L}^{NNCLR} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i^{NNCLR}$

NNCLR


Contrastive Loss
NN(z1)
nearest neighbor in support set
support set
z1
z2
encoder
encoder
random augmentations
x1
x2
image

NNCLR


SimCLR
InfoNCE Loss
view 1
view 2
Nearest-neighbor CLR
view 1
NNs of view 1
InfoNCE Loss
view 2
nearest-neighbors from the support set
Support set
random augmentation
mini-batch
random augmentation
manifold of all samples
encoder
encoder
view 1
support set
NNs of view 1
InfoNCE Loss
view 2
nearest neighbor of view 1 in the support set

ORE

- Open-World Object Detection, identifies new objects without supervision in an incremental way
- It uses contrastive clustering in the latent space
- For each known class $i \in \mathcal{K}$, it stores a prototype vector p_i, let $f_c \in R^d$ be a feature vector for an object of class c, the contrastive loss is:

$$\mathcal{L}_{cont}(f_c) = \sum_{i=0}^{C} I(f_c, p_i)$$

where

$$I(f_c, p_i) = \begin{cases} D(f_c, p_i) & i = c \\ \max\{0, \triangle - D(f_c, p_i)\} & \text{otherwise} \end{cases}$$

- Where D is any distance function and $\triangle$ defines how close a similar and dissimilar item can be

ORE

- The prototypes are created by the mean of the feature vectors of each class
- When new prototypes are added, the prototypes are updated with a momentum parameter
- They use Faster R-CNN (just the region proposal network) to generate bounding boxes with a corresponding objectness score
- Then select the top-k region proposals, based on their objectness scores, as unknown objects

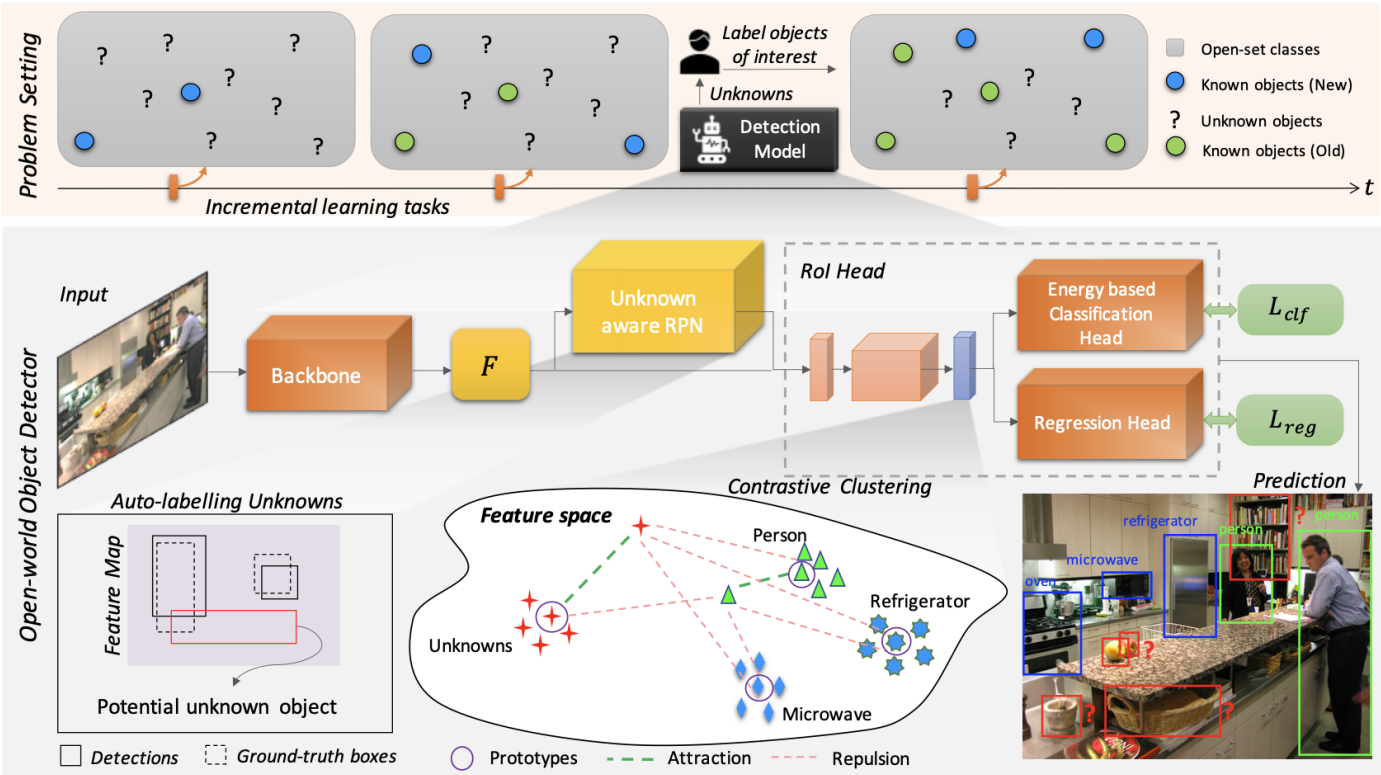
ORE

- Transforms the classifiction head into an energy function:

$$E(f;g) = -T \log \sum_{i=1}^{C} \exp(\frac{g_i(f)}{T})$$

- And models the energy distribution of the known and unknown energy values as Weibull distributions $\psi_{kn}(f)$ and $\psi_{unkn}(f)$
- The predictions are labelled as unknowns if $\psi_{kn}(f) < \psi_{unkn}(f)$
- They store a balanced set of examplars for each class for finetuning the model

ORE


Problem Setting
Label objects of interest
Unknowns
Detection Model
Open-set classes
Known objects (New)
Unknown objects
Known objects (Old)
t
Incremental learning tasks
Open-world Object Detector
Input
Backbone
F
Unknown aware RPN
RoI Head
Energy based Classification Head
Regression Head
L_{clf}
L_{reg}
Prediction
Auto-labelling Unknowns
Feature Map
Potential unknown object
Contrastive Clustering
Feature space
Person
Refrigerator
Unknowns
Microwave
Detections
Ground-truth boxes
Prototypes
Attraction
Repulsion

MoCo

- Some researchers have stored, what is caled, a *memory bank* which consists on the embedding of the data for unsupervised learning with contrastive loss
- Momentum Contrast (MoCo) maintains a dictionary as a queue of data samples, where the samples in the dictionary are progressively replaced, the current mini-batch is added and the oldest mini-batch is removed
- Denoting the parameters of f_k (key) and θ_k and those of f_q (query) and θ_q, θ_k is updated as;
  $\theta_k \leftarrow m\theta_k + (1 - m)\theta_q$
- Only the parameters of θ_q are updated by backpropagation

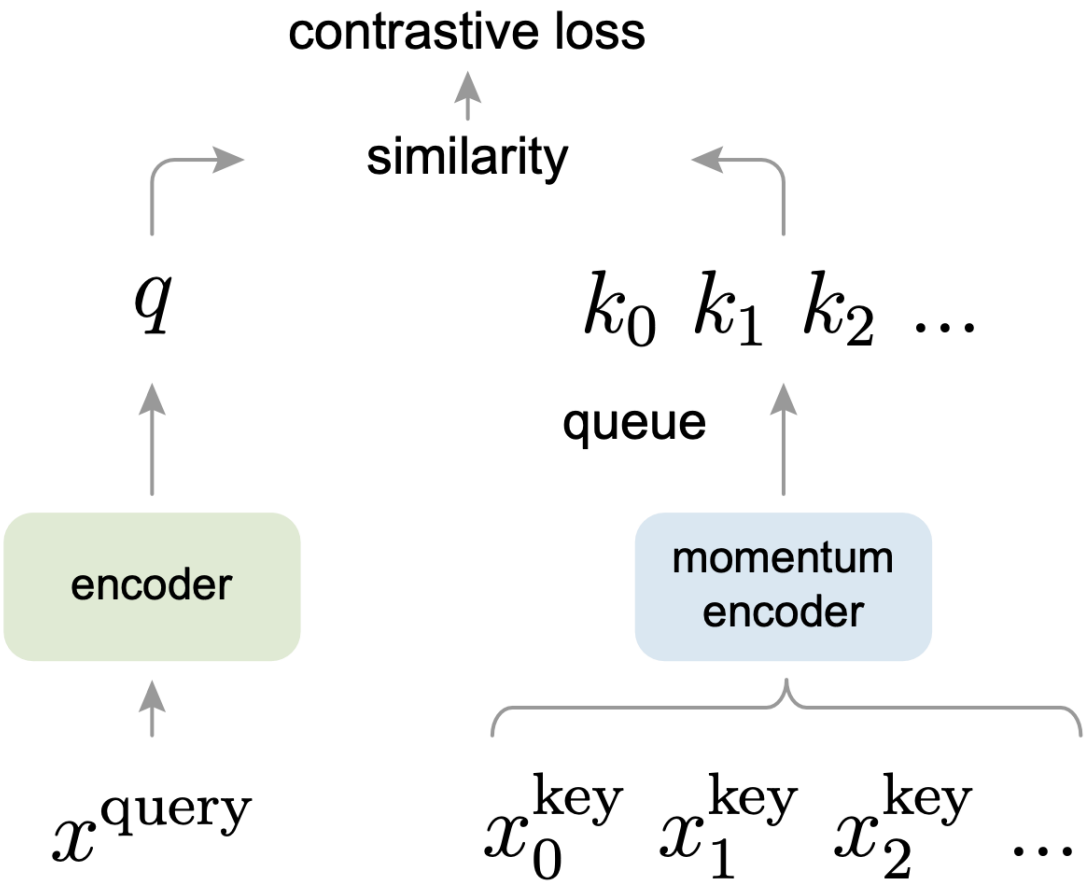
MoCo

- They use the InfoNCE contrastive loss function:

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

  the sum is over one positive and K negatives
- Can be used for different pretext tasks: images, patches, set of patches
- The networks f_q and f_k can be identical, partially shared or different

MoCo


contrastive loss
similarity
q
k_0 k_1 k_2 ...
queue
encoder
momentum encoder
x^{query}
x_0^{key} x_1^{key} x_2^{key} ...

PCL

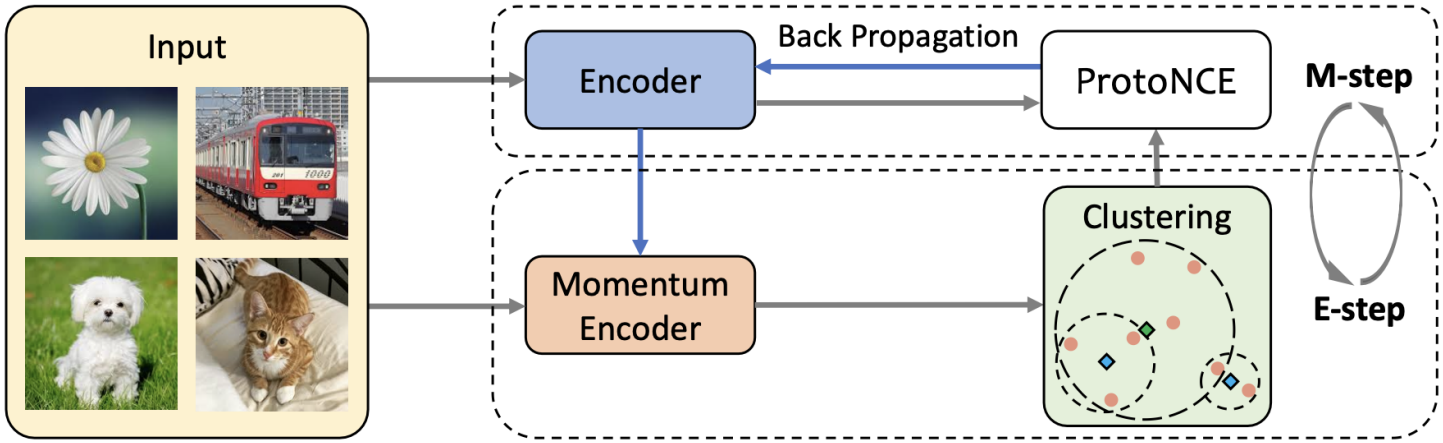
- In instance contrastive learning, they learn an embedding with a constractive loss function (e.g., InfoNCE) that includes one positive and N negative samples
- Prototypical Contrastive Learning (PCL) uses prototypes and bridges contrastive learning with clustering
- Perform $k-$means on the features $v_i' = f_{\theta'}(x_i)$ given the momentum encoder to obtain k clusters
- Prototype c_i is the centroid of the $i-$th cluster
- Uses a similar loss as InfoNCE, considering r negative prototypes, and clustering the samples M times with different number of clusters: $K = \{k_m\}_{m=1}^{M}$

PCL

$$\mathcal{L}_{ProtoNCE} = \sum_{i=1}^{n} - \left(\log \frac{e^{(v_i \cdot v_i' / \tau)}}{\sum_{j=0}^{r} e^{(v_i \cdot v_j' / \tau)}} + \frac{1}{M} \sum_{m=1}^{M} \log \frac{e^{(v_i \cdot c_s^m / \phi_s^m)}}{\sum_{j=0}^{r} e^{(v_i \cdot c_j^m / \phi_j^m)}} \right)$$

- E-step: finds the distribution of prototypes via clustering
- M-step: optimizes the network via contrastive learning

PCL


Input
Encoder
Back Propagation
ProtoNCE
M-step
Momentum Encoder
Clustering
E-step

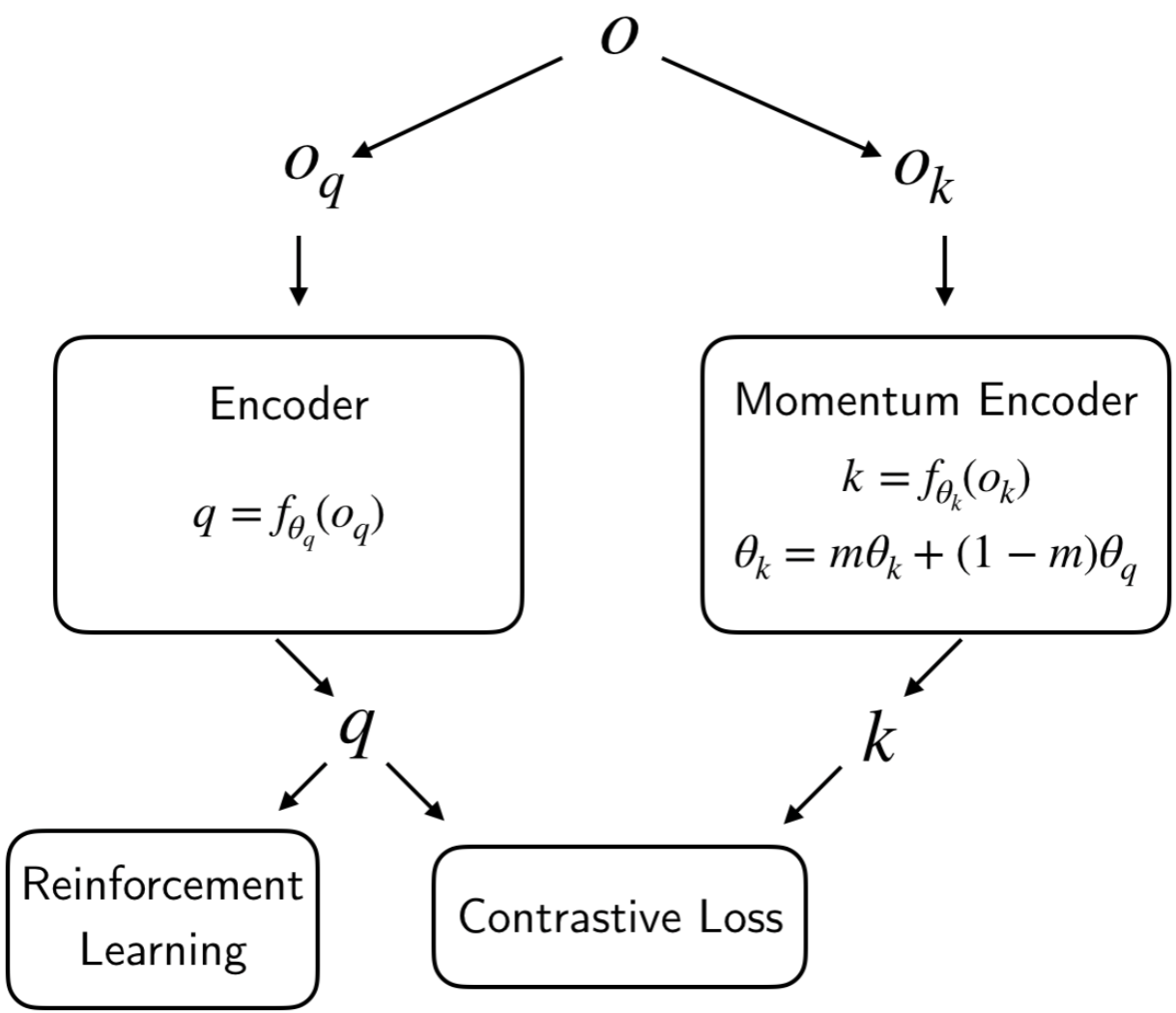
CURL

- Contrastive Unsupervised Representation for Reinforcement Learning (CURL) uses contrastive learning to maximize agreement between augmented versions of the same observation (temporally sequential frames)
- Main idea: Extract high-level features from raw images using constractive learning and performs off-policy control on top of the extracted features
- Instance discrimination is performed over frame stacks (instead of a single image)
- Employ random augmentation (cropping) across the batch retaining spatio-temporal information (anchor and positive)

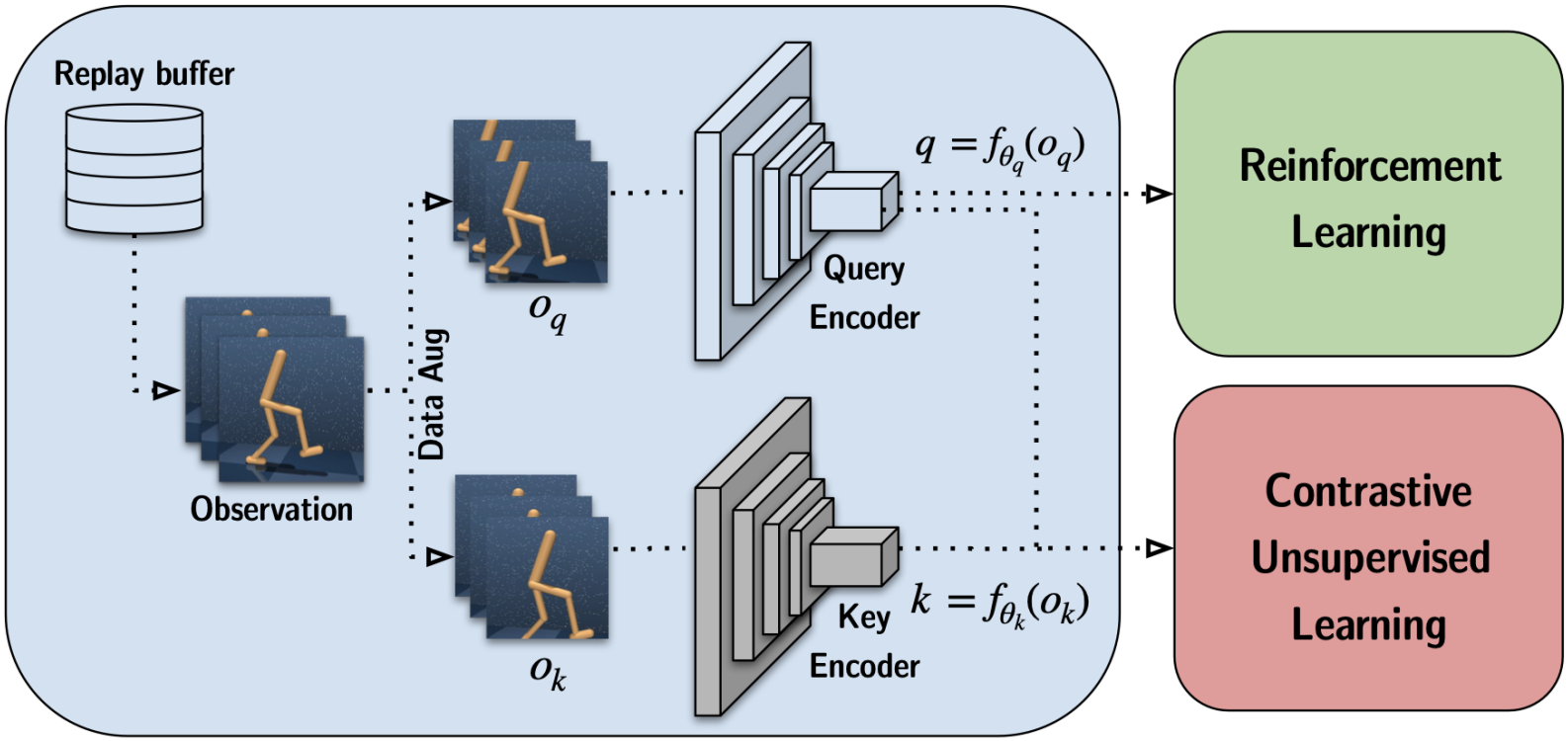
CURL

- The constrastive representation is trained jointly with the RL algorithm
- The similarity measure is: $sim(q, k) = q^T Wk$, where W is a learned parameter matrix
- InfoNCE loss learns encoders f_q and f_k mapping raw anchors (query) x_q and targets (keys) x_k into latents $q = f_q(x_q)$ and $k = f_k(x_k)$ on which to apply the similarity dot product, where it is common to have $f_q = f_k$
- They used momentum contrastive (MoCo) for encoding the keys: $\theta_k = m\theta_k + (1 - m)\theta_q$

CURL


o
o_q
o_k
Encoder
$q = f_{\theta_q}(o_q)$
Momentum Encoder
$k = f_{\theta_k}(o_k)$
$\theta_k = m\theta_k + (1-m)\theta_q$
q
k
Reinforcement Learning
Contrastive Loss

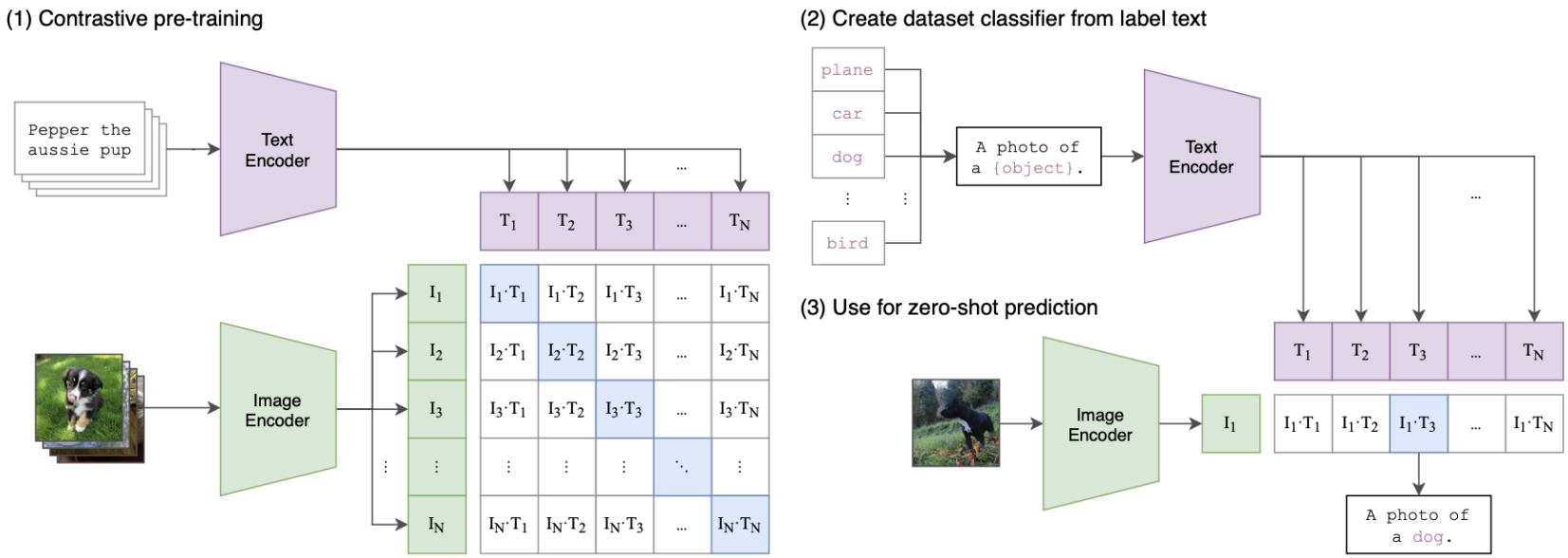
CURL


Replay buffer
Observation
Data Aug
o_q
o_k
Query Encoder
Key Encoder
$q = f_{\theta_q}(o_q)$
$k = f_{\theta_k}(o_k)$
Reinforcement Learning
Contrastive Unsupervised Learning

CLIP

- Learns from scratch on a database of 400 million (image,text) pairs
- Jointly train an image CNN and a text transformer to predict the caption of an image
- For each batch of N (image,text) pairs, CLIP is trained to predict all $N \times N$ pairs learning image and text encoders to maximize cosine similarity of real pairs and minimize cosine similarity of incorrect pairings

CLIP


(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
A photo of a dog.

Constrastive Learning

- It has been used for audio, text and speech
- There are some works on multimodal domains
- In unsupervised learning it is still an open research question how to distinguish between positive and negative samples

References

- A. Radford, J.W. Kim, C. Hallacy, A. Ramesh, G. Goh. S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever (2021). Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020v1 [cs.CV]
- L. Weng (2021). Contrastive representation learning. https://lilianweng.github.io/posts/2021-05-31-contrastive/
- P. Kumar, P. Rawat, S. Chauhan (2022). Contrastive self-supervised learning: review, progress, challenges and future research directions. International Journal of Multimedia Information Retrieval 11:461–488
- A. Srinivas, M. Laskin, P. Abbeel (2020). CURL: Contrastive Unsupervised Representations for Reinforcement Learning arXiv:2004.04136v4 [cs.LG]
- A. van den Oord. Y. Li, O. Vinyals (2019). Representation Learning with Contrastive Predictive Coding arXiv:1807.03748v2 [cs.LG]
- K. He, H. Fan, Y. Wu, S. Xie, R. Girshick (2020). Momentum Contrast for Unsupervised Visual Representation Learning CVPR 2020, 9729-9738
- D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, A. Zisserman (2021). With a Little Help from My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations. arXiv:2104.14548v2 [cs.CV]
- K.J. Joseph, S. Khan, F.S. Khan, V.N. Balasubramanian (2021). Towards Open World Object Detection arXiv:2103.02603v2 [cs.CV]
- J. Li, P. Zhou, C. Xiong, S.C.H. Hoi (2021). Prototypical Contrastive Learning of Unsupervised Representations. arXiv:2005.04966v5 [cs.CV]
- T. Chen, S. Kornblith, M. Norouzi, G. Hinton (2020). A Simple Framework for Contrastive Learning of Visual Representations. arXiv:2002.05709v3 [cs.LG]