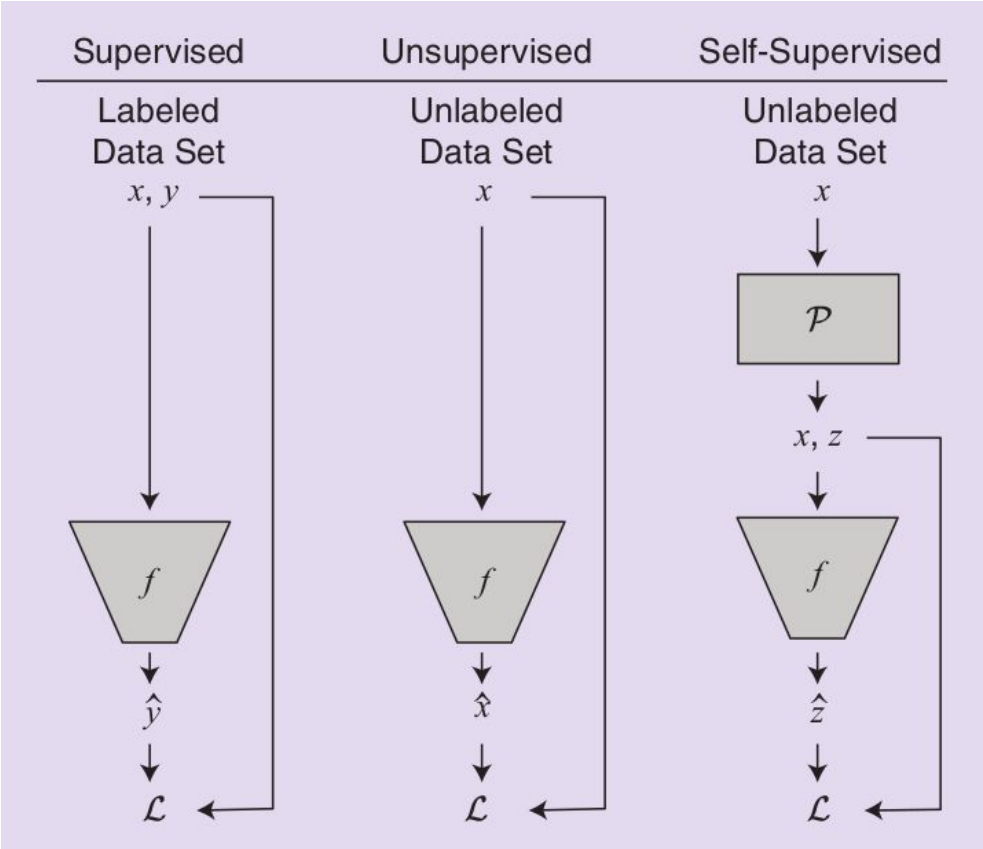
Self-Supervised Representation Learning (SSRL)

Overview


Supervised
Unsupervised
Self-Supervised
Labeled Data Set
Unlabeled Data Set
Unlabeled Data Set
x, y
x
x
P
x, z
f
f
f
ŷ
x̂
ẑ
ℒ
ℒ
ℒ

Supervised Model

- Utilizes labeled dataset D_t for training a predictive model $f(x) = g_\phi(h_\theta(x))$.
- The model comprises a representation extractor h_θ and a classifier/regression function g_ϕ.
- Training aims to minimize a loss function $\mathcal{L}$, optimizing parameters θ and ϕ.

Supervised Model

We train this predictive model by minimizing a loss function $\mathcal{L}$, such as the negative log likelihood

$$\underset{\theta,\phi}{\operatorname{argmin}} \sum_{\left(x_i^{(t)},y_i^{(t)}\right)\in D_t} \mathcal{L}\left(g_\phi\left(h_\theta\left(x_i^{(t)}\right)\right), y_i^{(t)}\right). \tag{1}$$

However, h_θ may have hundreds of millions of parameters, requiring millions of labeled data points in D_t to fit this correctly.

Unsupervised Models

Challenges with Large Parameter Spaces:

- h_θ can have hundreds of millions of parameters, requiring vast labeled datasets (D_t) for fitting.
- Labeled data (D_t) is often scarce, whereas unlabeled data (D_s) is abundant in many applications.

Utilizing Unlabeled Data through Unsupervised Learning:

- Leverages unlabeled data (D_s) to build generative models or learn compact latent representations.

Unsupervised Models

The common unsupervised methods, such as autoencoders and clustering learn compact latent representations. For example, autoencoders often optimize the following reconstruction objective:

$$\underset{\theta,\phi}{\operatorname{argmin}} \sum_{x_i^{(i)} \in D_t} \mathcal{L}\left(g_\phi\left(h_\theta\left(x_i^{(t)}\right)\right), x_i^{(t)}\right), \tag{2}$$

where $h_\theta(\cdot)$ extracts a compact feature from the input, and $g_\phi(\cdot)$ uses it to reconstruct the original input.

Self-Supervised Representation Learning (SSRL)

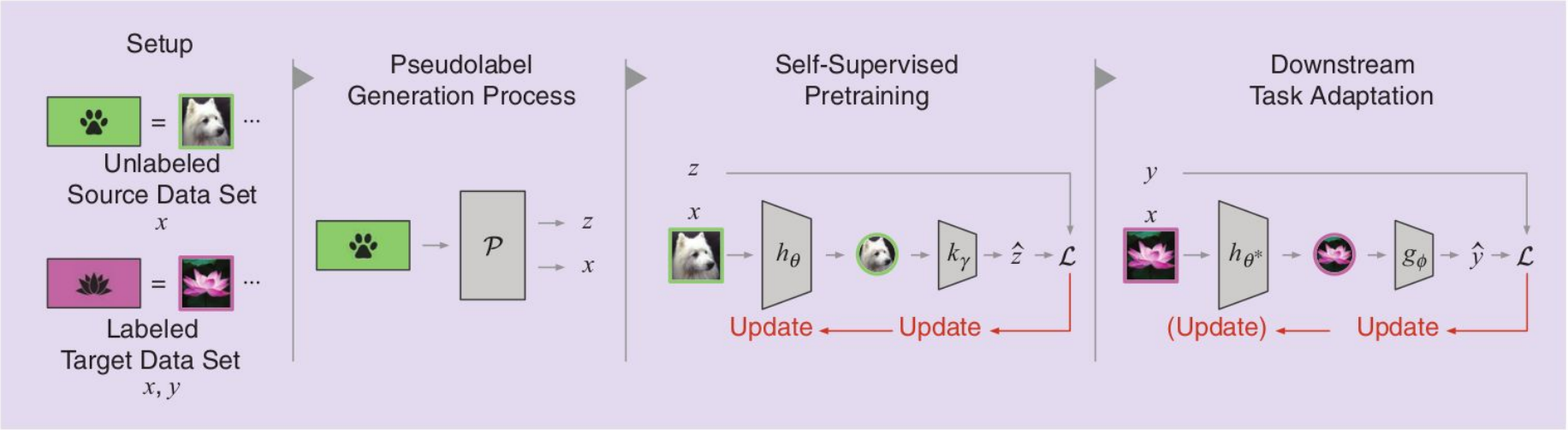


FIGURE 2. The self-supervised workflow starts with an unlabeled source data set and a labeled target data set. As defined by the pretext task, pseudolabels are programmatically generated from the unlabeled set. The resulting inputs, x and pseudolabels z, are used to pretrain the model $k_\gamma(h_\theta(\cdot))$—composed of feature extractor h_θ and output k_γ modules—to solve the pretext task. After pretraining is complete, the learned weights θ^* of the feature extractor h_{θ^*} are transferred and used together with a new output module g_ϕ to solve the downstream target task.

SSRL

- **Data Sets:**
  - Labeled target data set, D_t, for the specific task.
  - Larger unlabeled source data set, D_s, readily available.
- **Pseudolabeled Data Set Generation:**
  - Pretext task creates new pseudolabeled data set, $\bar{D}_s = \{x_i, z_i\}_{i=1}^{M} = \mathcal{P}(D_s)$.
  - Process $\mathcal{P}$ typically involves transformation or masking parameters.
  - Repeated at each training epoch's start for fresh sample generation.
- **Pretext Model Training:**
  - Trains model $k_\gamma(h_\theta(\cdot))$ to optimize the self-supervised objective on $\bar{D}_s$.
  - Objective: Minimize the loss $\mathcal{L}(k_\gamma(h_\theta(x_i)), z_i)$ across pseudolabeled data.
  - Results in optimal parameter estimation θ^* without the need for labeled data.
- **Input and Pseudolabel Nature:**
  - Typically, input x_i is a single data point.
  - Pseudolabel z_i usually represents a class label with scalar value.

Linear classifier (on downstream dataset)

For linear readout, let (θ, γ) be the weights of the pretrained model, consisting of a feature extractor, h_θ, followed by a task-specific head, k_γ.

The simplest way to reuse h_θ for a new task is to replace head with a new one, g_ϕ, designed for new task.

This head is then trained with the feature extractor frozen. Given a target data set of N instances, $D_t = \left\{x_i^{(t)}, y_i^{(t)}\right\}_{i=1}^{N}$, the training objective is

$$\underset{\phi}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(g_\phi\left(h_\theta\left(x_i^{(t)}\right)\right), y_i^{(t)}\right) \tag{4}$$

Pretext tasks

SSRL in Computer Vision

- The aim is to leverage self-supervised learning for representation learning, enabling models to learn meaningful and useful representations from unlabeled data.
- SSRL operates within the computer vision domain by utilizing freely available labels to train deep representations, thereby addressing the scalability bottleneck of supervised learning.

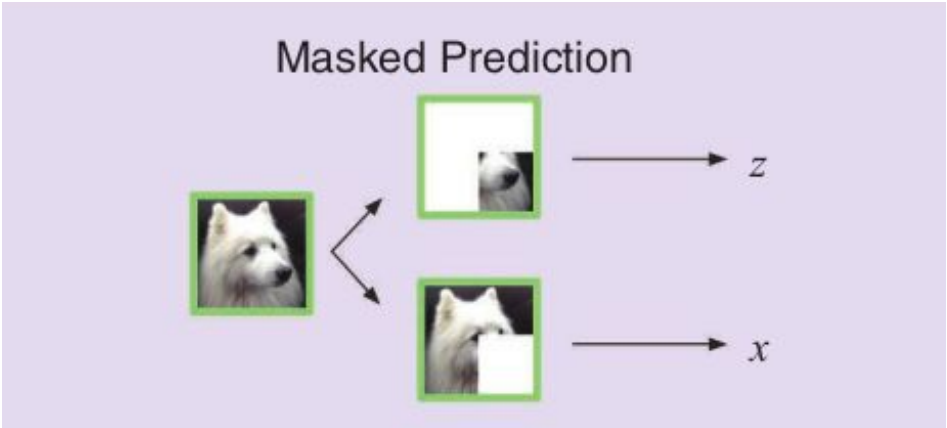
SSRL in Computer Vision

- SSRL achieves this by training models on **pretext tasks** that do not necessitate manual annotation.
- Pretext tasks involve learning to predict certain properties or relationships within the data, such as relative position or viewing angle of training images.
- Through training on these pretext tasks, models learn to extract meaningful features from the data without explicit labels.

Interesting tasks in SSRL

Masked Prediction:

- Model predicts missing information in input data with some elements masked.
- Minimizing reconstruction loss approach
- Examples: missing words in sentences, time slices in speech, or regions in images.



Masked Prediction

Algorithm 1. The pseudolabel generation process $\mathcal{P}$ for masked prediction.

Input: Unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^{M}$.

for i from 1 to M **do**

Generate indices, I, of elements to remove from $x_i^{(s)}$

$z_i \leftarrow \{x_{i,j}^{(s)} : j \in I\}$

$x_i \leftarrow \{x_{i,j}^{(s)} : j \notin I\}$

end for

Output: $\{x_i, z_i\}_{i=1}^{M}$.

Generating pseudolabels for masked prediction

- **Input**: Start with an unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^{M}$, where M is the number of data points.
- **Process**:
  - For each data point i from 1 to M:
    - Generate a set of indices I, which represent elements to be removed from $x_i^{(s)}$.
    - Create the pseudolabel z_i by selecting elements from $x_i^{(s)}$ corresponding to indices in I.
    - Define the modified input x_i by excluding elements from $x_i^{(s)}$ that are indexed by I.
- **Output**: Produce a set of pairs $\{x_i, z_i\}_{i=1}^{M}$, where each pair consists of the modified input and its corresponding pseudolabel.

Objective Function Overview

- Demonstrates using real data where a square region of an image is masked.
- Within this region, I denotes the set of indices inside the square mask.
- The masked pixels correspond to z_i, and pixels outside the masked region are x_i.
- Minimize a reconstruction loss, typically using mean square error for the masked region prediction.

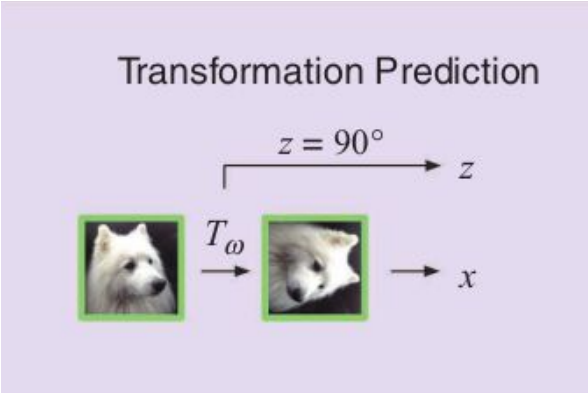
$$\theta^* = \underset{\theta,\gamma}{\operatorname{argmin}} \frac{1}{|\mathcal{P}(D_s)|} \sum_{(x_i,z_i)\in\mathcal{P}(D_s)} (k_\gamma(h_\theta(x_i)) - z_i)^2.$$

- This setup aims to predict the contents of the masked region accurately leveraging the surrounding pixels' information.

Interesting tasks in SSRL

Transformation Prediction:

- Model predicts applied transformation to input data, aiming for invariant representations.
- Valuable where labeled data is scarce, offering meaningful feature learning without explicit annotations.
- Examples: rotation angle of an image, original order of shuffled frames in a video.



Transformation Prediction

Algorithm 2. The pseudolabel generation process $\mathcal{P}$ for TP.

Input: Unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^{M}$.
  for i from 1 to M **do**
    Sample $\omega \sim \Omega$
    $x_i \leftarrow T_\omega(x_i^{(s)})$ ▷ Apply transformation to raw input
    $z_i \leftarrow \omega$
  end for
Output: $\{x_i, z_i\}_{i=1}^{M}$.

Pseudolabel generation process for Transformation Prediction (TP)

- **Input**: An unlabeled dataset $D_s = \{x_i^{(s)}\}_{i=1}^{M}$, which consists of M samples.
- **Process**:
  - For each sample i from 1 to M:
    - A transformation parameter ω is sampled from a predefined set of transformations Ω.
    - The transformation T_ω corresponding to the sampled parameter is applied to the input sample $x_i^{(s)}$ to generate a transformed sample x_i.
    - The transformation parameter ω is then assigned as the pseudolabel z_i for the transformed sample x_i.
- **Output**: The algorithm outputs a set of transformed samples and their corresponding pseudolabels $\{x_i, z_i\}_{i=1}^{M}$.

Learning Objective:

- The objective might include a cross-entropy loss for categorical transformation parameters.

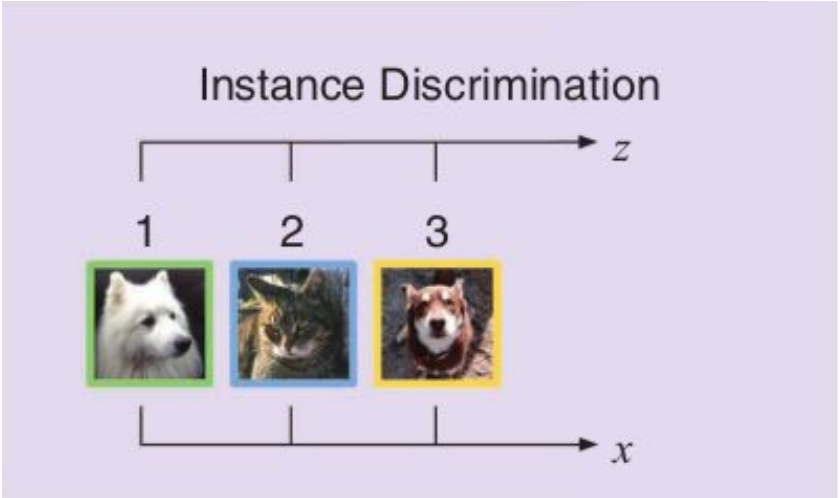
$$\theta^* = \underset{\theta,\gamma}{\operatorname{argmin}} \sum_{(x_i,z_i)\in\mathcal{P}(D_s)} \mathcal{L}_{CE}(k_\gamma(h_\theta(x_i)), z_i).$$

- The full process involves generating several views of each $x_i^{(s)}$, each with a different set of transformation parameters.

Interesting tasks in SSRL

Instance Discrimination:

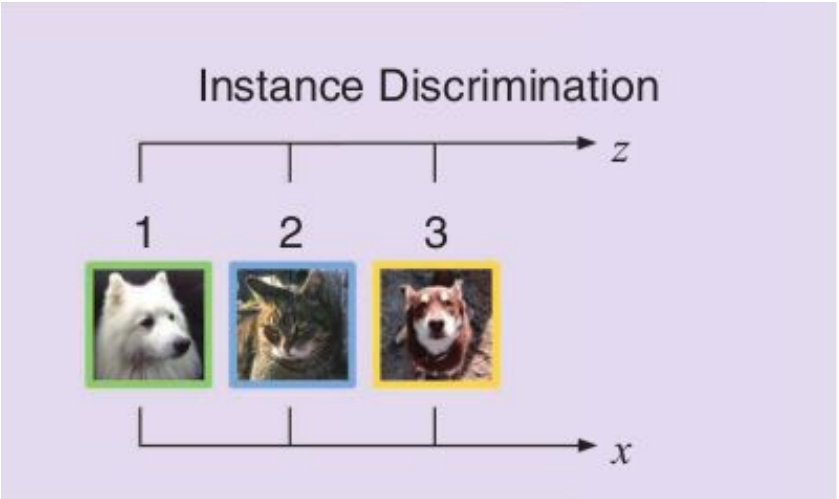
- Model treats each instance in the dataset as its own class and learns to discriminate between them.



Interesting tasks in SSRL

Instance Discrimination:

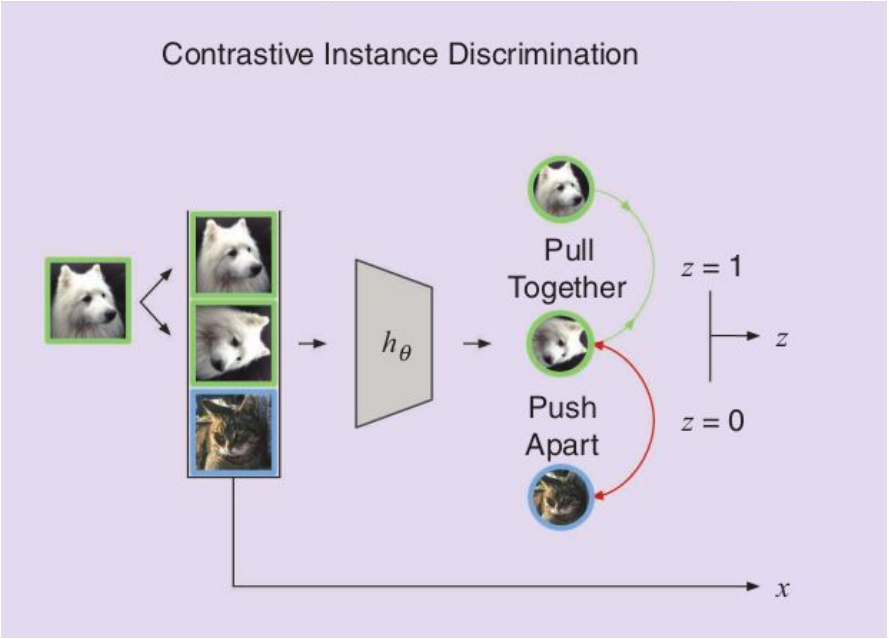
- **Cross Entropy:** Assign one-hot encoding of class label, train with categorical cross-entropy loss.
- **Contrastive Learning**: Train model to predict whether pairs of inputs belong to same or different classes, pulling positive pairs closer and pushing negative pairs apart.
- **Regularization-based Approaches**: Use regularization techniques to prevent feature collapse without negative examples.



Interesting tasks in SSRL

Contrastive Instance Discrimination:

- Instance discrimination that leverages contrastive loss.
- Pulls together same-instance vectors and pushes apart different-instance vectors.



Instance Discrimination

Algorithm 3. The pseudolabel generation process $\mathcal{P}$ for contrastive-instance discrimination.

Input: Unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^{M}$.
  for i from 1 to M **do**
    Sample $x^a \sim T(x_i^{(s)})$
    Sample $x^+ \sim T(x_i^{(s)})$
    for k from 1 to K **do**
      Sample $j \sim \mathcal{U}(1, M)$ ▷ Pick another raw input.
      Sample $x_k^- \sim T(x_j^{(s)})$ ▷ Get a random transform
    end for
    $x_i \leftarrow \{(x^a, x^+), (x^a, x_1^-), \ldots, (x^a, x_K^-)\}$.
    $z_i \leftarrow \{1, 0, \ldots, 0\}$.
  end for
Output: $\{x_i, z_i\}_{i=1}^{M}$.

Pseudolabel Generation for Contrastive-Instance Discrimination

- **Input**: An unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^{M}$.
- **Process** for each data point i from 1 to M:
  - Generate a transformed version of x_i, labeled x^a, using a transformation T.
  - Generate another transformed version of x_i, labeled x^+, with the same transformation T.
  - For k from 1 to K (number of negative samples):
    - Randomly select a different data point j from the data set.
    - Generate a transformed version of x_j, labeled x_k^-, with the transformation T.
- Create a tuple of transformed versions for x_i: $\{(x^a, x^+), (x^a, x_1^-), \ldots, (x^a, x_K^-)\}$.
- Assign a pseudolabel z_i with '1' for the positive pair (x^a, x^+) and '0's for all negative pairs (x^a, x_k^-).
- **Output**: A set of tuples $\{(x_i, z_i)\}_{i=1}^{M}$, each containing an anchor, a positive, and K negative samples with corresponding pseudolabels.

Objective Function

- **Feature Extraction:**
  - Samples are encoded by a feature extractor to obtain representations: $r^a = h_\theta(x^a)$, $r^+ = h_\theta(x^+), r_j^- = h_\theta(x_j^-)$.
- **Similarity Function:**
  - A similarity function Φ measures the similarity between positive pairs (the anchor with a positive sample) and negative pairs (the anchor with a negative sample).
- **Contrastive Loss:**

$$\mathcal{L}_{\text{con}} = -\mathbb{E}\left[\log \frac{\Phi(r^a, r^+)}{\Phi(r^a, r^+) + \sum_{j=1}^{k} \Phi(r^a, r_j^-)}\right],$$

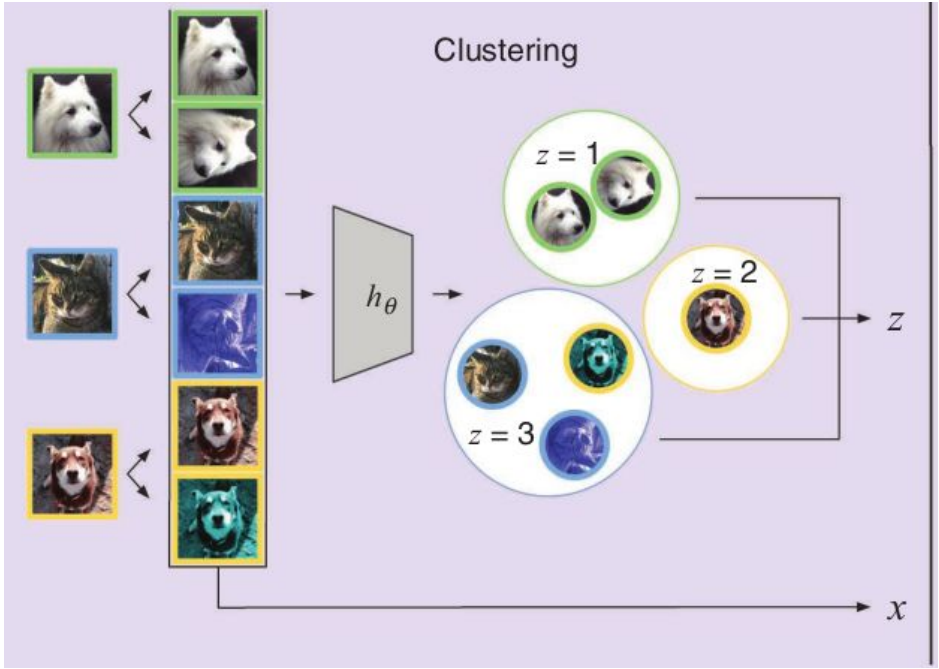
  where k different negative samples are contrasted with the anchor.
- **Model Update:**

$$\theta^* = \underset{\theta,\gamma}{\operatorname{argmin}} \sum_{(x_i,z_i)\in\mathcal{P}(D_x)} \mathcal{L}_{\text{con}}(k_\gamma(h_\theta(x_i)), z_i).$$

Interesting tasks in SSRL

Clustering:

- Model divides training data into groups with high intra-group similarity and low inter-group similarity.
- Methods involve joint learning of feature extractors and clustering.
- Goal: Find meaningful similarities for grouping instances.



Clustering

Algorithm 4. The pseudolabel generation process $\mathcal{P}$ for clustering.

Input: Unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^{M}$.
Input: Representations $\{r_i\}_{i=1}^{M}$, where $r_i \leftarrow h_\theta(x_i^{(s)})$
Input: Cluster centers $\{c_j\}_{j=1}^{k}$, via clustering on $\{r_i\}_{i=1}^{M}$.
  for i from 1 to M **do**
    Sample $x_i \sim T(x_i^{(s)})$
    $z_i \leftarrow \operatorname{argmin}_{j \in [k]} \| c_j - r_i \|$
  end for
Output: $\{x_i, z_i\}_{i=1}^{M}$.

Pseudolabel generation process for Clustering

- **Inputs**:
  - Unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^{M}$.
  - Representations $\{r_i\}_{i=1}^{M}$, where each r_i is derived from $h_\theta(x_i^{(s)})$.
  - Cluster centers $\{c_j\}_{j=1}^{k}$, obtained through clustering on the representations $\{r_i\}$.
- **Process**:
  - For each data point $x_i^{(s)}$ in D_s:
    - Transform $x_i^{(s)}$ using a transformation function T to get a sample x_i.
    - Assign a pseudolabel z_i by finding the nearest cluster center c_j to the representation r_i.
- **Output**:
  - The transformed samples and their corresponding pseudolabels $\{x_i, z_i\}_{i=1}^{M}$.

Objective Function for Clustering

- Involves assigning each input x_i to a cluster class z_i.
- The model is optimized using a cross-entropy loss based on these cluster assignments.
- Cross-Entropy Loss Optimization:

$$\theta^* = \underset{\theta,\gamma}{\operatorname{argmin}} \sum_{(x_i,z_i)\in\mathcal{P}(D_s)} \mathcal{L}_{CE}(k_\gamma(h_\theta(x_i)), z_i).$$

- After optimizing with cross-entropy loss, the process returns to the clustering step.
- Uses newly updated model representations for reassigning cluster classes, facilitating an iterative refinement of both model parameters and cluster assignments.