Lecture 9: Self-Supervised Learning
Recall: Supervised

**Data:** \((x, y)\)
x is the input data, y is the output label.

**Goal:** Learn a function \(f: x \rightarrow y\)

**Example:** in image classification, x is the image and y is the object category
Problem: Supervised Learning is Expensive!

Assume that we want to label re-label ImageNet’s 1.4 Million images.

How much will it cost?
Problem: Supervised Learning is Expensive!

Assume that we want to label re-label ImageNet’s 1.4 Million images.

How much will it cost?

\[(1,400,000 \text{ images}) \times (10 \text{ seconds/image}) \times (1/3600 \text{ hours/second}) \times ($15 / \text{ hour})\]  

(Small to medium sized dataset)  
(Fast annotation)  
(Low wage paid to annotator)
Problem: Supervised Learning is Expensive!

Assume that we want to label re-label ImageNet’s 1.4 Million images.

How much will it cost?

\[
\begin{align*}
(1,400,000 \text{ images}) & \quad \times (10 \text{ seconds/image}) & \quad (1/3600 \text{ hours/second}) & \quad \times ($15 / \text{ hour}) & \quad = $58,333 \\
\text{(Small to medium sized dataset)} & \quad \text{(Fast annotation)} & \quad \text{(Low wage paid to annotator)}
\end{align*}
\]

Assumptions:
- one annotator per image,
- no benefits / payroll tax / crowdsourcing fee for annotators;
- not accounting for front end developer time to set up tasks for annotators.
- Real costs could easily be 3x this or more: >$175,000
Problem: Supervised Learning is Expensive!

Assume that we want to label CLIP’s 1B images. (GPT also needs billions of documents)

How much will it cost?

\[
\begin{align*}
(1,000,000,000 \text{ images}) & \times (10 \text{ seconds/image}) \times (1/3600 \text{ hours/second}) \times ($15 / \text{ hour}) \\
& = \$41,666,667
\end{align*}
\]

41 Million dollars (again, not including all other costs)
Supervised Learning is Not How We Learn

Babies don’t get supervision for everything they see!
Solution: self-supervised learning

Let's build methods that learn from "raw" data – no annotations required

**Unsupervised Learning:** Model isn’t told what to predict. Older terminology, not used as much today.

**Self-Supervised Learning:** Model is trained to predict some naturally occurring signal in the raw data rather than human annotations.
Solution: self-supervised learning

Let's build methods that learn from "raw" data – no annotations required.

**Unsupervised Learning**: Model isn’t told what to predict. Older terminology, not used as much today.

**Self-Supervised Learning**: Model is trained to predict some naturally occurring signal in the raw data rather than human annotations.

**Semi-Supervised Learning**: Train jointly with some labeled data and (a lot) of unlabeled data.
Self-Supervised Learning: Pretext then Transfer

Step 1: Pretrain a network on a pretext task that doesn’t require supervision

Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

Downstream tasks:
Image classification, object detection, semantic segmentation
Goal of Self-Supervised Learning: Define pre-text tasks that do better than supervised learning

**Step 1:** Pretrain a network on a pretext task that doesn’t require supervision

**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

**Downstream tasks:** Image classification, object detection, semantic segmentation
Self-Supervised Learning: Pretext Tasks

**Generative**: Predict part of the input signal
- Autoencoders (sparse, denoising, masked)
- Autoregressive
- GANs
- Colorization
- Inpainting

**Discriminative**: Predict something about the input signal
- Context prediction
- Rotation • Clustering
- Contrastive

**Multimodal**: Use some additional signal in addition to RGB images
- Video
- 3D
- Sound
- Language
Self-supervised pretext tasks

Example: learn to predict image transformations / complete corrupted images

1. Solving the pretext tasks allow the model to learn good features.
2. We can automatically generate labels for the pretext tasks.
Generative Self-supervised Learning


Learning to generate pixel-level details is often unnecessary; learn high-level semantic features with pretext tasks instead

Source: Anand, 2020
How to evaluate a self-supervised learning method?

We usually don’t care about the performance of the self-supervised learning task, e.g., we don’t care if the model learns to predict image rotation perfectly.

Evaluate the learned feature encoders on downstream target tasks
How to evaluate a self-supervised learning method?

Step 1: **Pretrain** a network on a pretext task that doesn’t require supervision
How to evaluate a self-supervised learning method?

**Step 1:** Pretrain a network on a pretext task that doesn’t require supervision

**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning
Broader picture

computer vision

robot / reinforcement learning

language modeling

speech synthesis

Dense Object Net (Florence and Manuelli et al., 2018)

GPT3 (Brown, Mann, Ryder, Subbiah et al., 2020)

Doersch et al., 2015

Wavenet (van den Oord et al., 2016)
Today’s Agenda

Pretext tasks from image transformations
- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning
- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
Today’s Agenda

Pretext tasks from image transformations
- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning
- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
Pretext task: predict rotations

**Hypothesis**: a model could recognize the correct rotation of an object only if it has the “visual commonsense” of what the object should look like unperturbed.

(Image source: Gidaris et al. 2018)
Pretext task: predict rotations

Self-supervised learning by rotating the entire input images.

The model learns to predict which rotation is applied (4-way classification)

(Image source: Gidaris et al. 2018)
Pretext task: predict rotations

Self-supervised learning by rotating the entire input images.

The model learns to predict which rotation is applied (4-way classification)

(Image source: Gidaris et al. 2018)
Evaluation on semi-supervised learning

Self-supervised learning on **CIFAR10** (entire training set).

Freeze conv1 + conv2
Learn **conv3 + linear** layers with subset of labeled CIFAR10 data (classification).

(Image source: Gidaris et al. 2018)
Transfer learned features to supervised learning

<table>
<thead>
<tr>
<th>Source</th>
<th>Classification (%mAP)</th>
<th>Detection (%mAP)</th>
<th>Segmentation (%mIoU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trained layers</td>
<td>fc6-8</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td><strong>ImageNet labels</strong></td>
<td>78.9</td>
<td>79.9</td>
<td>56.8</td>
</tr>
<tr>
<td><strong>Random</strong></td>
<td></td>
<td>53.3</td>
<td>43.4</td>
</tr>
<tr>
<td>Random rescaled Krähenbühl et al. (2015)</td>
<td>39.2</td>
<td>56.6</td>
<td>45.6</td>
</tr>
<tr>
<td>Egomotion (Agraval et al., 2015)</td>
<td>31.0</td>
<td>54.2</td>
<td>43.9</td>
</tr>
<tr>
<td>Context Encoders (Pathak et al., 2016b)</td>
<td>34.6</td>
<td>56.5</td>
<td>44.5</td>
</tr>
<tr>
<td>Tracking (Wang &amp; Gupta, 2015)</td>
<td>55.6</td>
<td>63.1</td>
<td>47.4</td>
</tr>
<tr>
<td>Context (Doersch et al., 2015)</td>
<td>55.1</td>
<td>65.3</td>
<td>51.1</td>
</tr>
<tr>
<td>Colorization (Zhang et al., 2016a)</td>
<td>61.5</td>
<td>65.6</td>
<td>46.9</td>
</tr>
<tr>
<td>BIGAN (Donahue et al., 2016)</td>
<td>52.3</td>
<td>60.1</td>
<td>46.9</td>
</tr>
<tr>
<td>Jigsaw Puzzles (Noroozi &amp; Favaro, 2016)</td>
<td>-</td>
<td>67.6</td>
<td>53.2</td>
</tr>
<tr>
<td>NAT (Bojanowski &amp; Joulin, 2017)</td>
<td>56.7</td>
<td>65.3</td>
<td>49.4</td>
</tr>
<tr>
<td>Split-Brain (Zhang et al., 2016b)</td>
<td>63.0</td>
<td>67.1</td>
<td>46.7</td>
</tr>
<tr>
<td>ColorProxy (Larsson et al., 2017)</td>
<td>65.9</td>
<td>-</td>
<td>51.4</td>
</tr>
<tr>
<td>Counting (Noroozi et al., 2017)</td>
<td>-</td>
<td>67.7</td>
<td>36.6</td>
</tr>
<tr>
<td>(Ours) RotNet</td>
<td><strong>70.87</strong></td>
<td><strong>72.97</strong></td>
<td><strong>54.4</strong></td>
</tr>
</tbody>
</table>

Self-supervised learning on ImageNet (entire training set) with AlexNet.

Finetune on labeled data from Pascal VOC 2007.

Pretrained with full ImageNet supervision

No pretraining

Self-supervised learning with rotation prediction

source: Gidaris et al. 2018
Visualize learned visual attentions

(a) Attention maps of supervised model

(b) Attention maps of our self-supervised model

(Image source: Gidaris et al. 2018)
Pretext task: predict relative patch locations

Example:

Question 1:

Question 2:

(Image source: Doersch et al., 2015)
Pretext task: solving “jigsaw puzzles”

(Image source: Noroozi & Favaro, 2016)
Transfer learned features to supervised learning

Table 1: Results on PASCAL VOC 2007 Detection and Classification. The results of the other methods are taken from Pathak et al. [30].

<table>
<thead>
<tr>
<th>Method</th>
<th>Pretraining time</th>
<th>Supervision</th>
<th>Classification</th>
<th>Detection</th>
<th>Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krizhevsky et al. [25]</td>
<td>3 days</td>
<td>1000 class labels</td>
<td>78.2%</td>
<td>56.8%</td>
<td>48.0%</td>
</tr>
<tr>
<td>Wang and Gupta [39]</td>
<td>1 week</td>
<td>motion</td>
<td>58.4%</td>
<td>44.0%</td>
<td>-</td>
</tr>
<tr>
<td>Doersch et al. [10]</td>
<td>4 weeks</td>
<td>context</td>
<td>55.3%</td>
<td>46.6%</td>
<td>-</td>
</tr>
<tr>
<td>Pathak et al. [30]</td>
<td>14 hours</td>
<td>context</td>
<td>56.5%</td>
<td>44.5%</td>
<td>29.7%</td>
</tr>
<tr>
<td>Ours</td>
<td>2.5 days</td>
<td>context</td>
<td>67.6%</td>
<td>53.2%</td>
<td>37.6%</td>
</tr>
</tbody>
</table>

“Ours” is feature learned from solving image Jigsaw puzzles (Noroozi & Favaro, 2016). Doersch et al. is the method with relative patch location

(source: Noroozi & Favaro, 2016)
Pretext task: predict missing pixels (inpainting)

Context Encoders: Feature Learning by Inpainting (Pathak et al., 2016)

Source: Pathak et al., 2016
Learning to inpaint by reconstruction

Source: Pathak et al., 2016
Inpainting evaluation

Input (context)  reconstruction

Source: Pathak et al., 2016
Learning to inpaint by reconstruction

Loss = reconstruction + adversarial learning

\[ L(x) = L_{\text{recon}}(x) + L_{\text{adv}}(x) \]

\[ L_{\text{recon}}(x) = \| M \ast (x - F_{\theta}((1 - M) \ast x)) \|_2^2 \]

\[ L_{\text{adv}} = \max_D \mathbb{E}[\log(D(x))] + \log(1 - D(F((1 - M) \ast x))) \]

Adversarial loss between “real” images and inpainted images

Source: Pathak et al., 2016
Inpainting evaluation

Source: Pathak et al., 2016
Transfer learned features to supervised learning

<table>
<thead>
<tr>
<th>Pretraining Method</th>
<th>Supervision</th>
<th>Pretraining time</th>
<th>Classification</th>
<th>Detection</th>
<th>Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet [26]</td>
<td>1000 class labels</td>
<td>3 days</td>
<td>78.2%</td>
<td>56.8%</td>
<td>48.0%</td>
</tr>
<tr>
<td>Random Gaussian</td>
<td>initialization</td>
<td>&lt; 1 minute</td>
<td>53.3%</td>
<td>43.4%</td>
<td>19.8%</td>
</tr>
<tr>
<td>Autoencoder</td>
<td>-</td>
<td>14 hours</td>
<td>53.8%</td>
<td>41.9%</td>
<td>25.2%</td>
</tr>
<tr>
<td>Agrawal et al. [1]</td>
<td>egomotion</td>
<td>10 hours</td>
<td>52.9%</td>
<td>41.8%</td>
<td>-</td>
</tr>
<tr>
<td>Wang et al. [39]</td>
<td>motion</td>
<td>1 week</td>
<td>58.7%</td>
<td>47.4%</td>
<td>-</td>
</tr>
<tr>
<td>Doersch et al. [7]</td>
<td>relative context</td>
<td>4 weeks</td>
<td>55.3%</td>
<td>46.6%</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td>context</td>
<td>14 hours</td>
<td>56.5%</td>
<td>44.5%</td>
<td>30.0%</td>
</tr>
</tbody>
</table>

Self-supervised learning on ImageNet training set, transfer to classification (Pascal VOC 2007), detection (Pascal VOC 2007), and semantic segmentation (Pascal VOC 2012)

Source: Pathak et al., 2016
Pretext task: image coloring

Grayscale image: \( L \) channel

\[
X \in \mathbb{R}^{H \times W \times 1}
\]

\[
L \xrightarrow{\mathcal{F}} \hat{Y} \in \mathbb{R}^{H \times W \times 2}
\]

Color information: \( ab \) channels

Source: Richard Zhang / Phillip Isola
Pretext task: image coloring

Grayscale image: $L$ channel

$X \in \mathbb{R}^{H \times W \times 1}$

Concatenate $(L, ab)$ channels

$(X, \hat{Y})$

Source: Richard Zhang / Phillip Isola
Learning features from colorization: Split-brain Autoencoder

**Idea**: cross-channel predictions

[Diagram of Split-Brain Autoencoder]
Learning features from colorization: Split-brain Autoencoder

Source: Richard Zhang / Phillip Isola
Learning features from colorization: 
Split-brain Autoencoder

Source: Richard Zhang / Phillip Isola
Transfer learned features to supervised learning

Self-supervised learning on **ImageNet** (entire training set).

Use *concatenated features* from $F_1$ and $F_2$.

Labeled data is from the **Places** (Zhou 2016).

Source: *Zhang et al., 2017*
Real world application: image coloring

Source: Richard Zhang / Phillip Isola
Pretext task: image coloring

Source: Richard Zhang / Phillip Isola
Pretext task: video coloring

Idea: model the *temporal coherence* of colors in videos

reference frame

how should I color these frames?

$t = 0$

$t = 1$

$t = 2$

$t = 3$

...
Pretext task: video coloring

**Idea:** model the *temporal coherence* of colors in videos

reference frame

how should I color these frames?

Should be the same color!

**Hypothesis:** learning to color video frames should allow model to learn to track regions or objects without labels!

Source: Vondrick et al., 2018
Learning to color videos

Learning objective:

Establish mappings between reference and target frames in a learned feature space.

Use the mapping as “pointers” to copy the correct color (LAB).

Source: Vondrick et al., 2018
Learning to color videos

Grayscale Video

Reference Frame

Target Frame

CNN

Embeddings

Reference Colors

Predicted Colors

attention map on the reference frame

\[ A_{ij} = \frac{\exp \left( f_i^T f_j \right)}{\sum_k \exp \left( f_k^T f_j \right)} \]

Source: Vondrick et al., 2018
Learning to color videos

attention map on the reference frame

\[ A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)} \]

predicted color = weighted sum of the reference color

\[ y_j = \sum_i A_{ij} c_i \]

Source: Vondrick et al., 2018
Learning to color videos

Attention map on the reference frame

Predicted color = weighted sum of the reference color

Loss between predicted color and ground truth color

\[ A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)} \]

\[ y_j = \sum_i A_{ij} c_i \]

\[ \min_{\theta} \sum_j \mathcal{L}(y_j, c_j) \]

Source: Vondrick et al., 2018
Colorizing videos (qualitative)

reference frame  target frames (gray)  predicted color

Source: Google AI blog post
Colorizing videos (qualitative)

reference frame  target frames (gray)  predicted color

Source: Google AI blog post
Tracking emerges from colorization
Propagate segmentation masks using learned attention

Source: Google AI blog post
Tracking emerges from colorization

Propagate pose keypoints using learned attention

Source: Google AI blog post
Summary: pretext tasks from image transformations

- Pretext tasks focus on “visual common sense”, e.g., predict rotations, inpainting, rearrangement, and colorization.
- The models are forced learn good features about natural images, e.g., semantic representation of an object category, in order to solve the pretext tasks.
- We don’t care about the performance of these pretext tasks, but rather how useful the learned features are for downstream tasks (classification, detection, segmentation).
Summary: pretext tasks from image transformations

- Pretext tasks focus on “visual common sense”, e.g., predict rotations, inpainting, rearrangement, and colorization.

- The models are forced learn good features about natural images, e.g., semantic representation of an object category, in order to solve the pretext tasks.

- We don’t care about the performance of these pretext tasks, but rather how useful the learned features are for downstream tasks (classification, detection, segmentation).

- Problems: 1) coming up with individual pretext tasks is tedious, and 2) the learned representations may not be generally useful for all downstream tasks.
Pretext tasks from image transformations

- image completion
- rotation prediction
- “jigsaw puzzle”
- colorization

Learned representations may be tied to a specific pretext task!

Can we come up with a more general pretext task?
A more general pretext task?

same object
A more general pretext task?

?  

same object  

different object  

θ = ?
Contrastive Representation Learning

attract

repel
Today’s Agenda

Pretext tasks from image transformations
- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning
- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC
Self-Supervised Learning: Pretext then Transfer

**Step 1:** Pretrain a network on a pretext task that doesn’t require supervision

- **Input Image:** $x$
- **Features:** $\phi(x)$
- **Prediction:** $\hat{y}$
- **Loss:** $L(\hat{y}, y)$

**Step 2:** Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

**Downstream tasks:** Image classification, object detection, semantic segmentation
Summary: pretext tasks from image transformations

- Pretext tasks focus on “visual common sense”, e.g., predict rotations, inpainting, rearrangement, and colorization.
- The models are forced to learn good features about natural images, e.g., semantic representation of an object category, in order to solve the pretext tasks.
- We don’t care about the performance of these pretext tasks, but rather how useful the learned features are for downstream tasks (classification, detection, segmentation).
- Problems: 1) coming up with individual pretext tasks is tedious, and 2) the learned representations may not be generally useful for all downstream tasks.
Contrastive Representation Learning

? attract

repel
Contrastive Representation Learning

\[ x^+ \] reference

\[ x^+ \] positive

\[ x^- \] negative

\[ x \]
A formulation of contrastive learning

What we want:

$$\text{score}(f(x), f(x^+)) \gg \text{score}(f(x), f(x^-))$$

$x$: reference sample; $x^+$ positive sample; $x^-$ negative sample

Given a chosen score function, we aim to learn an encoder function $f$ that yields high score for positive pairs $(x, x^+)$ and low scores for negative pairs $(x, x^-)$. 
A formulation of contrastive learning

Loss function given 1 positive sample and N - 1 negative samples:

\[
L = -\mathbb{E}_x \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]
\]

Q. What does this loss function remind you of?
A formulation of contrastive learning

Loss function given 1 positive sample and $N - 1$ negative samples:

$$L = -\mathbb{E}_X \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x^-_j)))} \right]$$
A formulation of contrastive learning

Loss function given 1 positive sample and N - 1 negative samples:

$$L = -\mathbb{E}_X \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]$$

score for the positive pair

score for the N-1 negative pairs

This seems familiar …
A formulation of contrastive learning
Loss function given 1 positive sample and N - 1 negative samples:

\[
L = -\mathbb{E}_x \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]
\]

This seems familiar …
Cross entropy loss for a N-way softmax classifier!
I.e., learn to find the positive sample from the N samples
A formulation of contrastive learning

Loss function given 1 positive sample and \( N - 1 \) negative samples:

\[
L = -\mathbb{E}_x \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]
\]

Very similar to the softmax classifier we talked about a few lectures ago.

- We want to compare the reference image against all other positive and negative images.
- We can exponentiate and normalize these scores like we did with the softmax classifier.
- And we get the above similar equation.
A formulation of contrastive learning

Loss function given 1 positive sample and N - 1 negative samples:

\[ L = -\mathbb{E}_X \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right] \]

Commonly known as the InfoNCE loss (van den Oord et al., 2018)

A lower bound on the mutual information between \( f(x) \) and \( f(x^+) \):

\[ MI[f(x), f(x^+)] - \log(N) \geq -L \]

The larger the negative sample size (N), the tighter the bound

Detailed derivation: Poole et al., 2019
SimCLR: A Simple Framework for Contrastive Learning

Cosine similarity as the score function:

$$s(u, v) = \frac{u^T v}{||u|| ||v||}$$

Use a projection network $h(\cdot)$ to project features to a space where contrastive learning is applied.

Generate positive samples through data augmentation:
- random cropping, random color distortion, and random blur.

Source: Chen et al., 2020
SimCLR: generating positive samples from data augmentation

Source: Chen et al., 2020
SimCLR

Generate a positive pair by sampling data augmentation functions

```
Algorithm 1 SimCLR’s main learning algorithm.

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

*We use a slightly different formulation in the assignment. You should follow the assignment instructions.

Source: Chen et al., 2020
SimCLR

*We use a slightly different formulation in the assignment. You should follow the assignment instructions.

**Algorithm 1** SimCLR’s main learning algorithm.

- **input:** batch size $N$, constant $\tau$, structure of $f$, $g$, $\mathcal{T}$.
- for sampled minibatch $\{x_k\}_{k=1}^N$
  - for all $k \in \{1, \ldots, N\}$
    - draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
      - the first augmentation
        - $\tilde{x}_{2k-1} = t(x_k)$
        - $h_{2k-1} = f(\tilde{x}_{2k-1})$
        - $z_{2k-1} = g(h_{2k-1})$
      - the second augmentation
        - $\tilde{x}_{2k} = t'(x_k)$
        - $h_{2k} = f(\tilde{x}_{2k})$
        - $z_{2k} = g(h_{2k})$
  - for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$
    - $s_{i,j} = z_i^T z_j / (\|z_i\| \|z_j\|)$
      - pairwise similarity
  - 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 $\mathcal{L}$

- return encoder network $f(\cdot)$, and throw away $g(\cdot)$

Source: Chen et al., 2020

InfoNCE loss: Use all non-positive samples in the batch as $x^-$

Generate a positive pair by sampling data augmentation functions
SimCLR

**Algorithm 1** SimCLR’s main learning algorithm.

**input:** batch size $N$, constant $\tau$, structure of $f$, $g$, $\mathcal{T}$.

for sampled minibatch $\{x_k\}_{k=1}^N$
do
  for all $k \in \{1, \ldots, N\}$
do
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{x}_{2k-1} = t(x_k)$
    $h_{2k-1} = f(\tilde{x}_{2k-1})$ # representation
    $z_{2k-1} = g(h_{2k-1})$ # projection
  end for

  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$
do
    $s_{i,j} = z_i^\top z_j / (\|z_i\|\|z_j\|)$ # pairwise similarity
  end for

  define $\ell(i, j)$ as
  \[
  \ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \exp(s_{i,k}/\tau)}
  \]
  $L = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$

  update networks $f$ and $g$ to minimize $L$
end for

return encoder network $f(\cdot)$, and throw away $g(\cdot)$

*We use a slightly different formulation in the assignment. You should follow the assignment instructions.*

**Source:** [Chen et al., 2020](#)
SimCLR: mini-batch training

Each 2k and 2k + 1 element is a positive pair

We use a slightly different formulation in the assignment.
You should follow the assignment instructions.

\[
S_{i,j} = \frac{z_i^T z_j}{||z_i|| \cdot ||z_j||}
\]

"Affinity matrix"
SimCLR: mini-batch training

Each 2k and 2k + 1 element is a positive pair

*We use a slightly different formulation in the assignment. You should follow the assignment instructions.

\[ s_{i,j} = \frac{z_i^T z_j}{\|z_i\| \|z_j\|} \]

"Affinity matrix"

= classification label for each row

\[ z \in \mathbb{R}^{2N \times D} \]
SimCLR: what a batch looks like

Batch of N images | Two augmentations for each image | Extract features
Training linear classifier on SimCLR features

Train feature encoder on ImageNet (entire training set) using SimCLR.

Freeze feature encoder, train a linear classifier on top with labeled data.

Source: Chen et al., 2020
Semi-supervised learning on SimCLR features

<table>
<thead>
<tr>
<th>Method</th>
<th>Architecture</th>
<th>Label fraction 1%</th>
<th>Label fraction 10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised baseline</td>
<td>ResNet-50</td>
<td>48.4</td>
<td>80.4</td>
</tr>
<tr>
<td>Methods using other label-propagation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pseudo-label</td>
<td>ResNet-50</td>
<td>51.6</td>
<td>82.4</td>
</tr>
<tr>
<td>VAT+Entropy Min.</td>
<td>ResNet-50</td>
<td>47.0</td>
<td>83.4</td>
</tr>
<tr>
<td>UDA (w. RandAug)</td>
<td>ResNet-50</td>
<td>-</td>
<td>88.5</td>
</tr>
<tr>
<td>FixMatch (w. RandAug)</td>
<td>ResNet-50</td>
<td>-</td>
<td>89.1</td>
</tr>
<tr>
<td>S4L (Rot+VAT+En. M.)</td>
<td>ResNet-50 (4×)</td>
<td>-</td>
<td>91.2</td>
</tr>
<tr>
<td>Methods using representation learning only:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InstDisc</td>
<td>ResNet-50</td>
<td>39.2</td>
<td>77.4</td>
</tr>
<tr>
<td>BigBiGAN</td>
<td>RevNet-50 (4×)</td>
<td>55.2</td>
<td>78.8</td>
</tr>
<tr>
<td>PIRL</td>
<td>ResNet-50</td>
<td>57.2</td>
<td>83.8</td>
</tr>
<tr>
<td>CPC v2</td>
<td>ResNet-161(*)</td>
<td>77.9</td>
<td>91.2</td>
</tr>
<tr>
<td>SimCLR (ours)</td>
<td>ResNet-50</td>
<td>75.5</td>
<td>87.8</td>
</tr>
<tr>
<td>SimCLR (ours)</td>
<td>ResNet-50 (2×)</td>
<td>83.0</td>
<td>91.2</td>
</tr>
<tr>
<td>SimCLR (ours)</td>
<td>ResNet-50 (4×)</td>
<td><strong>85.8</strong></td>
<td><strong>92.6</strong></td>
</tr>
</tbody>
</table>

Table 7. ImageNet accuracy of models trained with few labels.

Train feature encoder on ImageNet (entire training set) using SimCLR.

Finetune the encoder with 1% / 10% of labeled data on ImageNet.

Source: Chen et al., 2020
SimCLR design choices: projection head

Linear / non-linear projection heads improve representation learning.

A possible explanation:

- contrastive learning objective may discard useful information for downstream tasks
- representation space $z$ is trained to be invariant to data transformation.
- by leveraging the projection head $g(\cdot)$, more information can be preserved in the $h$ representation space

Source: Chen et al., 2020
SimCLR design choices: large batch size

Large training batch size is crucial for SimCLR!

Large batch size causes large memory footprint during backpropagation: requires distributed training on TPUs (ImageNet experiments)

Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.\textsuperscript{10}

Source: Chen et al., 2020
Momentum Contrastive Learning (MoCo)

Key differences to SimCLR:

- Keep a running queue of keys (negative samples).
- Compute gradients and update the encoder only through the queries.
- Decouple min-batch size with the number of keys: can support a large number of negative samples.

Source: He et al., 2020
Momentum Contrastive Learning (MoCo)

Key differences to SimCLR:

- Keep a running queue of keys (negative samples).
- Compute gradients and update the encoder only through the queries.
- Decouple min-batch size with the number of keys: can support a large number of negative samples.
- The key encoder is slowly progressing through the momentum update rules:

\[
\theta_k \leftarrow m\theta_k + (1 - m)\theta_q
\]

Source: He et al., 2020
MoCo

Generate a positive pair by sampling data augmentation functions

No gradient through the negative samples

Update the FIFO negative sample queue

Use the running queue of keys as the negative samples

InfoNCE loss

Update \( f_k \) through momentum

Source: He et al., 2020

---

Algorithm 1: Pseudocode of MoCo in a PyTorch-like style.

```python
# 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

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.
```

---

Source: He et al., 2020
“MoCo V2”

Improved Baselines with Momentum Contrastive Learning

Xinlei Chen    Haoqi Fan    Ross Girshick    Kaiming He
Facebook AI Research (FAIR)

A hybrid of ideas from SimCLR and MoCo:
- **From SimCLR**: non-linear projection head and strong data augmentation.
- **From MoCo**: momentum-updated queues that allow training on a large number of negative samples (no TPU required!).

Source: Chen et al., 2020
MoCo vs. SimCLR vs. MoCo V2

Key takeaways:

- Non-linear projection head and strong data augmentation are crucial for contrastive learning.

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.

<table>
<thead>
<tr>
<th>case</th>
<th>unsup. pre-train</th>
<th>ImageNet acc.</th>
<th>VOC detection AP50</th>
<th>AP</th>
<th>AP75</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MLP aug+ cos epochs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>supervised</td>
<td></td>
<td>76.5</td>
<td>81.3 55.5 58.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MoCo v1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td>√</td>
<td>200</td>
<td>60.6 55.9 62.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td>200</td>
<td>66.2 56.4 62.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>√ √</td>
<td>200</td>
<td>63.4 56.8 63.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d)</td>
<td>√ √ √</td>
<td>200</td>
<td>67.3 57.2 63.9</td>
<td></td>
<td>82.5</td>
</tr>
<tr>
<td>(e)</td>
<td>√ √ √</td>
<td>800</td>
<td>67.5 57.0 63.6</td>
<td></td>
<td>71.1</td>
</tr>
</tbody>
</table>

Source: Chen et al., 2020
MoCo vs. SimCLR vs. MoCo V2

Key takeaways:

- Non-linear projection head and strong data augmentation are crucial for contrastive learning.

- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).

<table>
<thead>
<tr>
<th>case</th>
<th>MLP</th>
<th>unsup. pre-train</th>
<th>ImageNet acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo v1 [6]</td>
<td></td>
<td></td>
<td>60.6</td>
</tr>
<tr>
<td>SimCLR [2]</td>
<td>✓</td>
<td>✓</td>
<td>61.9</td>
</tr>
<tr>
<td>SimCLR [2]</td>
<td>✓</td>
<td>✓</td>
<td>66.6</td>
</tr>
<tr>
<td>MoCo v2</td>
<td>✓</td>
<td>✓</td>
<td>67.5</td>
</tr>
</tbody>
</table>

Table 2. MoCo vs. SimCLR: ImageNet linear classifier accuracy (ResNet-50, 1-crop 224×224), trained on features from unsupervised pre-training. “aug+” in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

Source: Chen et al., 2020
## MoCo vs. SimCLR vs. MoCo V2

### Key takeaways:

- Non-linear projection head and strong data augmentation are crucial for contrastive learning.

- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).

- … all with much smaller memory footprint! ("end-to-end" means SimCLR here)

<table>
<thead>
<tr>
<th>mechanism</th>
<th>batch</th>
<th>memory / GPU</th>
<th>time / 200-ep.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo</td>
<td>256</td>
<td>5.0G</td>
<td>53 hrs</td>
</tr>
<tr>
<td>end-to-end</td>
<td>256</td>
<td>7.4G</td>
<td>65 hrs</td>
</tr>
<tr>
<td>end-to-end</td>
<td>4096</td>
<td>93.0G†</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.

Source: [Chen et al., 2020](#)
Problem with MoCoV2: Need to keep around a set of negatives

Do we need these negatives?
Solution: DINO: self-distillation with no labels

- Similar to SimCLR and MOCO but with one big difference: no negatives
- Reformulates contrastive learning as knowledge distillation between a student and a teacher model.

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021
Solution: DINO: self-distillation with no labels

- The teacher model is not trained: \textit{sg} stands for stop-gradient: meaning that gradients are prevented from flowing back.

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021
Problem: But how do we choose the teacher model?

- The teacher model is like the momentum encoder. It is a running average of the student model

\[ \theta_t \leftarrow \lambda \theta_t + (1 - \lambda) \theta_s \]

- The teacher sees a global view augmentation of the image
- Student only sees augmented local crops of the image

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021
Problem: But how do we choose the teacher model?

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021
Problem: But how do we choose the teacher model?

Training tricks:
- **Centering**: prevents one dimension from dominating.
  - A constant value $c$ is added to all dimensions of the teacher’s output.
  - $c$ is a running average of outputs
    \[
    g_t(x) \leftarrow g_t(x) + c, \quad c \leftarrow mc + (1 - m) \frac{1}{B} \sum_{i=1}^{B} g_{\theta_t}(x_i)
    \]

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021
Problem: But how do we choose the teacher model?

Training tricks:
- **Sharpening**: Opposite of centering.
  - A temperature (Tau) hyperparameter is used to sharpen the distributions towards one dimension.

\[
\frac{\exp(g_{\theta_s}(x)^{(i})/\tau_s)}{\sum_{k=1}^{K} \exp(g_{\theta_s}(x)^{(k})/\tau_s)}
\]

Source: Caron et al. Emerging Properties in Self-Supervised Vision Transformers. 2021
**Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.**

```python
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
# gt.params = gs.params
for x in loader:  # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x)  # random views
    s1, s2 = gs(x1), gs(x2)  # student output n-by-K
    t1, t2 = gt(x1), gt(x2)  # teacher output n-by-K
    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward()  # back-propagate

# student, teacher and center updates
update(gs)  # SGD
gt.params = l*gt.params + (1-l)*gs.params
C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach()  # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1)  # center + sharpen
    return -(t * log(s)).sum(dim=1).mean()
```

---

**Diagram:**

- **1.** Softmax operations and centering for student and teacher networks.
- **2.** Loss calculation using entropy (H).
Results: DINO. Also DINO V2 just released last week

<table>
<thead>
<tr>
<th>Method</th>
<th>Arch.</th>
<th>Param.</th>
<th>im/s</th>
<th>Linear</th>
<th>k-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>79.3</td>
<td>79.3</td>
</tr>
<tr>
<td>SCLR [12]</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>69.1</td>
<td>60.7</td>
</tr>
<tr>
<td>MoCov2 [15]</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>71.1</td>
<td>61.9</td>
</tr>
<tr>
<td>InfoMin [67]</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>73.0</td>
<td>65.3</td>
</tr>
<tr>
<td>BarlowT [81]</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>73.2</td>
<td>66.0</td>
</tr>
<tr>
<td>OBoW [27]</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>73.8</td>
<td>61.9</td>
</tr>
<tr>
<td>BYOL [30]</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>74.4</td>
<td>64.8</td>
</tr>
<tr>
<td>DCv2 [10]</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>75.2</td>
<td>67.1</td>
</tr>
<tr>
<td>SwAV [10]</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>75.3</td>
<td>65.7</td>
</tr>
<tr>
<td>DINO</td>
<td>RN50</td>
<td>23</td>
<td>1237</td>
<td>75.3</td>
<td>67.5</td>
</tr>
</tbody>
</table>
Instance vs. Sequence Contrastive Learning

**Instance-level contrastive learning:**
contrastive learning based on positive & negative instances.
Examples: SimCLR, MoCo

**Sequence-level contrastive learning:**
contrastive learning based on sequential / temporal orders.
Example: **Contrastive Predictive Coding (CPC)**

Source: van den Oord et al., 2018
Contrastive Predictive Coding (CPC)

Contrastive: contrast between “right” and “wrong” sequences using contrastive learning.

Predictive: the model has to predict future patterns given the current context.

Coding: the model learns useful feature vectors, or “code”, for downstream tasks, similar to other self-supervised methods.

Figure source

Source: van den Oord et al., 2018.
Contrastive Predictive Coding (CPC)

1. Encode all samples in a sequence into vectors $z_t = g_{\text{enc}}(x_t)$
Contrastive Predictive Coding (CPC)

1. Encode all samples in a sequence into vectors $z_t = g_{enc}(x_t)$

2. Summarize context (e.g., half of a sequence) into a context code $c_t$ using an auto-regressive model ($g_{ar}$). The original paper uses GRU-RNN here.
Contrastive Predictive Coding (CPC)

1. Encode all samples in a sequence into vectors $z_t = g_{\text{enc}}(x_t)$

2. Summarize context (e.g., half of a sequence) into a context code $c_t$ using an auto-regressive model ($g_{\text{ar}}$)

3. Compute InfoNCE loss between the context $c_t$ and future code $z_{t+k}$ using the following time-dependent score function:

$$s_k(z_{t+k}, c_t) = z_{t+k}^T W_k c_t$$

, where $W_k$ is a trainable matrix.

Figure source: van den Oord et al., 2018
CPC example: modeling audio sequences

Source: van den Oord et al., 2018,
CPC example: modeling audio sequences

Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

<table>
<thead>
<tr>
<th>Method</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phone classification</strong></td>
<td></td>
</tr>
<tr>
<td>Random initialization</td>
<td>27.6</td>
</tr>
<tr>
<td>MFCC features</td>
<td>39.7</td>
</tr>
<tr>
<td>CPC</td>
<td>64.6</td>
</tr>
<tr>
<td>Supervised</td>
<td>74.6</td>
</tr>
<tr>
<td><strong>Speaker classification</strong></td>
<td></td>
</tr>
<tr>
<td>Random initialization</td>
<td>1.87</td>
</tr>
<tr>
<td>MFCC features</td>
<td>17.6</td>
</tr>
<tr>
<td>CPC</td>
<td>97.4</td>
</tr>
<tr>
<td>Supervised</td>
<td>98.5</td>
</tr>
</tbody>
</table>

Linear classification on trained representations (LibriSpeech dataset)

Source: van den Oord et al., 2018.
CPC example: modeling visual context

Idea: split image into patches, model rows of patches from top to bottom as a sequence. I.e., use top rows as context to predict bottom rows.

Source: van den Oord et al., 2018,
CPC example: modeling visual context

- Compares favorably with other pretext task-based self-supervised learning method.
- Doesn’t do as well compared to newer instance-based contrastive learning methods on image feature learning.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Using AlexNet conv5</strong></td>
<td></td>
</tr>
<tr>
<td>Video [28]</td>
<td>29.8</td>
</tr>
<tr>
<td>BiGan [35]</td>
<td>34.8</td>
</tr>
<tr>
<td>Colorization [10]</td>
<td>35.2</td>
</tr>
<tr>
<td>Jigsaw [29] *</td>
<td>38.1</td>
</tr>
<tr>
<td><strong>Using ResNet-V2</strong></td>
<td></td>
</tr>
<tr>
<td>Motion Segmentation [36]</td>
<td>27.6</td>
</tr>
<tr>
<td>Exemplar [36]</td>
<td>31.5</td>
</tr>
<tr>
<td>Relative Position [36]</td>
<td>36.2</td>
</tr>
<tr>
<td>Colorization [36]</td>
<td>39.6</td>
</tr>
<tr>
<td>CPC</td>
<td>48.7</td>
</tr>
</tbody>
</table>

Table 3: ImageNet top-1 unsupervised classification results. *Jigsaw is not directly comparable to the other AlexNet results because of architectural differences.

Source: [van den Oord et al., 2018](#).
Summary: Contrastive Representation Learning

A general formulation for contrastive learning:

\[
\text{score}(f(x), f(x^+)) \gg \text{score}(f(x), f(x^-))
\]

InfoNCE loss: N-way classification among positive and negative samples

\[
L = -\mathbb{E}_x \left[ \log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x^-_j)))} \right]
\]

Commonly known as the InfoNCE loss (van den Oord et al., 2018)

A lower bound on the mutual information between \(f(x)\) and \(f(x^+)\)

\[
MI[f(x), f(x^+)] - \log(N) \geq -L
\]
Summary: Contrastive Representation Learning

**SimCLR**: a simple framework for contrastive representation learning
- **Key ideas**: non-linear projection head to allow flexible representation learning
- Simple to implement, effective in learning visual representation
- Requires large training batch size to be effective; large memory footprint
Summary: Contrastive Representation Learning

**MoCo** (v1, v2): contrastive learning using momentum sample encoder
- Decouples negative sample size from minibatch size; allows large batch training without TPU
- MoCo-v2 combines the key ideas from SimCLR, i.e., nonlinear projection head, strong data augmentation, with momentum contrastive learning
Summary: Contrastive Representation Learning

**CPC**: sequence-level contrastive learning
- Contrast “right” sequence with “wrong” sequence.
- InfoNCE loss with a time-dependent score function.
- Can be applied to a variety of learning problems, but not as effective in learning image representations compared to instance-level methods.
Other examples

Contrastive learning between image and natural language sentences

1. Contrastive pre-training

CLIP (Contrastive Language–Image Pre-training) Radford et al., 2021
Other examples

Contrastive learning on pixel-wise feature descriptors

(c) Background Randomization  (d) Cross Object Loss  (e) Direct Multi Object  (f) Synthetic Multi Object

Dense Object Net, Florence et al., 2018
Other examples

Dense Object Net, Florence et al., 2018
Our robot then grasps the best match for an instance-specific descriptor.
Next time: Vision and Language + RNNs
Today’s Agenda

Pretext tasks from image transformations
- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning
- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC

Frontier:
- Contrastive Language Image Pre-training (CLIP)
Frontier: Contrastive Language–Image Pre-training (CLIP)
Self-Supervised Learning

General idea: pretend there is a part of the data you don’t know and train the neural network to predict that.

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible.
- Pretend there is a part of the input you don’t know and predict that.

Source: Lecun 2019 Keynote at ISSCC
“The Cake of Learning”

How Much Information is the Machine Given during Learning?

- **“Pure” Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - A few bits for some samples

- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - 10→10,000 bits per sample

- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - Millions of bits per sample

Source: Lecun 2019 Keynote at ISSCC
Can we do better?

SimCLR

Momentum Contrast (MoCo)

Source: Chen et al., 2020b