EE 046746 - Technion - Computer Vision

Tal Daniel (https://taldatech.github.io) (adapted from Tal's Tutorials in ECE046211 - Deep Learning)

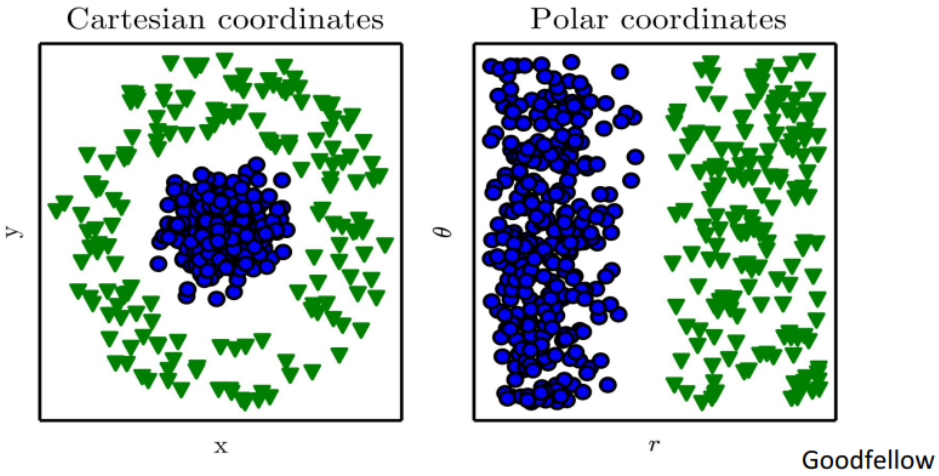
Tutorial 12 - Self-Supervised Learning

Agenda

Representation and Self-Supervised Learning

- Data is usually abundant and cheap, it's the labels that are expensive.
  - Can we use the unlabeled data to gain knowledge that is usable for downstream supervised tasks?

- Maybe we can learn rich and useful features from raw unlabeled data
  - We can learn a representation of the data!



- **The way we represent the data has a great impact on the performance and compelxity**.

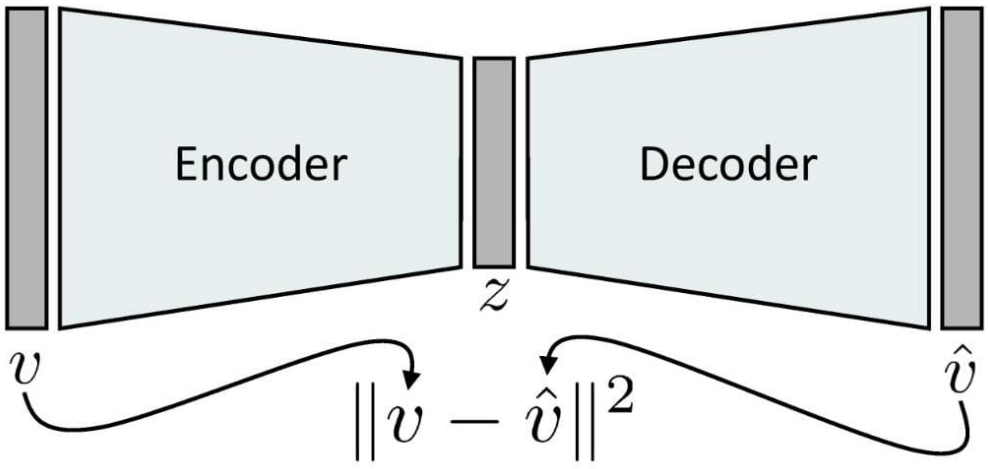
- What are the various general tasks that can be used to learn representations from unlabelled data?
  - **Deep Unsupervised Learning** - learn representations without lables, subset of deep learning, which is a subset of representation learning, which is a subset of machine learning.
  - **Self-supervised Learning** - often used interchangeably with unsupervised learning. Self-supervised: **create your own supervision through pretext tasks**.

Deep Unsupervised Learning - Deep Autoencoders

- **Motivation:** Most of the natural data is high-dimensional, such as images. Consider the MNIST (hand-written digits) dataset, where each image has $28x28 = 784$ pixels, which means it can be represented by a vector of length 784.
  - But do we really need 784 values to represent a digit? The answer is probably no. We believe that the data lies on a low-dimensional manifold which is enough to describe the observations. In the case of MNIST, we known that there are 10 digits - so we can represent the digits as one-hot vectors, which means we only need 10 dimensions.
  - So we can **encode** high-dimensional observations in a low-dimensional space.
  - But how can we learn meaningful low-dimensional representations?
  - The general idea is to reconstruct or, **decode** the low-dimensional representation to the high-dimensional representation, and use the reconstruction error to learn the best representations. This is the core idea behind **autoencoders**.

- Image from [Wikipedia (https://en.wikipedia.org/wiki/MNIST_database)]

- **Autoencoders** - models which take data as input and discover some latent state representation of that data. The input data is converted into an encoding vector where each dimension represents some learned attribute about the data. The most important detail to grasp here is that our encoder network is outputting a single value for each encoding dimension. The decoder network then subsequently takes these values and attempts to recreate the original input. Autoencoders have **three parts**: an encoder, a decoder, and a 'loss' function that maps one to the other. For the simplest autoencoders - the sort that compress and then reconstruct the original inputs from the compressed representation - we can think of the 'loss' as describing the amount of information lost in the process of reconstruction.



Self-Supervised Learning

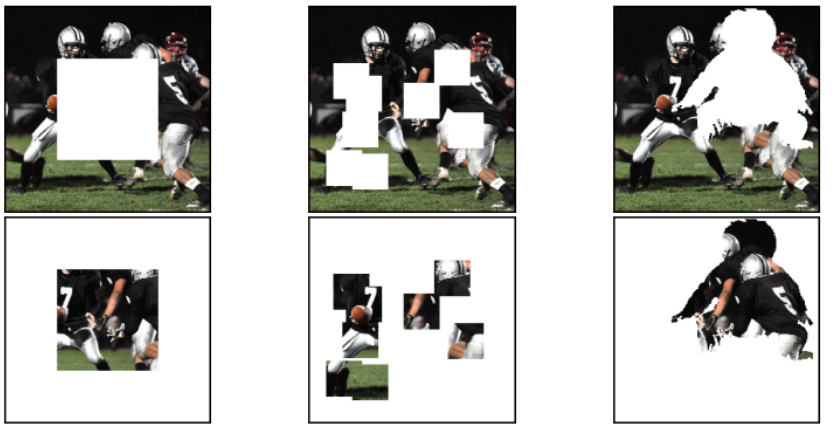
- A version of unsupervised learning where **data provides the supervision**.
- **Idea**: withhold some part of the data and then task a neural network to predict it from the remaining parts.
- Details decide what proxy loss or pretext task the network tries to solve, and depending on the quality of the task, good semantic features can be obtained without actual labels.
- Advantages over supervised learning:
  - Large cost of producing a new dataset for each task (prepare labeling manuals, categories, hiring humans, creating GUIs, storage pipelines, etc).
  - Good supervision may not be cheap (e.g., medicine, legal).
  - Take advantage of vast amount of unlabeled data on the Internet (images, videos, language).

Self-Supervised Learning Methods

- *Reconstruct from a corrupted (or partial) version*
  - Denoising Autoencoders - "Withholded" data: Clean image
  - In-painting -
  - Colorization, Split-Brain Autoencoder
- *Visual common sense tasks*
  - Relative patch prediction
  - Jigsaw puzzles
  - Rotation prediction
- **Contrastive Learning** (our focus)
  - word2vec
  - Contrastive Predictive Coding (CPC)
  - Instance Discrimination
  - Simple Framework for Contrastive Learning of Visual Representations (SimCLR), Momentum Contrast (MoCo), Bootstrap Your Own Latent (BYOL)

Corrupted Version Reconstruction & Visual Common Sense Tasks

Code Demos - [Self-Supervised Learning Demos (https://colab.research.google.com/github/rll/deepul/blob/master/demos/lecture7_selfsupervised_demos.ipynb)](https://colab.research.google.com/github/rll/deepul/blob/master/demos/lecture7_selfsupervised_demos.ipynb)

- **Context Encoder** - Try to predict a hidden mask in the image
  - 

(a) Central region (b) Random block (c) Random region

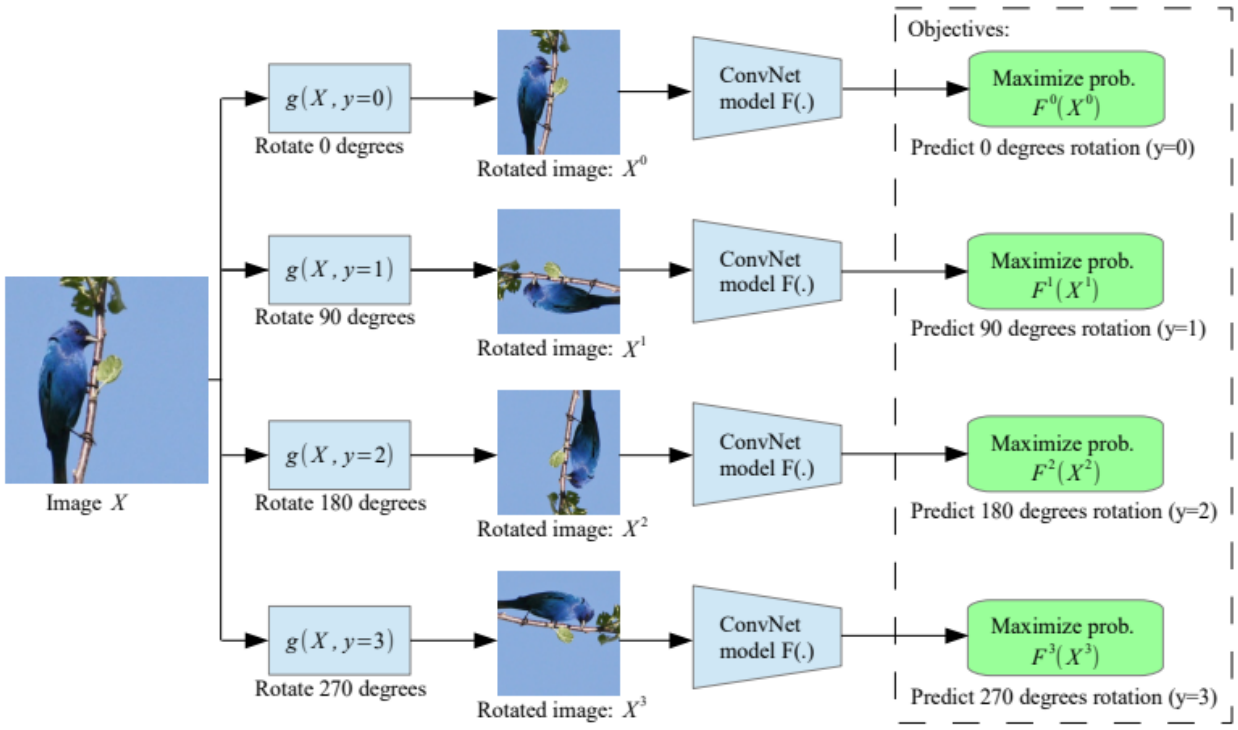
- The reconstruction is mediocre

- We care about the learned representation!
  - Specifically - is it useful for a downstream task?

Top 1 Accuracy on CIFAR-10	Top 5 Accuracy CIFAR-10
45.77	90.29

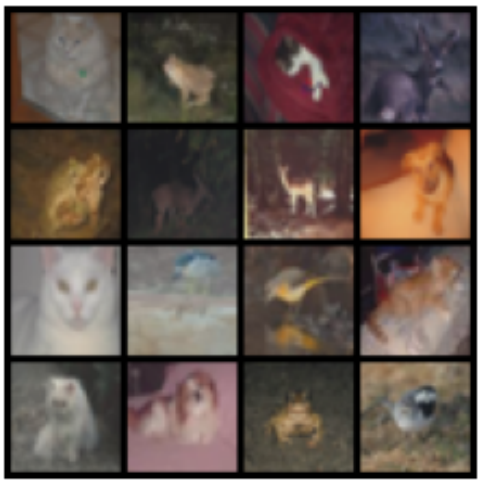
- Did it learn anything?
  - We can look the the nearest neighbors in the latent space to see if that's the case.

- **Rotation Prediction** - Try to predict the rotation "class" of a given image.



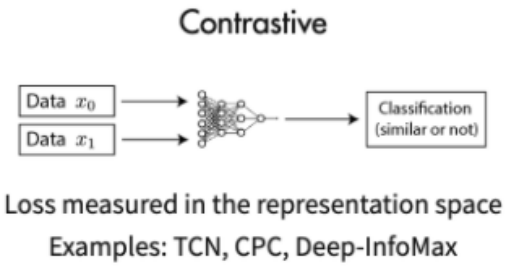
- What about the learned representations?

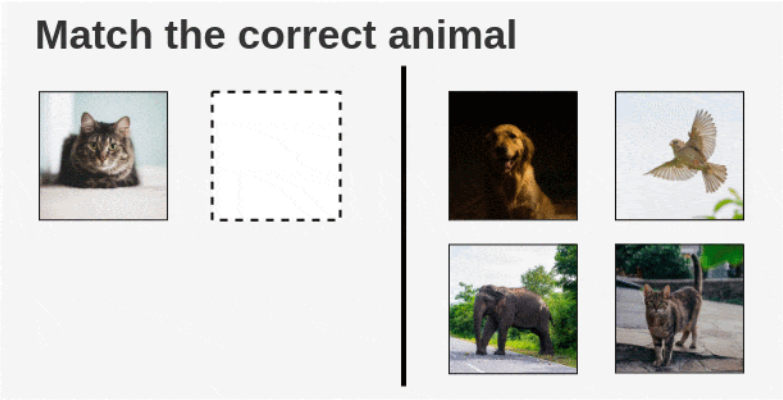
	Top 1 Accuracy on CIFAR-10	Top 5 Accuracy on CIFAR-10
Context Encoder	45.77	90.29
Rotation Prediction	79.91	99.12



Contrastive Learning

- Contrastive learning is an approach to formulate the task of **finding similar and dissimilar things for a ML model (basically what classification does when given labels)**.
- Contrastive methods, as the name implies, learn representations by contrasting **positive and negative** examples.
- Using this approach, one can train a machine learning model to classify between similar and dissimilar images.





- [Image Source (https://analyticsindiamag.com/contrastive-learning-self-supervised-ml)](https://analyticsindiamag.com/contrastive-learning-self-supervised-ml)

- More formally, for any data point x, contrastive methods aim to learn an encoder f such that:
  - x^+ is a data point similar to x, referred to as a *positive* sample.
  - x^- is a data point dissimilar to x, referred to as a *negative* sample.
  - The **score function** is a metric that measures the similarity between two features:

$$score(f(x), f(x^+)) >> score(f(x), f(x^-))$$

- How can we sample "similar" or "different" images?

- The most common loss function to implement the score paradigm is **InfoNCE** loss, which looks similar to softmax.

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

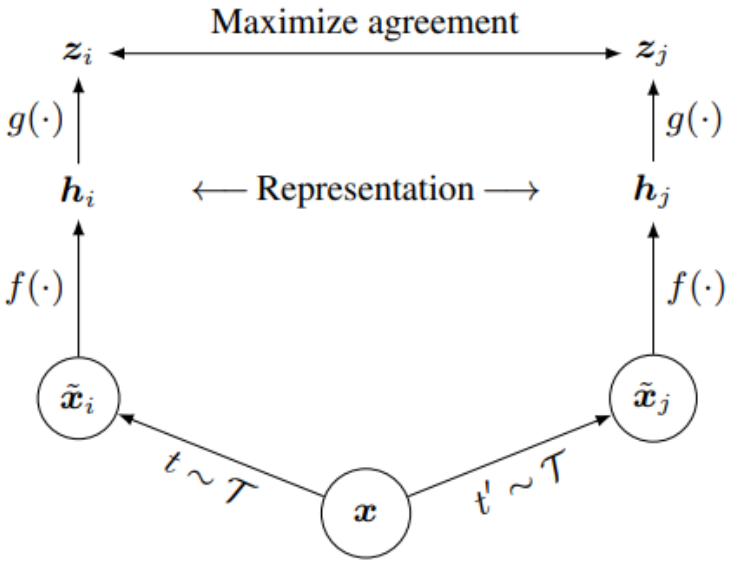
- The denominator terms consist of one positive sample, and N−1 negative samples.
- Compare this to softmax:

$$Softmax(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{M} e^{y_j}}, \quad i \in [1, \ldots, M], y \in \mathbb{R}^M$$

  - Where y_i is the angle between the normalized representation of the images $u^T w = ||u||||w|| \cos \angle(u, w)$

Simple Framework for Contrastive Learning of Visual Representations (SimCLR)

- [**Simple Framework for Contrastive Learning of Visual Representations (SimCLR)**](https://arxiv.org/abs/2002.05709) is a framework for contrastive learning of *visual* representations.
- It learns representations by maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space.



- A **stochastic data augmentation module** that transforms any given data example randomly resulting in two correlated views of the same example, denoted $\tilde{x}_i$ and $\tilde{x}_j$, which is considered a **positive pair**.
- SimCLR sequentially applies three simple augmentations: random cropping followed by resize back to the original size, random color distortions, and random Gaussian blur. The authors find **random crop and color distortion** is crucial to achieve good performance.
- A neural network base encoder $f(\cdot)$ that extracts **representation vectors** from augmented data examples. The framework allows various choices of the network architecture without any constraints.
  - For simplicity ResNet is used to obtain $h_i = f(\tilde{x}_i) \in \mathcal{R}^d$ where h_i is the output after the average pooling layer.
- A small neural network projection head $g(\cdot)$ that maps representations to the space where contrastive loss is applied.
- MLP with one hidden layer is used to obtain $z_i = g(h_i)$.
- **The authors find it beneficial to define the contrastive loss on z_i's rather than h_i's**.

- A minibatch of N examples is randomly sampled and the contrastive prediction task is defined on pairs of augmented examples derived from the minibatch, resulting in $2N$ data points.
- Negative examples are not sampled explicitly. Instead, given a positive pair, the other $2(N-1)$ augmented examples within a minibatch are treated as negative examples.
- A NT-Xent (the normalized temperature-scaled cross entropy loss) loss function is used:

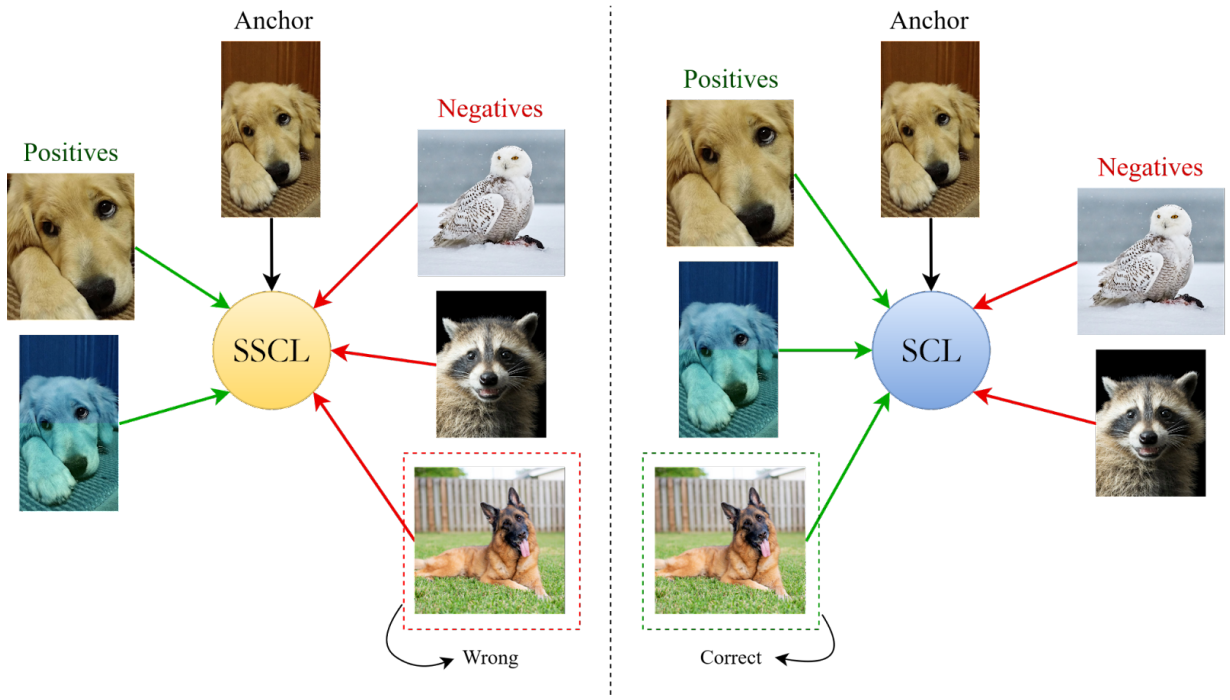
$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(\text{sim}(z_i, z_k)/\tau)}$$

  where $\text{sim}(z_i, z_j) = \frac{z_i^T z_j}{\|z_i\| \|z_j\|}$
- PyTorch Code (https://github.com/sthalles/SimCLR)

- Image Source (https://ai.googleblog.com/2020/04/advancing-self-supervised-and-semi.html)

- Why can't we just use the positive samples?

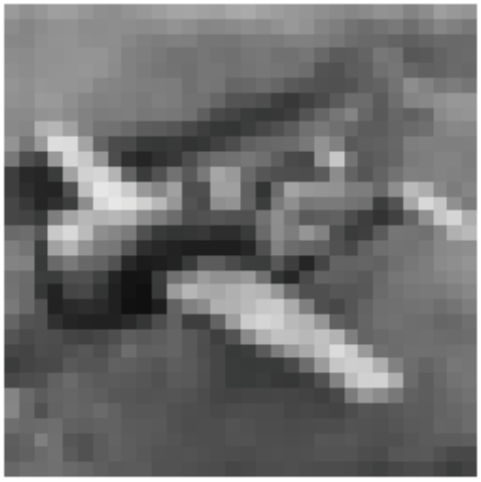
- How likely is our negative sample to actually be negative?



- Image Source (https://www.v7labs.com/blog/contrastive-learning-guide)

- What about the representations?

	Top 1 Accuracy on CIFAR-10	Top 5 Accuracy on CIFAR-10	Top 1 Accuracy on Imagenet	Top 5 Accuracy on Imagenet
Context Encoder	45.77	90.29		
Rotation Prediction	79.91	99.12		
SimCLR	92.84	99.86	69.3	89.0



Using the Learned Representation for Downstream Tasks

- Is classification the only downstream task our learned representation can help?

- Example: **Segmentation** on PASCAL VOC2012

- Images from [Berkeley's Deep Unsupervised Learning Course (https://colab.research.google.com/github/rll/deepul/blob/master/demos/lecture7_selfsupervised_demos.ipynb](https://colab.research.google.com/github/rll/deepul/blob/master/demos/lecture7_selfsupervised_demos.ipynb)

Momentum Contrast (MoCo)

- **Momentum Contrast (MoCo)** (https://arxiv.org/abs/1911.05722) is a self-supervised learning algorithm with a contrastive loss.
- Contrastive loss methods can be thought of as **building dynamic dictionaries**.
- The **"keys" (tokens)** in the dictionary are sampled from data (e.g., images or patches) and are represented by an encoder network.
- Unsupervised learning trains encoders (by minimizing a contrastive loss) to perform dictionary look-up: an encoded “query” should be similar to its matching key and dissimilar to others.
- In MoCo, we maintain the dictionary as a queue of data samples: the encoded representations of the current mini-batch are enqueued, and the oldest are dequeued.
- The queue decouples the dictionary size from the mini-batch size, allowing it to be large.
- Moreover, as the dictionary keys come from the preceding several mini-batches, a slowly progressing key encoder, implemented as a momentum-based moving average of the query encoder, is proposed to maintain consistency.
- PyTorch Code (https://github.com/facebookresearch/moco)
  - Colab Demo (https://colab.research.google.com/github/facebookresearch/moco/blob/colab-notebook/colab/moco_cifar10_demo.ipynb)

- The positive samples part remains the same as in SimCLR.
- For the negative samples - We build a **momentum encoder**
  - This encoder's architecture is the same as the normal encoder, but its weights are an **exponential moving average**:

$$W^{k}_{\text{momentum encoder}} = \beta \cdot W^{k-1}_{\text{momentum encoder}} + (1-\beta) \cdot W_{\text{encoder}}$$

  - The momentum encoder doesn't have backpropegation - so we can save memory and apply it on a large amount of negative samples

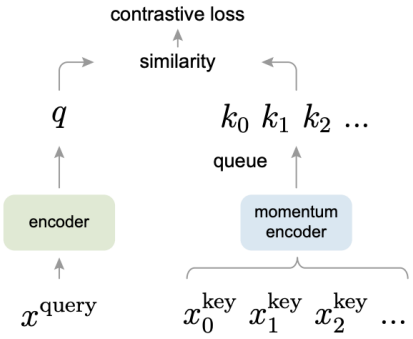


Figure 1. Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys $\{k_0, k_1, k_2, ...\}$ are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from the mini-batch size. The keys are encoded by a slowly progressing encoder, driven by a momentum update with the query encoder. This method enables a large and consistent dictionary for learning visual representations.

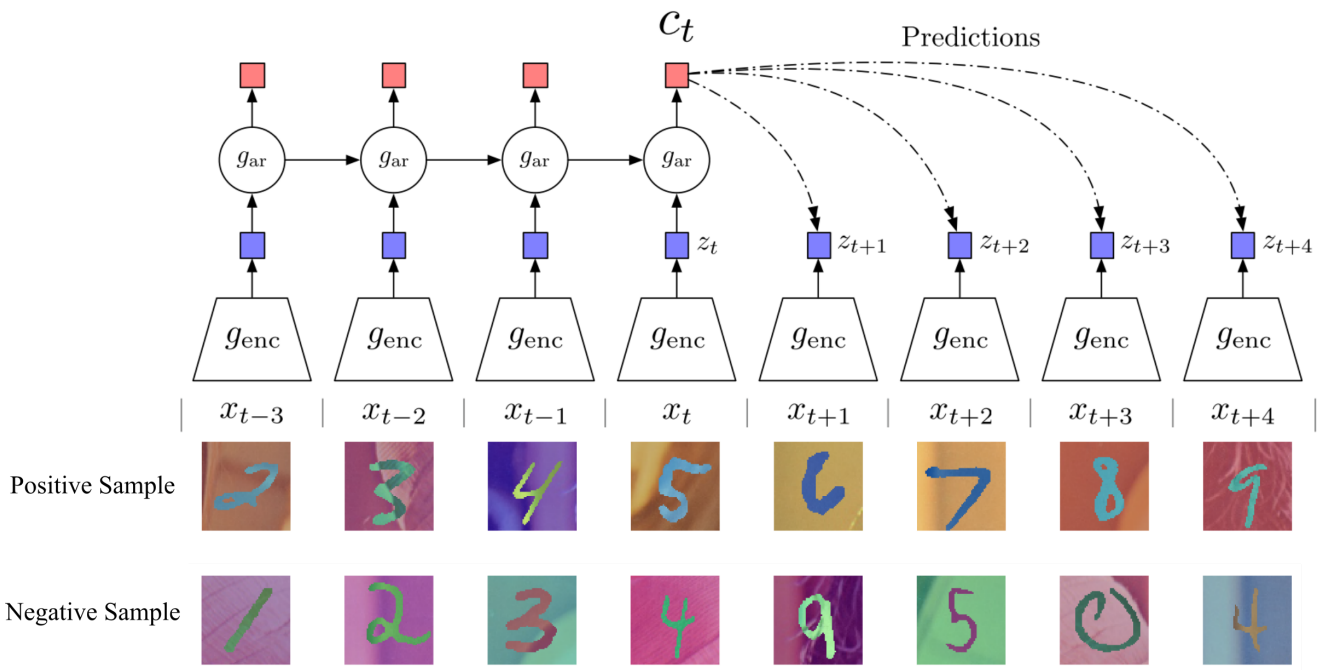
Short training	Top 1 Accuracy on Imagenet	Batch size
SimCLR	61.9	256
SimCLR	66.6	8192
MoCo	**60.6**	**256**
MoCo v2	67.5	256

Long training	Top 1 Accuracy on Imagenet	Batch size
SimCLR	69.3	4096
MoCo v2	71.1	256

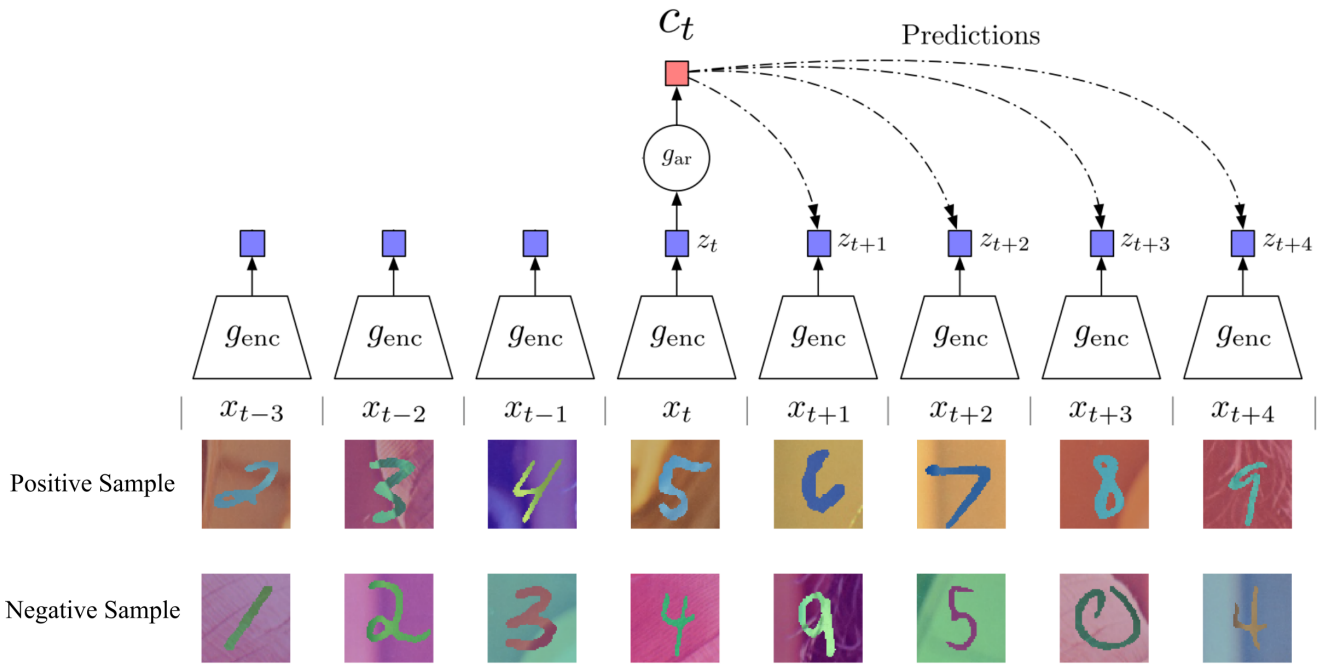
- Both MoCo and SimCLR can be categorized (along with other newer methods such as SwAV, BYOL, etc.) as Instance Based (Discrimination) Contrastive Learning.

Contrastive Predictive Coding (CPC)

- **Contrastive Predictive Coding (CPC)** (https://arxiv.org/abs/1807.03748) learns self-supervised **representations** (Coding) by **predicting the future** (Predictive) in a learned *latent space* by using powerful autoregressive models that **Contrast** (Contrastive) "right" and "wrong" sequences.
- The model uses a probabilistic contrastive loss which induces the latent space to capture information that is maximally useful to predict future samples.



- [Image Source (https://github.com/davidtellez/contrastive-predictive-coding)](https://github.com/davidtellez/contrastive-predictive-coding)


c_t
Predictions
g_{ar}
z_t
z_{t+1}
z_{t+2}
z_{t+3}
z_{t+4}
g_{enc}
x_{t-3}
x_{t-2}
x_{t-1}
x_t
x_{t+1}
x_{t+2}
x_{t+3}
x_{t+4}
Positive Sample
Negative Sample

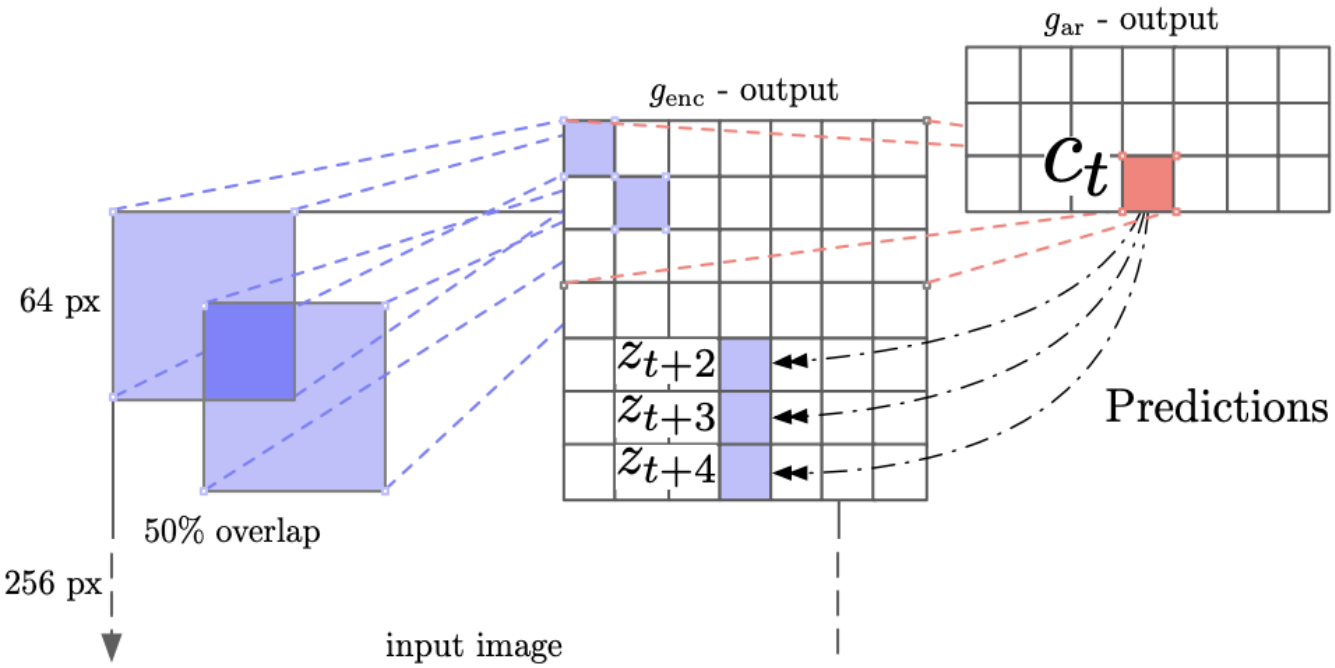
1. A non-linear encoder g_{enc} maps the input sequence of observations x_t to a sequence of latent representations $z_t = g_{enc}(x_t)$, potentially with a lower resolution.
2. An autoregressive model g_{ar} summarizes all $z \leq t$ in the latent space and produces a context latent representation $c_t = g_{ar}(z \leq t)$.
    - In the original paper they used a single sample window $k = 1$ (Predict the future based on the last sample)
3. Compute the InfoNCE loss between the future code z_{t+k} and the predicted future code $\hat{z}_{t+k}$ based on the context c_t
    - Remember in InfoNCE we have $f(x)^T f(x_j)$, so here $f_k(x_{t+k}, c_t) = \exp(z_{t+k}^T W_k c_t)$
        - f is modeled to preserves the mutual information between x_{t+k} and c_t ($f_k(x_{t+k}, c_t) \propto \frac{\mathbb{P}(x_{t+k}|c_t)}{\mathbb{P}(x_{t+k}}$))
        - W_k are learned weights, f can be unnormalized (does not have to integrate to 1)
    - Negative samples are an incorrect "future prediction"

- Any type of encoder and autoregressive can be used.
    - For example: strided convolutional layers with RNN and GRUs.
- [PyTorch Code (https://github.com/jefflai108/Contrastive-Predictive-Coding-PyTorch)](https://github.com/jefflai108/Contrastive-Predictive-Coding-PyTorch)

- How can we "predict the future" in images?

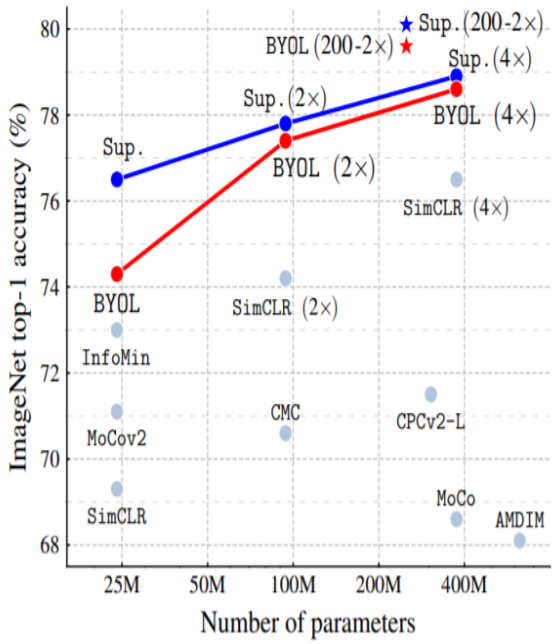
- [Image Source (https://towardsdatascience.com/a-framework-for-contrastive-self-supervised-learning-and-designing-a-new-approach-3caab5d29619)](https://towardsdatascience.com/a-framework-for-contrastive-self-supervised-learning-and-designing-a-new-approach-3caab5d29619)

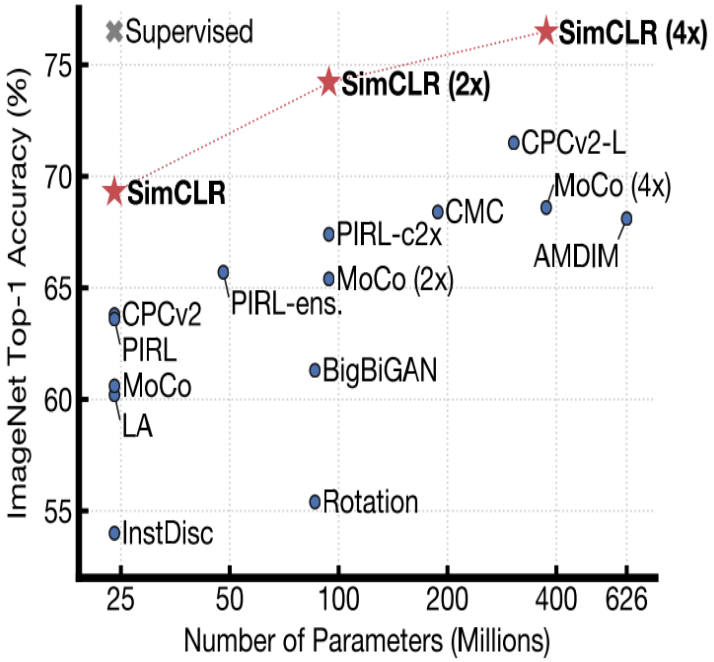
1. Create a sequence from the image: Split the image into overlapping patches, and model rows of patches from top to bottom as a sequence.
2. Use an encoder g_{enc} that's fit for images (e.g. Resnet-50 architecture)
3. Use an autoregressive model fit for images (e.g. PixelCNN) to create a context vector c_t from the first k rows.
4. Compute the InfoNCE loss between the context c_t and the predicted rows z_{t+k}
    - Negative samples will be incorrect rows



Performance Comparison

Performance on ImageNet (linear evaluation) using ResNet-50 and ResNet200 (2×), compared to other unsupervised and supervised (Sup.) baselines:





Recommended Videos

Warning!

- These videos do not replace the lectures and tutorials.
- Please use these to get a better understanding of the material, and not as an alternative to the written material.

Video By Subject

- General Self-Supervised Learning - Lecture 7 Self-Supervised Learning - UC Berkeley Spring 2020 - CS294-158 Deep Unsupervised Learning (https://www.youtube.com/watch?v=dMUes74-nYY)
- SimCLR - SimCLR Explained! (https://www.youtube.com/watch?v=APki8LmdJwY)
- MoCo - Momentum Contrastive Learning (https://www.youtube.com/watch?v=LvHwBQF14zs)

Credits

- EE 046211 Winter 22 - Original Tutorial - Tal Daniel (https://taldatech.github.io/)
- EE 046746 Spring 22 - Hila Manor (https://github.com/HilaManor)
- Icons made by Becris (https://www.flaticon.com/authors/becris) from www.flaticon.com (https://www.flaticon.com/)
- Icons from Icons8.com (https://icons8.com/) - https://icons8.com (https://icons8.com)
- Berkeley's CS294-158-SP20-Deep Unsupervised Learning (https://sites.google.com/view/berkeley-cs294-158-sp20/home)
- Stanford's CS2331n (Spring 2021) -Convolutional Neural Networks for Visual Recognition (http://cs231n.stanford.edu/2021/)
- Contrastive Predictive Coding (https://paperswithcode.com/method/contrastive-predictive-coding)
- Simple Framework for Contrastive Learning of Visual Representations (SimCLR) (https://paperswithcode.com/method/simclr)
- Momentum Contrast (https://paperswithcode.com/method/moco)
- A Framework For Contrastive Self-Supervised Learning And Designing A New Approach (https://arxiv.org/pdf/2009.00104.pdf)