Unsupervised Learning: Representation Learning

Earl Wong

Previously: Supervised Learning

ImageNet Images

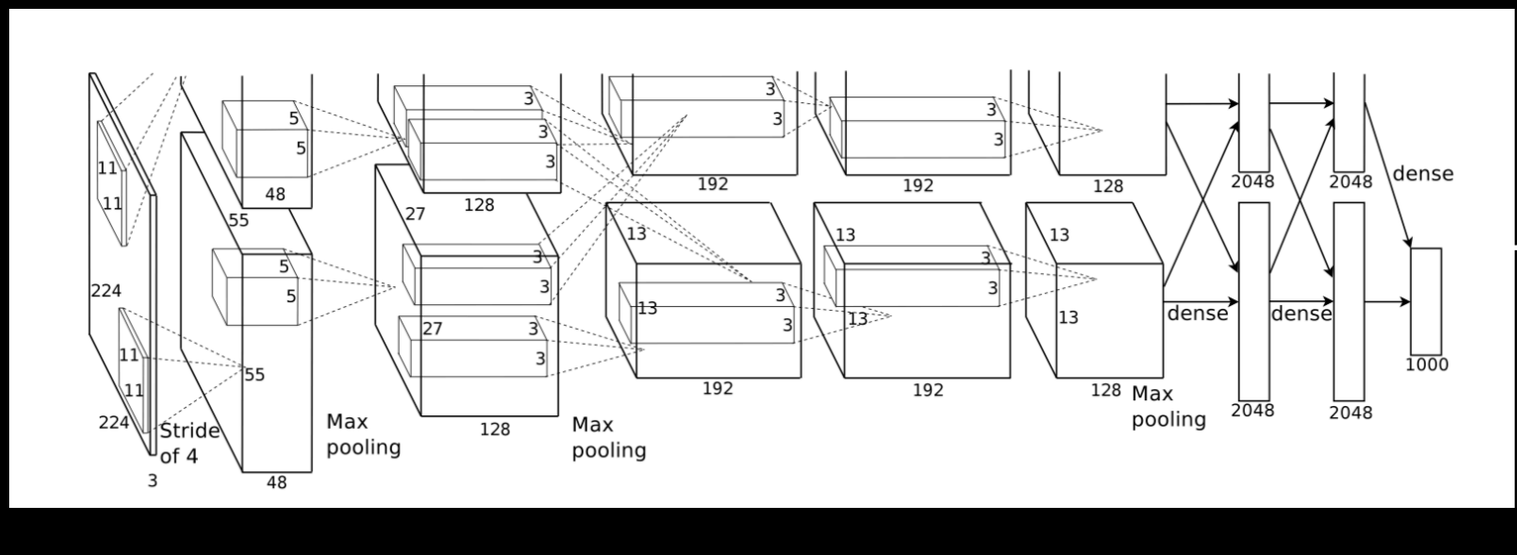


Image Class

Encoder network

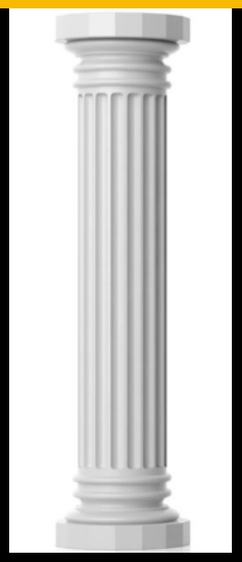
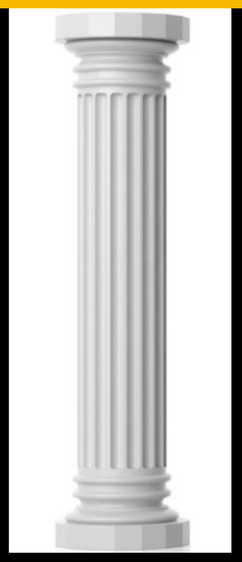
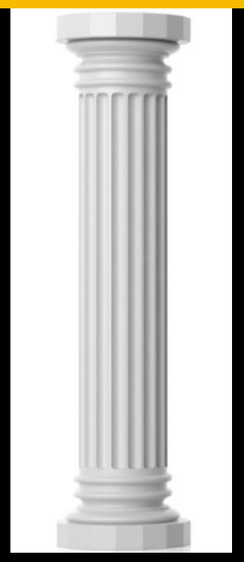
Although supervised learning was able to train deep networks containing rich feature representations, costly labeled training data was required.

This begged the following question: “Could we learn similar, rich feature representations using an identical deep network architecture, but without the costly labeled training data = an unlabeled dataset?”

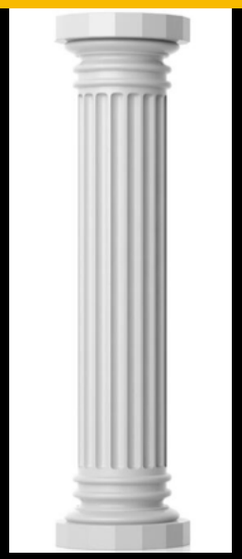
How would we accomplish this? How good would the learned representations be?

Unsupervised Learning: Representations

Unlabeled Dataset — **Algorithm** — **Loss Function**

Unlabeled Dataset



Unlabeled Dataset

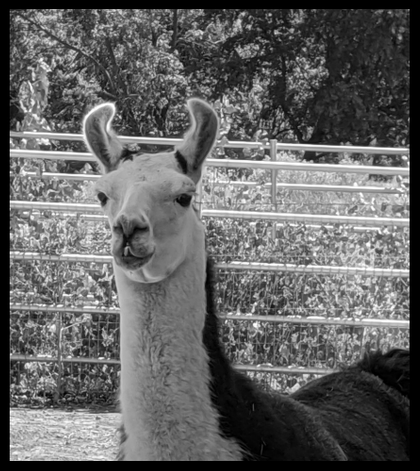
- Suppose we had access to a dataset, but we did not have access to the labels. What could we do?

- If we could construct test pairs between the data (similar versus different), we could then exploit this knowledge and use it to learn.

- This was the basis of learning by contrast = contrastive learning.

- Formally, contrastive learning can be viewed as training an encoder for a dictionary lookup task.

Unlabeled Dataset

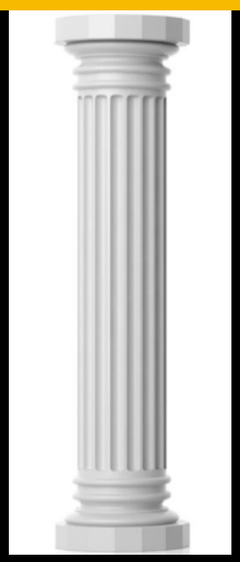


Similar

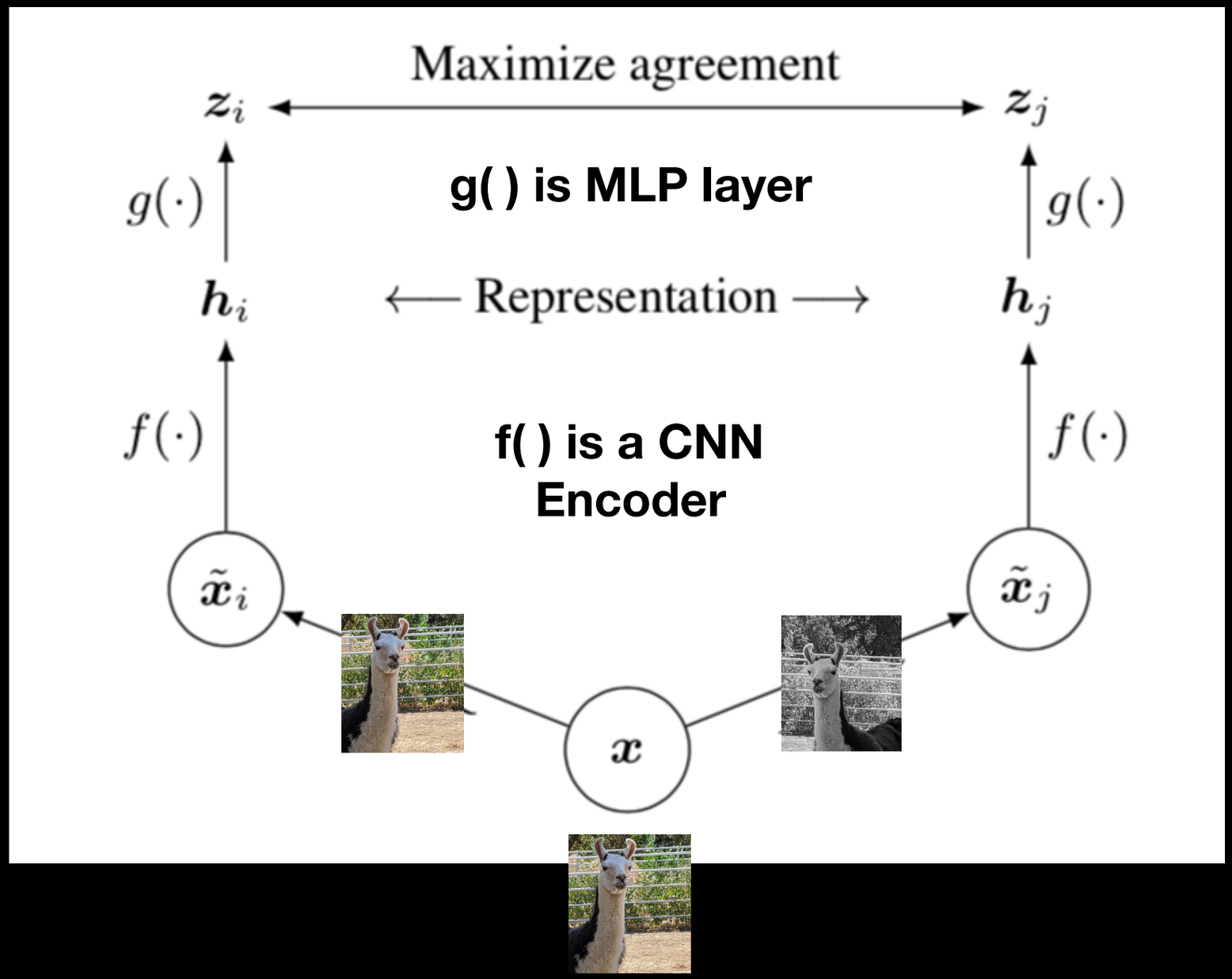
Dissimilar

In general, we can take any dataset and augment its images. The most effective augmentations (for producing similar image pairs) for contrastive learning included cropping and color alterations.

Algorithm



Algorithm simCLR



The simCLR algorithm represents the current state of the art for representation learning using unlabeled data.

The algorithm achieves learning by comparing the outputs in a siamese network, trained on image pairs.

The resulting loss is then used to update the network weights.

The Crux of Contrastive Learning

- Take a batch of N images from ImageNet, and generate two augmentations for every image.

- Run the augmented images through the network, generating two output vectors, z_{1i} and z_{2i}

- Compute the dot products between z_{1j} and z_{2k} for all possible combinations in the batch to determine similarity.

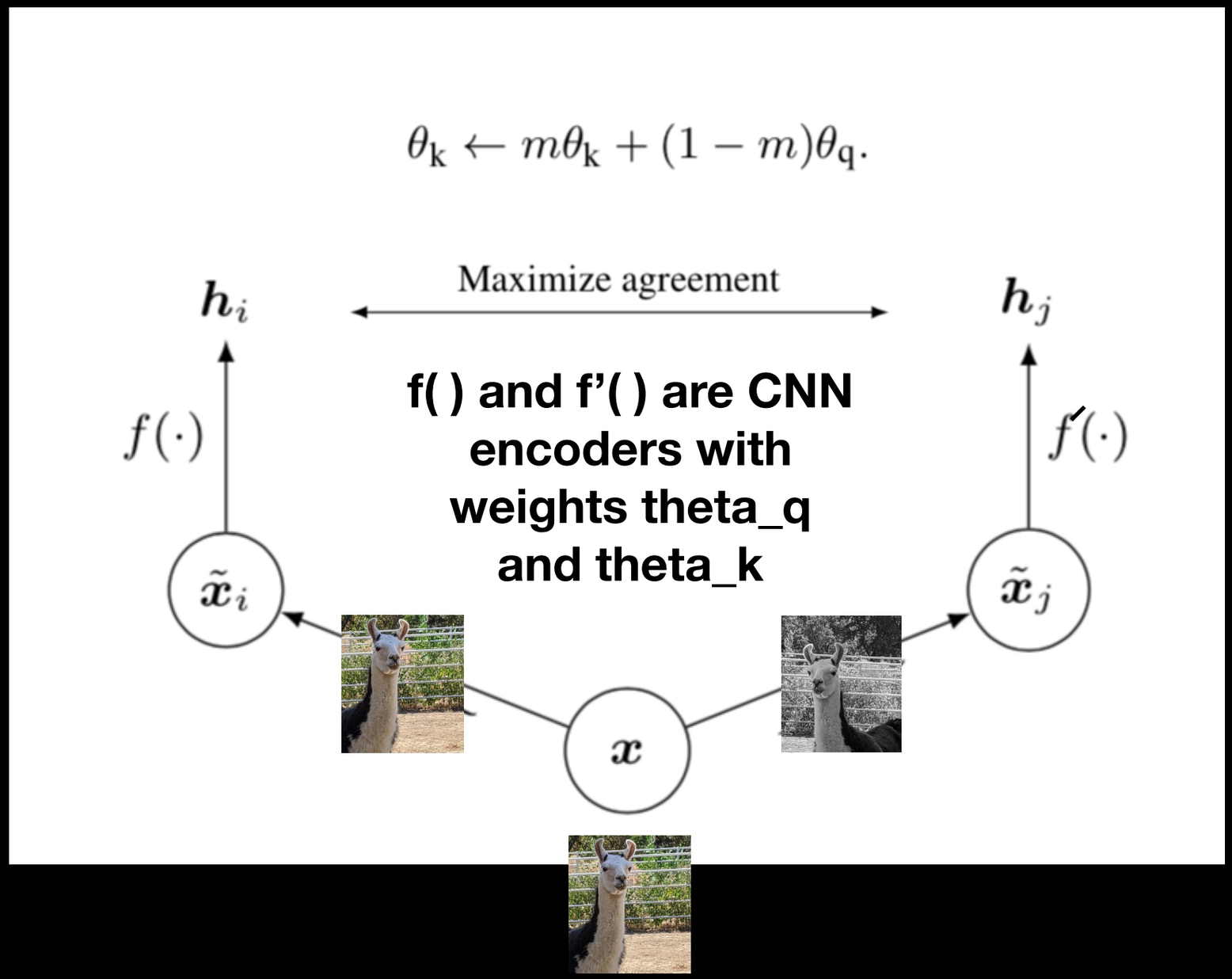
- Apply a contrastive loss function (similar to a softmax) to the resulting combinations, and back propagate the loss through the siamese network.

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 $g(\cdot)$

Algorithm
MoCo

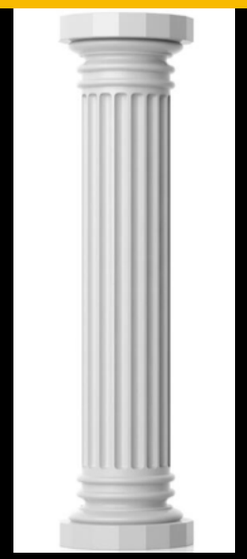


The MoCo (momentum contrast) algorithm is similar to simCLR.

However, it does not employ a MLP at the output.

In addition, there are two unique encoder networks that are updated at different rates.

Loss Function



Loss Function

Th contrastive loss function is defined as follows. (See “Contrastive Predictive Coding”)

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$

The loss function resembles a softmax function.

Tau is a tuned hyperparameter.

q denotes an output vector from the deep network associated with a reference query key (image).

The k’s represent output vectors associated with the dissimilar images k_i and the **s**imilar image consisting of a perturbed (via crop & color) reference query key (image).

Current Unsupervised Learning SOTA Comparisons

- Using equivalently sized networks, supervised learning results outperform self supervised / unsupervised learning learning results for image recognition.

- Although the gap has narrowed considerably, a true “apples to apples” comparison may not be appropriate, since SL is a form of specialized learning, while UL is generalized learning.

- By significantly widening and deepening the UL networks, comparable image recognition performance can be achieved.

- In addition, for downstream tasks (such as detection and segmentation), backbone architectures trained using UL and then fine tuned, frequently outperform backbones architectures trained using SL and then fine tuned.

Important History

- Many pre-text tasks (tasks to learn representations from unlabeled data) were investigated. They include:

- Denoising Autoencoder

- Context Encoder

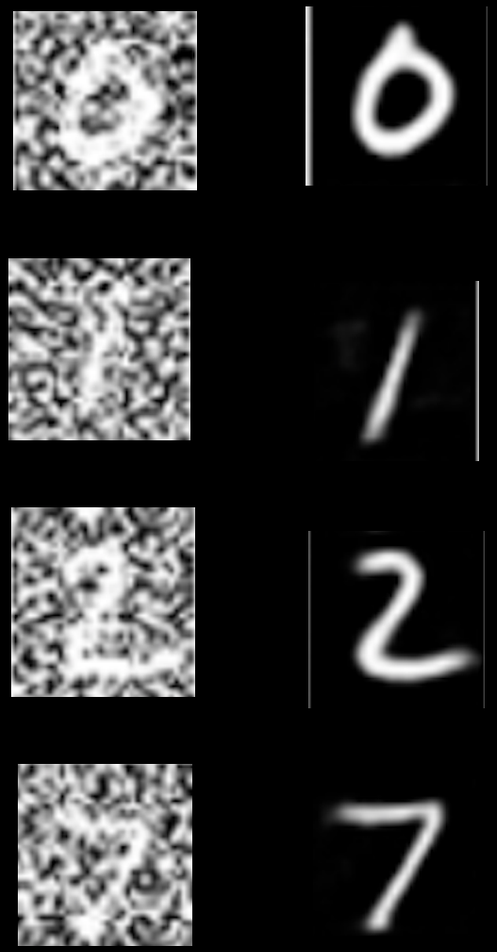
- Predicting positions of image patches, predicting image rotations, solving jigsaw puzzles …

- Contrastive Predictive Coding

- Their main components are now described and summarized.

Unlabeled Dataset

De-noising AutoEncoder



Input Images

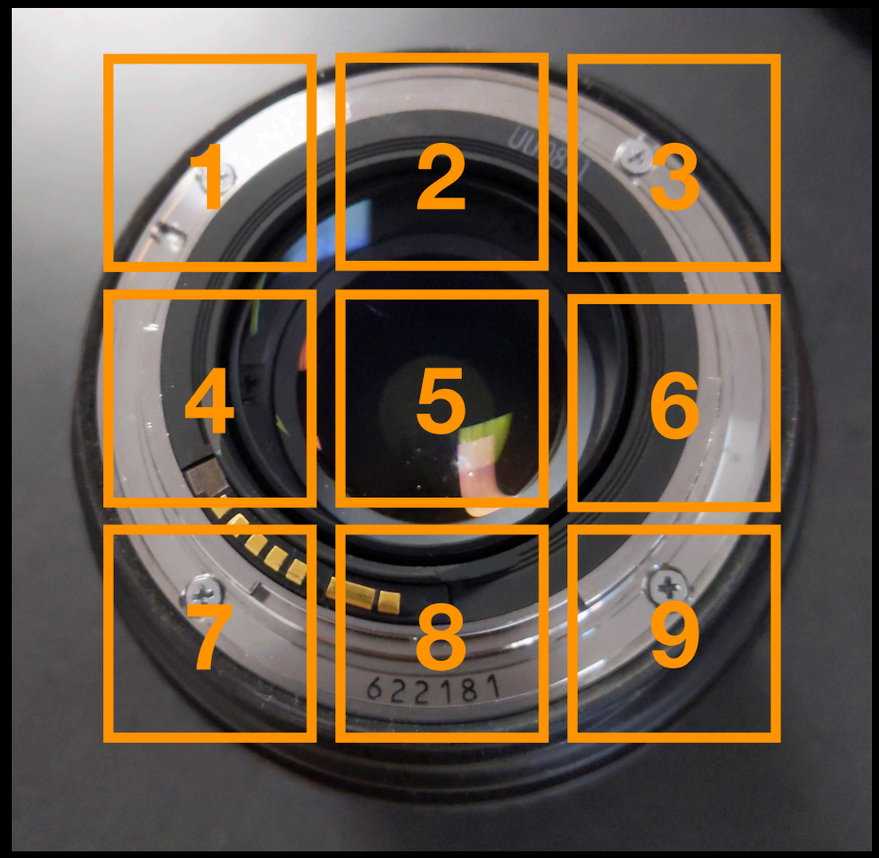
Desired Output Images

Context Encoder

Input Images

Desired Output Images

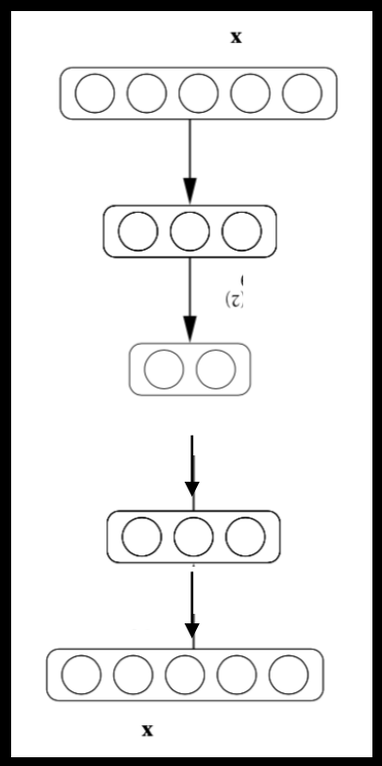
Predicting Image Patch Locations



Input: Any two unique images from blocks 1 to 9
Output: The position of the second selected block relative to the first

Algorithm (Architecture)

De-noising AutoEncoder



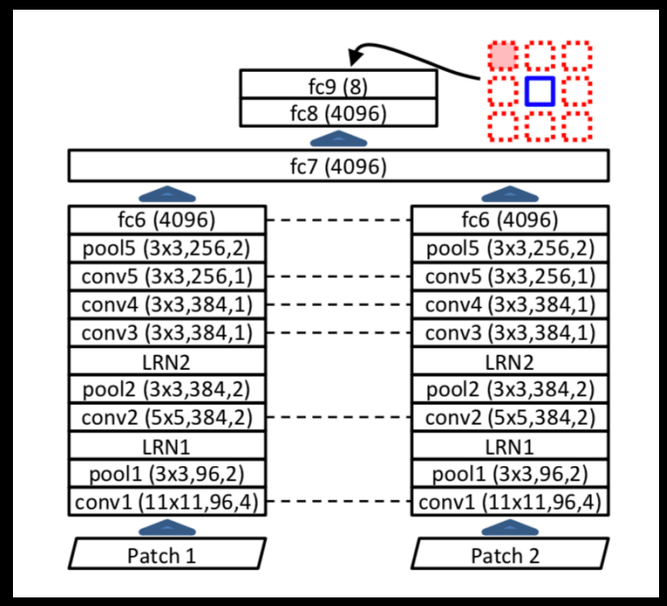
1) Stacked encoder - decoder network
2) A noisy input image is fed into the network. The goal is to produce a noise free output image.

Context Encoder

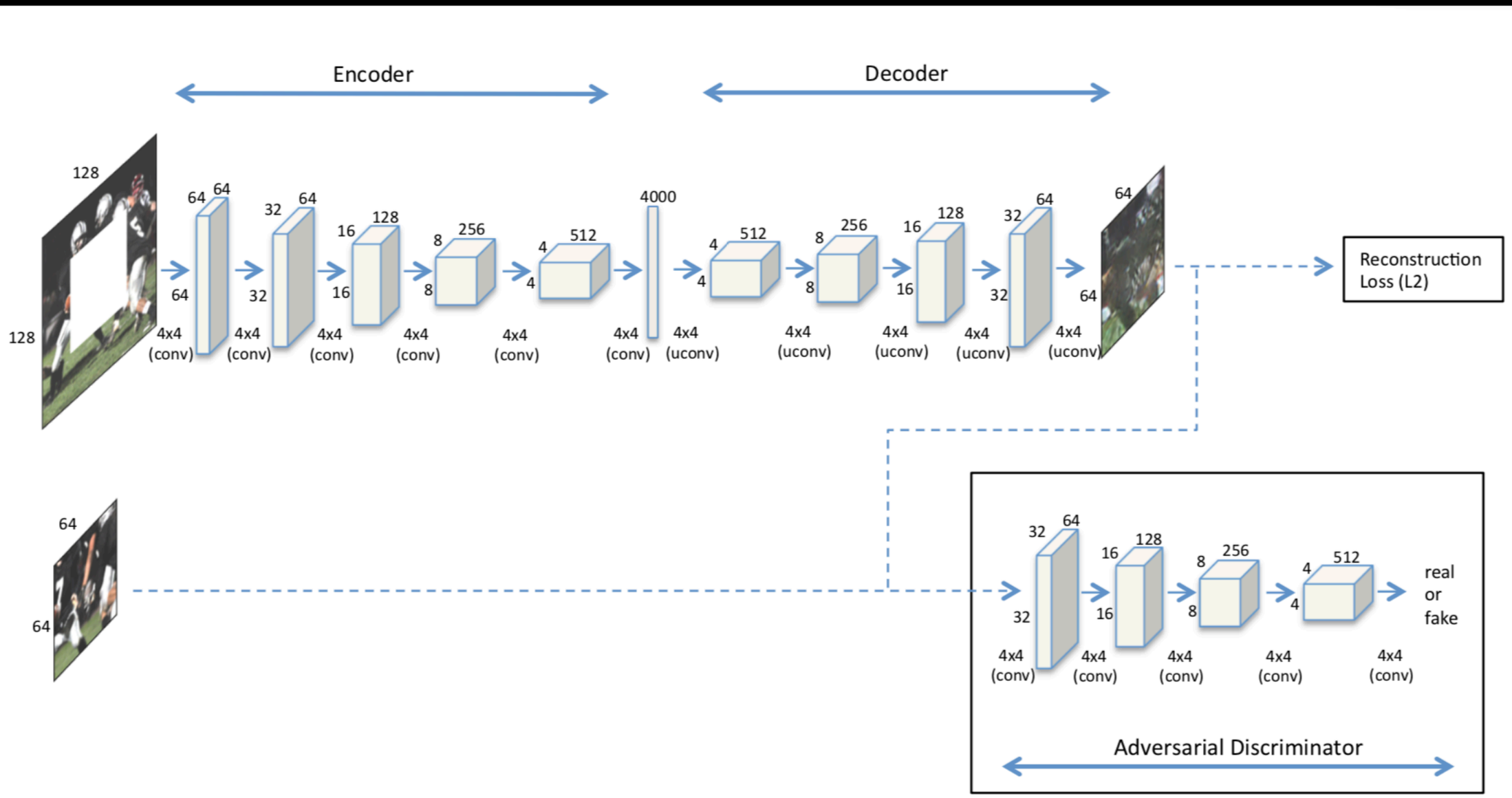
See Next Slide

The Encoder is AlexNet while the Decoder is the “Convolution Transpose”

Predicting Image Patch Locations



1) Two image blocks are fed into the siamese CNN.
2) The position of the second block is predicted, relative to the first block.


Encoder
Decoder
128
128
64
64
64
64
32
64
16
128
8
256
4
512
4000
4
512
8
256
16
128
32
64
64
64
4x4
(conv)
4x4
(uconv)
Reconstruction
Loss (L2)
64
64
32
64
16
128
8
256
4
512
real
or
fake
Adversarial Discriminator

Loss Function

De-noising AutoEncoder

Cross Entropy

MSE

Context Encoder

Total Loss

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}$$

Reconstruction Loss

$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2$$

Masked Region = 1

Elementwise Multiply

Adversarial Loss

$$\mathcal{L}_{adv} = \max_D \quad \mathbb{E}_{x \in \mathcal{X}}[\log(D(x)) + \log(1 - D(F((1 - \hat{M}) \odot x)))]$$

Predicting Image Patch Locations

Not discussed in the paper.

Random pairs of patches (from the same image) are sampled.

The algorithm must determine the position of one patch relative to the other.

The loss will be based on whether the resulting estimate is correct.

References

- A Simple Framework for Contrastive Learning of Visual Representations, Chen, et. al., arXiv February 2020
- Momentum Contrast for Unsupervised Visual Representation Learning, He, et. al. arXiv November 2019
- Representation Learning with Contrastive Predictive Coding, Oord, et. al., arXiv July 2018
- Stacked De-noising AutoEncoder, Vincent, et. al., JMLR 11 (2010)
- Contrast Encoders: Feature Learning by In-painting, Pathak, et. al., arXiv April 2016
- Unsupervised Visual Representation Learning by Context Prediction, Doersch, et. al. arXiv May 2015