# Knowledge Transfer and Adaptation

Self-Supervised Learning

Antonio Carta

antonio.carta@unipi.it

# Outline

*How do we pretrain a network?*

*Objective: forward transfer*

- what is self-supervised learning
- self-supervised methods in computer vision

(self-supervised)

$$\theta^{0} \xrightarrow{\text{pretraining}} \theta^{A} \xrightarrow{\text{finetuning}} \theta^{B}$$

# Motivations

- we want to learn discriminative features for a classification problem but we don't have labels
- we want features that are **transferable** to many downstream tasks

(self-supervised)

$$\theta^0 \xrightarrow{\text{pretraining}} \theta^A \xrightarrow{\text{finetuning}} \theta^B$$

# Definition

- **Self-supervised learning (SSL)** is predictive learning. Given a (large) unlabeled dataset, the system is trained on a task where the labels are self-generated.
- **pretext task**: the self-supervised task that defines a supervised loss given the unsupervised data.
- **evaluation**: we have a set of downstream tasks. We finetune the pretrained model on each task separately and evaluate the performance.
  - **Downstream task**: problems that we want to solve after pretraining
- **KEEP IN MIND**: we don’t care about the pretext task, it’s just a method to train the DNN representation.

# Evaluation

- **evaluation** on downstream tasks
- **finetuning** on all layers
- **linear probing**: finetune only the final classifier

# Example - BERT

- **Example**: BERT is a language model trained via self-supervised learning.
  - objective: masked language modeling (MLM)
    - input: $x$ is a sequence of tokens with some masked tokens
    - output: $y$ is the original sequence, where the masked tokens are predicted from their surrounding context
- **Why do we use the MLM objective?**
  - we have lots of data
  - MLM does not require expensive labels
  - MLM models transfer very well to downstream tasks in NLP

# Methods

- **augmentations**: exploit the properties of the domain (vision), such as invariances to transformations, to learn robust representations
- **contrastive learning**: learn representations by comparing and contrasting pairwise images.
- **encoding/autoregressive**: learn to reconstruct the input or predict a missing part of the input.
  - you have seen a lot of examples in ISPR and HLT, so we won’t cover them today
  - keep in mind that they are part of the SSL family

# Augmentations

# Invariance to augmentations

- **IDEA**: if we apply small augmentations (rotations, white noise, translations, …) to an image, its class doesn't change
- **pretext task**: given a batch of images resulting from different augmentations, predict whether they are the same image or not.
- The model must learn to
  - extract salient features of the images to recognize them
  - be invariant to augmentations

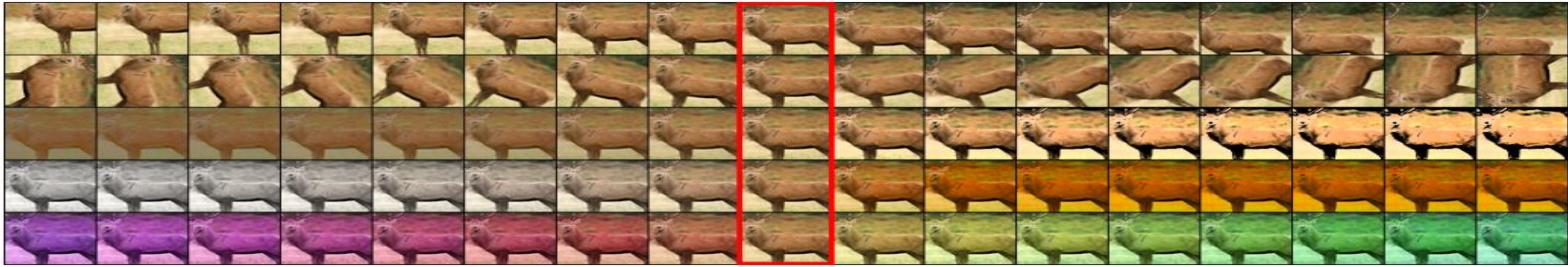


image: Dosovitskiy, Alexey, et al. "Discriminative unsupervised feature learning with exemplar convolutional neural networks (2015)." arXiv preprint arXiv:1406.6909.

# ExemplarCNN

- **surrogate class**: an instance (image) from the original dataset represent a surrogate class
- **instances of the class**: augmented version of the original instance. We create instances by repeatedly applying stochastic augmentations.
- **augmentations**: cropping, scaling, rotation, color, contrast
- **pretext task**: assign each augmented image to its corresponding surrogate class

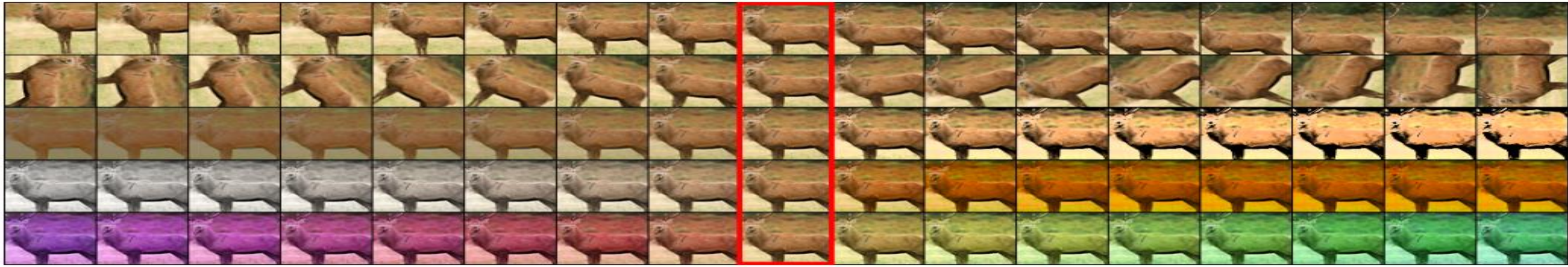


image: Dosovitskiy, Alexey, et al. "Discriminative unsupervised feature learning with exemplar convolutional neural networks (2015)." arXiv preprint arXiv:1406.6909.

# Learning from Image Patches

- **pretext task**: predict relative position of two random patches from the same image
- **challenge**: the task should be solvable but not too trivial
  - if patches are too far, their relative position may be unpredictable
  - if patches are neighbors, the model can track lines and edges to align the patches
- **solution**:
  - extract patches from a grid (not too difficult)
  - add noise and augmentations to break trivial solutions

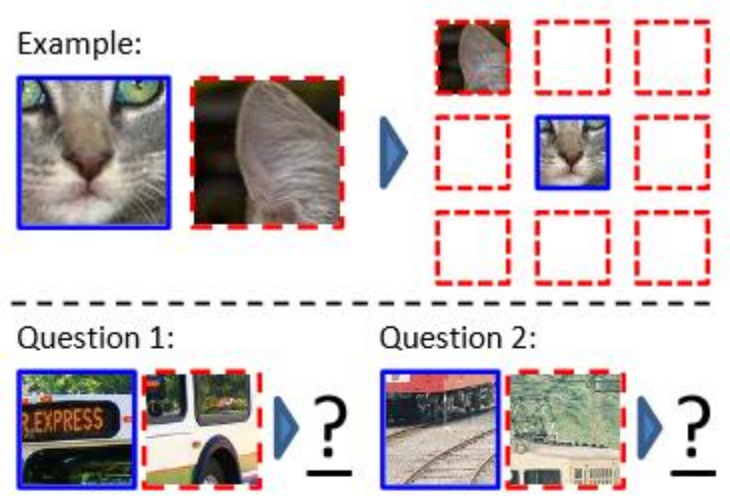



Figure 1. Our task for learning patch representations involves ra domly sampling a patch (blue) and then one of eight possib neighbors (red). Can you guess the spatial configuration for tl two pairs of patches? Note that the task is much easier once yc have recognized the object!

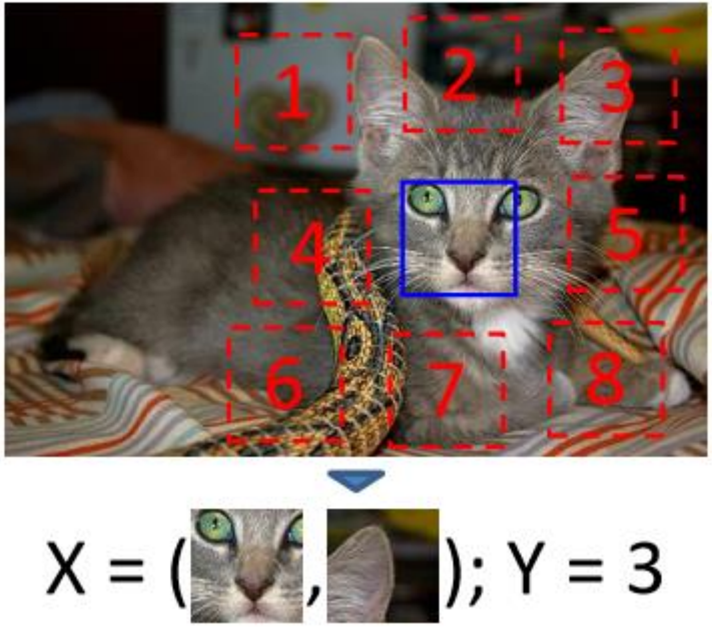
Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. "Unsupervised visual representation learning by context prediction." ICCV 2015.

# Chromatic Aberrations

- Defining pretext tasks is hard
- The model must not be able to solve the problem using trivial features
- Example: Chromatic Aberrations
  - Color distortion along boundaries due to poor lenses
  - Chromatic aberrations make the patch task trivial to solve

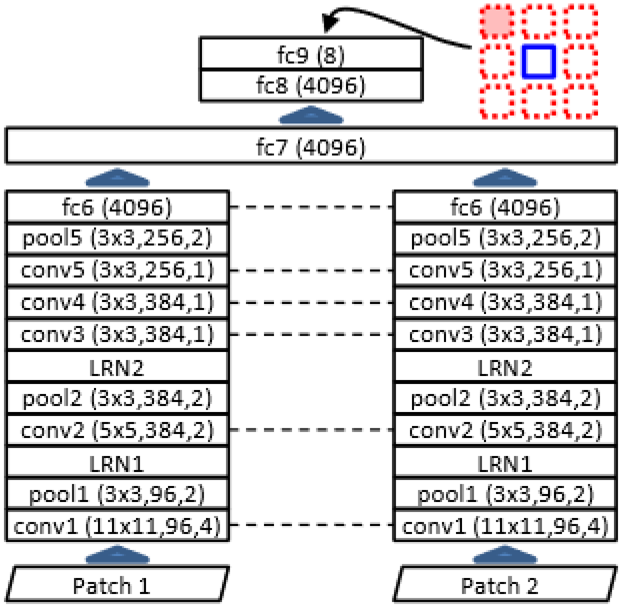
# Learning from Image Patches

- randomly sample the central patch (cat eyes in the picture)
- randomly pick one neighbor from the 3x3 grid centered around the first patch (8 choices)
- to make the task more difficult and encouraging learning nontrivial features
  - add small gaps and jitters (see the image, the grid is not perfectly aligned)
  - randomly upsample/downsample to introduce pixelation artifacts
  - apply color transformations




Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. "Unsupervised visual representation learning by context prediction." ICCV 2015.

# Model Architecture

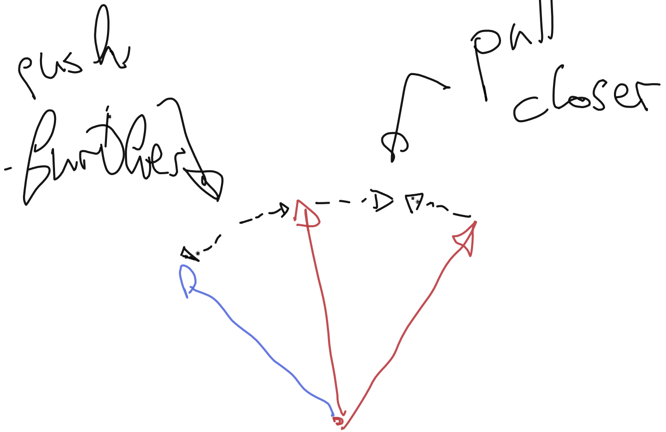



Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. "Unsupervised visual representation learning by context prediction." ICCV 2015.

# Contrastive Learning

# Contrastive Learning

- **Objective**: learn a latent representation space that is semantically meaningful (for downstream tasks)
- **Example**: in a downstream classification problem, we would like the examples to be linearly separable
- **IDEA**: similar images are close to each other, diverse images are far
- **contrastive learning**: compare and contrast pairs of images during training to learn good representations



# Contrastive vs Supervised Learning

- softmax + crossentropy also pushes+contrast
  - logits of the positive class are pushed up
  - logits of the negative classes are pushed down
  - this effect becomes problematic in imbalanced settings
- In contrastive learning we don’t need the labels

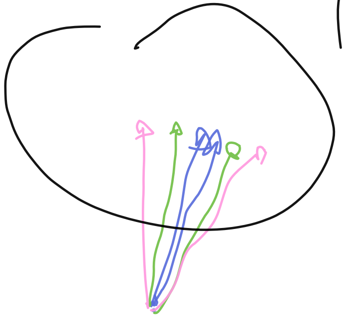
# Contrastive Learning

- we need to **compare and contrast**
- positive pairs → bring them closer
- negative pairs → push them further
- **Q1**: How do we sample positive/negative pairs?
- **Q2**: Which loss do we use?

# Representation Collapse

- Why do we need to push negative samples further?
- **representation collapse**: if we don't there is a trivial solution: collapse everything to the same representation

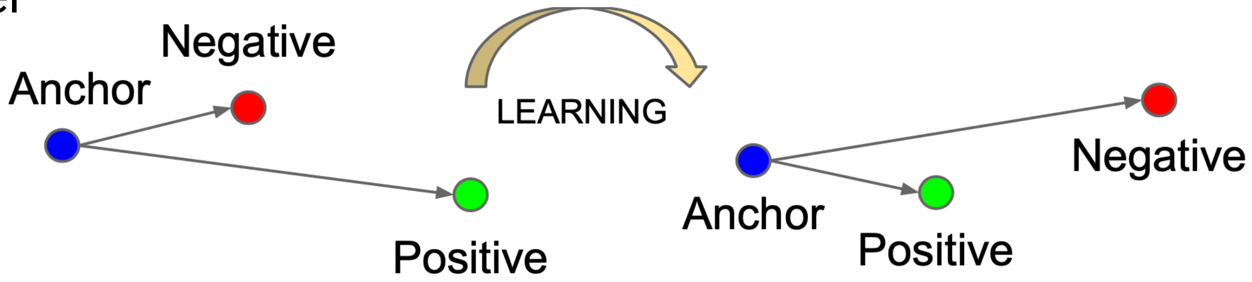
representation collapse



# Triplet Loss

*minimize the distance between the anchor and positive and maximize the distance between the anchor and negative*

- **anchor**: embedding representing the class/concept
- **positive**: embedding of an image with the same label as the anchor
- **negative**: embedding of an image from a different class
- **objective**: pull anchor and positive closer, push anchor and negative further



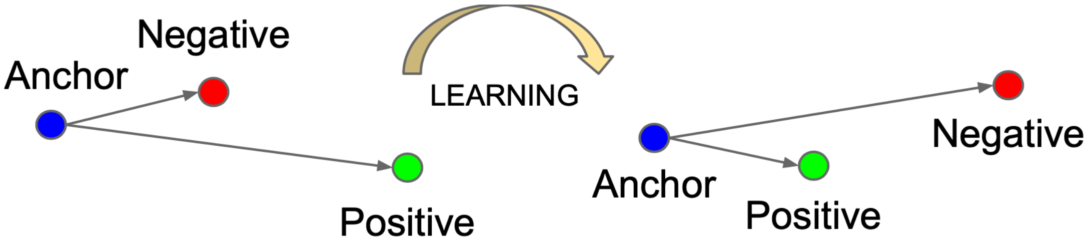

F. Schroff et al. "FaceNet: A Unified Embedding for Face Recognition and Clustering." CVPR 2015

# Triplet Loss

*minimize the distance between the anchor and positive and maximize the distance between the anchor and negative*

$$\mathcal{L}_{\text{triplet}}\left(\mathbf{x},\mathbf{x}^{+},\mathbf{x}^{-}\right)=\sum_{\mathbf{x}\in\mathcal{X}}\max\left(0,\left\|f(\mathbf{x})-f\left(\mathbf{x}^{+}\right)\right\|_2^2-\left\|f(\mathbf{x})-f\left(\mathbf{x}^{-}\right)\right\|_2^2+\epsilon\right)$$

- $f(x)$ embedding of image $x$
- $< x, x^{+}, x^{-} >$ anchor, positive, negative
- $\epsilon$ margin between positive and negative




F. Schroff et al. "FaceNet: A Unified Embedding for Face Recognition and Clustering." CVPR 2015

# Triplet Sampling

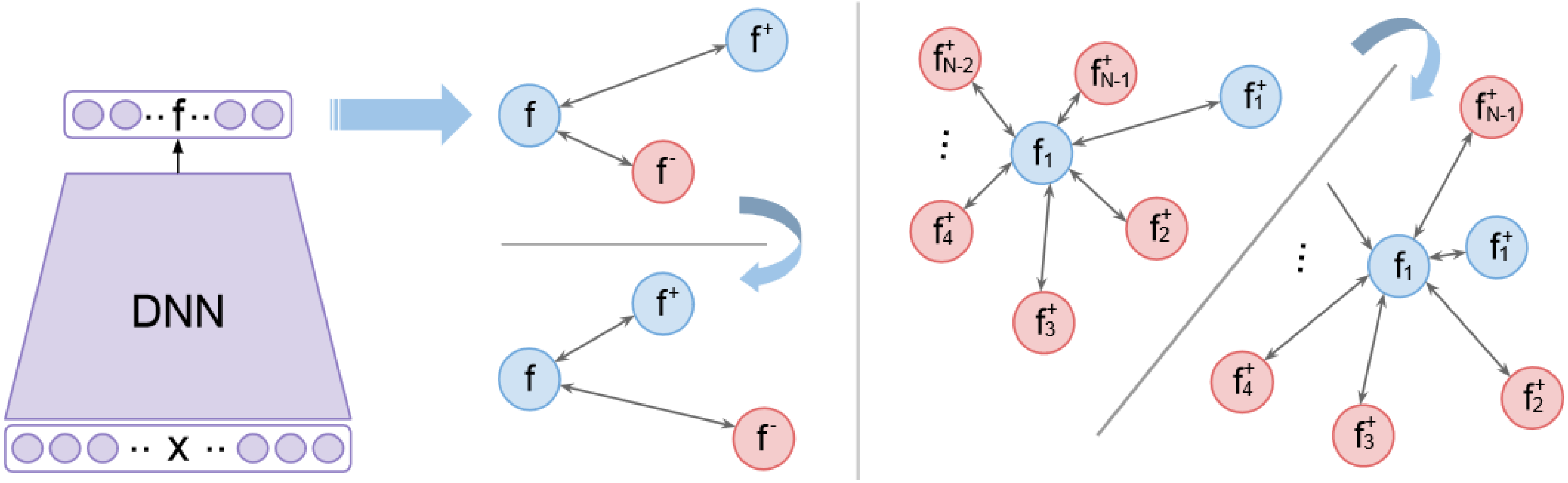
- problem: slow convergence due to instability
- triplet selection is crucial
- ideally, pick the most difficult examples
  - hard positive $\mathrm{argmax}_{x_i^p} \| f(x_i^a) - f\left(x_i^p\right) \|_2^2$ (further from anchor)
  - hard negative $\mathrm{argmin}_{x_i^n} \| f(x_i^a) - f(x_i^n) \|_2^2$ (closer to anchor)
  - expensive to compute
- in practice, use large mini-batches

# N-way Loss

- **PROBLEM**: triplet loss is slow to converge because it looks at a single <pos, neg> pair at each step
- **SOLUTION**: we can compare against more negative examples
- **N -pair loss** generalizes triplet loss by using N -1 negatives
- reduces computational burden by using only N pairs of examples, instead of (N +1)×N

Sohn, Kihyuk "Improved Deep Metric Learning with Multi-Class N-Pair Loss Objective." NIPS 2016

# N-Way Loss




Sohn, Kihyuk "Improved Deep Metric Learning with Multi-Class N-Pair Loss Objective." NIPS 2016

# N-way loss (2)

$$\mathcal{L}\left(\{x, x^+, \{x_i\}_{i=1}^{N-1}\}; f\right) = \log\left(1 + \sum_{i=1}^{N-1} \exp\left(f^\top f_i - f^\top f^+\right)\right)$$

- $x$ anchor
- $x^+$ positive
- $\{x_i\}_{i=1}^{N-1}$ negatives
- efficiency: the same N example are reused for all the mini-batch
  - only $N$ computations of the embeddings (expensive DNN forward pass)
  - $N^2$ dot products

Sohn, Kihyuk "Improved Deep Metric Learning with Multi-Class N-Pair Loss Objective." NIPS 2016

# Negative Mining

- we want to find the hard samples

**METHOD**: selection of negative samples for N-way loss

1. **Evaluate Embedding Vectors**:
   - choose randomly a large number of output classes C;
   - for each class, randomly pass a few (one or two) examples to extract their embedding vectors.
2. **Select Negative Classes**:
   - select one class randomly from C classes from step 1.
   - Next, greedily add a new class that **violates triplet constraint the most** w.r.t. the selected classes till we reach N classes. When a tie appears, we randomly pick one of tied classes
3. **Finalize N -pair**: draw two examples from each selected class from step 2.

Sohn, Kihyuk "Improved Deep Metric Learning with Multi-Class N-Pair Loss Objective." NIPS 2016

# Embedding Regularization

- the similarity $f^\top f^+$ depends on the direction and norm of the embeddings
- unconstrained optimization leads to unbounded norm growth
  - it pushes embeddings further without changing their direction
  - which we want to avoid
- normalization would solve the norm growth but it is too restrictive
- penalizing the embedding norm $||f||_2^2$ works well in practice
  - avoids unbounded growth by penalizing it
  - it doesn't constrain too much
- **KEEP IN MIND**: this is a common issue with embedding-based methods

Sohn, Kihyuk "Improved Deep Metric Learning with Multi-Class N-Pair Loss Objective." NIPS 2016

# SimCLR

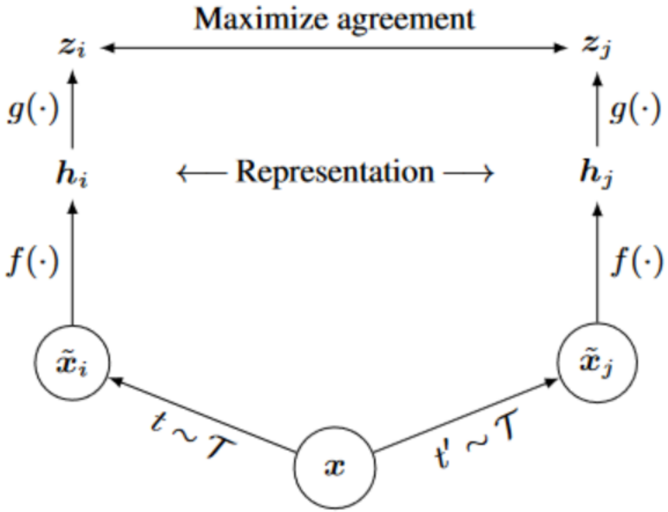
*SimCLR: a simple framework for contrastive learning of visual representations*

- **IDEA**: *maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space*
- (1) composition of data augmentations plays a critical role in defining effective predictive tasks
- (2) introducing a **learnable nonlinear transformation** between the representation and the contrastive loss substantially improves the quality of the learned representations, and
- (3) contrastive learning benefits from larger batch sizes and more training steps compared to supervised learning

T. Chen et al "A Simple Framework for Contrastive Learning of Visual Representations." 2020

# SimCLR - Components

- **Augmentations**: random crop, resize back to the original size, random color distortions, and random Gaussian blur. The combination of random crop and color distortion is crucial for performance.
- **base encoder**: $\mathbf{h}_i = f(\tilde{\mathbf{x}}_i)$, such as a ResNet without the linear classifier
- **projection head**: contrastive loss is not applied directly to the embeddings but to a projected space. 1-layer MLP projection $\mathbf{z}_i = g(\mathbf{h}_i) = W^{(2)}\mathrm{ReLU}\left(W^{(1)}\mathbf{h}_i\right)$




T. Chen et al "A Simple Framework for Contrastive Learning of Visual Representations." 2020

# NT-Xent Loss

- **contrastive loss**: given N samples, apply augmentations (get 2N samples), loss is a softmax on similarity scores
- **NT-Xent**: normalized temperature-scaled cross entropy loss
- $\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$
- cosine similarity $\mathrm{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v} / \| \mathbf{u} \| \| \mathbf{v} \|$
- $\tau$ softmax temperature
- doesn't use negative mining

T. Chen et al "A Simple Framework for Contrastive Learning of Visual Representations." 2020

# Algorithm

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

**input:** batch size $N$, constant $\tau$, 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 $g(\cdot)$

T. Chen et al "A Simple Framework for Contrastive Learning of Visual Representations." 2020

# Batch Size

## Contrastive Learning needs large batch sizes

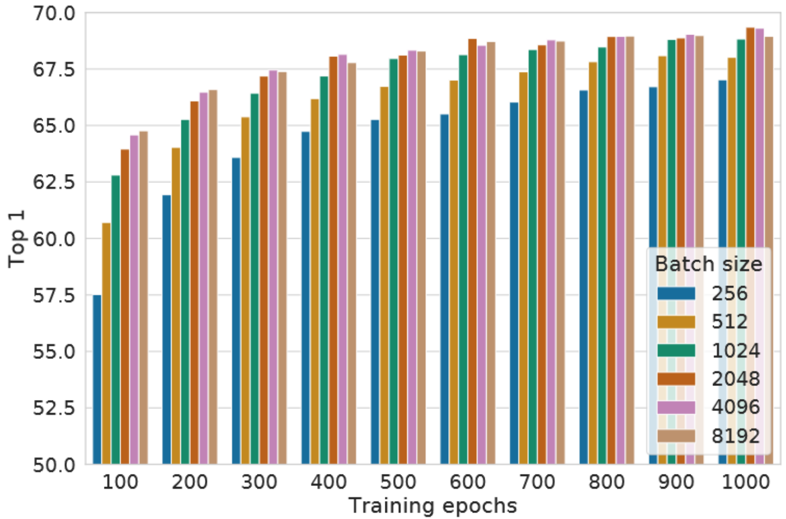



*Figure 9.* Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

T. Chen et al "A Simple Framework for Contrastive Learning of Visual Representations." 2020

# Performance

## SSL performance is close to Supervised Performance

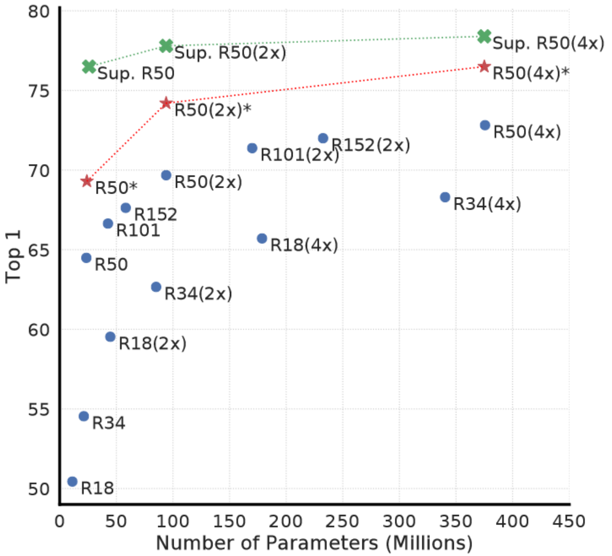
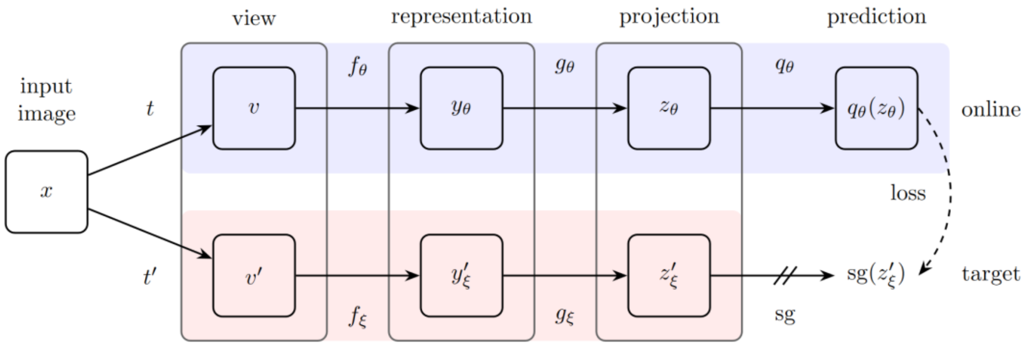



*Figure 7.* Linear evaluation of models with varied depth and width. Models in blue dots are ours trained for 100 epochs, models in red stars are ours trained for 1000 epochs, and models in green crosses are supervised ResNets trained for 90 epochs[7] (He et al., 2016).

T. Chen et al "A Simple Framework for Contrastive Learning of Visual Representations." 2020

# BYOL - Bootstrap Your Own Latent

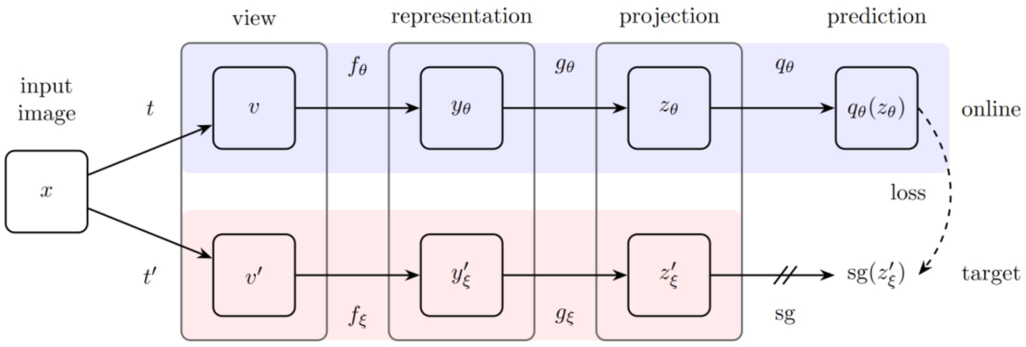
- **IDEA**: use a target network to provide target for the SSL model
- **BYOL**: bootstraps the outputs of a network to serve as targets for an enhanced representation
- no need for negative samples




J. Grill "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning." 2020

# BYOL - Components

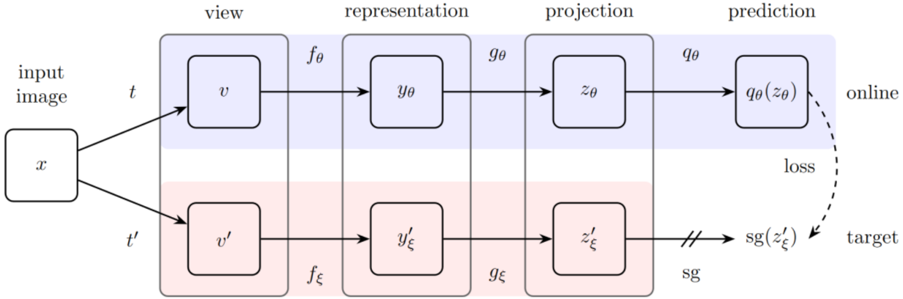
- two DNN, same architecture: *online* ($\theta$) and *target* ($\xi$) networks
- three modules: *encoder*, *projector*, *predictor*
- target network update with EMA: $\xi \leftarrow \tau\xi + (1-\tau)\theta$, $\tau$ decay rate




J. Grill "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning." 2020

# BYOL - Inputs

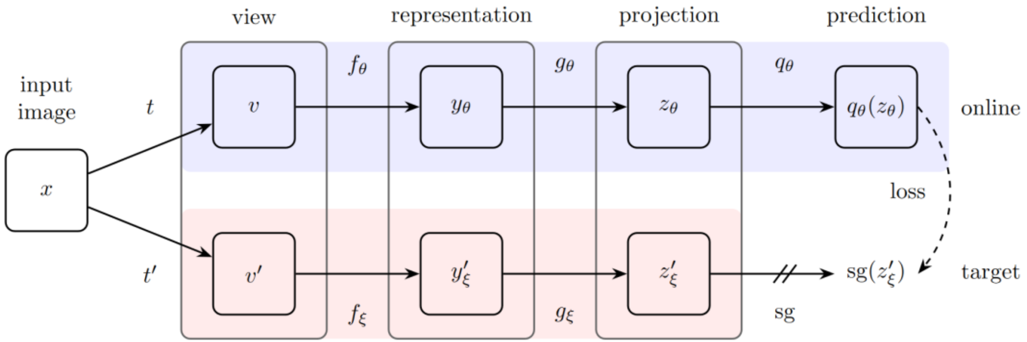
- two distributions of augmentations $\mathcal{T}$ and $\mathcal{T}'$ for the two networks
- two augmented views as input to the two networks
  - $v \triangleq t(x)$ for the online net $t \sim \mathcal{T}$
  - $v' \triangleq t'(x)$ for the target net $t' \sim \mathcal{T}'$
- outputs:
  - target net output $y'_\xi \triangleq f_\xi(v')$
  - target projection $z'_\xi \triangleq g_\xi(y')$
  - online net output $q_\theta(z_\theta)$
  - online net projection $q_\theta(z_\theta)$




J. Grill "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning." 2020

# BYOL - Loss

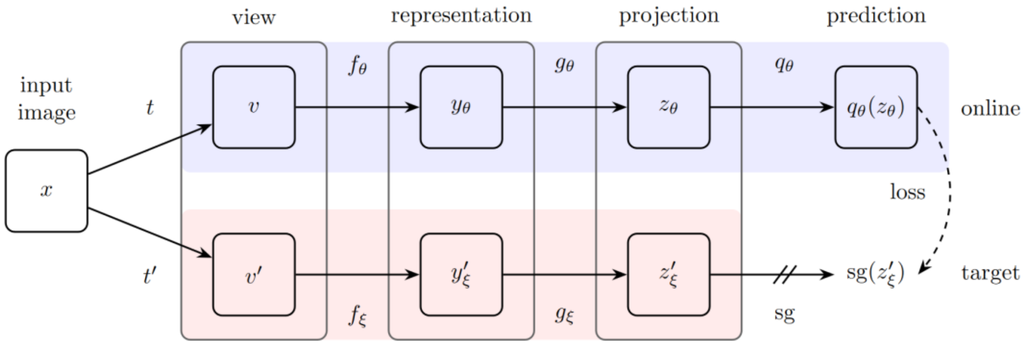
- mean squared error between the normalized predictions and target projections
- loss: $\mathcal{L}_{\theta,\xi} \triangleq \left\|\overline{q_\theta}(z_\theta) - \overline{z}'_\xi\right\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi \rangle}{\|q_\theta(z_\theta)\|_2 \cdot \|z'_\xi\|_2}$
- $\overline{z}$ normalized vectors
- $\mathcal{L}_{\theta,\xi}^{\text{BYOL}} = \mathcal{L}_{\theta,\xi} + \widetilde{\mathcal{L}}_{\theta,\xi}$
    - $\widetilde{\mathcal{L}}$ is the same as $\mathcal{L}$ with $v$ and $v'$ switched




J. Grill "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning." 2020

# BYOL - Backpropagation

- does not backpropagate through the target network (which is updated with the EMA)
- sg=stop-gradient, truncates the backprop
- $\theta \leftarrow \text{optimizer}\left(\theta, \nabla_\theta \mathcal{L}^{\text{BYOL}}_{\theta,\xi}, \eta\right)$
- $\xi \leftarrow \tau\xi + (1-\tau)\theta$




J. Grill "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning." 2020

# BYOL - Intuition

- the target network provides targets to the online network
- this works even if we use a randomly initialized network as target net
  - the random net gets 1.4% accuracy on ImageNet (linear evaluation)
  - the online net trained to predict the random target net outputs reaches 18.8%
  - much lower than BYOL but it still works
  - we just need to provide better targets than those provided by the random net

J. Grill "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning." 2020

# BYOL - Results

**Linear probing on ImageNet is close to supervised training**

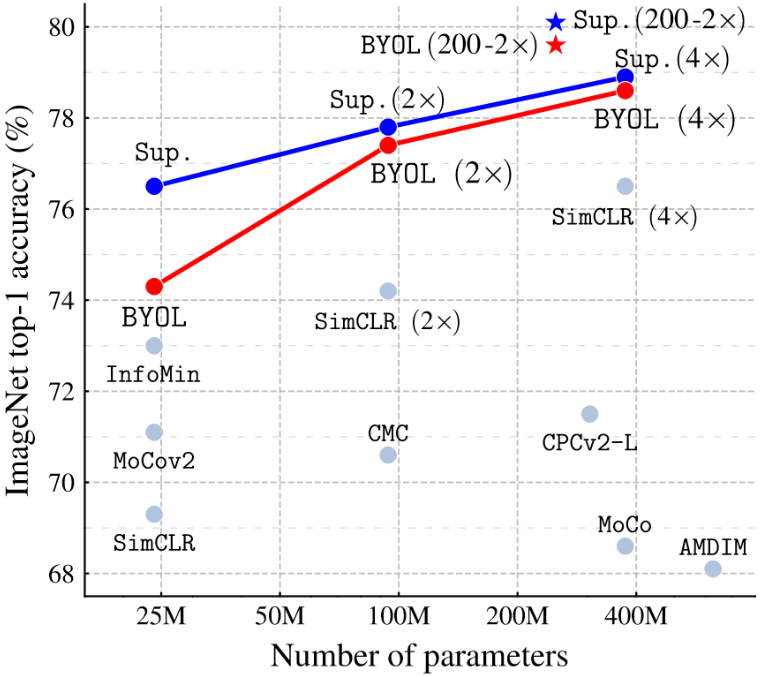



Figure 1: Performance of BYOL on ImageNet (linear evaluation) using ResNet-50 and our best architecture ResNet-200 (2×), compared to other unsupervised and supervised (Sup.) baselines [8].

J. Grill "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning." 2020

# Some Implementation Details

- **Hyperparemeters**:
  - many methods require large batch sizes and more iterations
  - in general, don’t expect hyperparameters optimized for the supervised setting to transfer to the SSL setting
- **IMPLEMENTATION DETAIL**: augmentations are expensive
  - they are often done on CPU. CPU←>GPU communication may become the bottleneck
  - heavy augmentations may be slow. The GPU has to wait, wasting GPU cycles.
  - several libraries are designed to optimized the preprocessing pipeline

# Recap - Contrastive Learning

- **IDEA**: pull positive closer, push negatives further
- **Triplet loss**: select one anchor, one positive and one negative. Mine for hard negatives
- **N-way loss**: select N samples and do all the pairwise comparisons
- **SimCLR**: augmentations + contrastive loss + linear projection
- **BYOL**: bootstraps the outputs of a network to serve as targets for an enhanced representation

# Take-Home Messages

- SSL methods learn robust representation without any supervision
- Augmentations are a fundamental tool to learn robust representations
- you can learn robust representations using contrastive learning (positives gets closer, negatives are pushed further)
- in vision, supervised models are still SotA, unlike NLP, but the gap is closing

# References

- papers in the footnotes
- good blogpost with an overview of many more methods: https://lilianweng.github.io/posts/2019-11-10-self-supervised/

# Next Lecture

- Definition of Meta-Learning
- Few-Shot Learning
- Deep Metric Learning