BYOL:

Bootstrap Your Own Latent
A New Approach to Self-Supervised Learning

Jean-Bastien Grill, F.Strub, F.Altche, C.Tallec, P.H.Richemond et al.

DeepMind, Imperial College

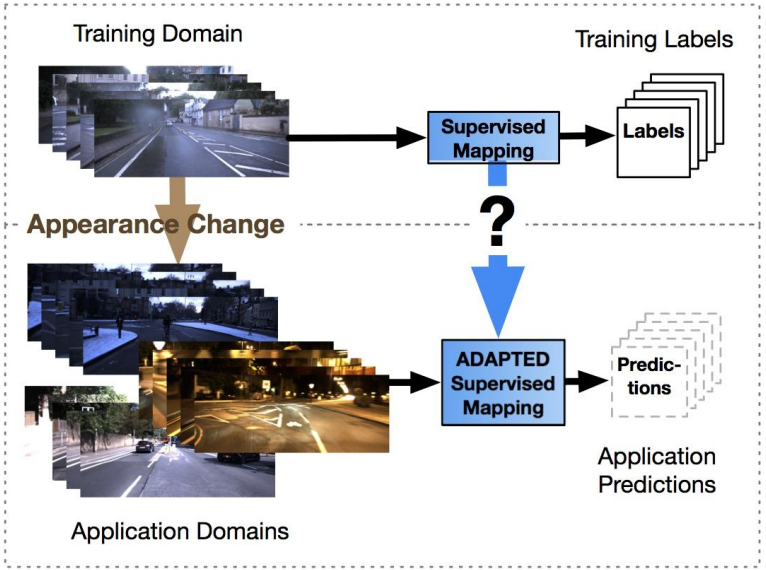
Sungman, Cho.

Appendix. Background

Why Self-Supervised ?

- To make a good model, we need to **large data.**

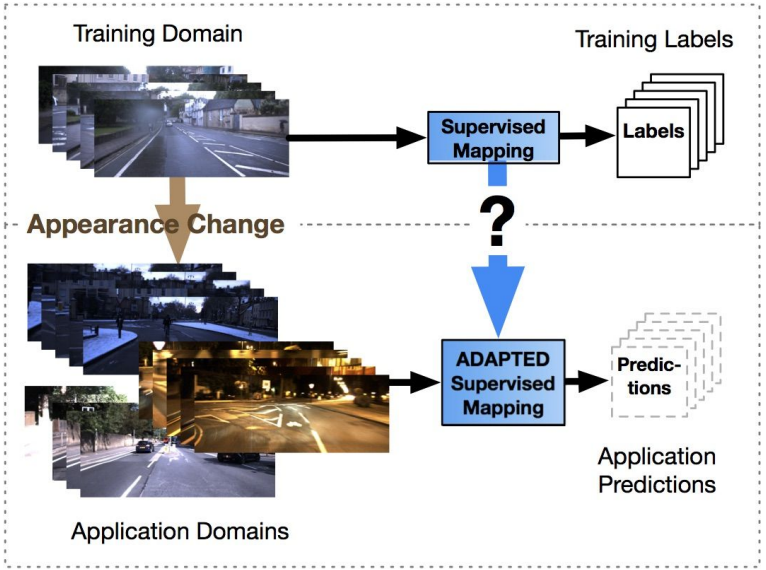
- If we can’t get large data ?
    - Transfer Learning
    - Domain Adaptation
    - Semi-Supervised Learning
    - Weakly-Supervised Learning
    - Self-Supervised Learning



Why Self-Supervised ?

- To make a good model, we need to **large data.**

- If we can’t get large data ?
  - Transfer Learning
  - Domain Adaptation
  - Semi-Supervised Learning
  - Weakly-Supervised Learning
  - **Self-Supervised Learning**



Self-Supervised ?

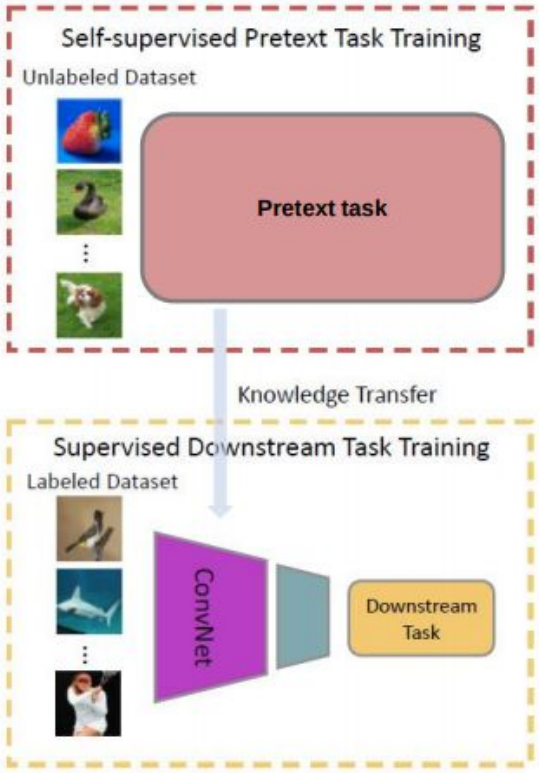
Yann LeCun
2019년 4월 30일 ·

I now call it "self-supervised learning", because "unsupervised" is both a loaded and confusing term.

In self-supervised learning, the system learns to predict part of its input from other parts of it input. In other words a portion of the input is used as a supervisory signal to a predictor fed with the remaining portion of the input.

Self-supervised learning uses way more supervisory signals than supervised learning, and enormously more than reinforcement learning. That's why calling it "unsupervised" is totally misleading. That's also why more knowledge about the structure of the world can be learned through self-supervised learning than from the other two paradigms: the data is unlimited, and amount of feedback provided by each example is huge.

Pretext Task & Downstream Task



Pretext Task

- Pre-designed tasks for networks to solve, and visual features are learned by learning objective functions of pretext tasks.

Downstream Task

- Computer vision applications that are used to evaluate the quality of features learned by self-supervised learning.

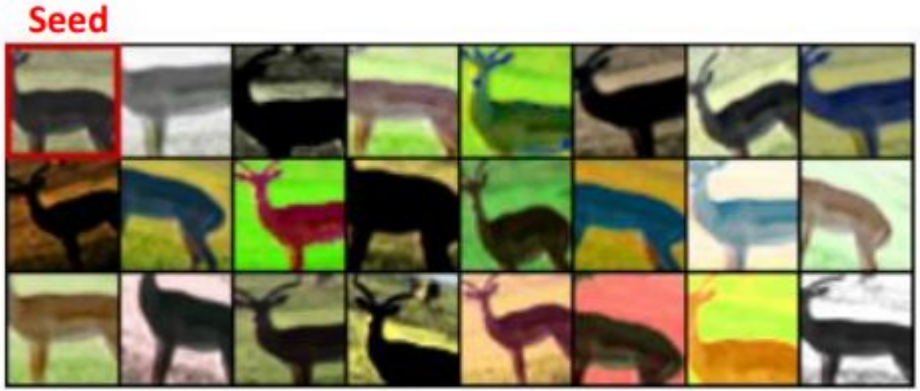
Appendix. Case Study

Image Transformation

Image Transformation

Conference / Journal	Paper	ImageNet Acc (Top 1).
CVPR 2018	Unsupervised feature learning via non-parametric instance discrimination (NPID++)	59.0 %
ICCV 2019	Scaling and Benchmarking Self-Supervised Visual Representation Learning (Jigsaw)	45.7 %
arXiv:1912.01991	Self-Supervised Learning of Pretext-Invariant Representations (PIRL)	63.6 %
CVPR 2020	Steering Self-Supervised Feature Learning Beyond Local Pixel Statistics	-
arXiv:2003.04298	Multi-modal Self-Supervision from Generalized Data Transformations	-

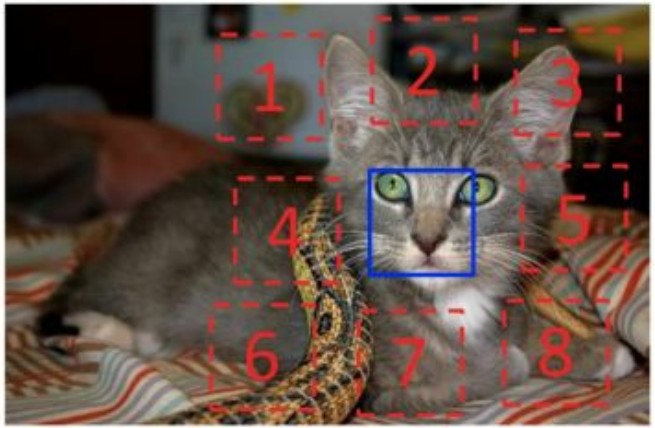
Exemplar, 2014 NIPS



Train with STL-10 dataset (96x96)

Algorithm	STL-10	CIFAR-10(400)	CIFAR-10	Caltech-101	Caltech-256(30)	#features
Convolutional K-means Network [32]	60.1 ± 1	70.7 ± 0.7	82.0	—	—	8000
Multi-way local pooling [33]	—	—	—	77.3 ± 0.6	41.7	1024×64
Slowness on videos [14]	61.0	—	—	74.6	—	556
Hierarchical Matching Pursuit (HMP) [34]	64.5 ± 1	—	—	—	—	1000
Multipath HMP [35]	—	—	—	82.5 ± 0.5	50.7	5000
View-Invariant K-means [16]	63.7	72.6 ± 0.7	81.9	—	—	6400
Exemplar-CNN (64c5-64c5-128f)	67.1 ± 0.2	69.7 ± 0.3	76.5	$79.8 \pm 0.5^{*}$	42.4 ± 0.3	256
Exemplar-CNN (64c5-128c5-256c5-512f)	72.8 ± 0.4	75.4 ± 0.2	82.2	$86.1 \pm 0.5^{\dagger}$	51.2 ± 0.2	960
Exemplar-CNN (92c5-256c5-512c5-1024f)	$\mathbf{74.2 \pm 0.4}$	$\mathbf{76.6 \pm 0.2}$	**84.3**	$\mathbf{87.1 \pm 0.7}^{\ddagger}$	$\mathbf{53.6 \pm 0.2}$	1884
Supervised state of the art	70.1[36]	—	92.0 [37]	91.44 [38]	70.6 [2]	—

Context Prediction, 2015 ICCV

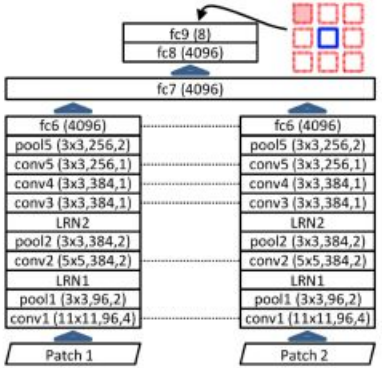


X = (,); Y = 3

Example:

Question 1:

Question 2:



Jigsaw Puzzle, 2016 ECCV

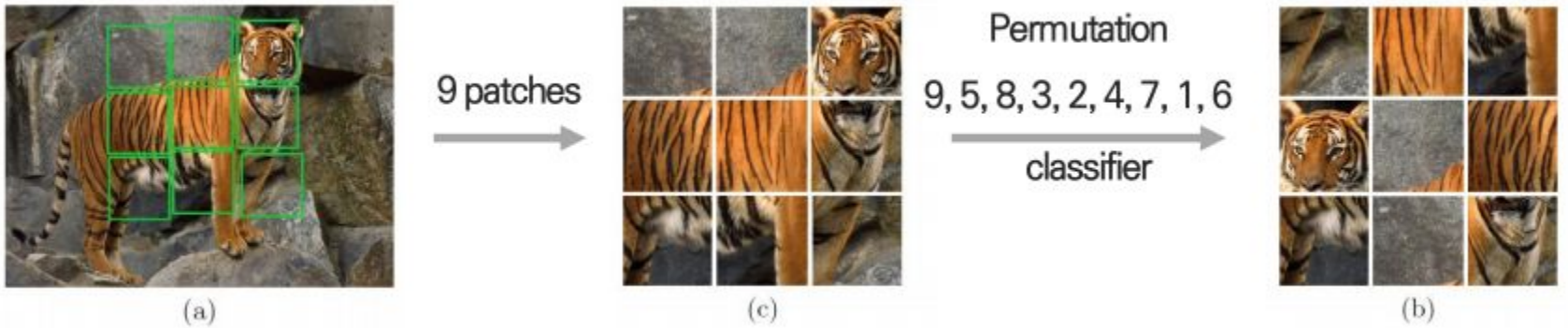
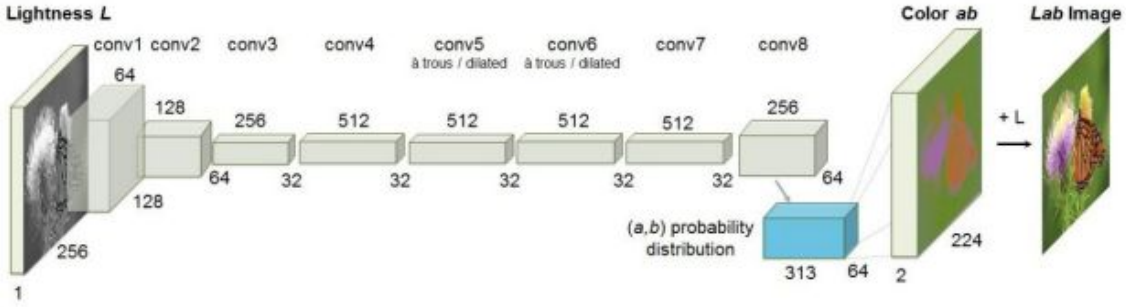
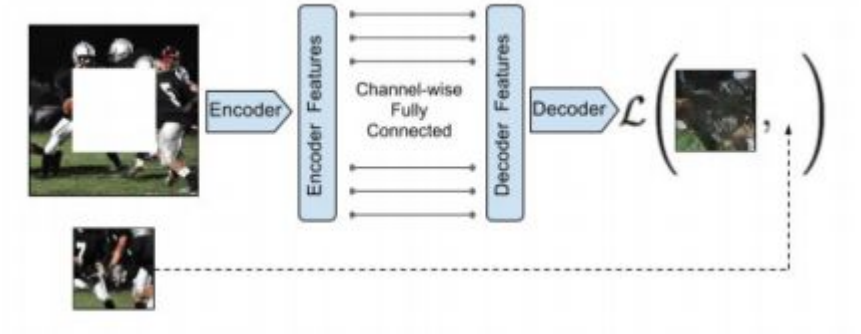
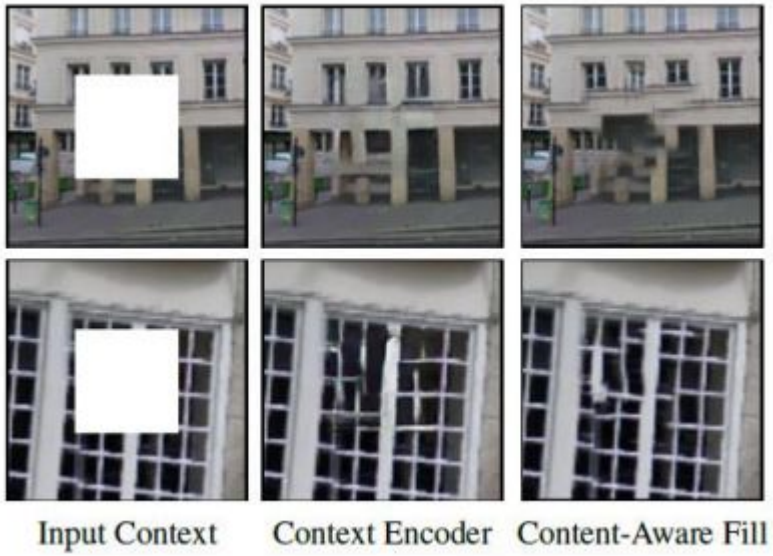


Image Colorization, 2016 ECCV



Context Autoencoder, 2016 CVPR





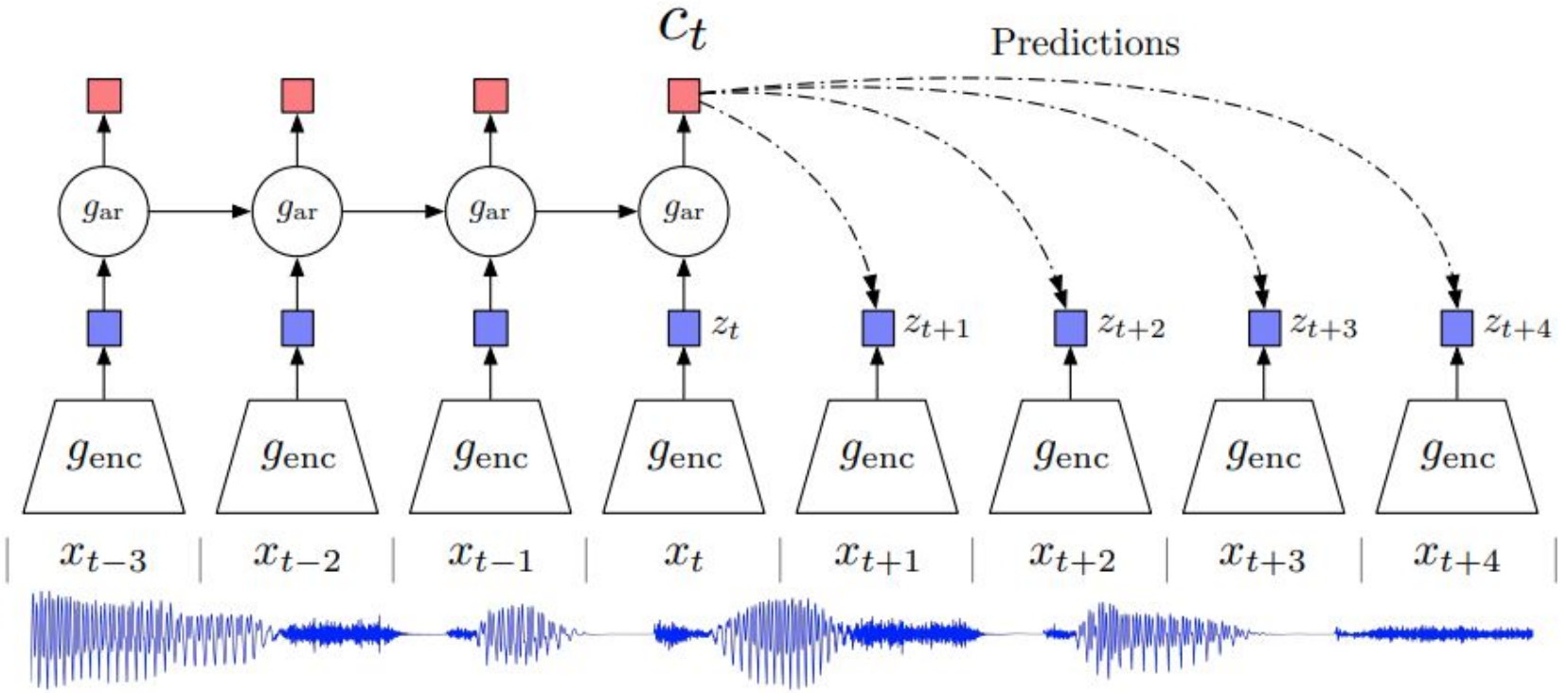
Appendix. Case Study

Contrastive

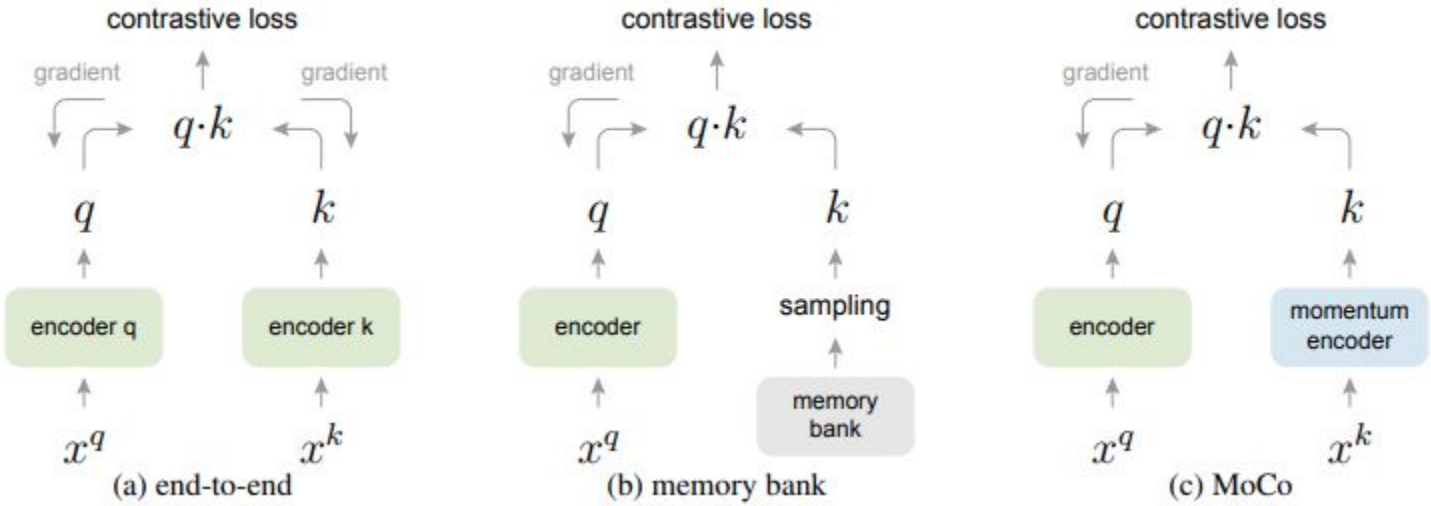
Contrastive Learning

Conference / Journal	Paper	ImageNet Acc (Top 1).
CVPR 2006	Dimensionality Reduction by Learning an Invariant Mapping	-
arXiv:1807.03748	Representation learning with contrastive predictive coding (CPC)	-
arXiv:1911.05722	Momentum Contrast for Unsupervised Visual Representation Learning (MoCo)	60.6 %
arXiv:1905.09272	Data-Efficient Image Recognition contrastive predictive coding (CPC v2)	63.8 %
arXiv:1906.05849	Contrastive Multiview Coding	66.2 %
arXiv:2002.05709	A Simple Framework for Contrastive Learning of Visual Representations (SimCLR)	69.3 %
arXiv:2003.12338	Improved Baselines with Momentum Contrastive Learning(MoCo v2)	71.1 %
arXiv:2003.05438	Rethinking Image Mixture for Unsupervised Visual Representation Learning	65.9 %
arXiv:2004.05554	Feature Lenses: Plug-and-play Neural Modules for Transformation-Invariant Visual Representations	
arXiv:2006.10029	Big Self-Supervised Models are Strong Semi-Supervised Learners	74.3 % (10% label)
arXiv:2006.07733	Bootstrap Your Own Latent A New Approach to Self-Supervised Learning	74.3 % (linear classification)

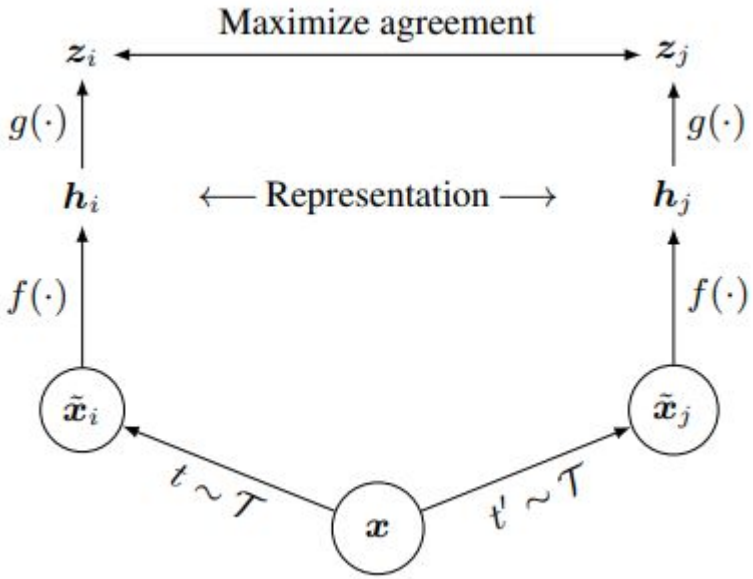
CPC, arXiv:1807.03748

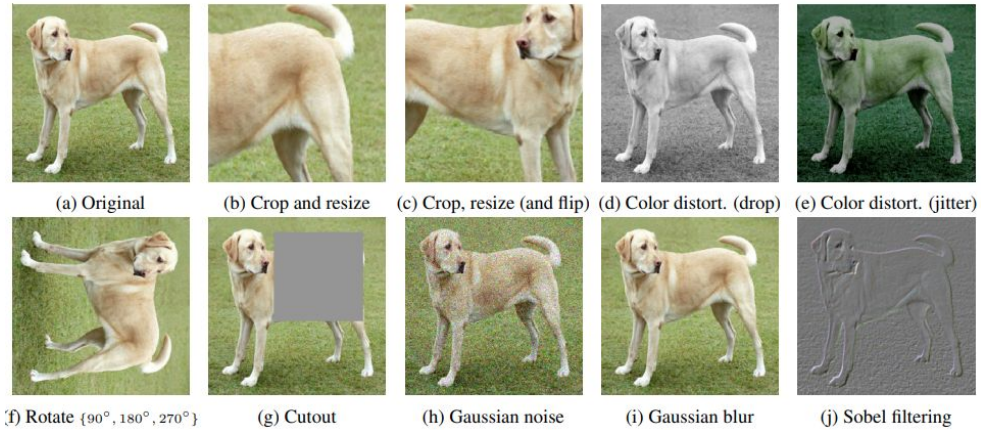


MoCo v2, 2020 CVPR



SimCLR, arXiv:2002.05709





References @ https://github.com/Sungman-Cho/Awesome-Self-Supervised-Papers

Contrastive Learning

Conference / Journal	Paper	ImageNet Acc (Top 1).
CVPR 2006	Dimensionality Reduction by Learning an Invariant Mapping	-
arXiv:1807.03748	Representation learning with contrastive predictive coding (CPC)	-
arXiv:1911.05722	Momentum Contrast for Unsupervised Visual Representation Learning (MoCo)	60.6 %
arXiv:1905.09272	Data-Efficient Image Recognition contrastive predictive coding (CPC v2)	63.8 %
arXiv:1906.05849	Contrastive Multiview Coding	66.2 %
arXiv:2002.05709	A Simple Framework for Contrastive Learning of Visual Representations (SimCLR)	69.3 %
arXiv:2003.12338	Improved Baselines with Momentum Contrastive Learning(MoCo v2)	71.1 %
arXiv:2003.05438	Rethinking Image Mixture for Unsupervised Visual Representation Learning	65.9 %
arXiv:2004.05554	Feature Lenses: Plug-and-play Neural Modules for Transformation-Invariant Visual Representations	
arXiv:2006.10029	Big Self-Supervised Models are Strong Semi-Supervised Learners	74.3 % (10% label)
arXiv:2006.07733	Bootstrap Your Own Latent A New Approach to Self-Supervised Learning	74.3 % (linear classification)

Image Transformation

Conference / Journal	Paper	ImageNet Acc (Top 1).
CVPR 2018	Unsupervised feature learning via non-parametric instance discrimination (NPID++)	59.0 %
ICCV 2019	Scaling and Benchmarking Self-Supervised Visual Representation Learning (Jigsaw)	45.7 %
arXiv:1912.01991	Self-Supervised Learning of Pretext-Invariant Representations (PIRL)	63.6 %
CVPR 2020	Steering Self-Supervised Feature Learning Beyond Local Pixel Statistics	-
arXiv:2003.04298	Multi-modal Self-Supervision from Generalized Data Transformations	-

Others (in Pretraining / Feature / Representation)

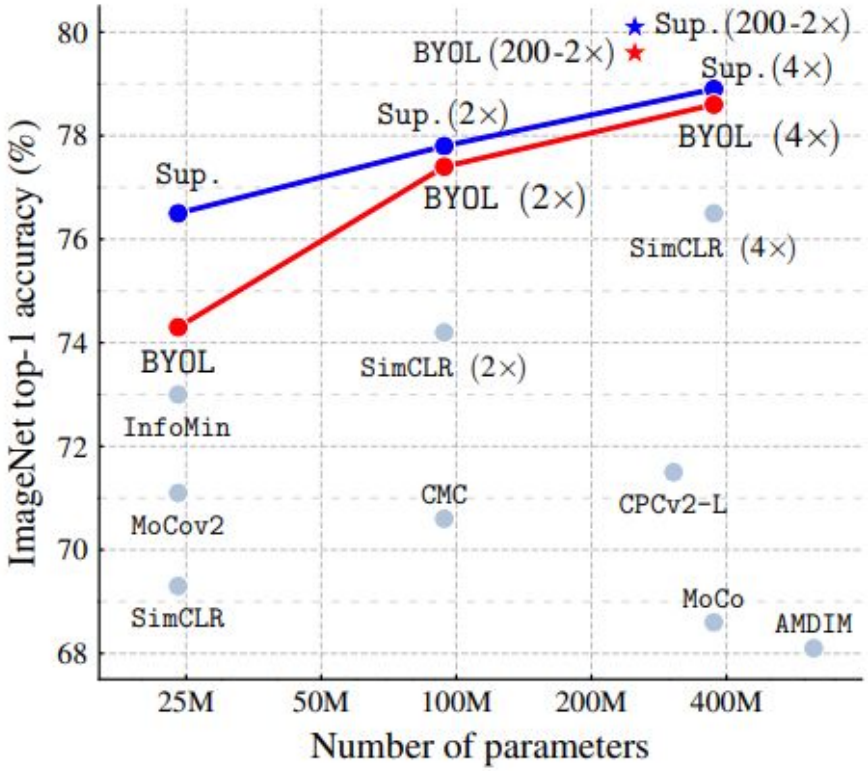
Conference / Journal	Paper	Method
ICML 2018	Mutual Information Neural Estimation	Mutual Information
NeurIPS 2019	Wasserstein Dependency Measure for Representation Learning	Mutual Information
ICLR 2019	Learning Deep Representations by Mutual Information Estimation and Maximization	Mutual Information
arXiv:1903.12355	Local Aggregation for Unsupervised Learning of Visual Embeddings	Local Aggregation
arXiv:1906.00910	Learning Representations by Maximizing Mutual Information Across Views	Mutual Information
ICLR 2020	On Mutual Information Maximization for Representation Learning	Mutual Information
CVPR 2020	How Useful is Self-Supervised Pretraining for Visual Tasks?	-
CVPR 2020	Adversarial Robustness: From Self-Supervised Pre-Training to Fine-Tuning	Adversarial Training
ICLR 2020	Self-Labeling via Simultaneous Clustering and Representation Learning	Information
arXiv:1912.11370	Big Transfer (BiT): General Visual Representation Learning	pre-training

Introduction

Introduction

- State-of-the-art contrastive methods
    - reducing the distance between positive pairs
    - increasing the distance between negative pairs
- Careful treatment of negative pairs
    - **Large batch size** : SimCLR(20.02) / Googlebrain
    - **Memory bank** : MoCo(19.11), MoCo v2(20.03) / FAIR
    - **Customized mining strategies**

Introduction



Contributions

- Achieves higher performance than state-of-the-art contrastive methods **without using negative pairs**.
- More robust to the choice of image augmentations than contrastive methods.
- Uses two neural networks, referred to as online and target networks, that interact and learn from each other.
- Trains its online network to predict the target network's representation of another augmented view of the same image.

Related work

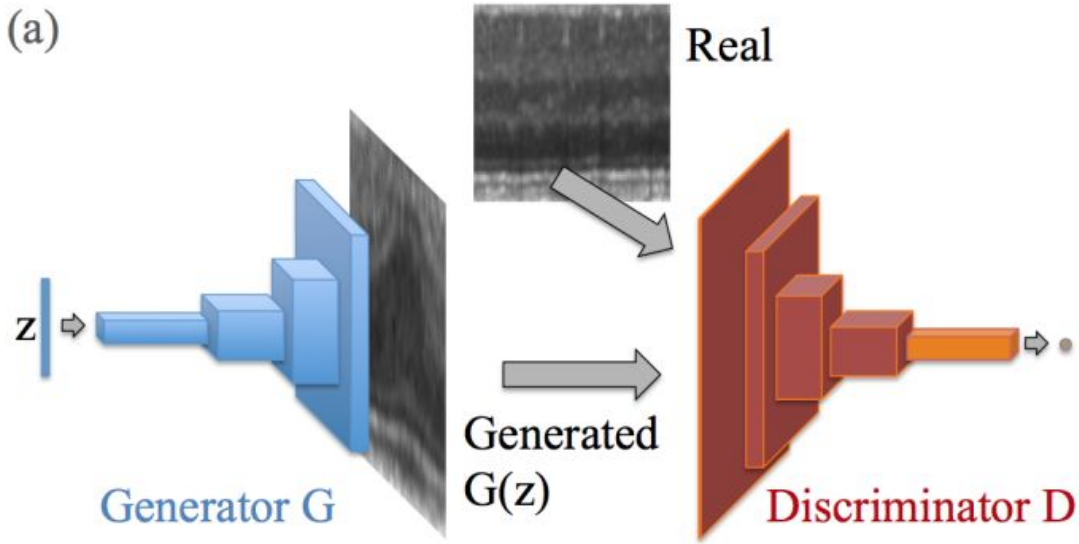
- Most unsupervised methods for representation learning:
    - **Generative :**

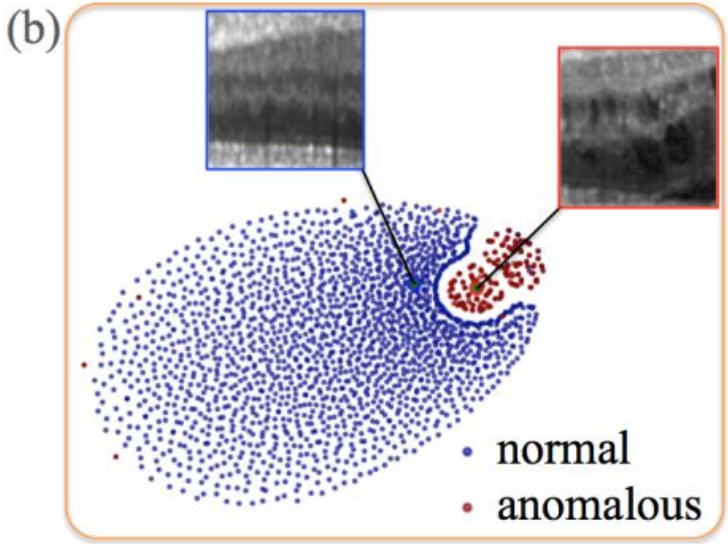
      build a distribution over data and latent embedding and use the learned embeddings as image representations.

      Operate directly in pixel space.

      → computationally expensive
      → high level of detail required for image generation may not be necessary for representation learning.

Related work - Generative (AnoGAN)





Related work

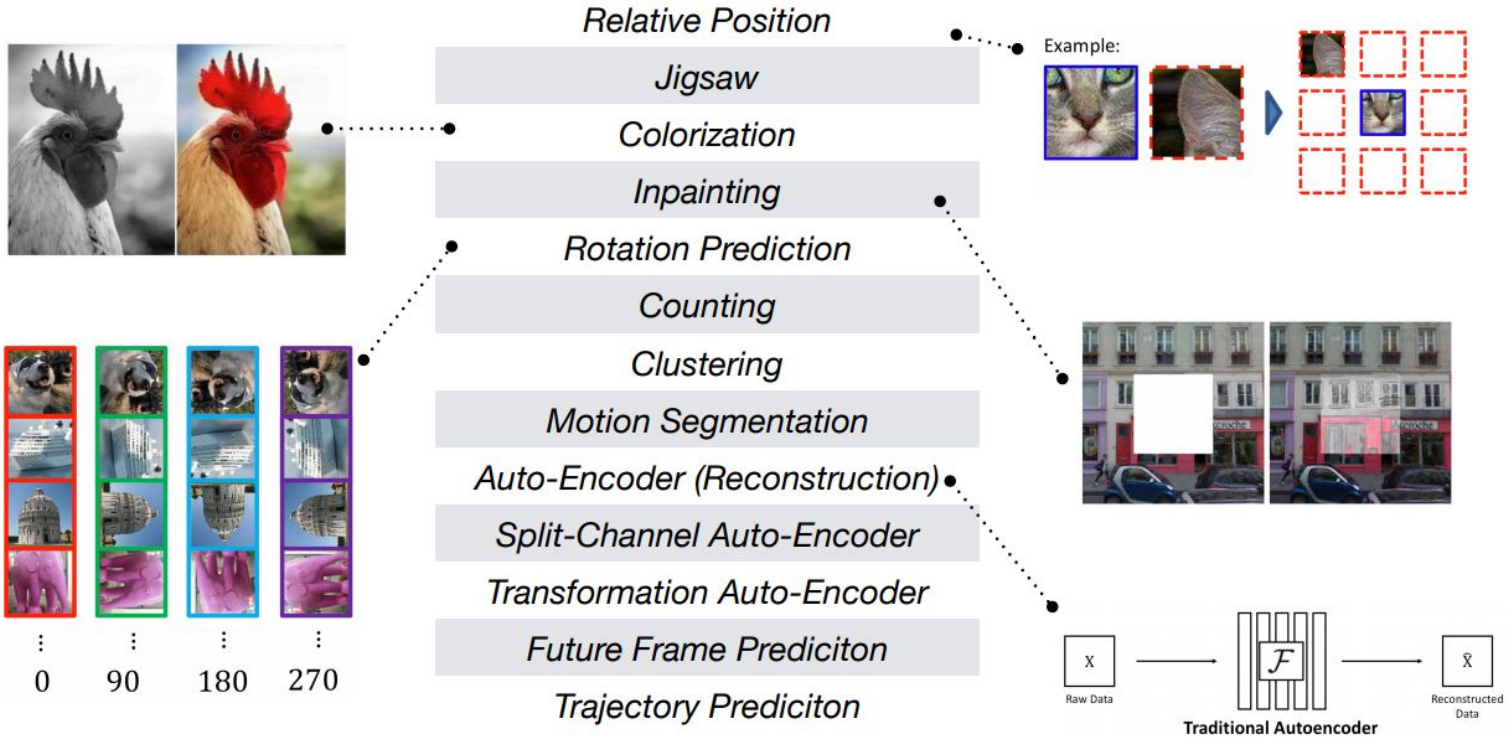
- Most unsupervised methods for representation learning:
    - **Discriminative :**

      → Contrastive approaches avoid a costly generation step in pixel space.

      → Deep Cluster uses bootstrapping on previous versions of its representation to produce targets for the next representation

      → Relative patch prediction, Colorizing gray-scale image, image inpainting, ...

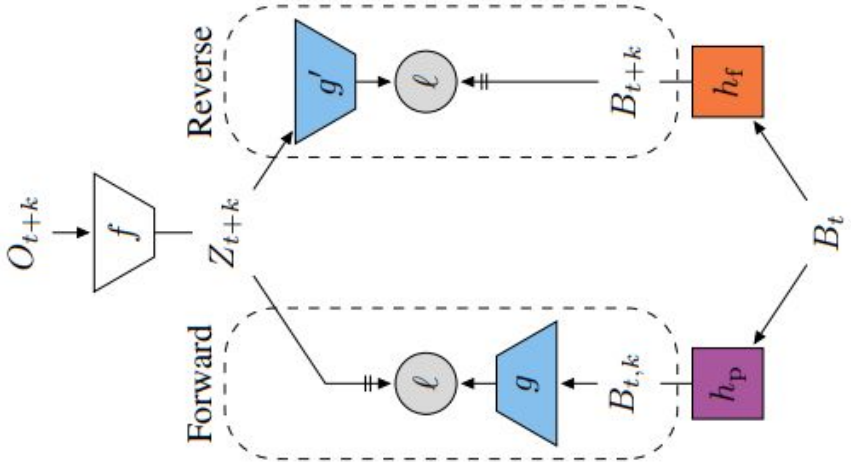
Related work - Discriminative



Related work

- Most unsupervised methods for representation learning:
  - **Predictions of Bootstrapped Latents (PBL: 20.04 / DeepMind)**

    → A self-supervised representation learning technique for **<u>reinforcement learning.</u>**



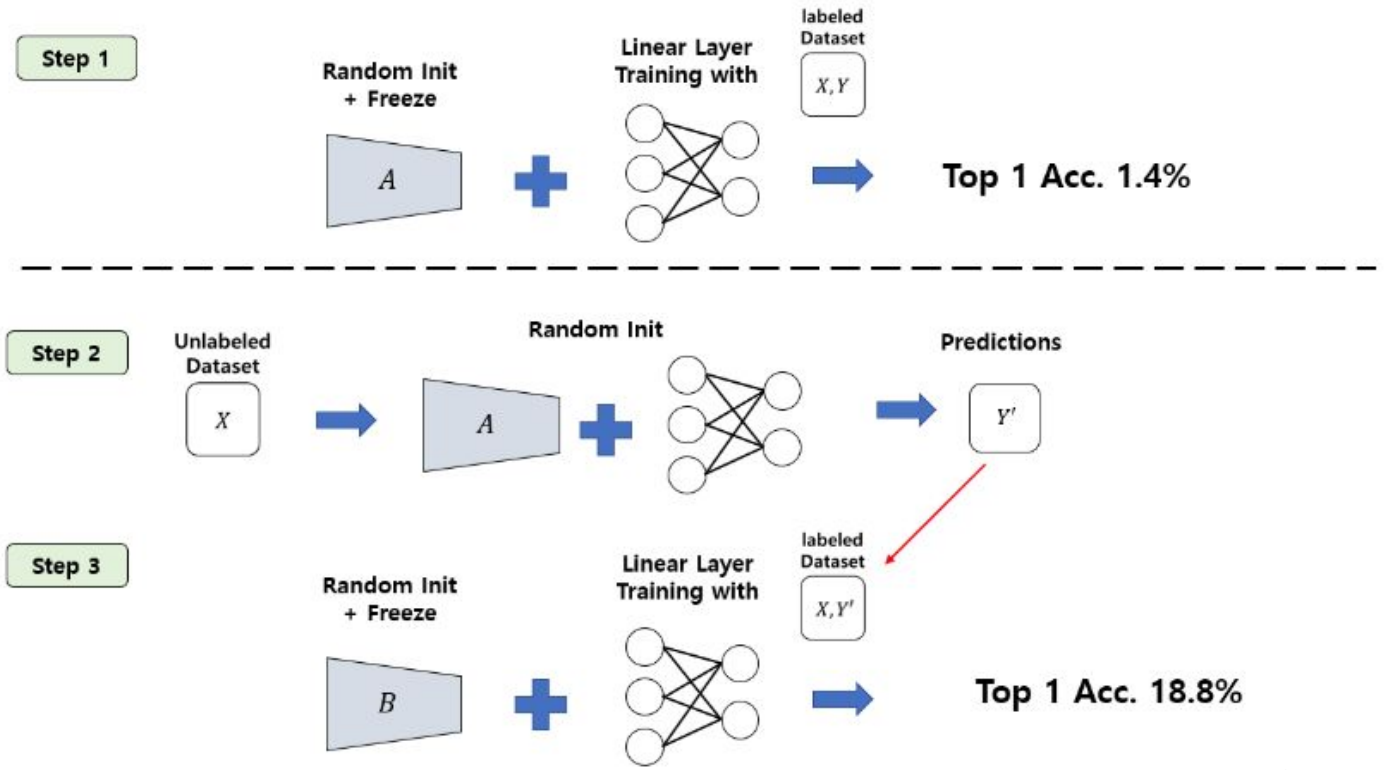
Method

- Many approaches cast the prediction problem directly in representation space: augmented views.

- Predicting directly in representation space can lead to collapsed representation: for instance, a representation that is constant across view is always **fully predictive of it self.**

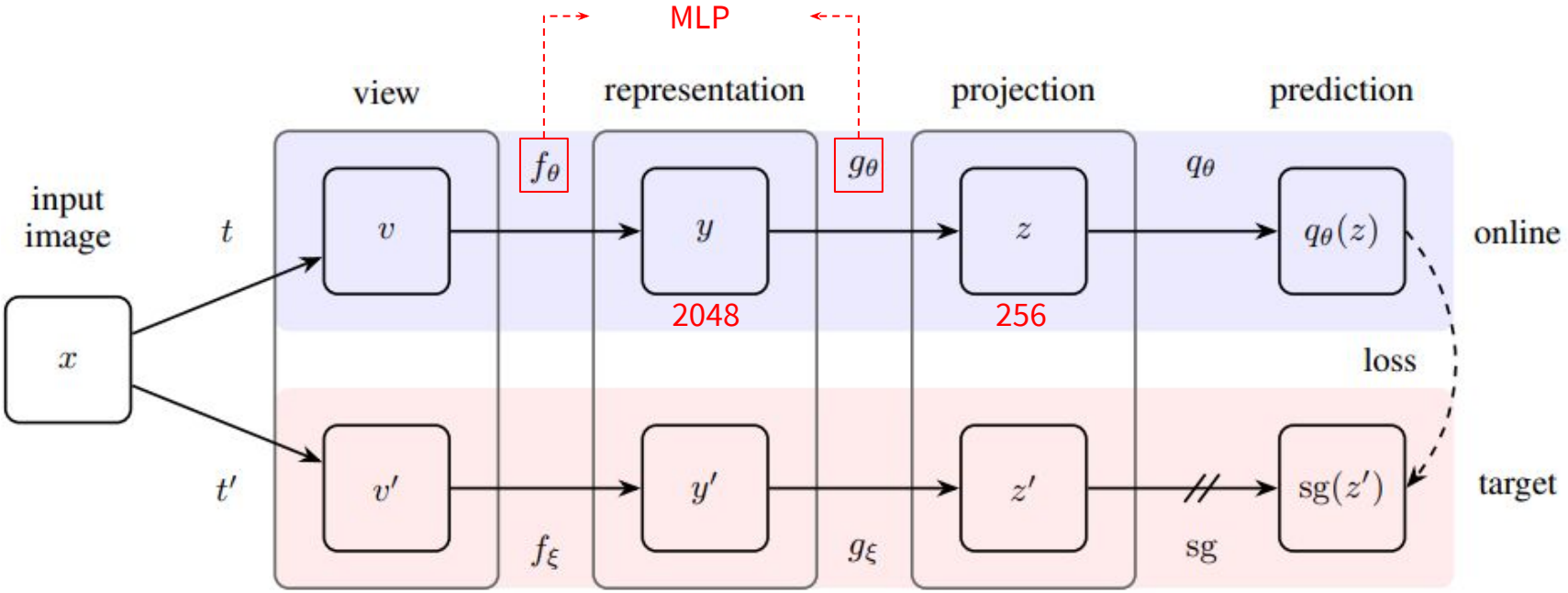
- Discriminative approach typically requires comparing each representation of an augmented view with many negative examples.

Method

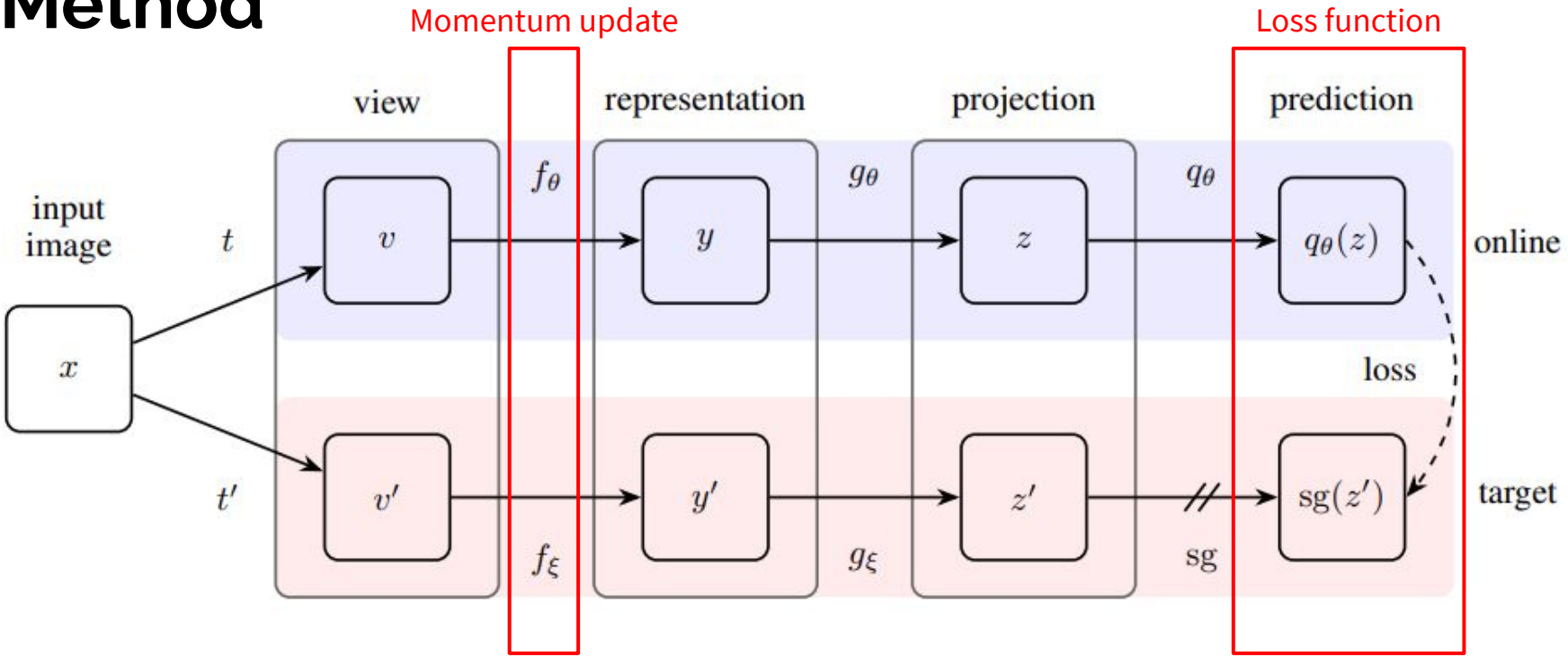


ref: https://hoya012.github.io/blog/byol/

Method


MLP
view
representation
projection
prediction
input image
t
t'
x
v
v'
f_θ
f_ξ
y
y'
2048
g_θ
g_ξ
z
z'
256
q_θ
sg
$q_\theta(z)$
$\mathrm{sg}(z')$
online
loss
target

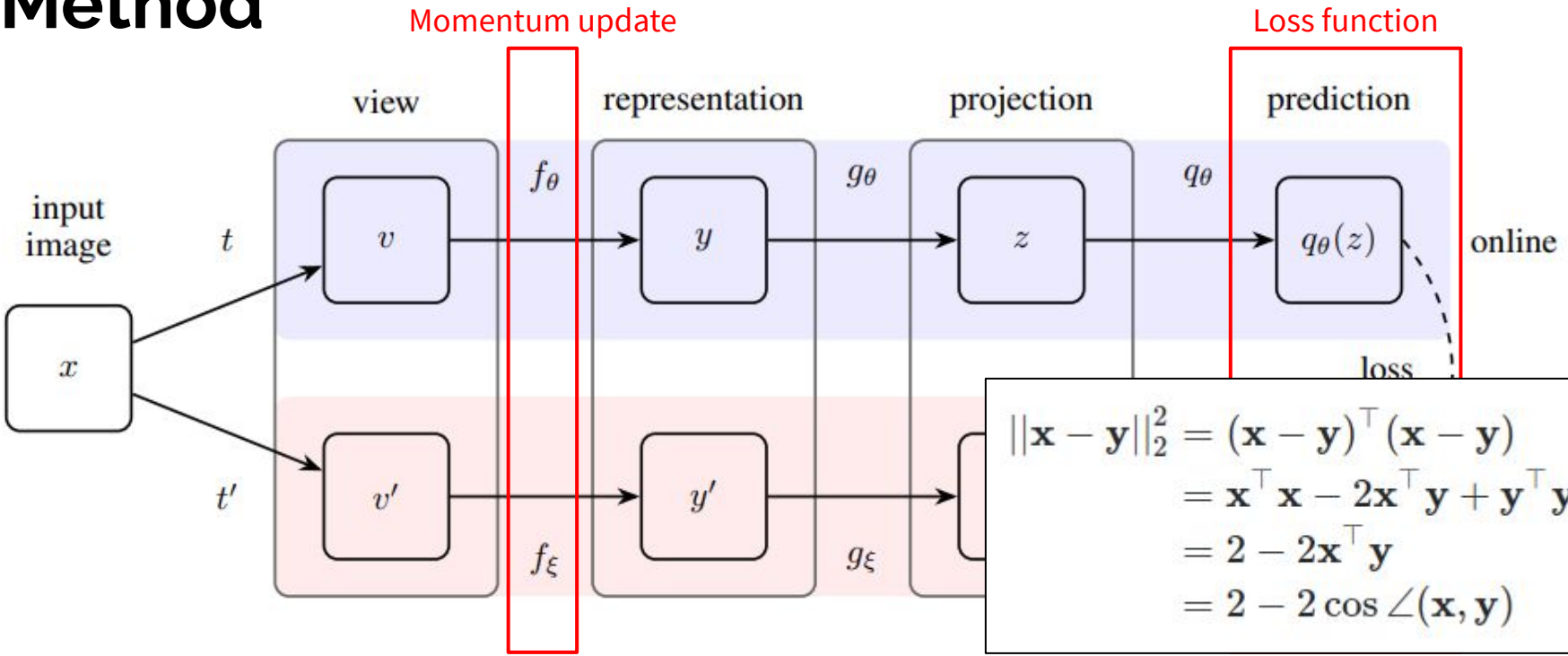
Method



$$\xi \leftarrow \tau\xi + (1-\tau)\theta.$$

$$\mathcal{L}_\theta^{\mathrm{BYOL}} \triangleq \left\|\overline{q_\theta}(z_\theta) - \overline{z}'_\xi\right\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi \rangle}{\left\|q_\theta(z_\theta)\right\|_2 \cdot \left\|z'_\xi\right\|_2}.$$

Method



$$||\mathbf{x}-\mathbf{y}||_2^2 = (\mathbf{x}-\mathbf{y})^\top(\mathbf{x}-\mathbf{y}) = \mathbf{x}^\top\mathbf{x} - 2\mathbf{x}^\top\mathbf{y} + \mathbf{y}^\top\mathbf{y} = 2 - 2\mathbf{x}^\top\mathbf{y} = 2 - 2\cos\angle(\mathbf{x},\mathbf{y})$$

$$\xi \leftarrow \tau\xi + (1-\tau)\theta.$$

$$\mathcal{L}_\theta^{\text{BYOL}} \triangleq \left\|\overline{q_\theta}(z_\theta) - \overline{z}'_\xi\right\|_2^2 = 2 - 2\cdot\frac{\langle q_\theta(z_\theta), z'_\xi\rangle}{\|q_\theta(z_\theta)\|_2 \cdot \|z'_\xi\|_2}.$$

Method (detail)

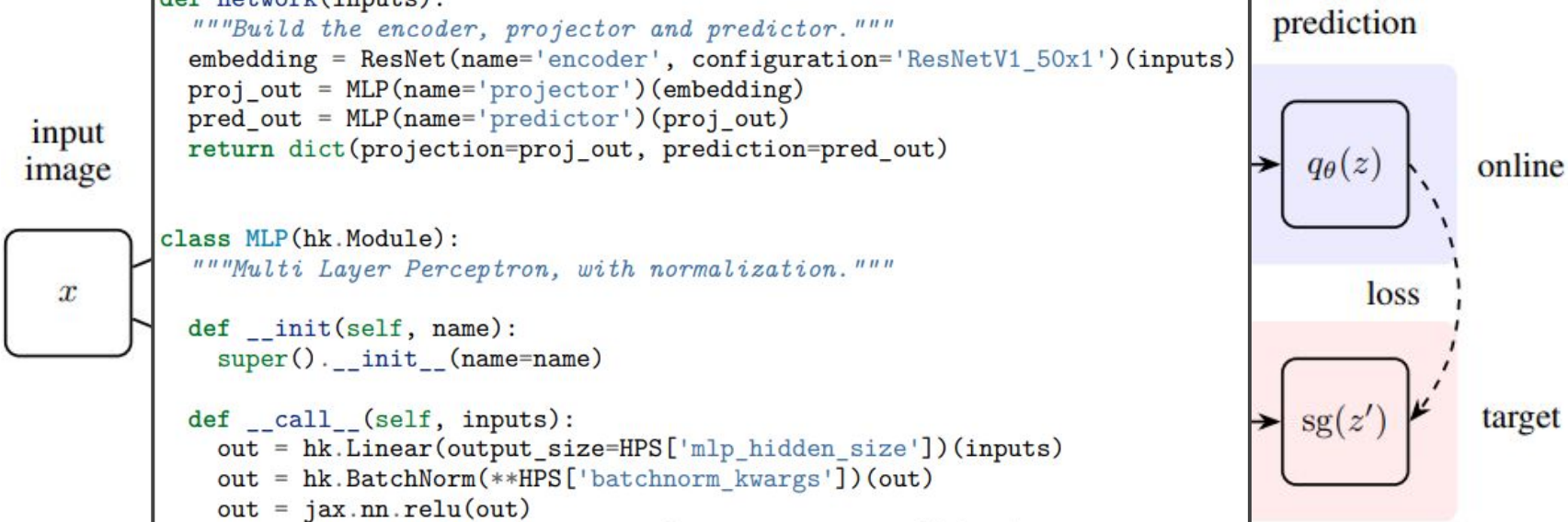
MLP

```python
def network(inputs):
  """Build the encoder, projector and predictor."""
  embedding = ResNet(name='encoder', configuration='ResNetV1_50x1')(inputs)
  proj_out = MLP(name='projector')(embedding)
  pred_out = MLP(name='predictor')(proj_out)
  return dict(projection=proj_out, prediction=pred_out)


class MLP(hk.Module):
  """Multi Layer Perceptron, with normalization."""

  def __init(self, name):
    super().__init__(name=name)

  def __call__(self, inputs):
    out = hk.Linear(output_size=HPS['mlp_hidden_size'])(inputs)
    out = hk.BatchNorm(**HPS['batchnorm_kwargs'])(out)
    out = jax.nn.relu(out)
    out = hk.Linear(output_size=HPS['projection_size'])(out)
    return out
```



Details

Proj. g_θ depth	Pred. q_θ depth	Top-1	Top-5
1	1	61.9	86.0
	2	65.0	86.8
	3	65.7	86.8
2	1	71.5	90.7
	2	**72.5**	**90.8**
	3	71.4	90.4
3	1	71.4	90.4
	2	72.1	90.5
	3	72.1	90.5

(a) Projector and predictor depth (i.e. the number of Linear layers).

Projector g_θ output dim	Top-1	Top-5
16	$69.9_{\pm 0.3}$	89.9
32	71.3	90.6
64	72.2	90.9
128	72.5	91.0
256	72.5	90.8
512	**72.6**	**91.0**

(b) Projection dimension.

Details

Learning rate	Top-1	Top-5
0.01	34.8±3.0	60.8±3.2
0.1	65.0	87.0
0.2	71.7	90.6
0.3	**72.5**	**90.8**
0.4	72.3	90.6
0.5	71.5	90.1
1	69.4	89.2

(a) Base learning rate.

Weight decay coefficient	Top-1	Top-5
$1 \cdot 10^{-7}$	72.1	90.4
$5 \cdot 10^{-7}$	**72.6**	**91.0**
$1 \cdot 10^{-6}$	72.5	90.8
$5 \cdot 10^{-6}$	71.0±0.3	90.0
$1 \cdot 10^{-5}$	69.6±0.4	89.3

(b) Weight decay.

Table 15: Effect of learning rate and weight decay. We note that BYOL's performance is quite robust within a range of hyperparameters.

Batch size	Top-1		Top-5	
	BYOL (ours)	SimCLR (repro)	BYOL (ours)	SimCLR (repro)
4096	**72.5**	67.9	**90.8**	88.5
2048	72.4	67.8	90.7	88.5
1024	72.2	67.4	90.7	88.1
512	72.2	66.5	90.8	87.6
256	71.8	64.3±2.1	90.7	86.3±1.0
128	69.6±0.5	63.6	89.6	85.9
64	59.7±1.5	59.2±2.9	83.2±1.2	83.0±1.9

Table 16: Influence of the batch size.

Pseudo Code

Algorithm 1: BYOL: **B**ootstrap **Y**our **O**wn **L**atent

Inputs :

$\mathcal{D}$, $\mathcal{T}$, and $\mathcal{T}'$	set of images and distributions of transformations
θ, f_θ, g_θ, and q_θ	initial online parameters, encoder, projector, and predictor
ξ, f_ξ, g_ξ	initial target parameters, target encoder, and target projector
optimizer	optimizer, updates online parameters using the loss gradient
K and N	total number of optimization steps and batch size
$\{\tau_k\}_{k=1}^{K}$ and $\{\eta_k\}_{k=1}^{K}$	target network update schedule and learning rate schedule

for $k = 1$ **to** K **do**

  $\mathcal{B} \leftarrow \{x_i \sim \mathcal{D}\}_{i=1}^{N}$ // sample a batch of N images

  for $x_i \in \mathcal{B}$ **do**

    $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}'$ // sample image transformations

    $z_1 \leftarrow g_\theta(f_\theta(t(x_i)))$ and $z_2 \leftarrow g_\theta(f_\theta(t'(x_i)))$ // compute projections

    $z'_1 \leftarrow g_\xi(f_\xi(t'(x_i)))$ and $z'_2 \leftarrow g_\xi(f_\xi(t(x_i)))$ // compute target projections

    $l_i \leftarrow -2 \cdot \left(\frac{\langle q_\theta(z_1), z'_1 \rangle}{\|q_\theta(z_1)\|_2 \cdot \|z'_1\|_2} + \frac{\langle q_\theta(z_2), z'_2 \rangle}{\|q_\theta(z_2)\|_2 \cdot \|z'_2\|_2} \right)$ // compute the loss for x_i

  end

  $\delta\theta \leftarrow \frac{1}{N} \sum_{i=1}^{N} \partial_\theta l_i$ // compute the total loss gradient w.r.t. θ

  $\theta \leftarrow \text{optimizer}(\theta, \delta\theta, \eta_k)$ // update online parameters

  $\xi \leftarrow \tau_k \xi + (1 - \tau_k)\theta$ // update target parameters

end

Output : encoder f_θ

Experiments - Linear Evaluation

Method	Top-1	Top-5
Local Agg.	60.2	-
PIRL [32]	63.6	-
CPC v2 [29]	63.8	85.3
CMC [11]	66.2	87.0
SimCLR [8]	69.3	89.0
MoCo v2 [34]	71.1	-
InfoMin Aug. [12]	73.0	91.1
BYOL (ours)	**74.3**	**91.6**

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1	Top-5
SimCLR [8]	ResNet-50 (2×)	94M	74.2	92.0
CMC [11]	ResNet-50 (2×)	94M	70.6	89.7
BYOL (ours)	ResNet-50 (2×)	94M	77.4	93.6
CPC v2 [29]	ResNet-161	305M	71.5	90.1
MoCo [9]	ResNet-50 (4×)	375M	68.6	-
SimCLR [8]	ResNet-50 (4×)	375M	76.5	93.2
BYOL (ours)	ResNet-50 (4×)	375M	78.6	94.2
BYOL (ours)	ResNet-200 (2×)	250M	**79.6**	**94.8**

(b) Other ResNet encoder architectures.

Table 1: Top-1 and top-5 accuracies (in %) under linear evaluation on ImageNet.

Experiments - Semi-supervised

Method	Top-1		Top-5	
	1%	10%	1%	10%
Supervised [64]	25.4	56.4	48.4	80.4
InstDisc	-	-	39.2	77.4
PIRL [32]	-	-	57.2	83.8
SimCLR [8]	48.3	65.6	75.5	87.8
BYOL (ours)	**53.2**	**68.8**	**78.4**	**89.0**

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1		Top-5	
			1%	10%	1%	10%
CPC v2 [29]	ResNet-161	305M	-	-	77.9	91.2
SimCLR [8]	ResNet-50 (2×)	94M	58.5	71.7	83.0	91.2
BYOL (ours)	ResNet-50 (2×)	94M	62.2	73.5	84.1	91.7
SimCLR [8]	ResNet-50 (4×)	375M	63.0	74.4	85.8	92.6
BYOL (ours)	ResNet-50 (4×)	375M	69.1	75.7	87.9	92.5
BYOL (ours)	ResNet-200 (2×)	250M	**71.2**	**77.7**	**89.5**	**93.7**

(b) Other ResNet encoder architectures.

Table 2: Semi-supervised training with a fraction of ImageNet labels.

Experiments - Transfer, Classification

Method	Food101	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
BYOL (ours)	**75.3**	91.3	**78.4**	**57.2**	**62.2**	**67.8**	60.6	82.5	75.5	90.4	94.2	**96.1**
SimCLR (repro)	72.8	90.5	74.4	42.4	60.6	49.3	49.8	81.4	**75.7**	84.6	89.3	92.6
SimCLR [8]	68.4	90.6	71.6	37.4	58.8	50.3	50.3	80.5	74.5	83.6	90.3	91.2
Supervised-IN [8]	72.3	**93.6**	78.3	53.7	61.9	66.7	**61.0**	**82.8**	74.9	**91.5**	**94.5**	94.7
Fine-tuned:												
BYOL (ours)	**88.5**	**97.8**	86.1	**76.3**	63.7	91.6	**88.1**	**85.4**	**76.2**	91.7	**93.8**	97.0
SimCLR (repro)	87.5	97.4	85.3	75.0	63.9	91.4	87.6	84.5	75.4	89.4	91.7	96.6
SimCLR [8]	88.2	97.7	85.9	75.9	63.5	91.3	88.1	84.1	73.2	89.2	92.1	97.0
Supervised-IN [8]	88.3	97.5	**86.4**	75.8	**64.3**	**92.1**	86.0	85.0	74.6	**92.1**	93.3	**97.6**
Random init [8]	86.9	95.9	80.2	76.1	53.6	91.4	85.9	67.3	64.8	81.5	72.6	92.0

Table 3: Transfer learning results from ImageNet (IN) with the standard ResNet-50 architecture.

Experiments - Transfer, Others

Method	AP_{50}	mIoU
Supervised-IN [9]	74.4	74.4
MoCo [9]	74.9	72.5
SimCLR (repro)	75.2	75.2
BYOL (ours)	**77.5**	**76.3**

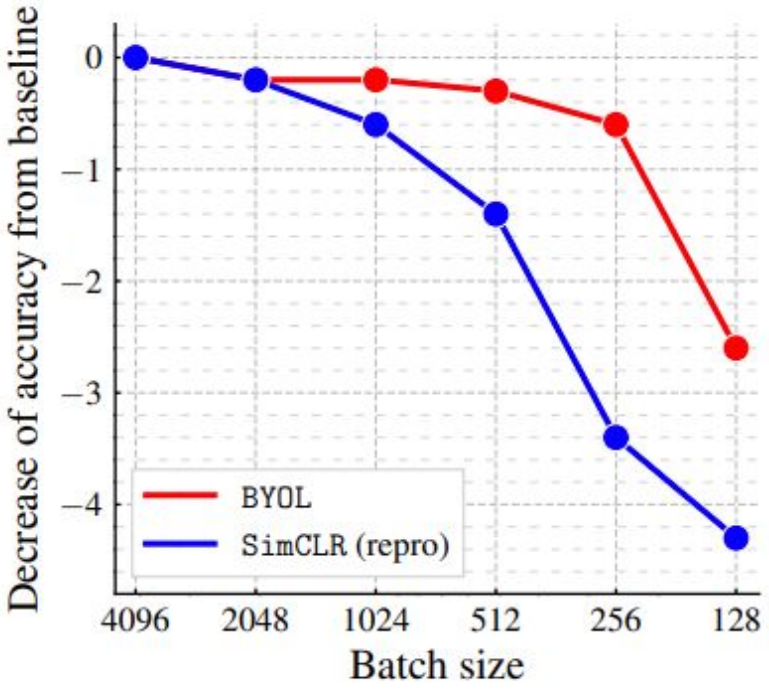
(a) Transfer results in semantic segmentation and object detection.

Method	Higher better			Lower better	
	pct.< 1.25	pct.$< 1.25^2$	pct.$< 1.25^3$	rms	rel
Supervised-IN [70]	81.1	95.3	98.8	0.573	**0.127**
SimCLR (repro)	83.3	96.5	99.1	0.557	0.134
BYOL (ours)	**84.6**	**96.7**	**99.1**	**0.541**	0.129

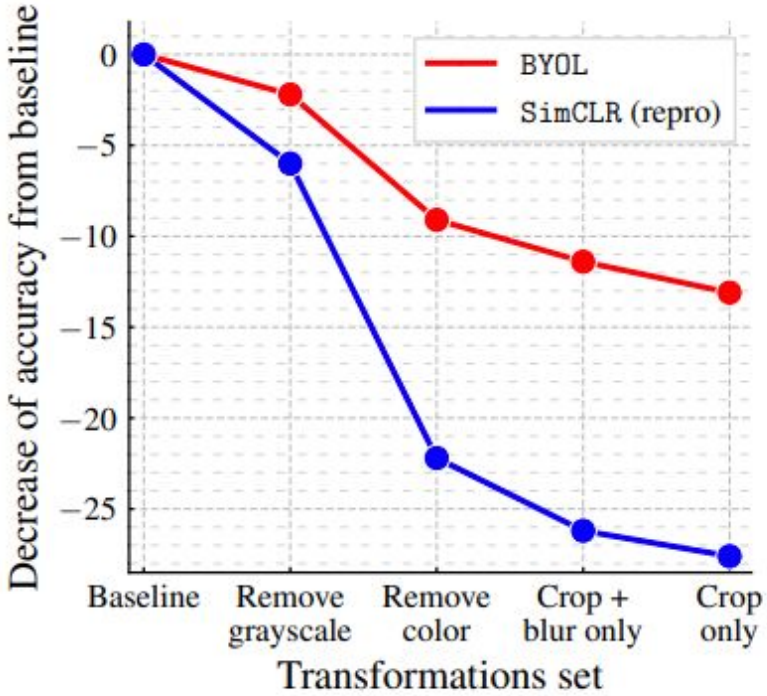
(b) Transfer results on NYU v2 depth estimation.

Table 4: Results on transferring BYOL's representation to other vision tasks.

Experiments - Batch, Transformation



(a) Impact of batch size



(b) Impact of progressively removing transformations

Experiments - Hyperparameters

Target	τ_{base}	Top-1
Constant random network	1	$18.8_{\pm 0.7}$
Moving average of online	0.999	69.8
Moving average of online	0.99	**72.5**
Moving average of online	0.9	68.4
Stop gradient of online[†]	0	0.3

(a) Results for different target modes. [†]In the *stop gradient of online*, $\tau = \tau_{\text{base}} = 0$ is kept constant throughout training.

Method	Predictor	Target network	β	Top-1
BYOL	✓	✓	0	**72.5**
	✓	✓	1	70.9
		✓	1	70.7
SimCLR			1	69.4
	✓		1	69.1
	✓		0	0.3
		✓	0	0.2
			0	0.1

(b) Intermediate variants between BYOL and SimCLR.

Table 5: Ablations with top-1 accuracy (in %) at 300 epochs under linear evaluation on ImageNet.

Appendix. InfoNCE

$$S_\theta(u_1, u_2) \triangleq \frac{\langle \phi(u_1), \psi(u_2) \rangle}{\|\phi(u_1)\|_2 \cdot \|\psi(u_2)\|_2}.$$

$$\text{InfoNCE}_\theta \triangleq \frac{2}{B} \sum_{i=1}^{B} S_\theta(v_i, v_i') - \beta \cdot \frac{2\alpha}{B} \sum_{i=1}^{B} \ln \left(\sum_{j \neq i} \exp \frac{S_\theta(v_i, v_j)}{\alpha} + \sum_{j} \exp \frac{S_\theta(v_i, v_j')}{\alpha} \right),$$

- NCE : Noise-contrastive estimation: A new estimation principle for unnormalized statistical models (jmlr 2010)
- Learning word embeddings efficiently with noise-contrastive estimation (NIPS 2013)

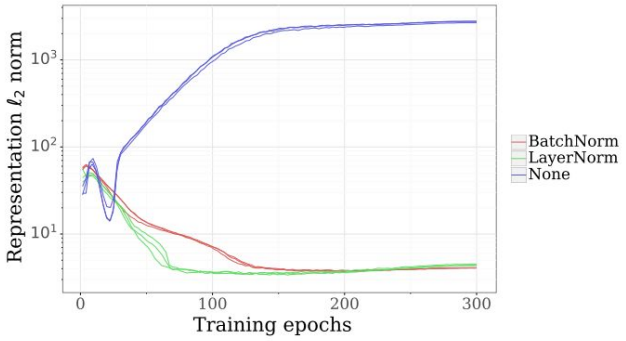
Experiments - Normalization

Normalization	Top-1	Top-5
ℓ_2-norm	**72.5**	**90.8**
LayerNorm	72.5±0.4	90.1
No normalization	67.4	87.1
BatchNorm	65.3	85.3

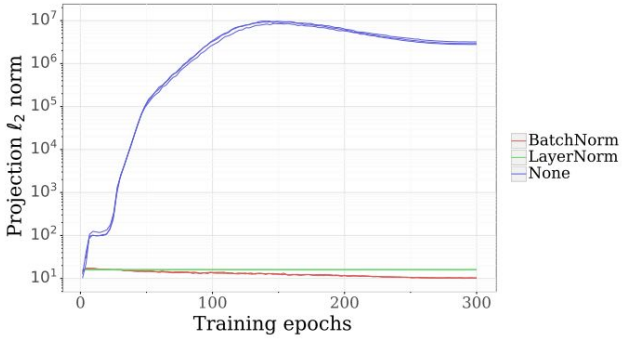
$$n^j_{\mathrm{BN}i} : x \to \frac{x^j_i - \mu^j_{\mathrm{BN}}(x)}{\sigma^j_{\mathrm{BN}}(x) \cdot \sqrt{d}}, \quad n^j_{\mathrm{LN}i} : x \to \frac{x^j_i - \mu_{\mathrm{LN}i}(x)}{\sigma_{\mathrm{LN}i}(x) \cdot \sqrt{d}}, \quad n_{\mathrm{ID}} : x \to x,$$

$$\mu^j_{\mathrm{BN}} : x \to \frac{1}{B}\sum_{i=1}^{B} x^j_i, \quad \sigma^j_{\mathrm{BN}} : x \to \sqrt{\frac{1}{B}\sum_{i=1}^{B}\left(x^j_i\right)^2 - \mu^j_{\mathrm{BN}}(x)^2},$$

$$\mu_{\mathrm{LN}i} : x \to \frac{1}{d}\sum_{j=1}^{d} x^j_i, \quad \sigma_{\mathrm{LN}i} : x \to \frac{\|x_i - \mu_{\mathrm{LN}i}(x)\|_2}{\sqrt{d}}$$



(a) Representation ℓ_2-norm



(b) Projection ℓ_2-norm

Figure 6: Effect of normalization on the ℓ_2 norm of network outputs.