Exploring Simple Siamese Representation Learning (and Beyond)

Xinlei Chen

Kaiming He

facebook
Artificial Intelligence Research

Self-/Unsupervised Representation Learning

- **Goal**: train useful representations from *unlabeled* data
- **Trending** topic

NLP

Revolutionized

Vision

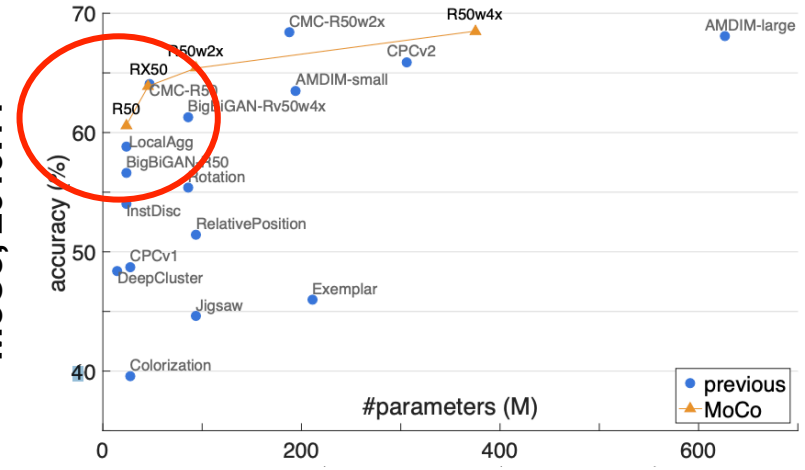
Getting there?

ImageNet, Supervised	76.5*
Unsupervised, 2019	65-
Unsupervised, 2020	75+

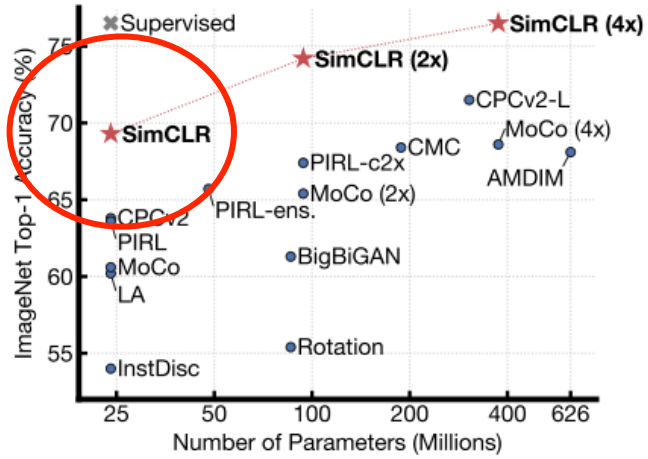
*: Relatively weak baseline, could be higher

What Happened: Many New Frameworks

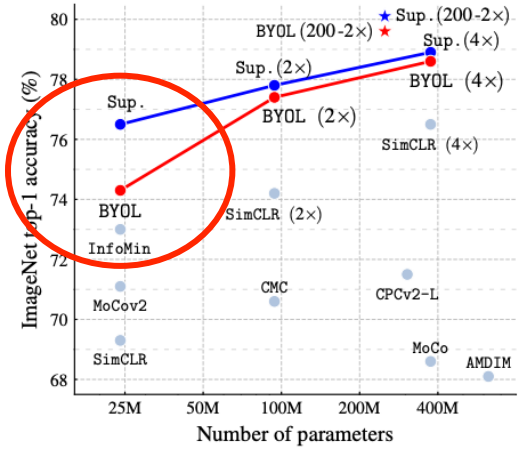
MoCo, 2019.11



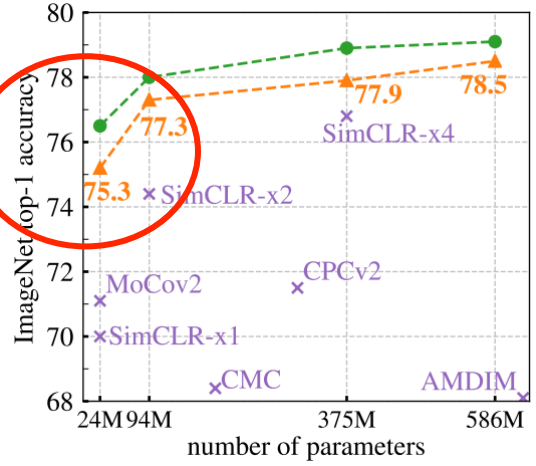
SimCLR, 2020.2



BYOL, 2020.6

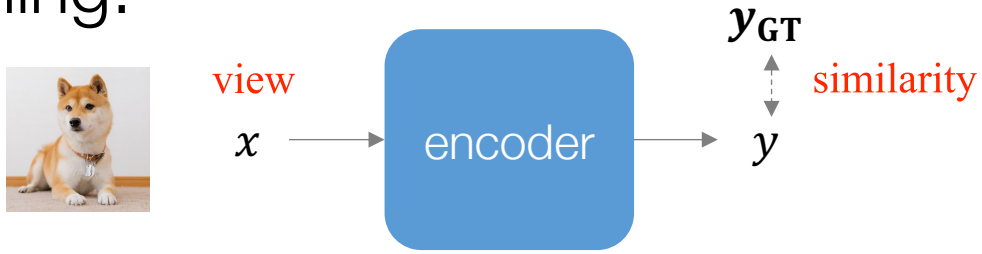


SwAV, 2020.6

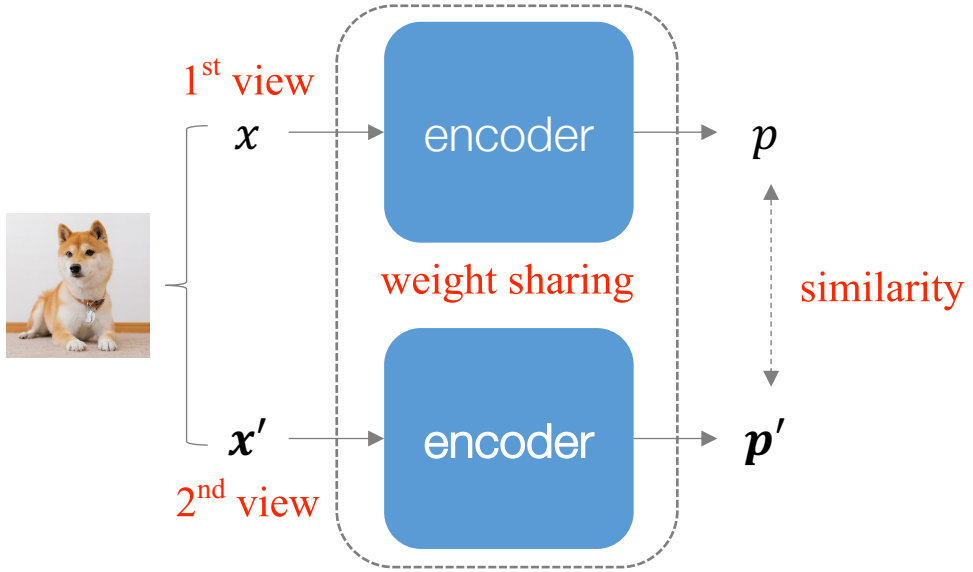


Common Theme: Siamese Networks

- Supervised learning:

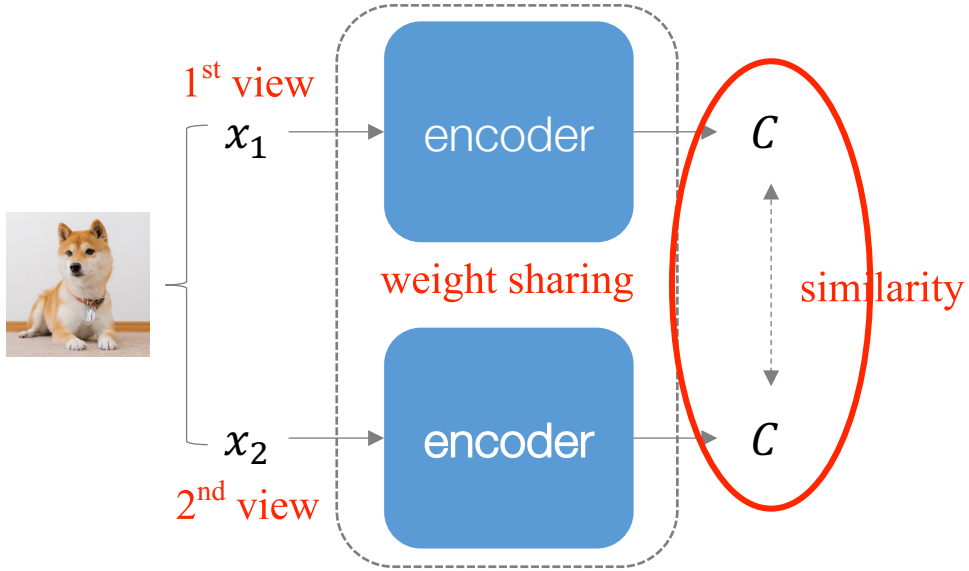


- *(these)* Self-supervised learning:



Well, Not Quite..

- Undesired *trivial* solution exist:
  - Predicting constant (C) for everything, representation collapses



- Countering strategies?

Contrastive Learning

- Explicitly requires **dissimilarity** for views from different images
  - Still requires similarity for views from
  - So, predicting constant is no longer optimal

- Popular loss function:
  - InfoNCE
    - $-\log \frac{\exp(p \cdot p'/\tau)}{\exp(p \cdot p'/\tau) + \sum_{n \in \mathcal{N}} \exp(p \cdot n/\tau)}$
    - $\mathcal{N}$ is the set of views from other images as negatives
    - τ is a temperature parameter

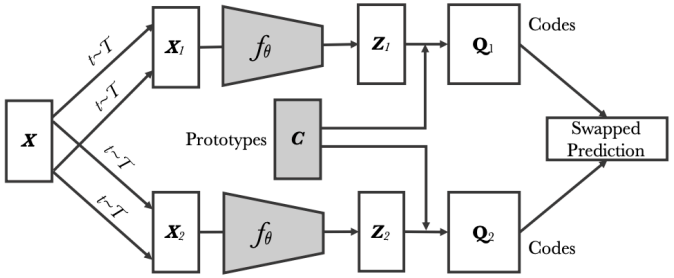
Contrastive Learning

- Drawback of InfoNCE:
  - Usually requires a sufficiently large # of negatives for good performance

- In practice:
  - SimCLR uses a large batch size (4096) to provide negatives within batch
    - Requires multi-node training (>>8 V100 GPUs)

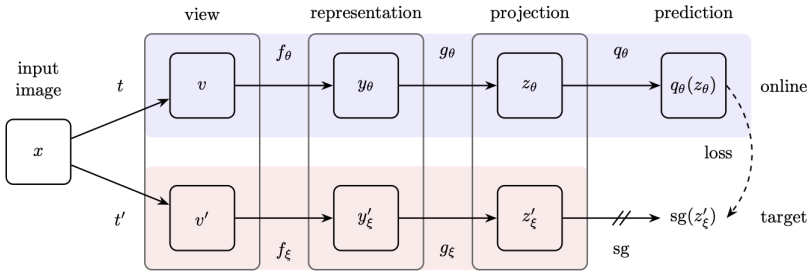
  - MoCo uses a momentum queue to store negatives
    - It *decouples* batch size from negative set size
    - Additional memory overhead, and implementation complexity

Other strategies



- **Balanced** online clustering (SwAV)
  - A cluster-center based output representation, p is used to pick center
  - Key: making sure that cluster sizes are balanced (Sinkhorn-Knopp)
    - Constant solution is less likely because otherwise all points are assigned to a singular cluster

- BYOL
  - Introduces an additional MLP (*predictor*), and uses *momentum encoder*
    - Momentum encoder
      - Exponential moving average (EMA) of base encoder weights
      - So, weights are *not* updated by gradients
      - But need to maintain *two copies* of weights



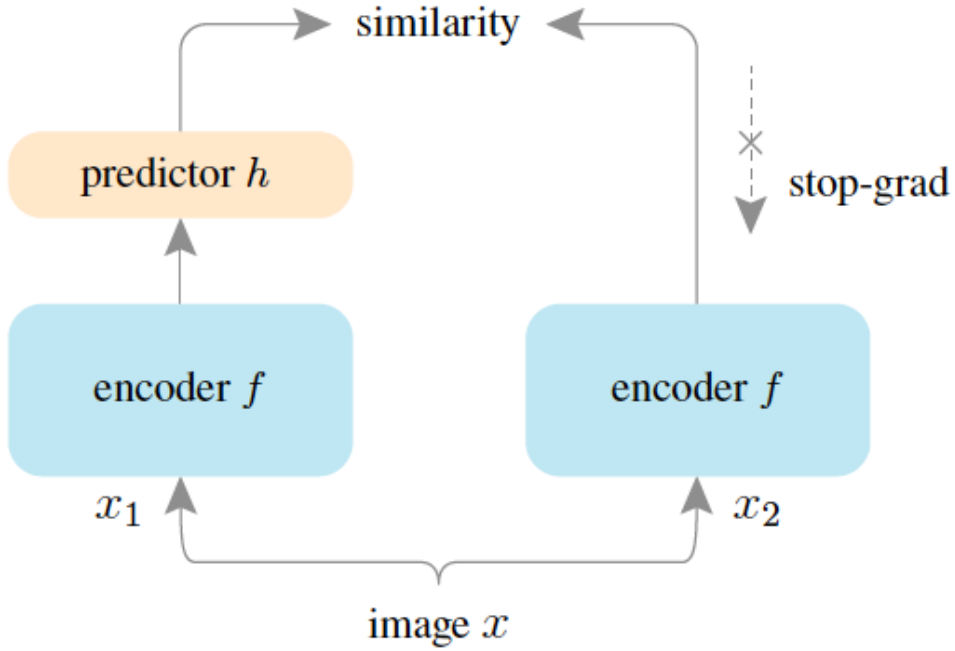
All These are Rich & Fancy..

Can a Simple Siamese Network just Work?

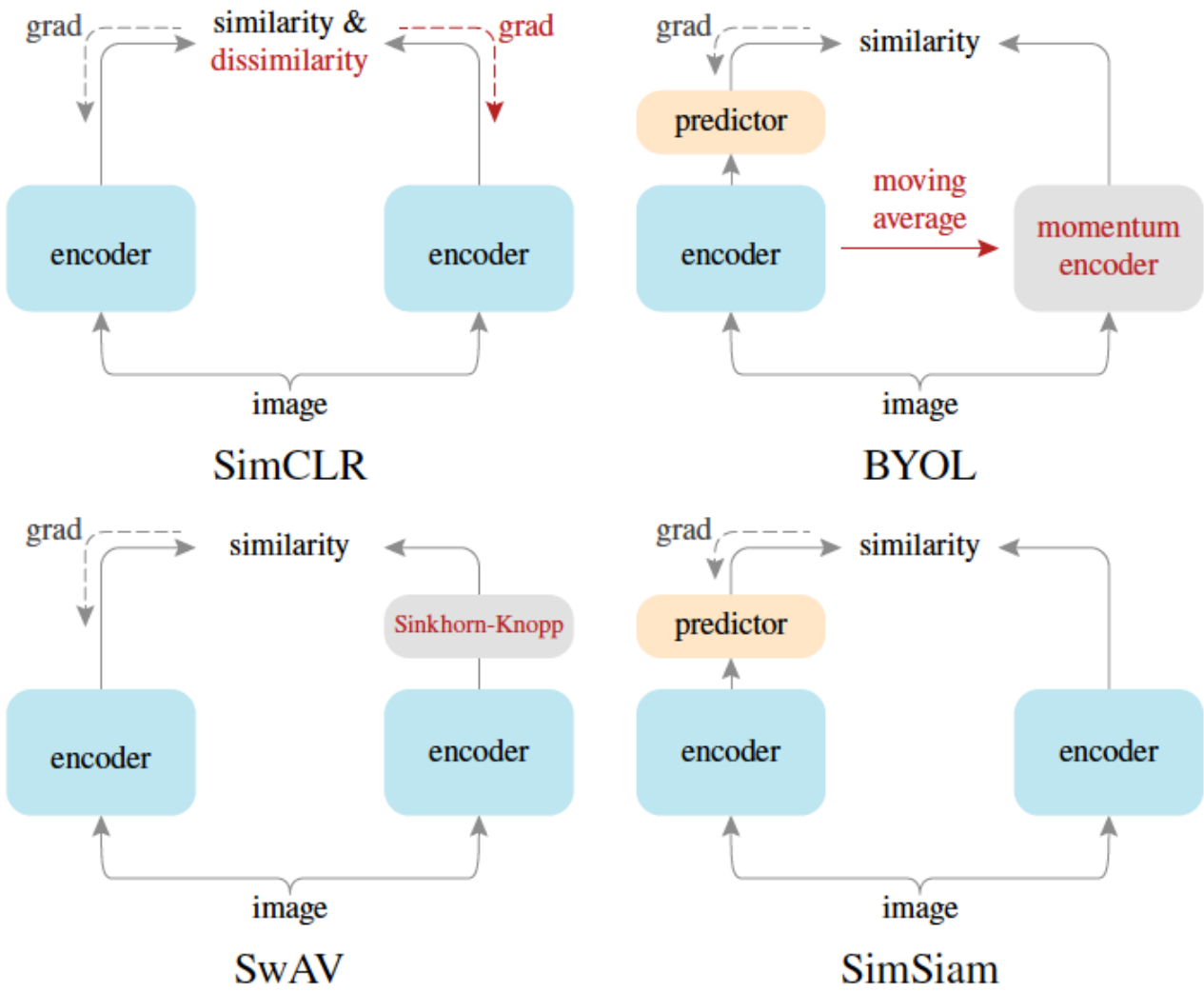
Yes, SimSiam!

- We show it indeed can and propose SimSiam

- Cuts core components from existing frameworks
  - SimCLR *w/o* negatives
  - SwAV *w/o* online clustering
  - BYOL *w/o* momentum encoder
  - MoCo *w/o* negatives or momentum encoder



Summary of Different Siamese Architectures



PyTorch-like Code for SimSiam

Algorithm 1 SimSiam Pseudocode, PyTorch-like

```
# f: backbone + projection mlp
# h: prediction mlp

for x in loader: # load a minibatch x with n samples
   x1, x2 = aug(x), aug(x) # random augmentation
   z1, z2 = f(x1), f(x2) # projections, n-by-d
   p1, p2 = h(z1), h(z2) # predictions, n-by-d

   L = D(p1, z2)/2 + D(p2, z1)/2 # loss

   L.backward() # back-propagate
   update(f, h) # SGD update

def D(p, z): # negative cosine similarity
   z = z.detach() # stop gradient

   p = normalize(p, dim=1) # l2-normalize
   z = normalize(z, dim=1) # l2-normalize
   return -(p*z).sum(dim=1).mean()
```

- Notes:
  - We use l_2 normalized cosine similarity by default
  - *Symmetrized* loss
  - Gradient is only back propagated through *predictor*
    - Stop-grad on other

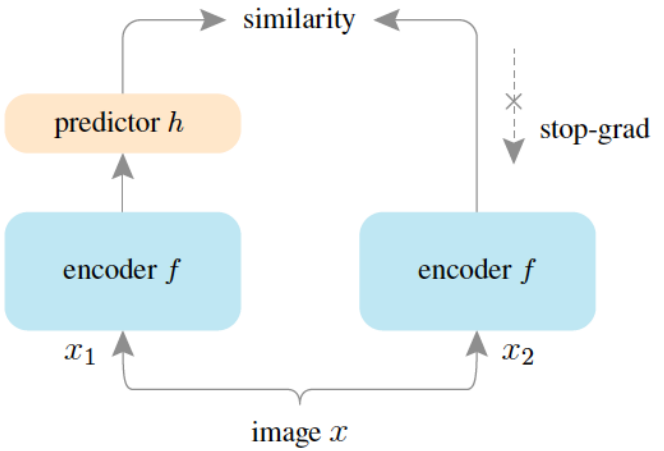
Empirical Study

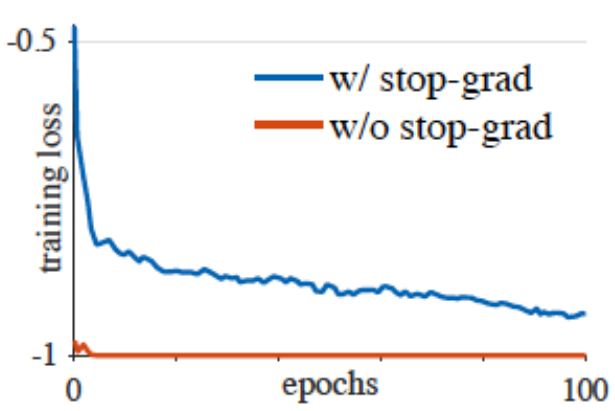
- Baseline settings:
  - ResNet-50 + 3-layer projector MLP as the default encoder
    - Projector MLP is a very helpful trick from SimCLR
    - Sync BatchNorm
  - Predictor MLP:
    - Bottleneck structure, with smaller hidden dimension (512) and larger input/output dimensions
  - Pre-training: SGD + momentum as the default optimizer
    - 512 batch size, fit in 8 GPUs
    - 0.05 base learning rate (follow linear scaling rule with base batch size 256)
    - 100-epoch pre-training, for analysis
  - Evaluation: linear classifier on top of frozen ResNet pool-5 features

Stop-Grad is Crucial for SimSiam

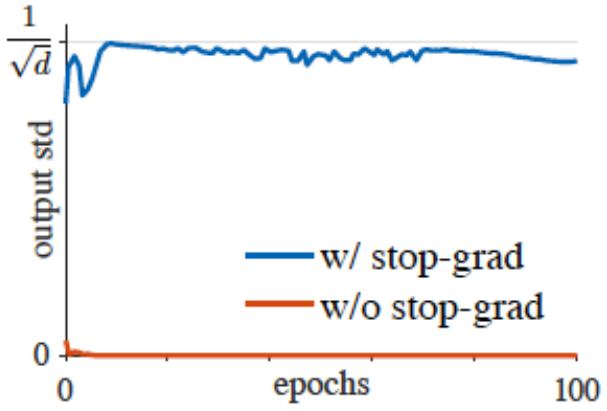
- Without it, representation collapses
  - *Implicit* for momentum encoder

setting	top-1
w/ stop-grad	67.7±0.1
w/o stop-grad	0.1

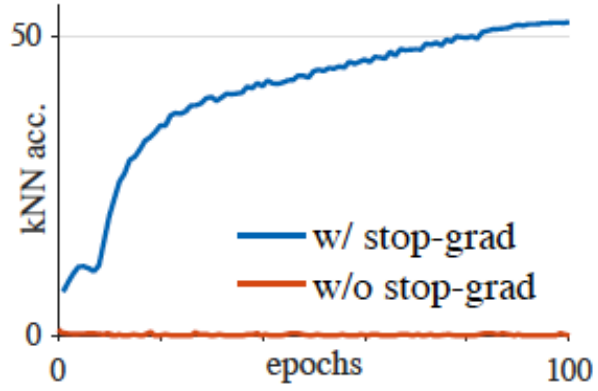




loss curve



monitor 1: std of p

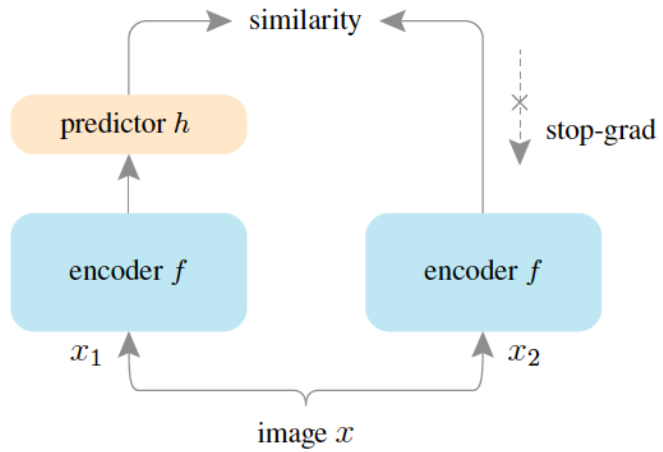


monitor 2: KNN classifier

Predictor is Important

- Tried different settings:

setting	top-1	
previous default	67.7	
w/o predictor	0.1	← effectively w/o stop-grad
random predictor	1.5	← does not converge
not decay predictor lr	68.1	← **default** for later explorations

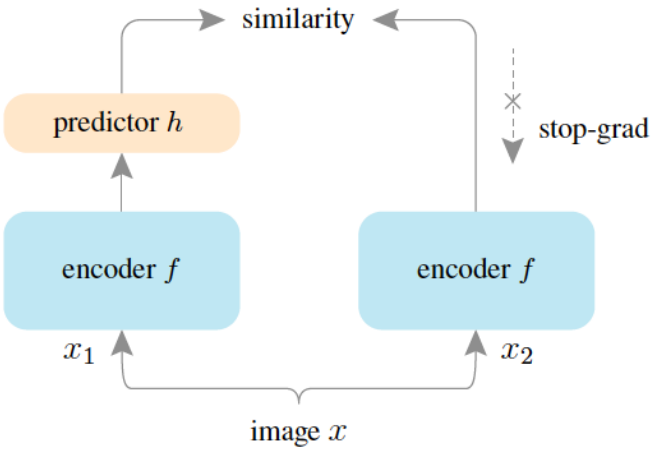


- Predictor **can** be removed and maintain reasonable performance with proper designs

Losses

- Cosine vs. soft-max cross-entropy
  - Can work out-of-box
  - Relates to SwAV that employs a similar loss

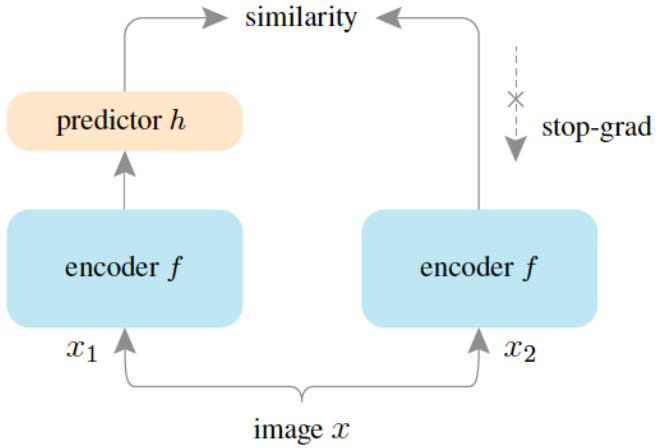
- Symmetrized vs. not
  - Symmetrized is better
    - Likely because it trains “longer”
  - SimSiam has advantage over BYOL:
    - Does not need to forward again on the momentum encoder



setting	top-1
cosine	68.1
cross-entropy	63.2

setting	top-1
symmetrized	68.1
asymmetric	64.8
asymmetric, 2x	67.3

Batch Normalization



- Batch normalization is required for SimSiam
  - SyncBN on each view *separately*
  - Weight decay applied to BN parameters (different from BYOL, SimCLR)

- Analysis of BN on MLPs

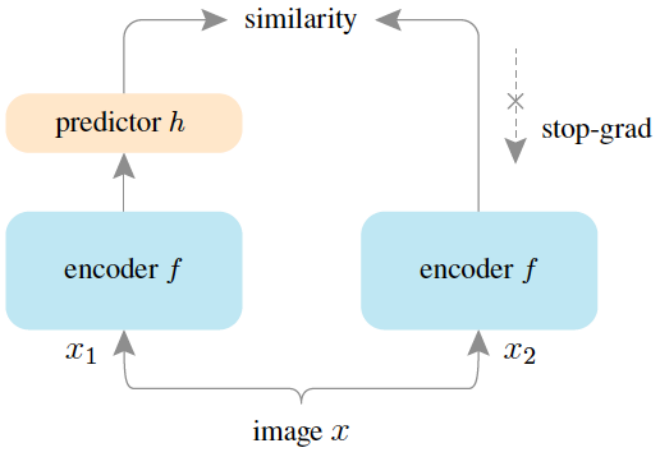
case	proj. hidden	proj. output	pred. hidden	pred. output	top-1
none					34.6
hidden-only	✅		✅		67.4
default	✅	✅	✅		68.1
all	✅	✅	✅	✅	unstable

Analysis on Other Basic Settings

- Batch size
  - Linearly scaled learning rates

	64	128	256	512	1024	2048	4096
top-1	66.1	67.3	68.1	68.1	68.0	67.9	64.0

- Learning rate & weight decay:
  - Again, relatively robust



The Role of Stop-Grad

- Hypothesis

  - Provides a different trajectory that alternates between optimizing *two* sets of variables:
    - θ, network parameters
    - η, hidden representation for an image x, indexed by x

  - Objective function:
    - $L(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}} \left[\left\| \mathcal{F}_\theta\left(\mathcal{T}(x)\right) - \eta_x \right\|_2^2 \right]$
    - $\mathcal{T}$ stands for transformations, or augmentations to the input image

The Role of Stop-Grad

- Optimization for $L(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}}\left[\left\|\mathcal{F}_\theta\left(\mathcal{T}(x)\right) - \eta_x\right\|_2^2\right]$
  - General alternative optimization:
    - Fix η, θ can be optimized with normal gradient decent
    - Fix θ, η can be updated with the expectation $\mathbb{E}_{\mathcal{T}}\left[\mathcal{F}_\theta\left(\mathcal{T}(x)\right)\right]$ over transformations
  - SimSiam: One-step alternation:
    - θ is updated with one-step of gradient compute
    - η is updated with *one sample* of $\mathcal{T}$ only $\mathcal{F}_\theta\left(\mathcal{T}(x)\right)$
- Hypothesis of the predictor
  - Fill in the **gap** between single-sample and expectation

Proof-of-Concept 1

- Multi-step alternation:
  - Update θ multiple times (with SGD) before updating η again

	1-step	10-step	100-step	1-epoch
top-1	68.1	68.7	68.9	67.0

  - Has a “momentum encoder” effect that uses predictions from previous weights

  - Suggest alternating optimization is a valid formulation

Proof-of-Concept 2

- Remove predictor
  - Replace it with a *moving average* of previous $\mathcal{F}_\theta(\mathcal{T}(x))$
  - This is to approximate the expectation $\mathbb{E}_\mathcal{T}[\mathcal{F}_\theta(\mathcal{T}(x))]$

setting	top-1
default, w/ predictor	68.1
w/o predictor	0.1
w/o predictor, w/ moving average	55.0

  - Supportive of the hypothesis that predictor is related to expectations

Comparisons to Others, ImageNet

method	batch size	negative pairs	momentum encoder	100-ep	200-ep	400-ep	800-ep
SimCLR	4096	✅		66.5	68.3	69.8	70.4
MoCo	256	✅	✅	67.4	69.9	71.0	72.2
BYOL	4096		✅	66.5	70.6	73.2	74.3
SwAV	4096			66.5	69.1	70.7	71.8
SimSiam	256			68.1	70.0	70.8	71.3

- SimSiam is batch size friendly, momentum encoder free, and competitive

Comparisons to Others, VOC Detection

Pre-train	AP50	AP75	AP
Supervised	74.4	42.4	42.7
SimCLR	75.9	46.8	50.1
MoCo	77.1	48.5	52.5
BYOL	77.1	47.0	49.9
SwAV	75.5	46.5	49.6
SimSiam (Optimal)	77.3	48.5	52.5

- All methods generally perform well, and *outperform* ImageNet supervised pre-training

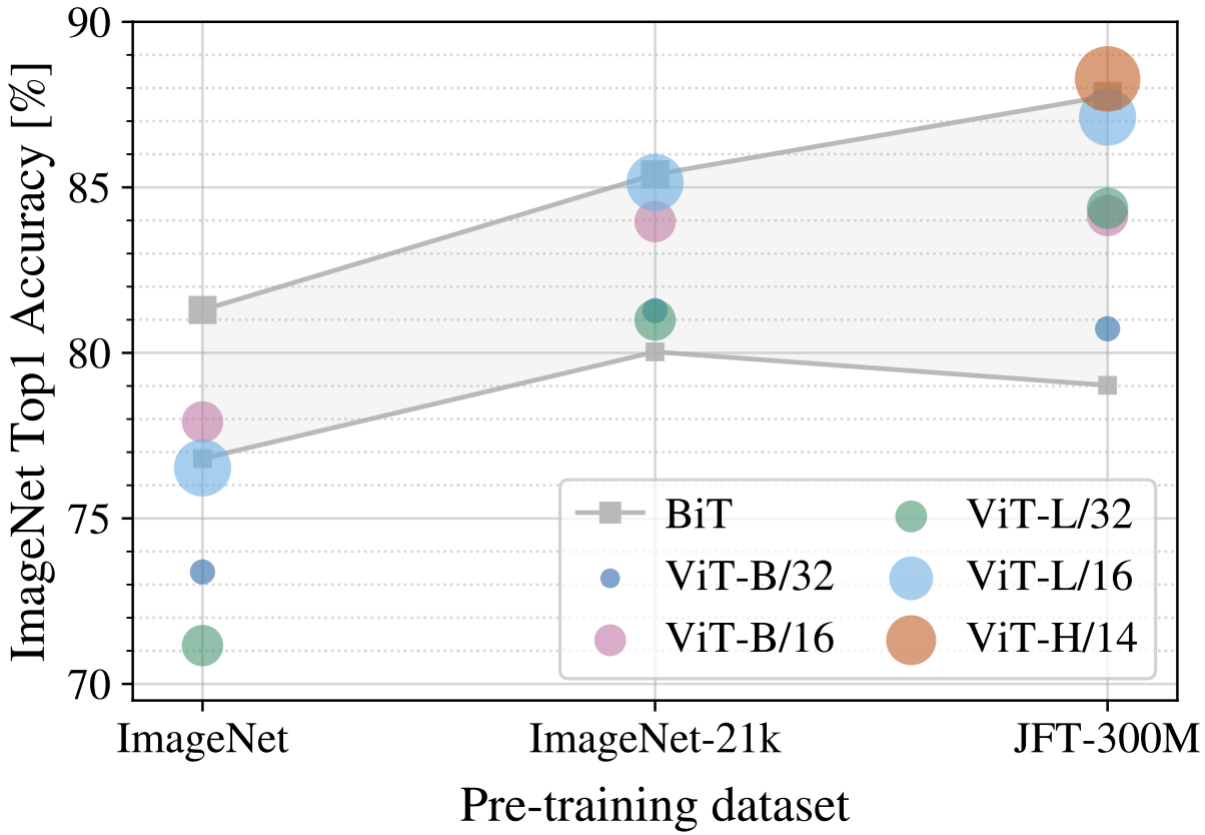
Are Siamese Networks the Bare Minimum?

- Siamese network is a natural and effective tool to learn invariance
  - It means two views of the same concept should *learn* to produce the same output with *data-driven* pre-training
  - While easy invariance like "*translation*" can be baked into "*convolutions*" as inductive biases, more complex transformations (e.g., color, scale, rotation) are harder to design the counterparts
  - If such invariance is an integral part of good visual representations, Siamese networks at least serves as a strong baseline

The Beyond: Vision Transformer + SSL

Saining Xie

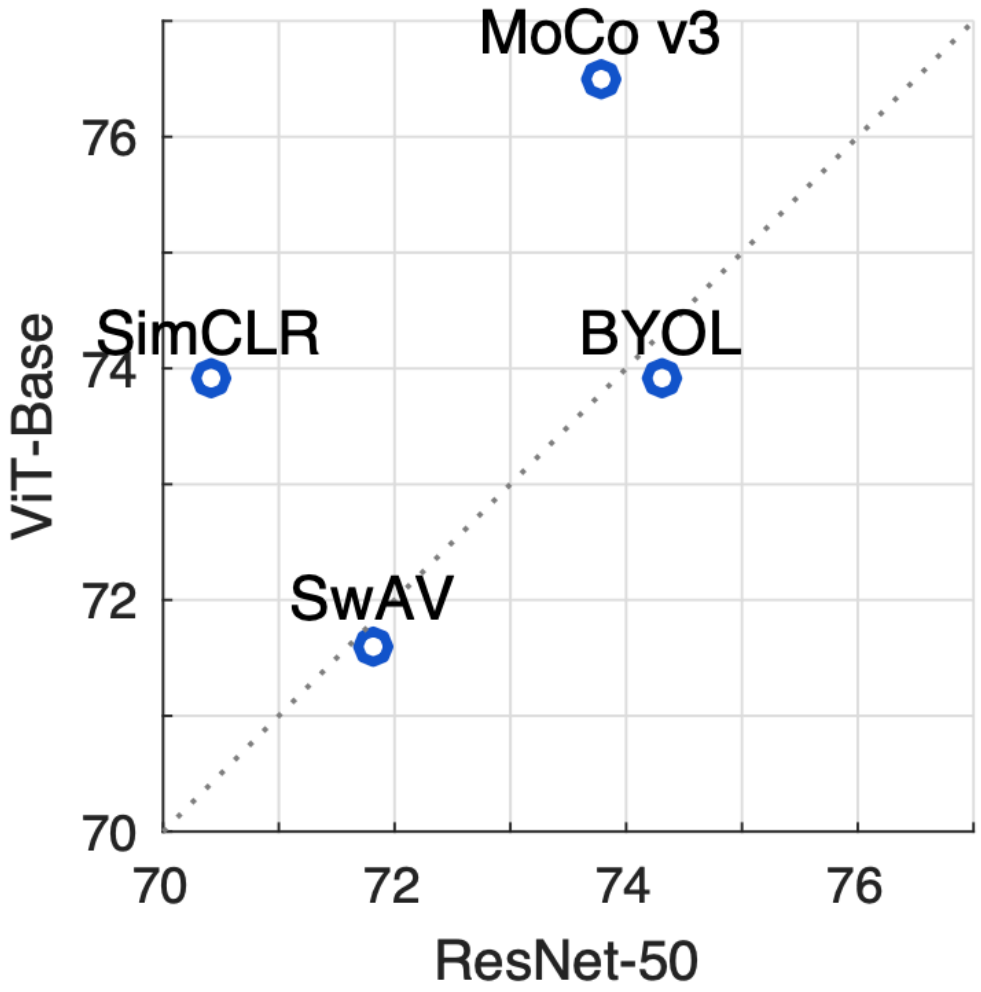
- Vision Transformer (ViT)
  - Inductive biases are less important, or even hurting given enough data
  - Translation invariance is still inherent with convolution-based encoders, which can be learned via Siamese networks
- Natural to explore recent self-supervised learning frameworks with ViT-based encoders



ViT Observation 1

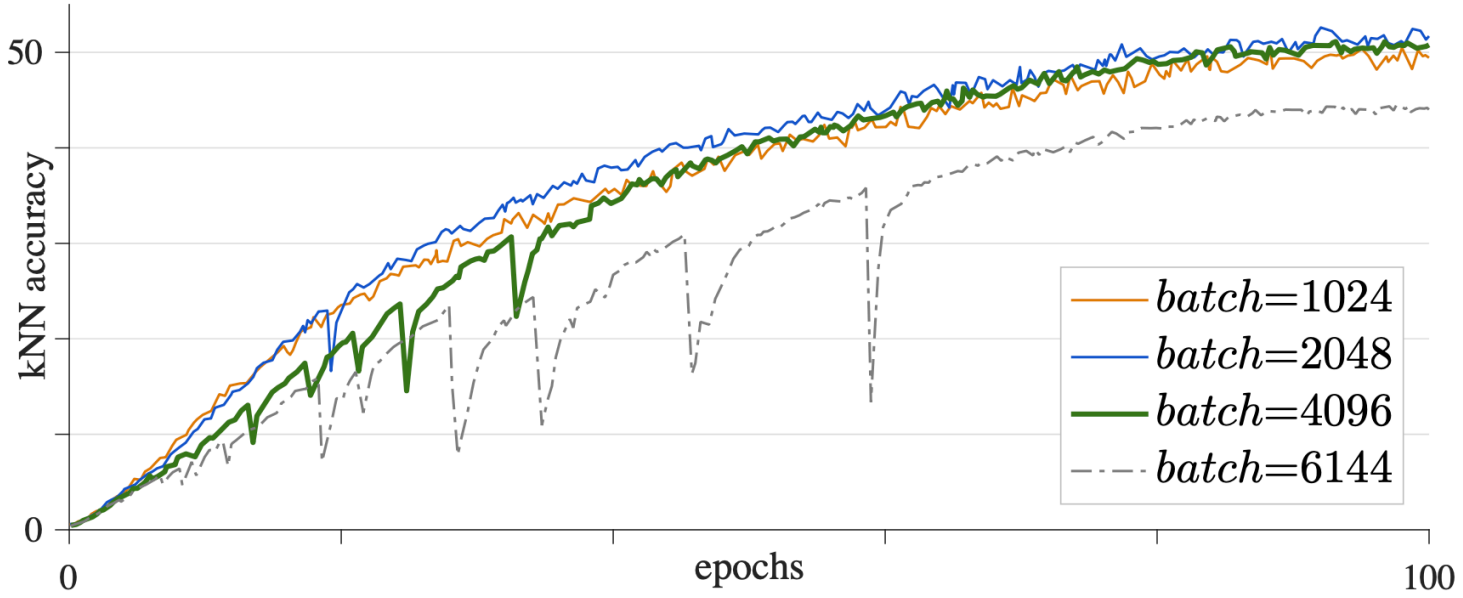
- Existing SSL frameworks generally transfer **well** to ViT and yield reasonable results

- However, they behave differently in different backbones
  - Contrastive learning-based methods have an edge on ViT



ViT Observation 2

- Large batch size, large lr training is more challenging for ViT
  - "Dips": instability influences training
  - Indicating training is only "partially" successful, and "partially" failed
  - LAMB does not fix the issue



KNN monitor curve for MoCo + ViT; AdamW optimizer

Thanks! Questions?

- Take-aways
  - Simple Siamese networks can work alone without (1) negatives; (2) large batches; (3) momentum encoders
  - With current SimSiam design, stop-gradient operation is crucial, suggesting an underlying alternative optimization trajectory
  - Siamese networks are a general and powerful tool to learn invariance with minimum inductive bias, and transfer well to other backbones