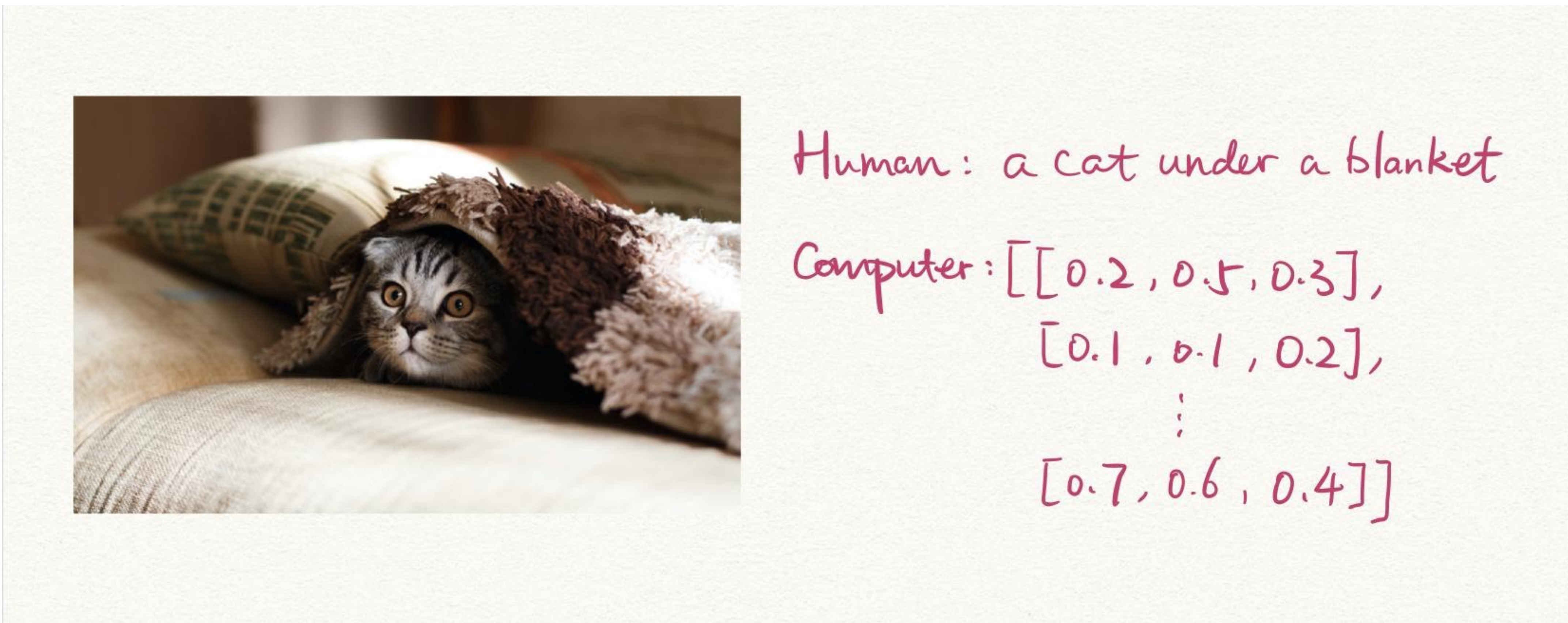
Self-supervised representation learning

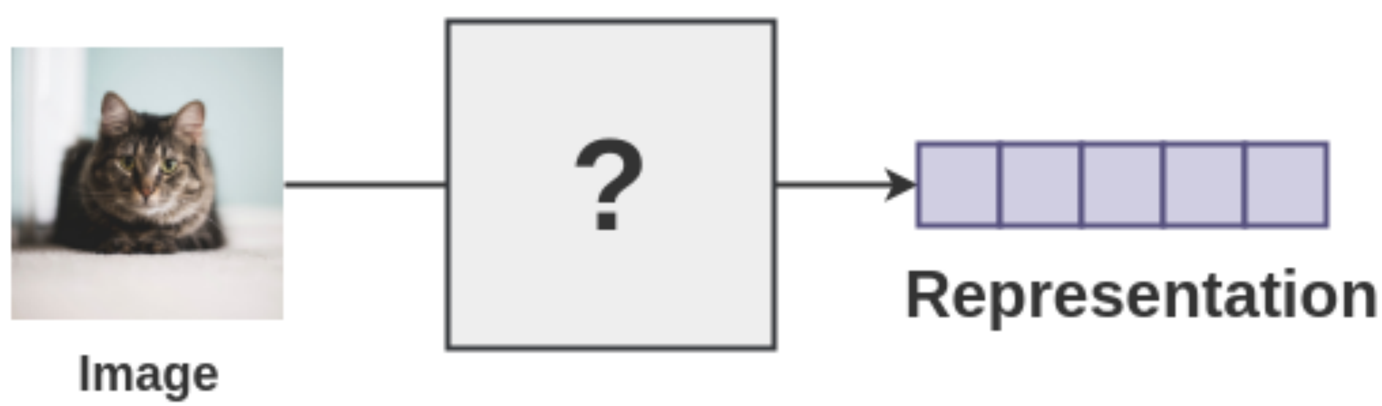
Part II: BYOL and SimSiam

I. Introduction

Self-supervised representation learning

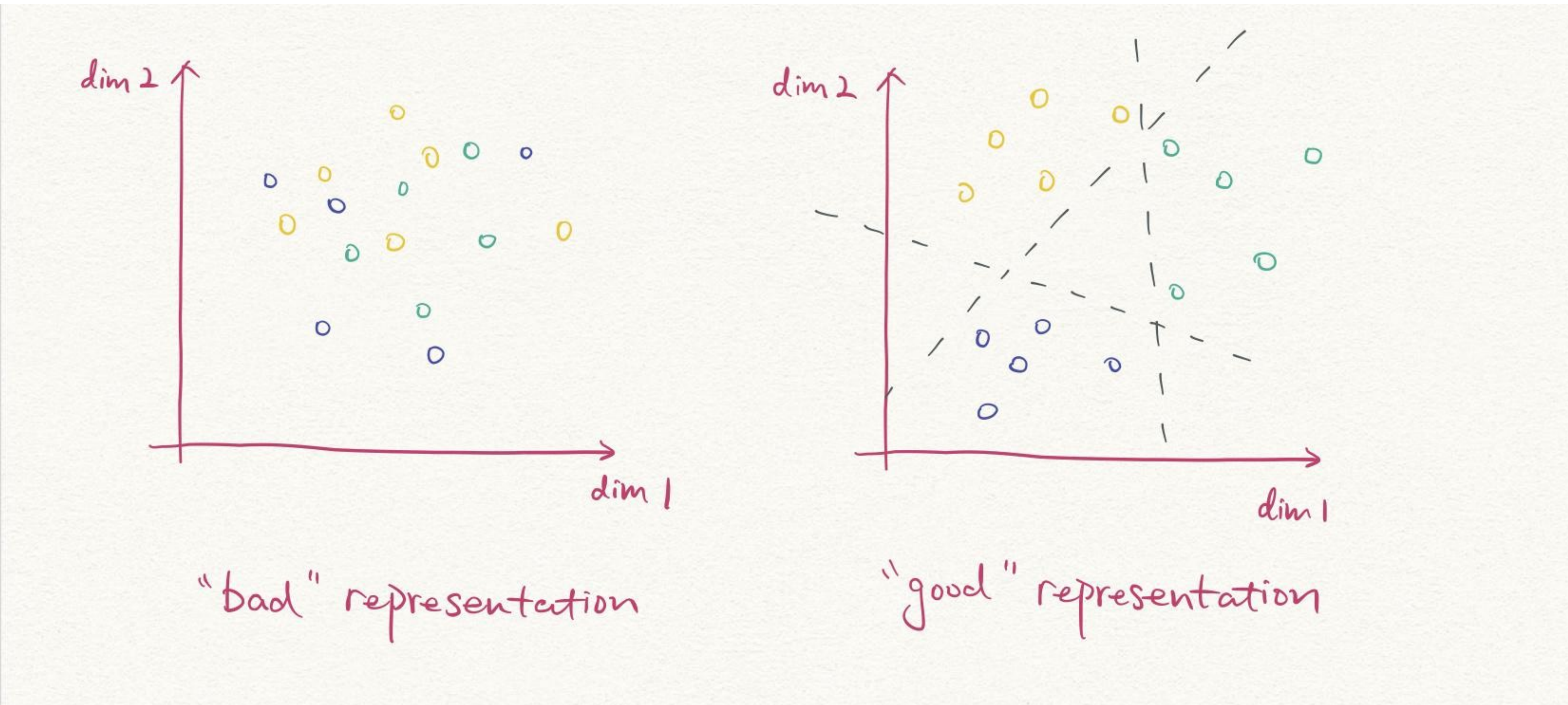
[1]



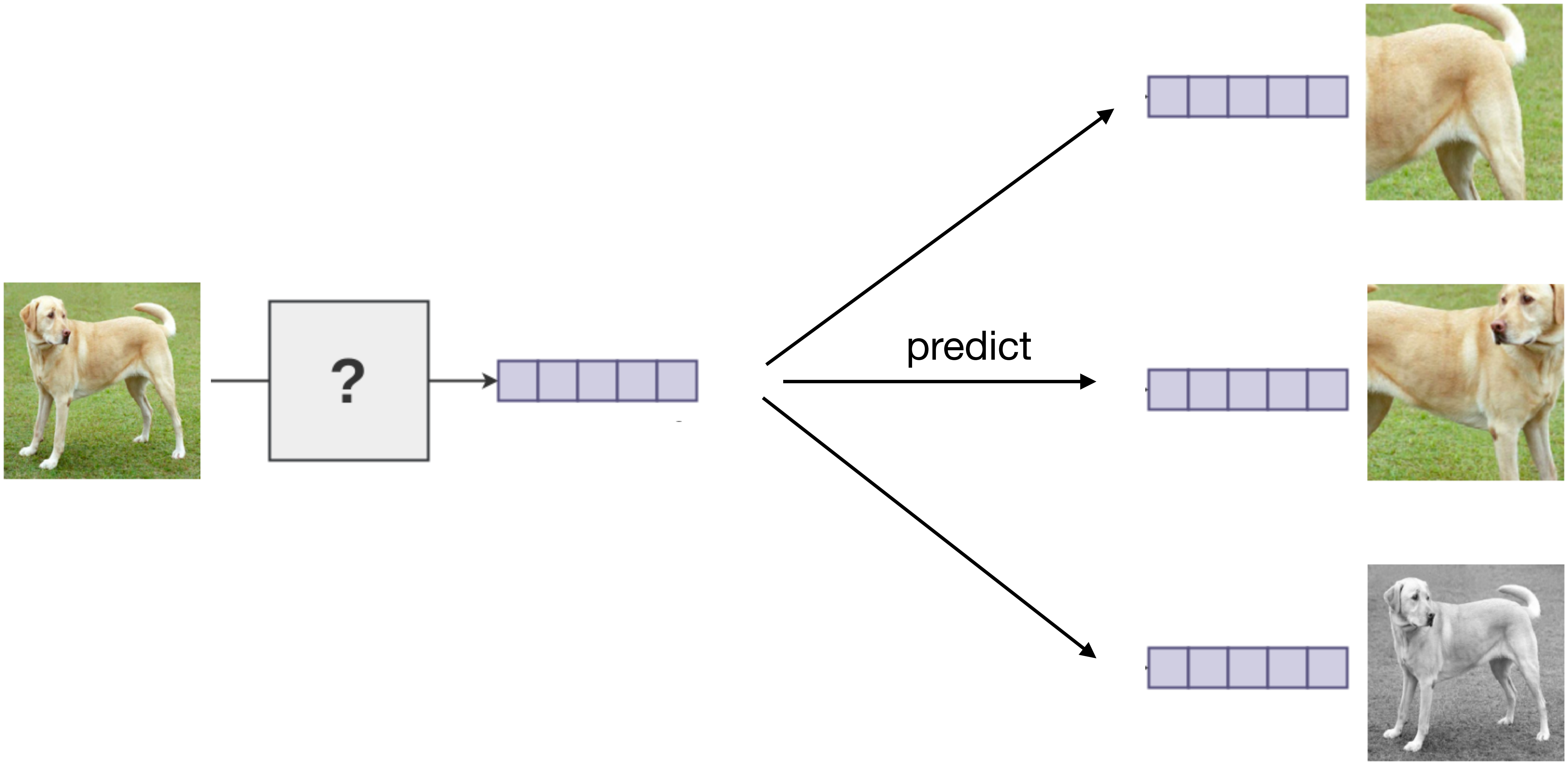


[1]: figures in introduction from ml.berkeley.edu/blog/posts/contrastive_learning/ *and* https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607

Self-supervised representation learning

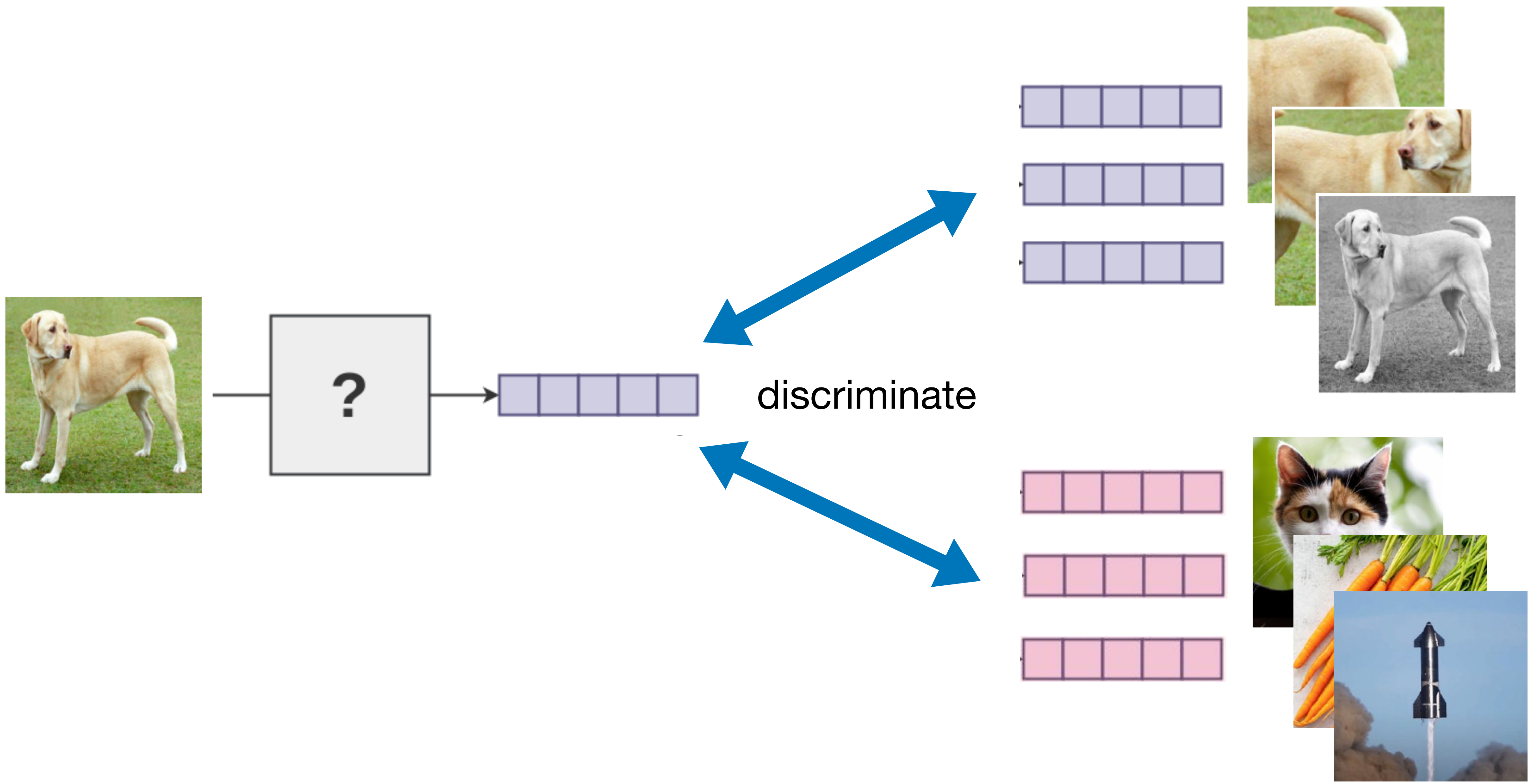


Cross-view prediction framework



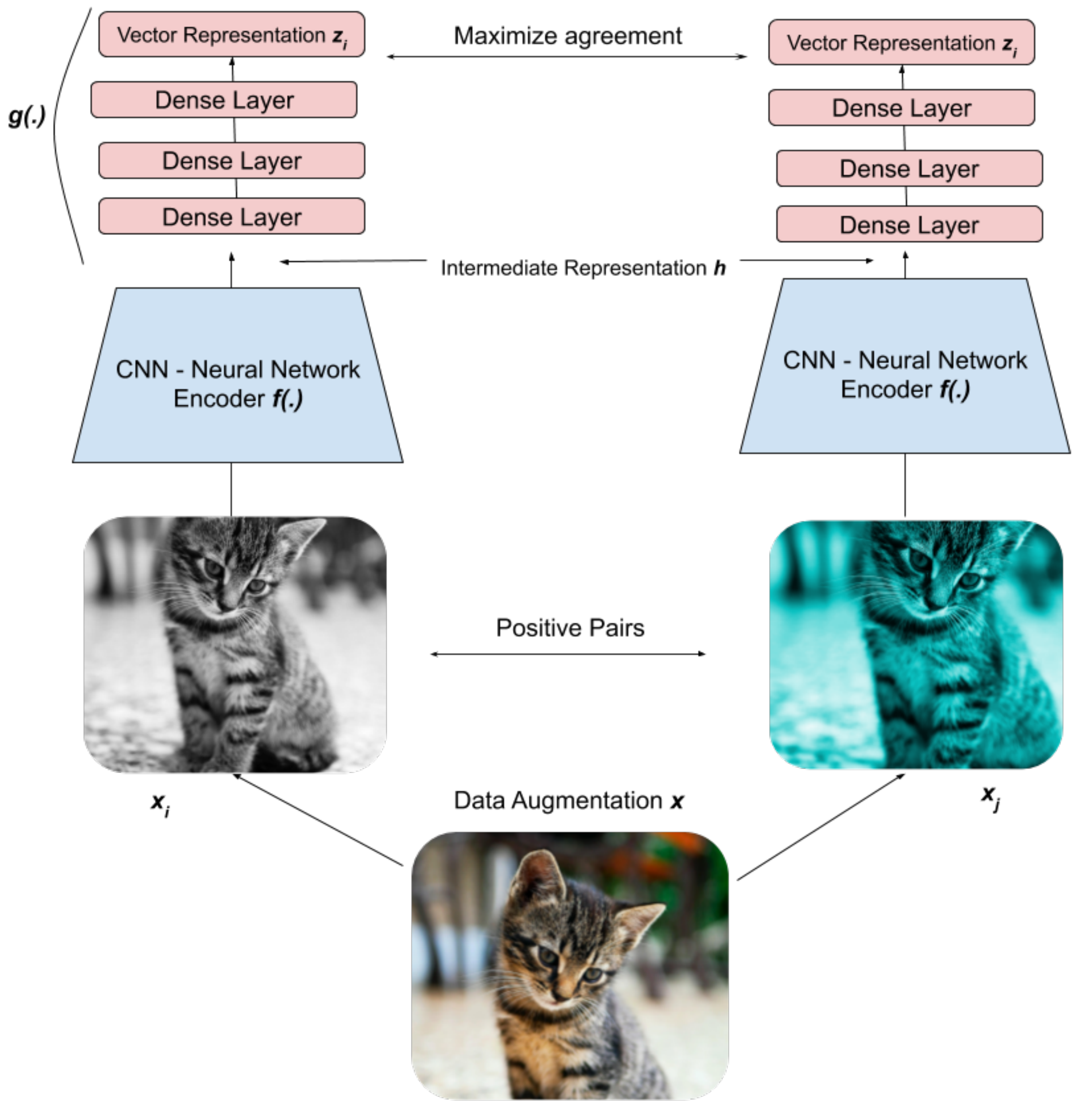
Collapse: constant representation across views is always predictive of itself!

Contrastive learning framework

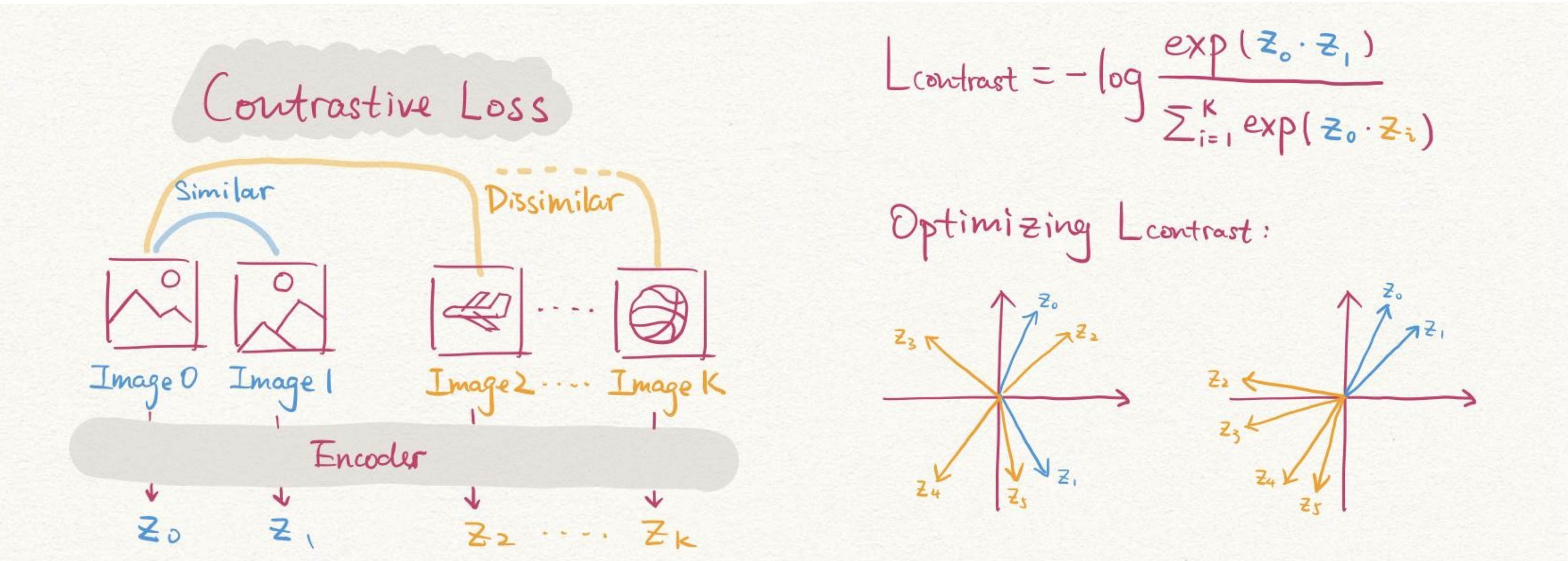


- **Avoids collapse**
- **Needs lots of challenging negative examples**

Contrastive learning



Contrastive learning



Are negative examples necessary?

II. Bootstrap Your Own Latent (BYOL)

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

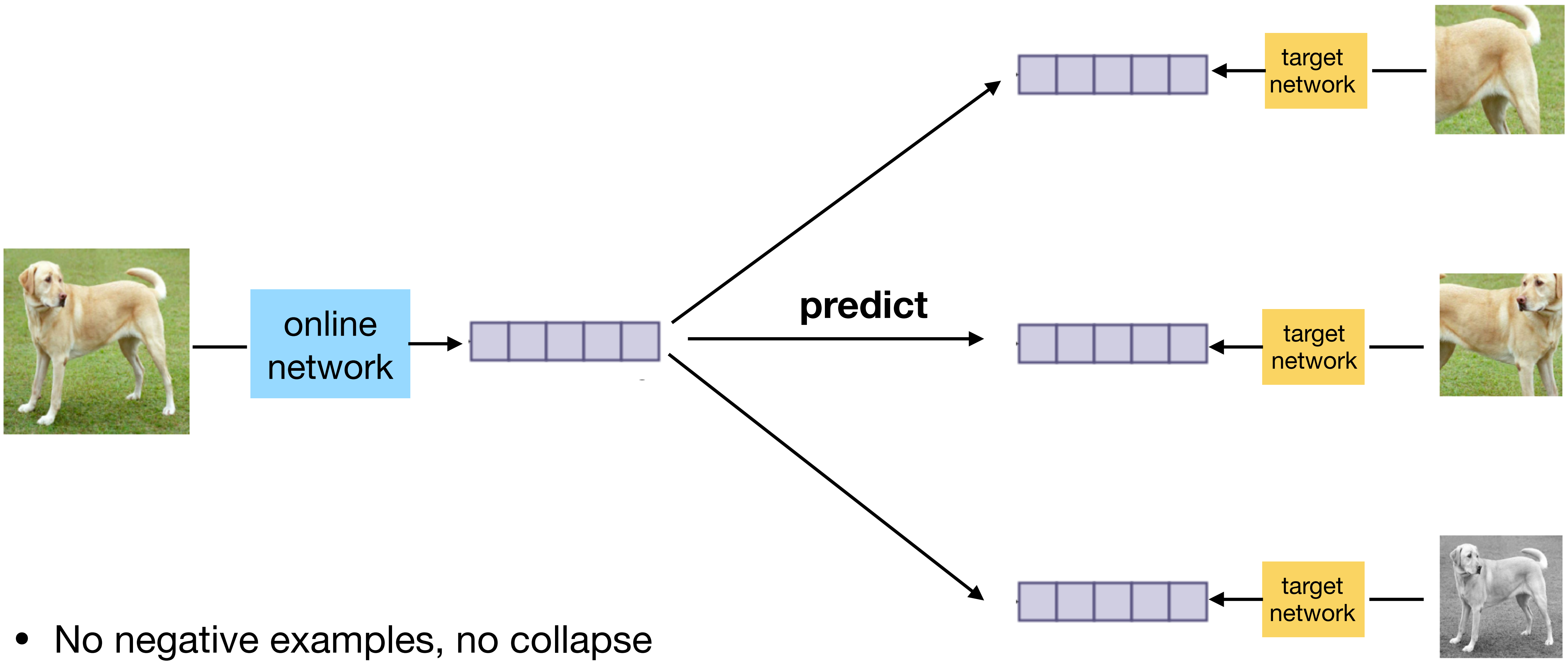
Jean-Bastien Grill[*,1] **Florian Strub**[*,1] **Florent Altché**[*,1] **Corentin Tallec**[*,1] **Pierre H. Richemond**[*,1,2]
Elena Buchatskaya[1] **Carl Doersch**[1] **Bernardo Avila Pires**[1] **Zhaohan Daniel Guo**[1]
Mohammad Gheshlaghi Azar[1] **Bilal Piot**[1] **Koray Kavukcuoglu**[1] **Rémi Munos**[1] **Michal Valko**[1]

[1]DeepMind [2]Imperial College

`[jbgrill,fstrub,altche,corentint,richemond]@google.com`

Learning without negative examples

- Cross-view prediction framework
- Separate network for prediction and to produce target representations



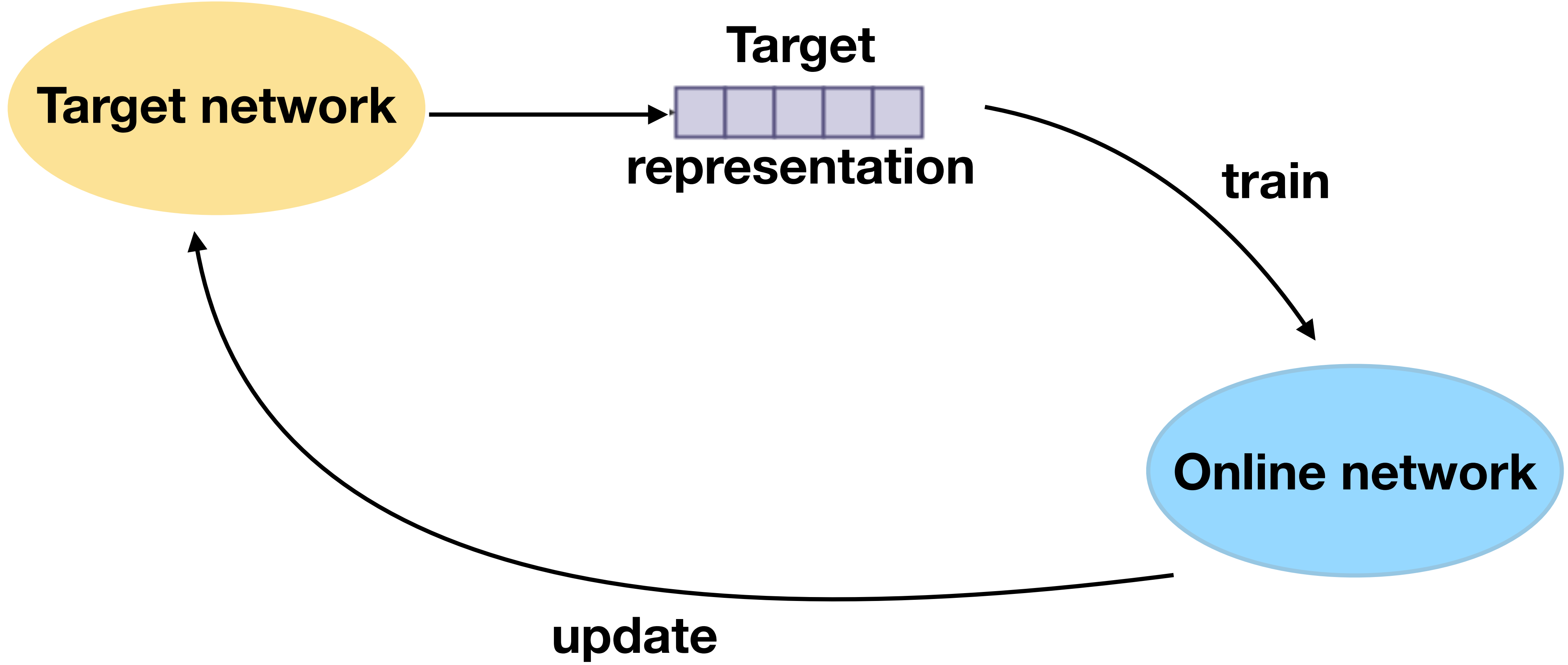
- No negative examples, no collapse

Fixed target network

Online network	Target network	ImageNet accuracy of linear classifier
None	None (random guessing)	0.1%
None	fixed ResNet50 (randomly initialized)	1.4%
ResNet50 (trained)	fixed ResNet50 (randomly initialized)	18.8%

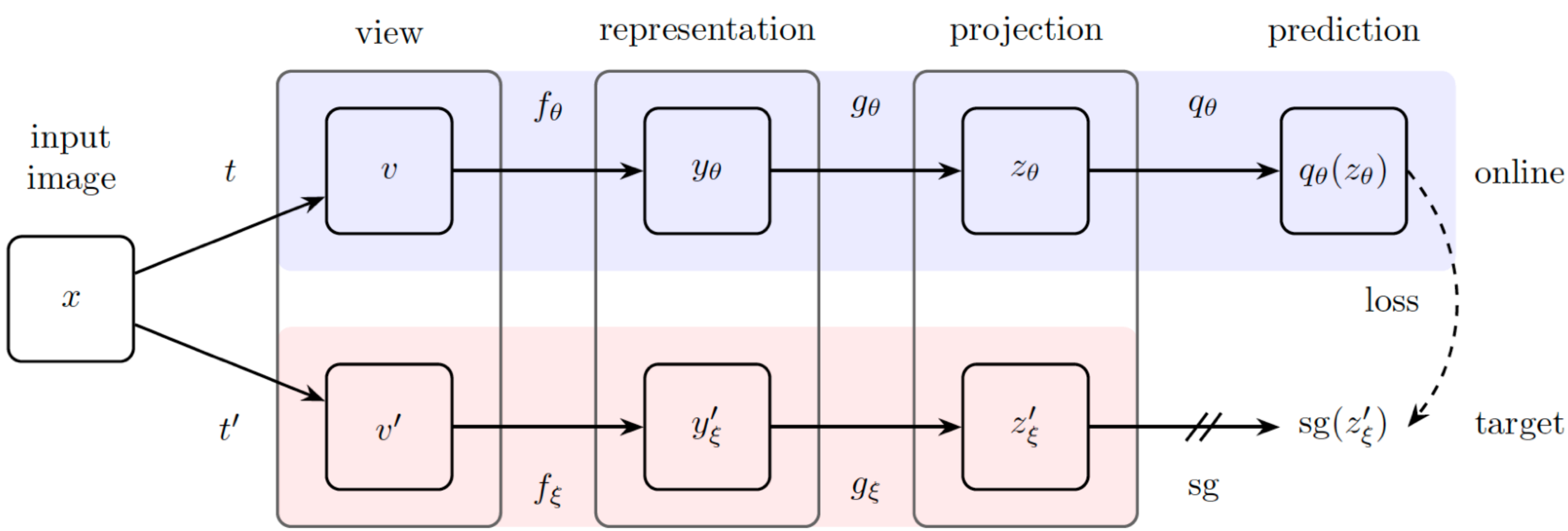
Can we do better with smarter target network?

Bootstrapping



Target in BYOL: exponential moving average of online network

BYOL architecture



- Loss:

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

$$\mathcal{L}^{\mathtt{BYOL}}_{\theta,\xi} = \mathcal{L}_{\theta,\xi} + \widetilde{\mathcal{L}}_{\theta,\xi}$$ (symmetrized loss)

- Dynamics:

1. Update online network: $\theta \leftarrow \mathrm{optimizer}\left(\theta, \nabla_\theta \mathcal{L}^{\mathtt{BYOL}}_{\theta,\xi}, \eta\right)$
2. Update target network: $\xi \leftarrow \tau\xi + (1-\tau)\theta$

How does BYOL avoid collapse?

- No explicit term to prevent collapse to constant representation, such as negative examples in SimCLR
- Important observation: BYOL dynamics will NOT necessarily converge to min of $\mathcal{L}^{\mathtt{BYOL}}_{\theta,\xi}$ w.r.t θ, ξ

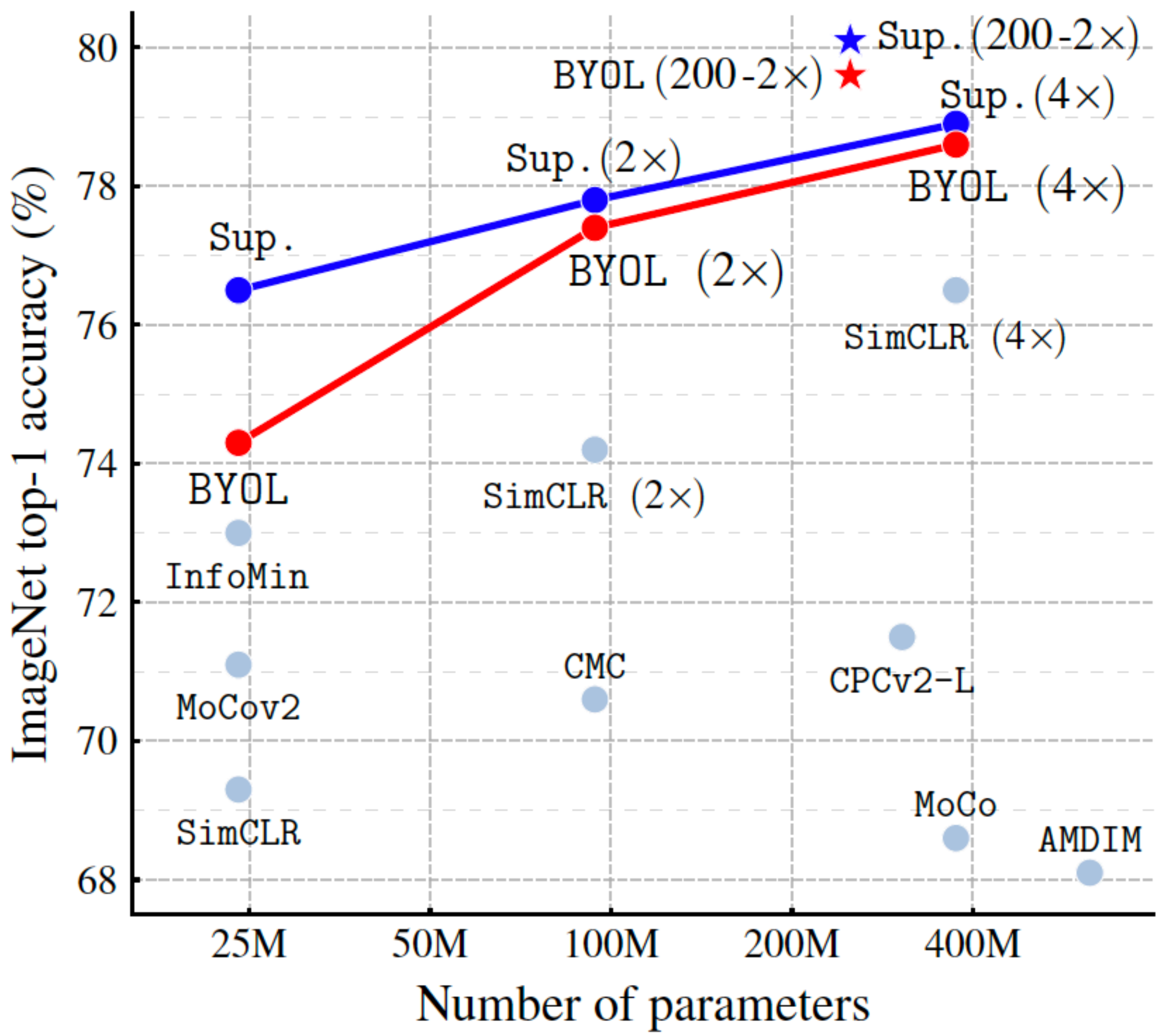
Target network update $\xi \leftarrow \tau\xi + (1-\tau)\theta$ **is not in the direction of** $\nabla_{\xi}\mathcal{L}^{\mathtt{BYOL}}_{\theta,\xi}$ **!**

- Hypothesis: there exists no $L_{\theta,\xi}$ such that BYOL dynamics is gradient descent on L jointly over θ, ξ
- There are still undesirable equilibria, but unstable

Experimental setup

- **Augmentations:** random flips, color distortion, Gaussian blur, solarization (same as SimCLR)
- **Architecture:**
  - Encoders: ResNet50(101, 150, 200) + MLP with BN
  - Predictor: MLP with BN
- **Optimization:**
  - LARS optimizer
  - 1000 epochs
  - 512 TPU cores (8 hours)
- **Evaluation protocols:**
  - Linear evaluation on ImageNet: freeze encoder + train linear classifier
  - Semi-supervised training on ImageNet: fine-tuning on 1% - 10% labelled data
  - Transfer to other classification tasks: linear evaluation and fine-tuning on other datasets

Linear evaluation



Method	Top-1	Top-5
Local Agg.	60.2	-
PIRL [35]	63.6	-
CPC v2 [32]	63.8	85.3
CMC [11]	66.2	87.0
SimCLR [8]	69.3	89.0
MoCo v2 [37]	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 [32]	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.

Semi-supervised

Method	Top-1		Top-5	
	1%	10%	1%	10%
Supervised [77]	25.4	56.4	48.4	80.4
InstDisc	-	-	39.2	77.4
PIRL [35]	-	-	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 [32]	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**

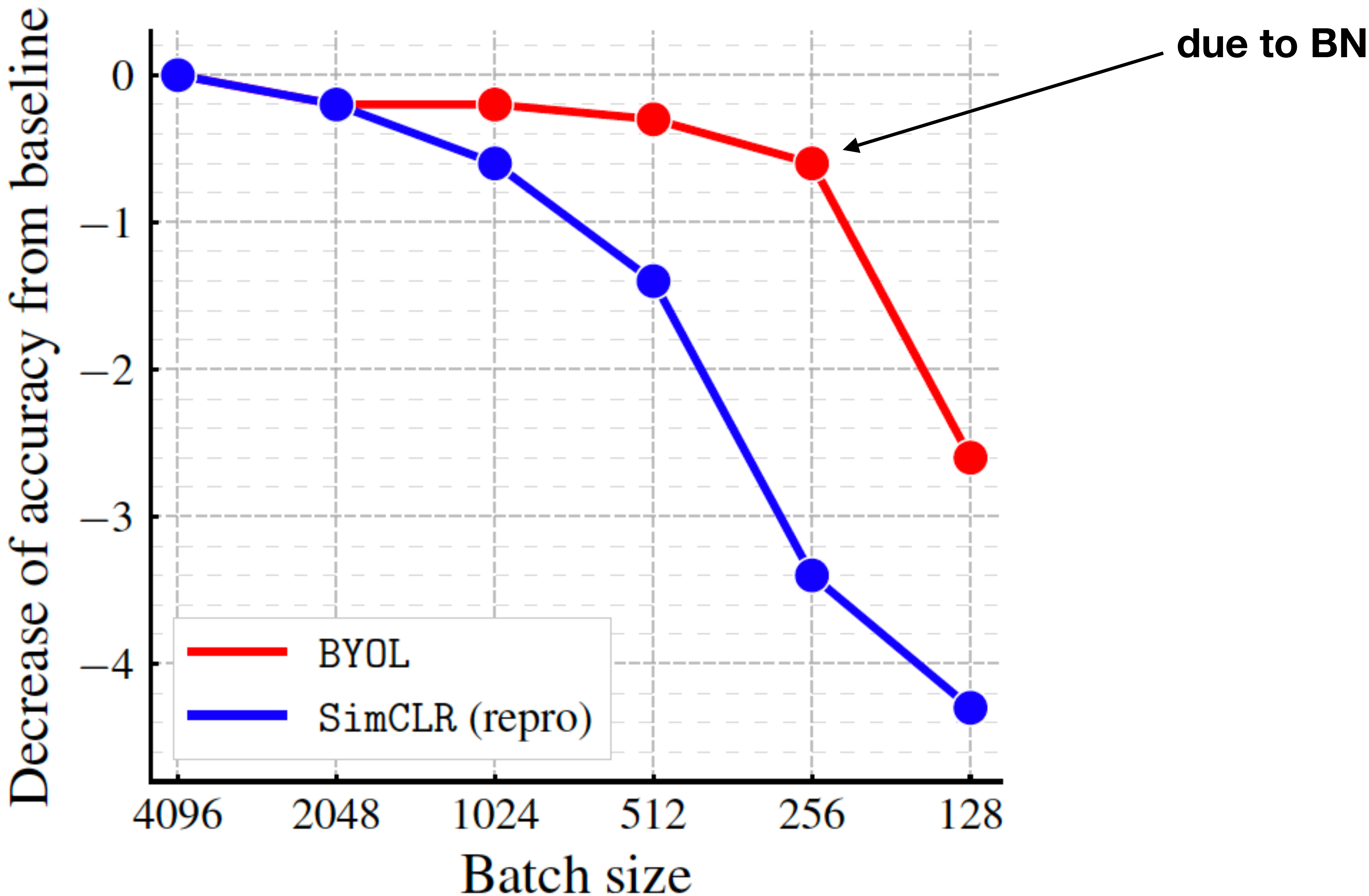
(b) Other ResNet encoder architectures.

Transfer learning

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

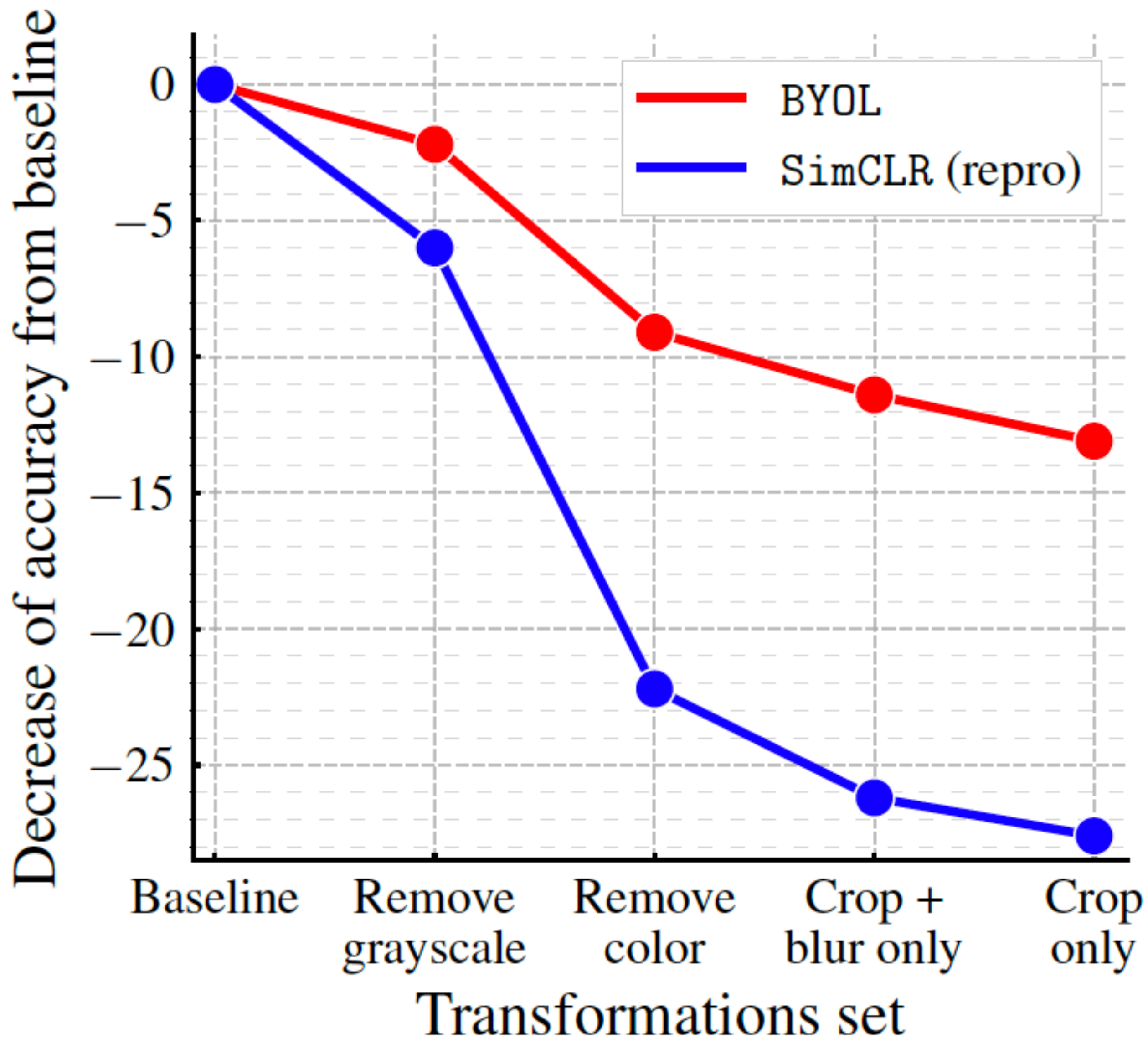
Table 3: Transfer learning results from ImageNet (IN) with the standard ResNet-50 architecture.

Effect of batch size



(a) Impact of batch size

Effect of augmentations



(b) Impact of progressively removing transformations

- Crops and flips of the same image share similar histogram
- Without color distortion, SimCLR relies on image histogram to differentiate between views of the same image and others
- SimCLR representations are not incentivized to retain information other than color histogram

What is necessary?

What do we need to learn useful representations (without collapse)?

Negative examples? ✗

Large batch size? ✗

Momentum encoder? ?

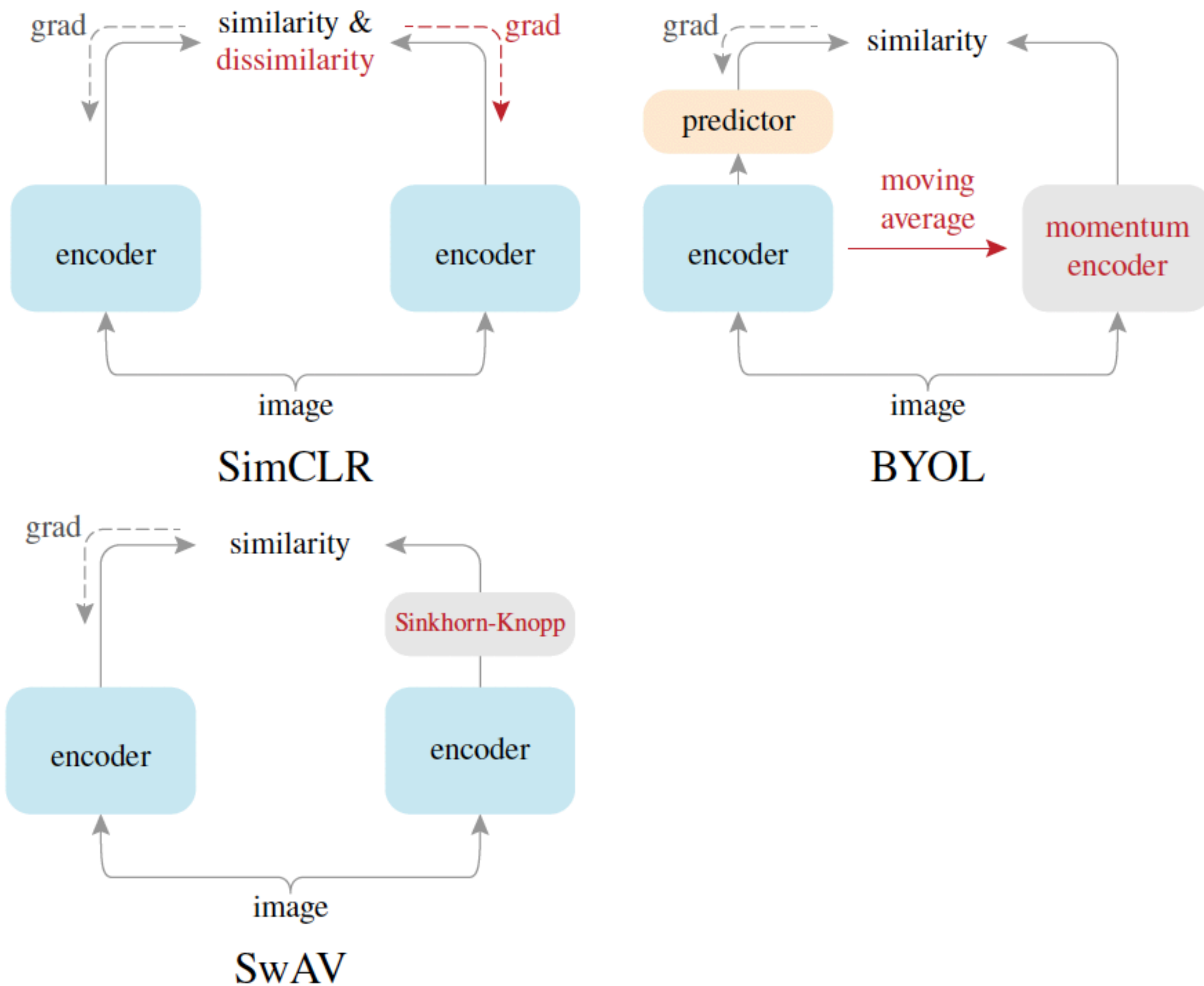
III. SimSiam

Exploring Simple Siamese Representation Learning

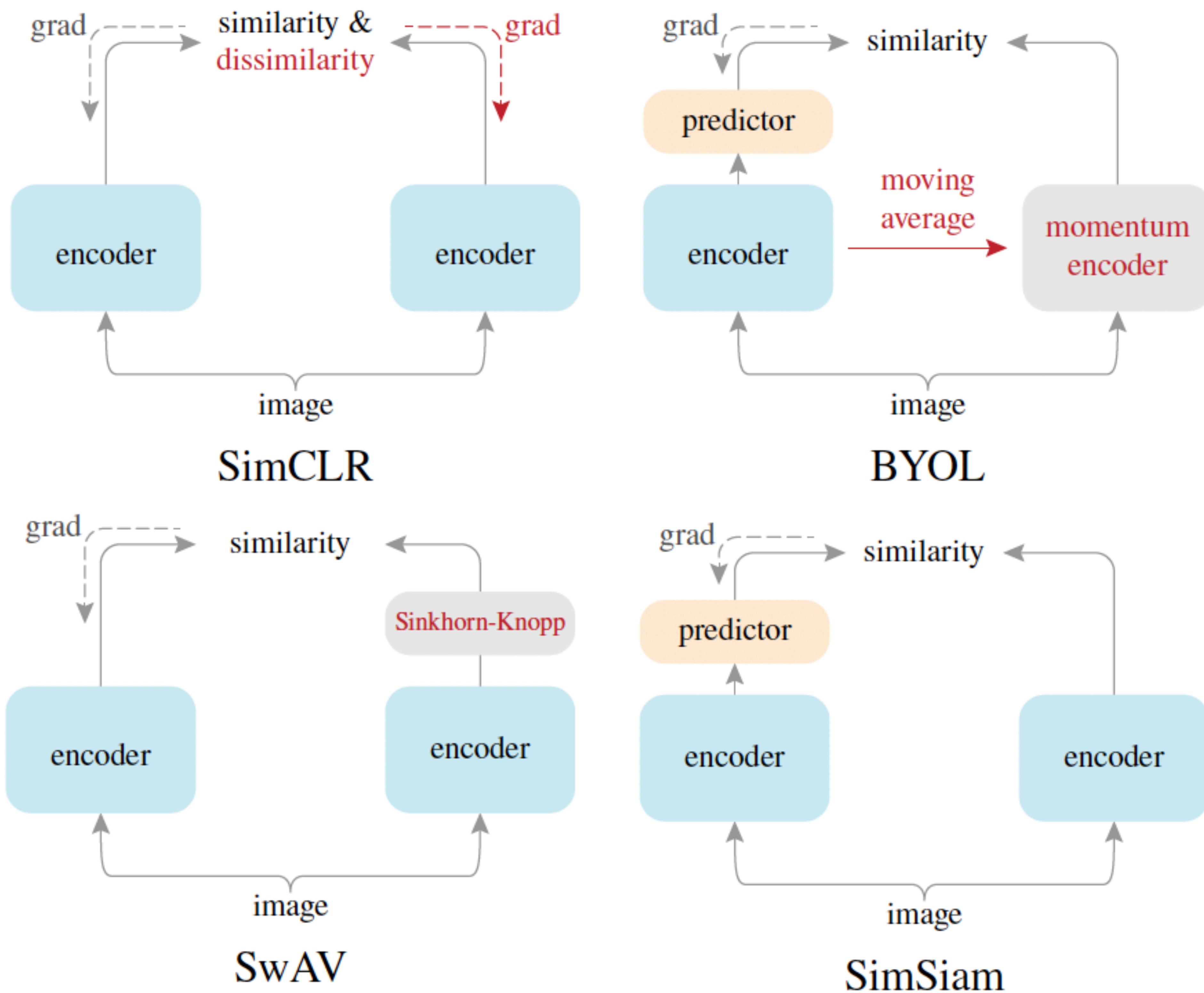
Xinlei Chen Kaiming He
Facebook AI Research (FAIR)

Siamese networks

Siamese networks

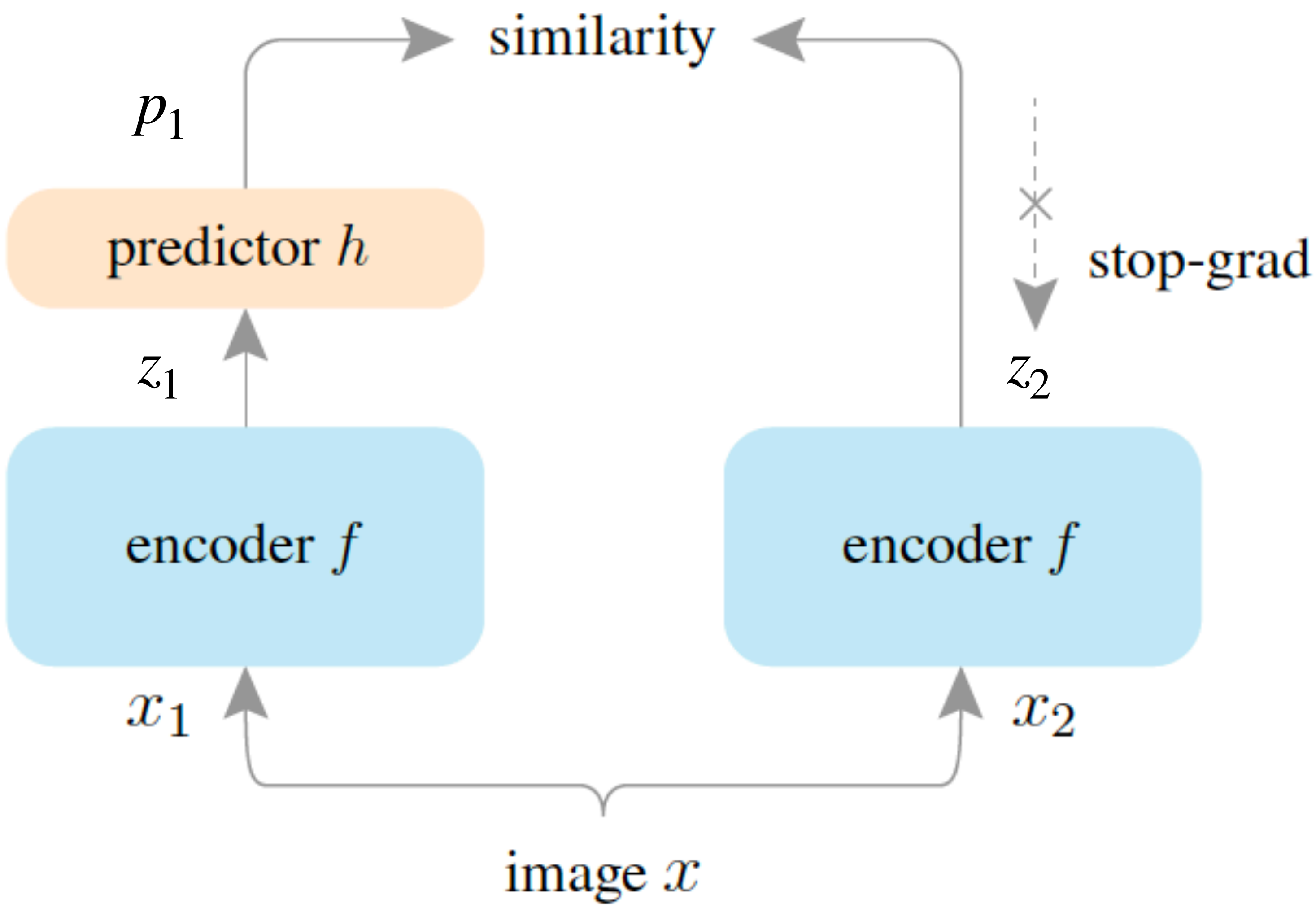


Siamese networks



Minimalistic design

What is the only necessary component? **Stop-gradient!**

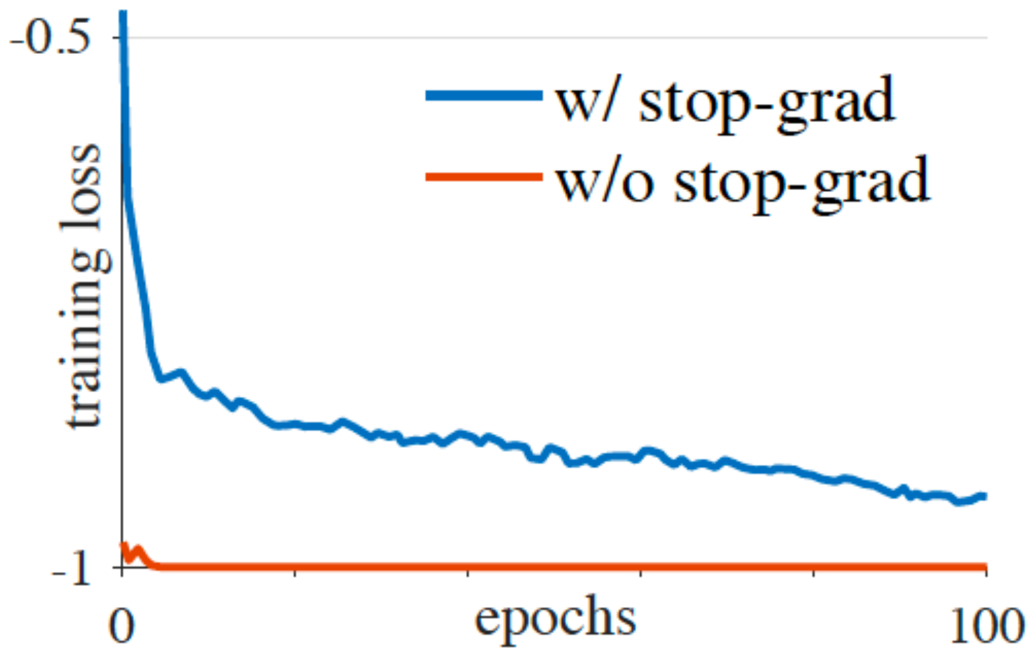


$$\mathcal{D}(p_1, z_2) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{z_2}{\|z_2\|_2}$$

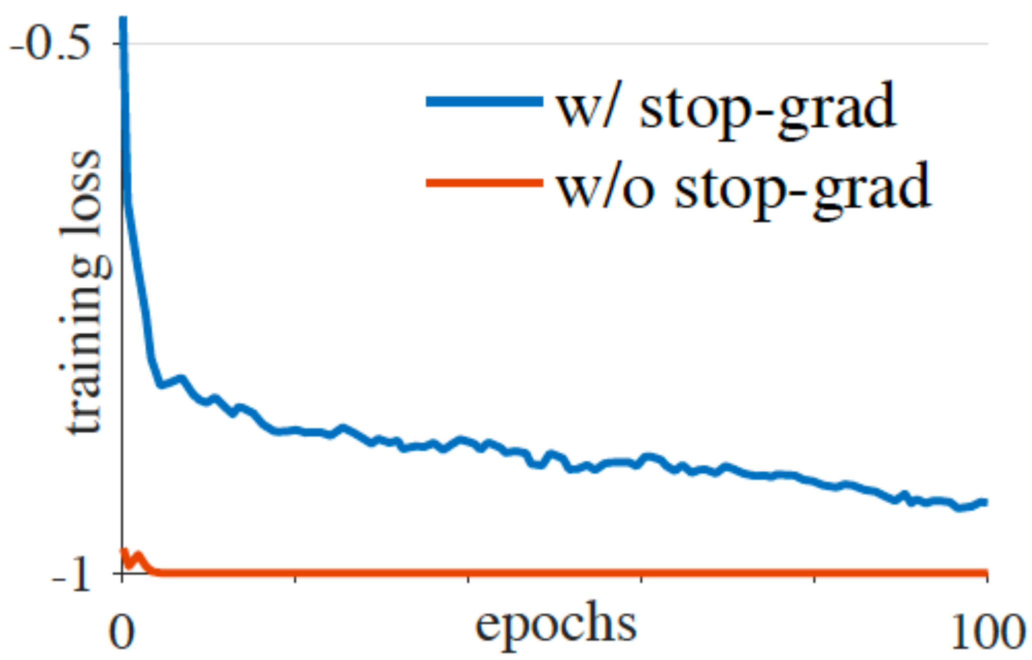
$$\mathcal{L} = \frac{1}{2}\mathcal{D}(p_1, \texttt{stopgrad}(z_2)) + \frac{1}{2}\mathcal{D}(p_2, \texttt{stopgrad}(z_1))$$

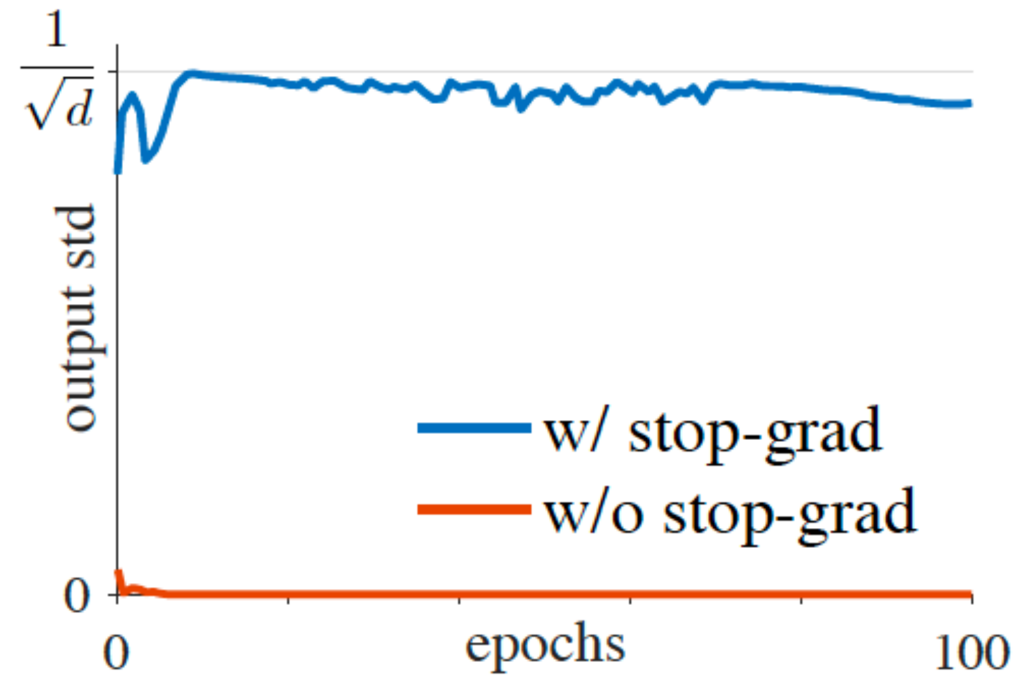
Importance of Stop-grad

Importance of Stop-grad

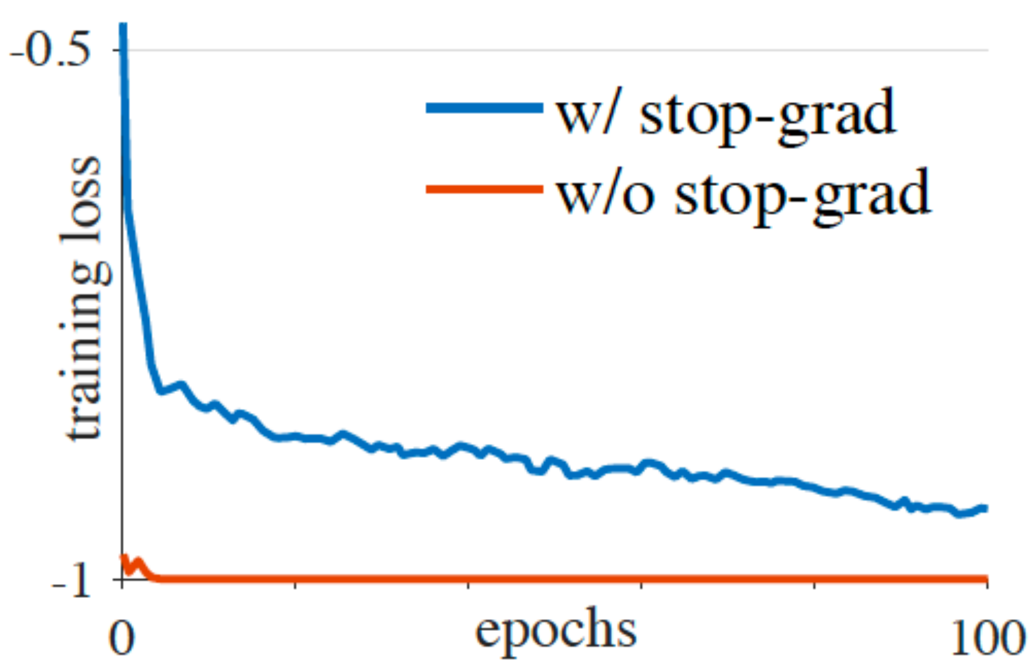


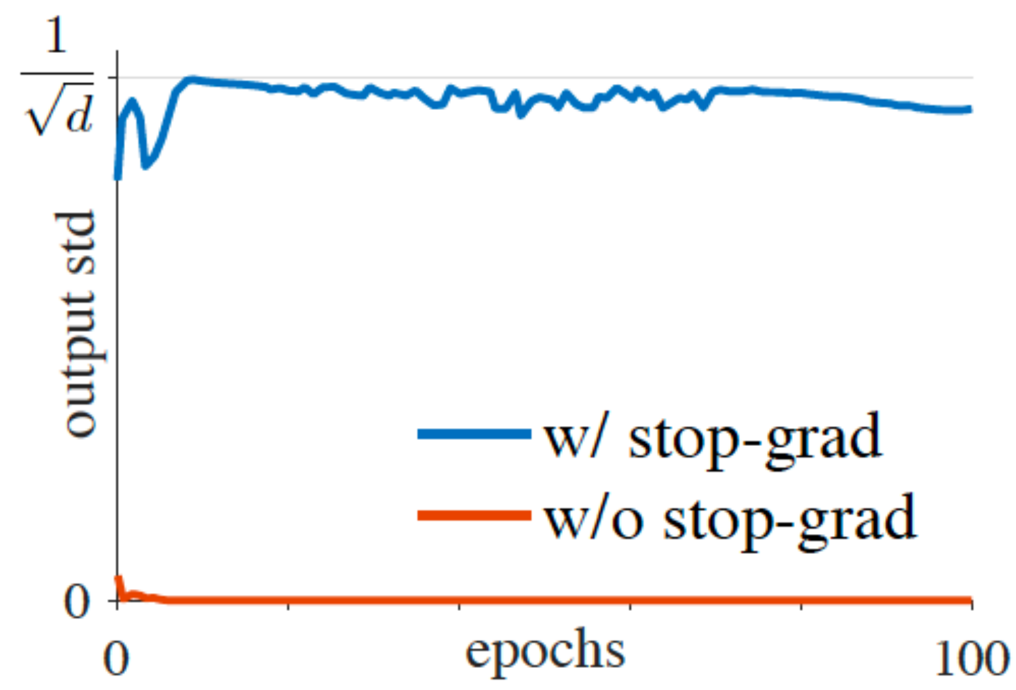
Importance of Stop-grad

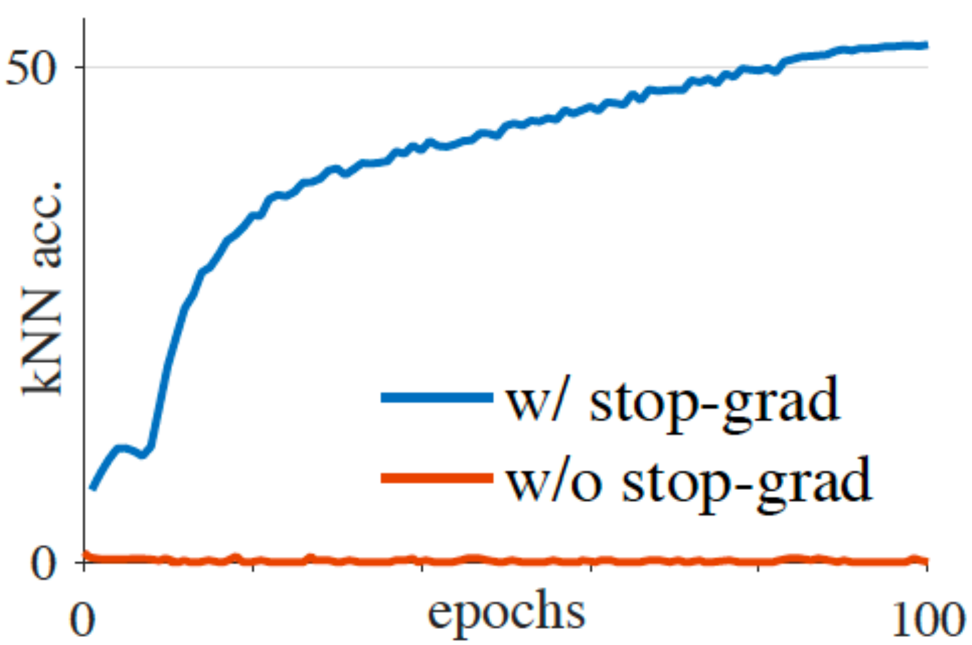




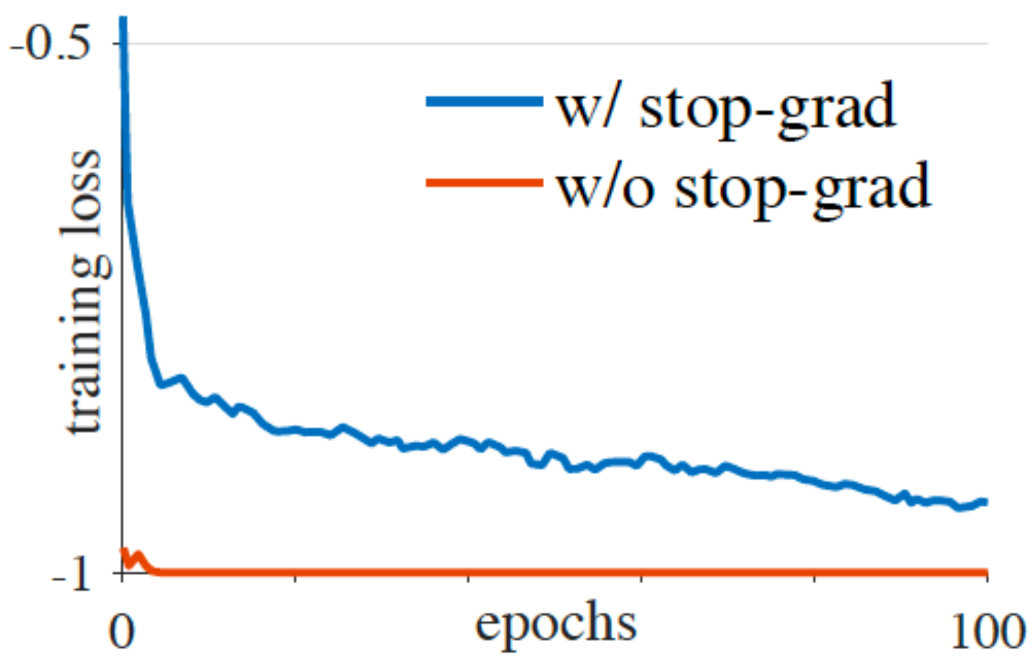
Importance of Stop-grad

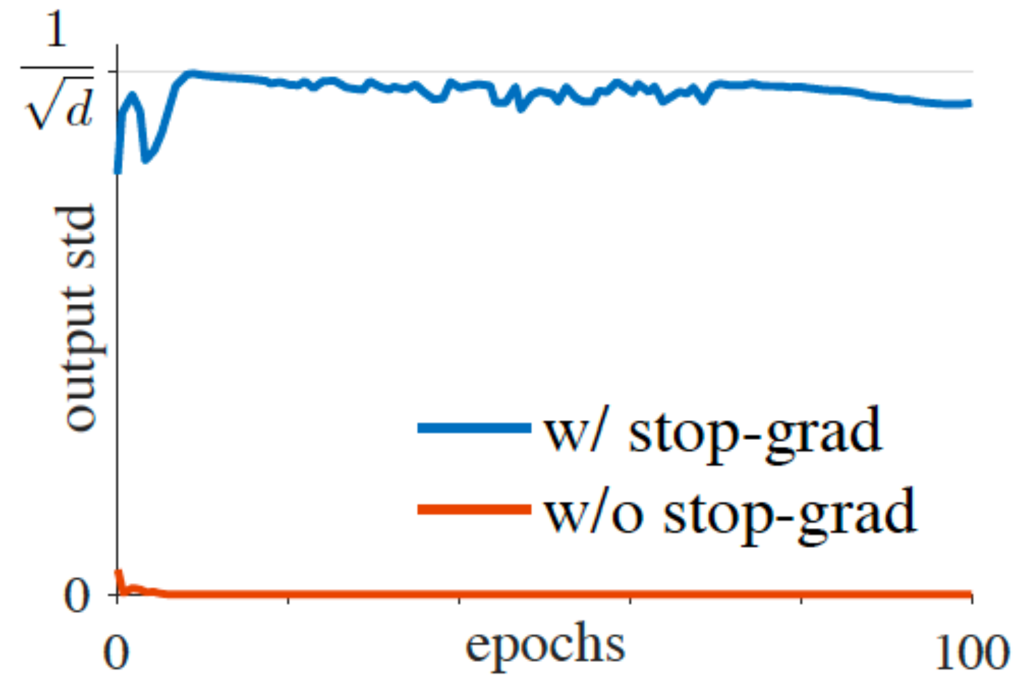


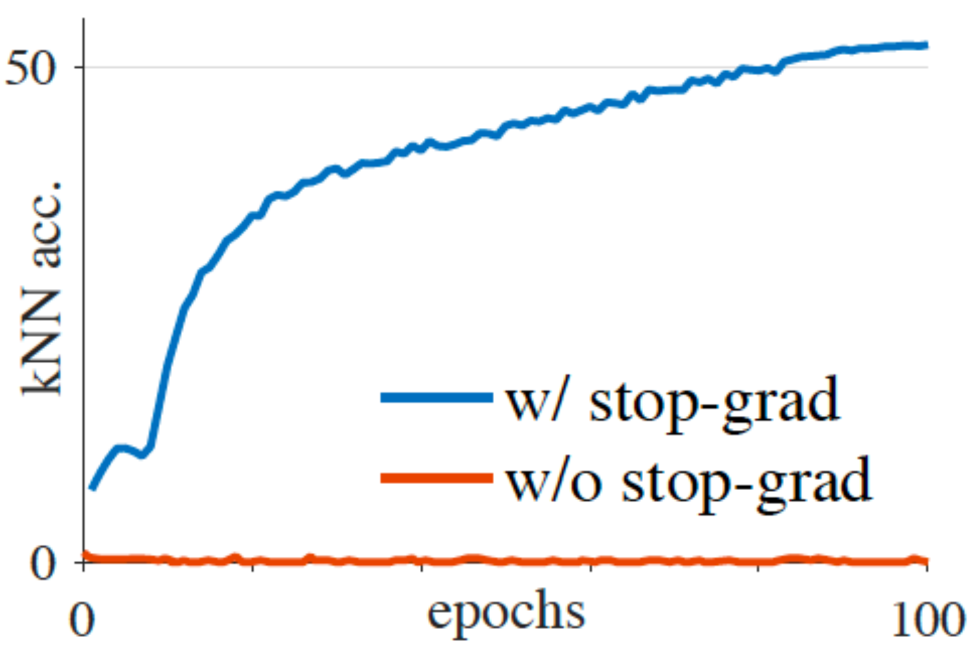




Importance of Stop-grad





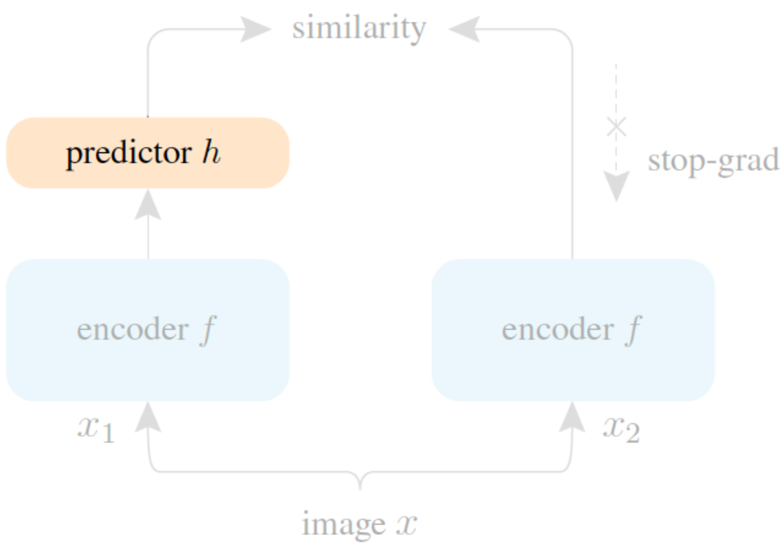


ImageNet linear evaluation

	acc. (%)
w/ stop-grad	67.7±0.1
w/o stop-grad	0.1

Other factors?

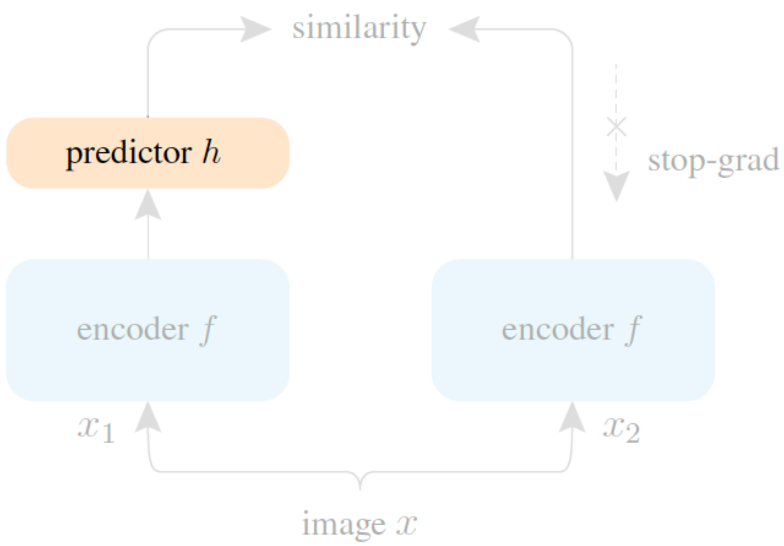
- Prediction head



	pred. MLP h	acc. (%)
baseline	lr with cosine decay	67.7
(a)	no pred. MLP	0.1
(b)	fixed random init.	1.5
(c)	lr not decayed	68.1

Other factors?

- Prediction head



	pred. MLP h	acc. (%)
baseline	lr with cosine decay	67.7
(a)	no pred. MLP	0.1
(b)	fixed random init.	1.5
(c)	lr not decayed	68.1

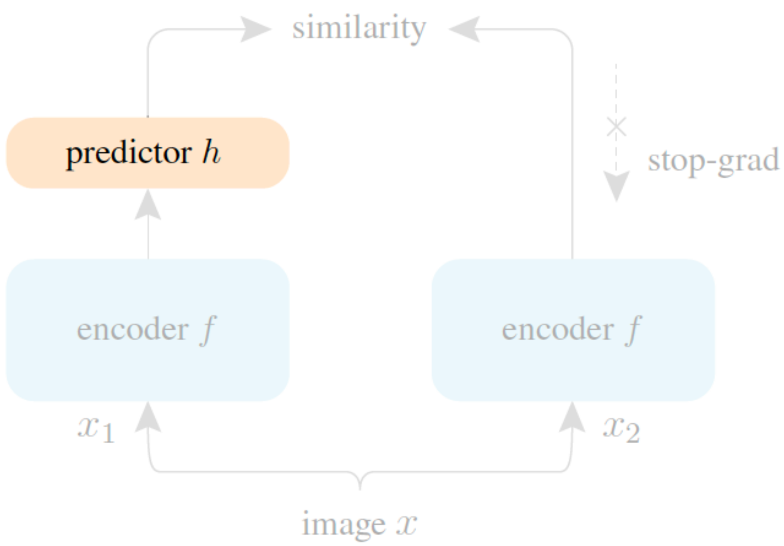
- Similarity function

	cosine	cross-entropy
acc. (%)	68.1	63.2

$$\mathcal{D}(p_1, z_2) = -\texttt{softmax}(z_2) \cdot \log \texttt{softmax}(p_1)$$

Other factors?

- Prediction head



	pred. MLP h	acc. (%)
baseline	lr with cosine decay	67.7
(a)	no pred. MLP	0.1
(b)	fixed random init.	1.5
(c)	lr not decayed	68.1

- Similarity function

	cosine	cross-entropy
acc. (%)	68.1	63.2

$$\mathcal{D}(p_1, z_2) = -\texttt{softmax}(z_2) \cdot \log \texttt{softmax}(p_1)$$

- Symmetrized loss

	sym.	asym.	asym. 2×
acc. (%)	68.1	64.8	67.3

Other factors?

- Batch size*

batch size	64	128	256	512	1024	2048	4096
acc. (%)	66.1	67.3	68.1	68.1	68.0	67.9	64.0

*SGD is used, not LARS

Other factors?

- Batch size*

batch size	64	128	256	512	1024	2048	4096
acc. (%)	66.1	67.3	68.1	68.1	68.0	67.9	64.0

*SGD is used, not LARS

- Batch normalization

	case	proj. MLP's BN hidden	proj. MLP's BN output	pred. MLP's BN hidden	pred. MLP's BN output	acc. (%)
(a)	none	-	-	-	-	34.6
(b)	hidden-only	✓	-	✓	-	67.4
(c)	default	✓	✓	✓	-	68.1
(d)	all	✓	✓	✓	✓	unstable

unstable training, **not** representation collapse

How does SimSiam work?

Hypothesis: SimSiam is an EM-like algorithm. It involves two sets of variables and solves two subproblems.

How does SimSiam work?

Hypothesis: SimSiam is an EM-like algorithm. It involves two sets of variables and solves two subproblems.

Consider loss:

$$\mathcal{L}(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}}\left[\left\|\mathcal{F}_\theta(\mathcal{T}(x)) - \eta_x\right\|_2^2\right]$$ (no predictor yet!)

How does SimSiam work?

Hypothesis: SimSiam is an EM-like algorithm. It involves two sets of variables and solves two subproblems.

Consider loss:

$$\mathcal{L}(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}}\left[\left\|\mathcal{F}_\theta(\mathcal{T}(x)) - \eta_x\right\|_2^2\right]$$

(no predictor yet!)

Optimization problem:

$$\min_{\theta,\eta} \mathcal{L}(\theta, \eta)$$

How does SimSiam work?

Hypothesis: SimSiam is an EM-like algorithm. It involves two sets of variables and solves two subproblems.

Consider loss:

$$\mathcal{L}(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}}\left[\left\|\mathcal{F}_\theta(\mathcal{T}(x)) - \eta_x\right\|_2^2\right]$$

(no predictor yet!)

Optimization problem:

$$\min_{\theta,\eta} \mathcal{L}(\theta, \eta)$$

Solution by alternating algorithm:

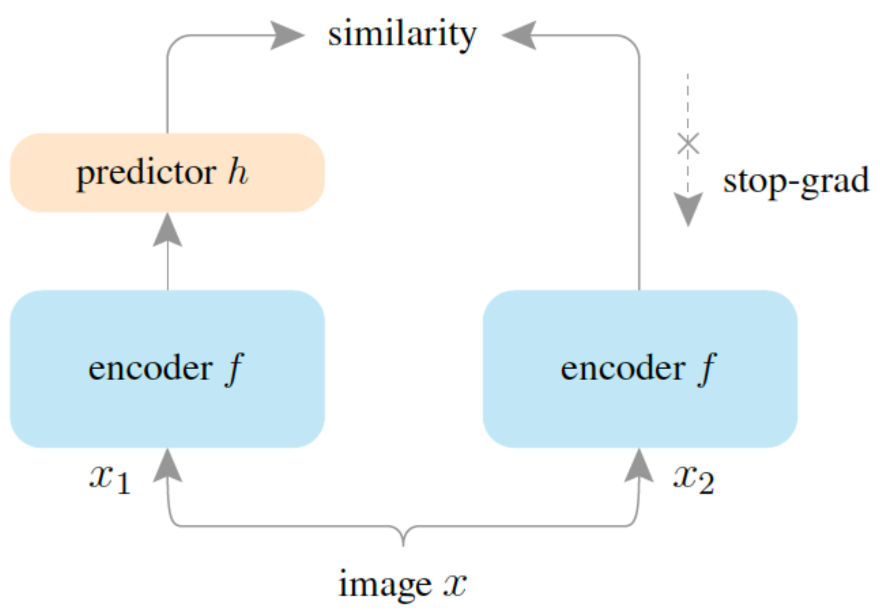
$$\begin{aligned} \theta^t &\leftarrow \arg\min_\theta \; \mathcal{L}(\theta, \eta^{t-1}) \\ \eta^t &\leftarrow \arg\min_\eta \; \mathcal{L}(\theta^t, \eta) \end{aligned}$$

How does SimSiam work?

Step 1.

$$\theta^t \leftarrow \arg\min_{\theta} \mathcal{L}(\theta, \eta^{t-1})$$

- Update encoder parameters
- Use SGD to solve sub-problem
- Stop-gradient: we don't optimize over η
- SimSiam: approx. solution by one step of SGD

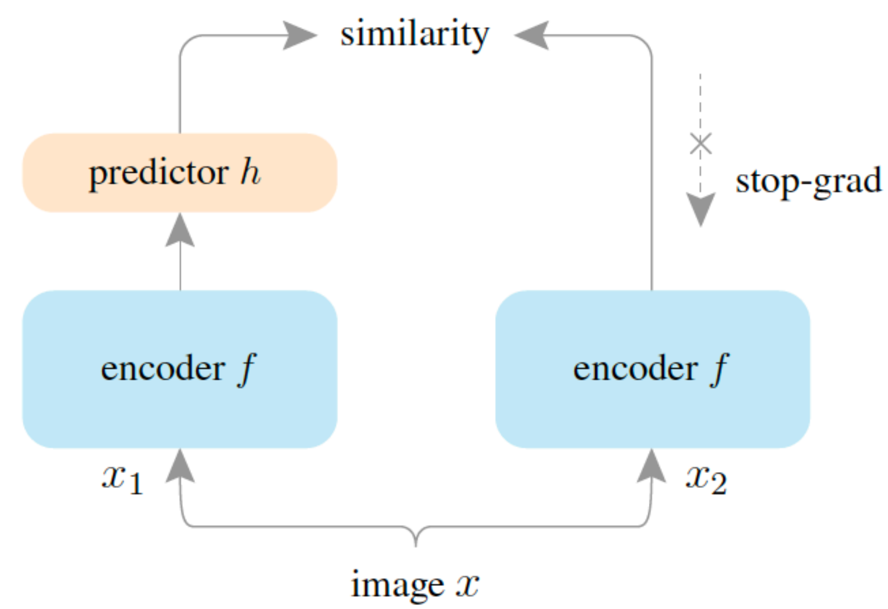


How does SimSiam work?

Step 1.

$$\theta^t \quad \leftarrow \quad \arg\min_{\theta} \ \mathcal{L}(\theta, \eta^{t-1})$$

- Update encoder parameters
- Use SGD to solve sub-problem
- Stop-gradient: we don't optimize over η
- SimSiam: approx. solution by one step of SGD



If this is true, we could run more iterations of SGD!

	1-step	10-step	100-step	1-epoch
acc. (%)	68.1	68.7	68.9	67.0

How does SimSiam work?

Step 2.

$$\eta^t \quad \leftarrow \quad \arg\min_{\eta} \; \mathcal{L}(\theta^t, \eta) \qquad\qquad \mathcal{L}(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}}\left[\left\|\mathcal{F}_\theta(\mathcal{T}(x)) - \eta_x\right\|_2^2\right]$$

How does SimSiam work?

Step 2.

$$\eta^t \quad \leftarrow \quad \arg\min_{\eta} \ \mathcal{L}(\theta^t, \eta) \qquad\qquad \mathcal{L}(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}}\Big[\big\|\mathcal{F}_\theta(\mathcal{T}(x)) - \eta_x\big\|_2^2\Big]$$

- Solution is

$$\eta_x^t \leftarrow \mathbb{E}_{\mathcal{T}}\Big[\mathcal{F}_{\theta^t}(\mathcal{T}(x))\Big]$$

How does SimSiam work?

Step 2.

$$\eta^t \quad \leftarrow \quad \arg\min_{\eta} \; \mathcal{L}(\theta^t, \eta) \qquad\qquad \mathcal{L}(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}}\Big[\big\|\mathcal{F}_{\theta}(\mathcal{T}(x)) - \eta_x\big\|_2^2\Big]$$

- Solution is

$$\eta_x^t \leftarrow \mathbb{E}_{\mathcal{T}}\Big[\mathcal{F}_{\theta^t}(\mathcal{T}(x))\Big]$$

- This is the average representation of x over the distribution of augmentations
- Approximate expectation by sampling *a single view*

$$\eta_x^t \leftarrow \mathcal{F}_{\theta^t}(\mathcal{T}'(x))$$

$$\theta^{t+1} \leftarrow \arg\min_{\theta} \mathbb{E}_{x,\mathcal{T}}\Big[\big\|\mathcal{F}_{\theta}(\mathcal{T}(x)) - \mathcal{F}_{\theta^t}(\mathcal{T}'(x))\big\|_2^2\Big]$$

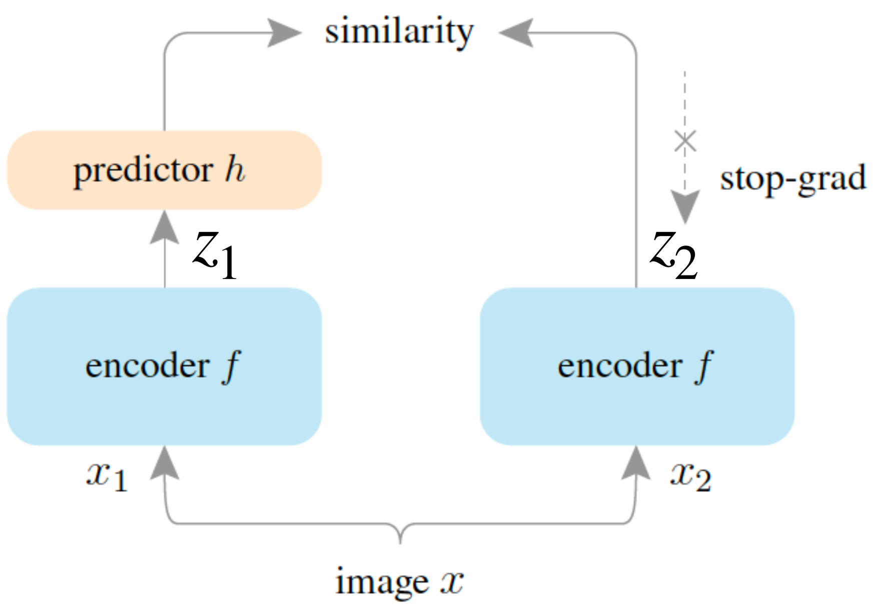
How does SimSiam work?

Adding predictor

- Predictor should minimize $\mathbb{E}_z\left[\left\|h(z_1) - z_2\right\|_2^2\right]$
- Minimizer:

$$h(z_1) = \mathbb{E}_z[z_2] = \mathbb{E}_{\mathcal{T}}\left[f(\mathcal{T}(x))\right]$$

- Predictor learns to estimate the expectation
- Sampling of $\mathcal{T}$ is distributed over the epochs



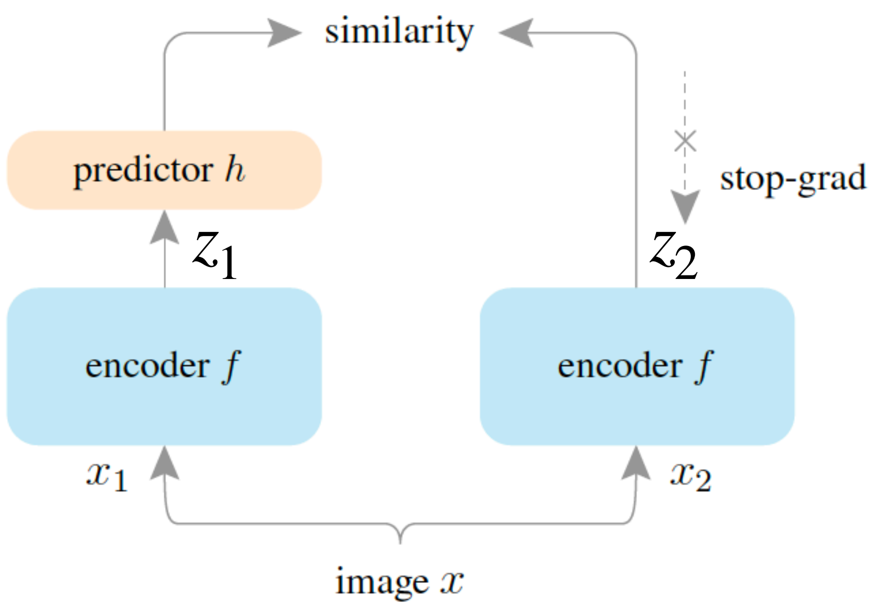
How does SimSiam work?

Adding predictor

- Predictor should minimize $\mathbb{E}_z\left[\left\|h(z_1) - z_2\right\|_2^2\right]$
- Minimizer:

$$h(z_1) = \mathbb{E}_z[z_2] = \mathbb{E}_{\mathcal{T}}\left[f(\mathcal{T}(x))\right]$$

- Predictor learns to estimate the expectation
- Sampling of $\mathcal{T}$ is distributed over the epochs



If this is true, we could estimate expectation using moving average!

$$\eta_x^t \leftarrow m * \eta_x^{t-1} + (1-m) * \mathcal{F}_{\theta^t}(\mathcal{T}'(x))$$

Predictor	Top-1 acc.
None	0.1%
MLP	68.1%
moving average	55.0%

Results

ImageNet linear evaluation

method	batch size	negative pairs	momentum encoder	100 ep	200 ep	400 ep	800 ep
SimCLR (repro.+)	4096	✓		66.5	68.3	69.8	70.4
MoCo v2 (repro.+)	**256**	✓	✓	67.4	69.9	71.0	72.2
BYOL (repro.)	4096		✓	66.5	**70.6**	**73.2**	**74.3**
SwAV (repro.+)	4096			66.5	69.1	70.7	71.8
SimSiam	**256**			**68.1**	70.0	70.8	71.3

Transfer learning

	VOC 07 detection			VOC 07+12 detection			COCO detection			COCO instance seg.		
pre-train	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP^{mask}_{50}	AP^{mask}	AP^{mask}_{75}
scratch	35.9	16.8	13.0	60.2	33.8	33.1	44.0	26.4	27.8	46.9	29.3	30.8
ImageNet supervised	74.4	42.4	42.7	81.3	53.5	58.8	58.2	38.2	41.2	54.7	33.3	35.2
SimCLR (repro.+)	75.9	46.8	50.1	81.8	55.5	61.4	57.7	37.9	40.9	54.6	33.3	35.3
MoCo v2 (repro.+)	**77.1**	**48.5**	**52.5**	**82.3**	**57.0**	**63.3**	**58.8**	**39.2**	**42.5**	**55.5**	**34.3**	**36.6**
BYOL (repro.)	**77.1**	47.0	49.9	81.4	55.3	61.1	57.8	37.9	40.9	54.3	33.2	35.0
SwAV (repro.+)	75.5	46.5	49.6	81.5	55.4	61.4	57.6	37.6	40.3	54.2	33.1	35.1
SimSiam, base	75.5	47.0	50.2	**82.0**	56.4	62.8	57.5	37.9	40.9	54.2	33.2	35.2
SimSiam, optimal	**77.3**	**48.5**	**52.5**	**82.4**	**57.0**	**63.7**	**59.3**	**39.2**	**42.1**	**56.0**	**34.4**	**36.7**

Conclusion

Conclusion

- **Self-supervised learning** is about learning good representations without human annotation

Conclusion

- **Self-supervised learning** is about learning good representations without human annotation
- **Contrastive methods** learn representations by discriminating between different views of the same image and views of a different image

Conclusion

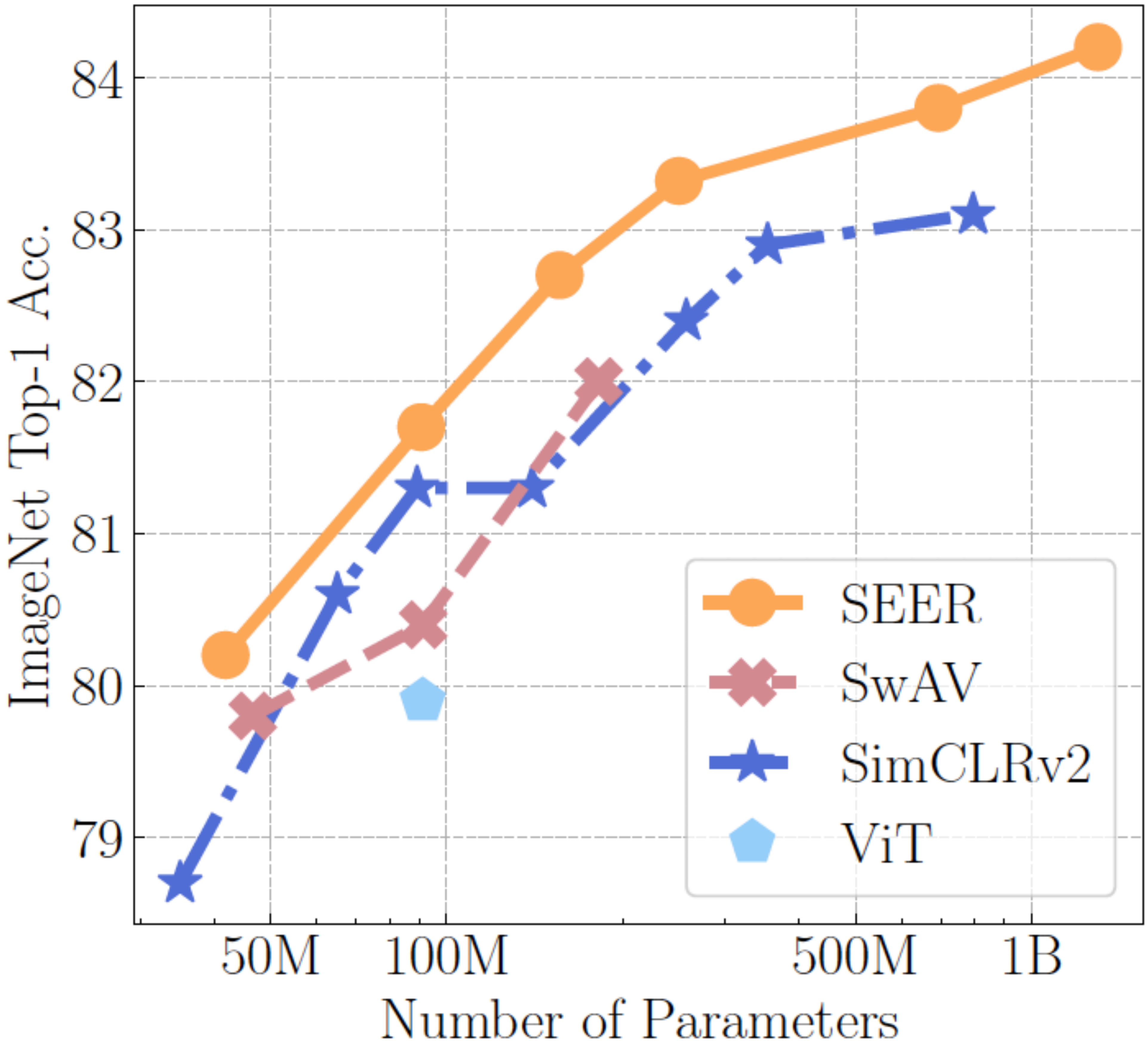
- **Self-supervised learning** is about learning good representations without human annotation
- **Contrastive methods** learn representations by discriminating between different views of the same image and views of a different image
- Now we know that **negative examples are not necessary** for good representation learning

Conclusion

- **Self-supervised learning** is about learning good representations without human annotation
- **Contrastive methods** learn representations by discriminating between different views of the same image and views of a different image
- Now we know that **negative examples are not necessary** for good representation learning
- **BYOL** and **SimSiam** learn to predict the representation of an image from the representation of another view of **the same image**

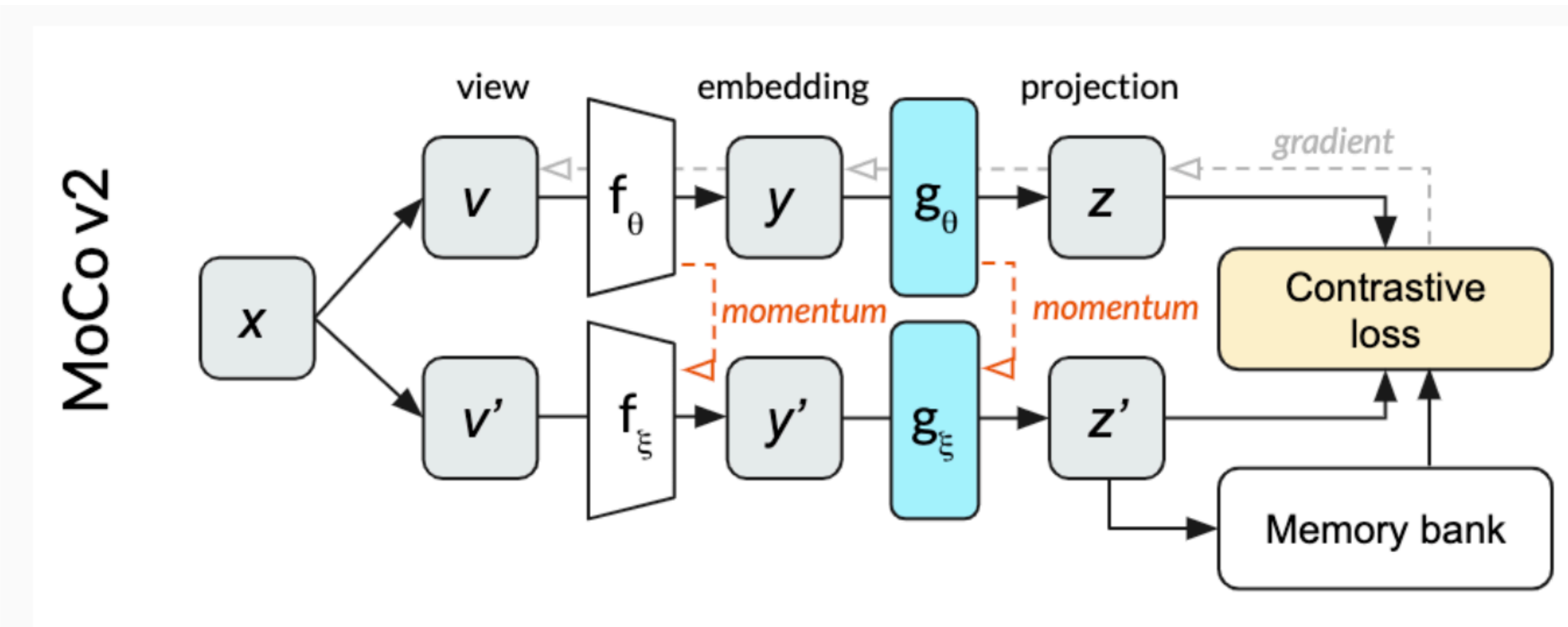
Conclusion

- **Self-supervised learning** is about learning good representations without human annotation
- **Contrastive methods** learn representations by discriminating between different views of the same image and views of a different image
- Now we know that **negative examples are not necessary** for good representation learning
- **BYOL** and **SimSiam** learn to predict the representation of an image from the representation of another view of **the same image**
- It is still an open question how these methods learn useful representations while avoiding representational collapse



ImageNet Top-1 Acc.
84
83
82
81
80
79
50M
100M
500M
1B
Number of Parameters
SEER
SwAV
SimCLRv2
ViT

MoCo v2



MoCo v2
view
embedding
projection
gradient
x
v
v'
fθ
fξ
y
y'
gθ
gξ
z
z'
momentum
momentum
Contrastive loss
Memory bank

Connection to EM

Likelihood function

$$L(\theta; X, Z) = p(X, Z \mid \theta)$$

observed data — latent variables — unknown parameters

Maximum Likelihood Estimate

$$\hat{\theta}_{ML} = arg\max_{\theta} p(X \mid \theta) = arg\max_{\theta} \int p(X, Z = z \mid \theta) dz$$

typically intractable

Connection to EM

Expectation-Maximization algorithm

E-step: obtain **E**xpectation of complete likelihood given current model

$$Q(\theta\,|\,\theta_t) := \mathbb{E}_{Z|X,\theta} \log L(\theta; X, Z)$$

M-step: update the model given the data by **M**aximization

$$\theta_{t+1} = arg\max_{\theta} Q(\theta\,|\,\theta_t)$$

Notes:

- This is equivalent to maximizing a lower bound of $\log L(\theta; X)$
- EM step always increases $\log L(\theta; X)$
- No guarantee that it converges to MLE (multi-modal distributions)

EM derivation

$$\begin{aligned}\max_\theta \log \int_x p(x,z;\theta)dx &= \max_\theta \log \int_x \frac{q(x)}{q(x)} p(x,z;\theta)dx \\ &= \max_\theta \log \int_x q(x) \frac{p(x,z;\theta)}{q(x)} dx \\ &= \max_\theta \log E_{X\sim q}\left[\frac{p(X,z;\theta)}{q(X)}\right] \\ &\geq \max_\theta E_{X\sim q} \log \left[\frac{p(X,z;\theta)}{q(X)}\right] \\ &= \max_\theta \int_x q(x)\log p(x,z;\theta)dx - \int_x q(x)\log q(x)dx\end{aligned}$$

Jensen's Inequality

Jensen's Inequality: equality holds when $f(x) = \log \frac{p(x,z;\theta)}{q(x)}$ *is an affine function. This is achieved for* $q(x) = p(x|z;\theta) \propto p(x,z;\theta)$

EM Algorithm: Iterate

1. E-step: Compute $q(x) = p(x|z;\theta)$
2. M-step: Compute $\theta = \arg\max_\theta \int_x q(x)\log p(x,z;\theta)dx$

https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/Likelihood_EM_HMM_Kalman.pdf