Self-Supervised Visual Representation Learning

Xiaolong Wang

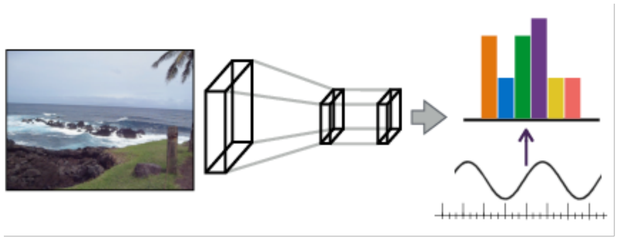
Self-Supervised Learning

- Pre-training and Fine-tuning
  - Pretext Tasks
  - Contrastive Learning
  - Masked Autoencoders
- Utilize self-supervision during Test Time
  - Adapting supervised task, RL task for out-of-distribution generalization

Pretext Tasks + Fine-tuning

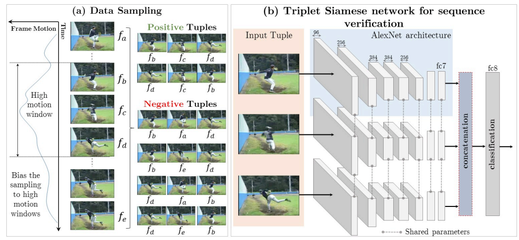
Pretext Task

The task being solved is not of genuine interest, but is solved only for the true purpose of learning a good data representation

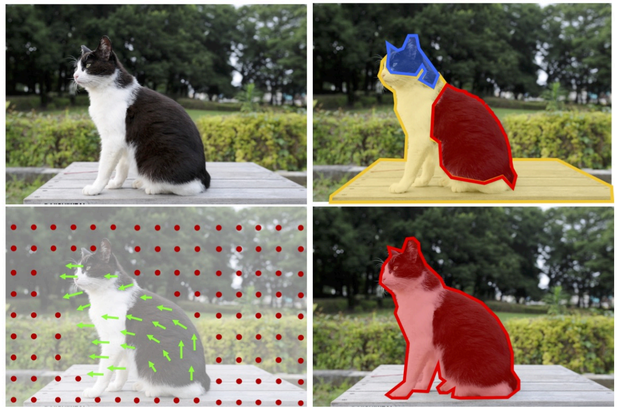


Owens et al. ECCV 2016

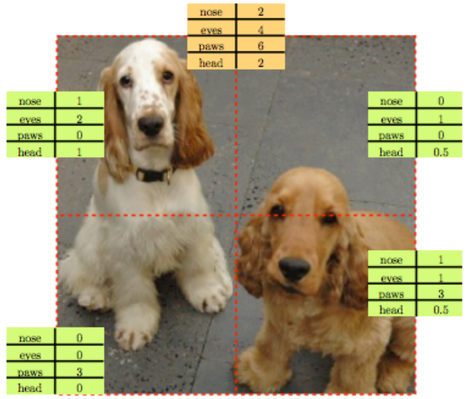
Zhang et al. ECCV 2016



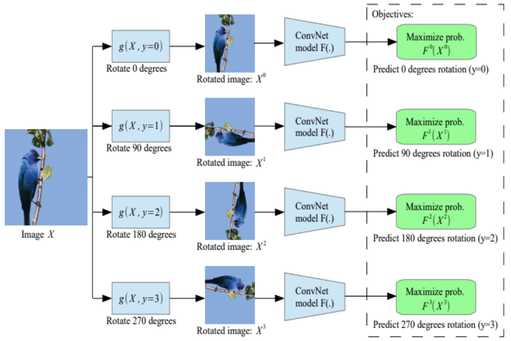
Misra et al. ECCV 2016



Pathak et al. CVPR 2017

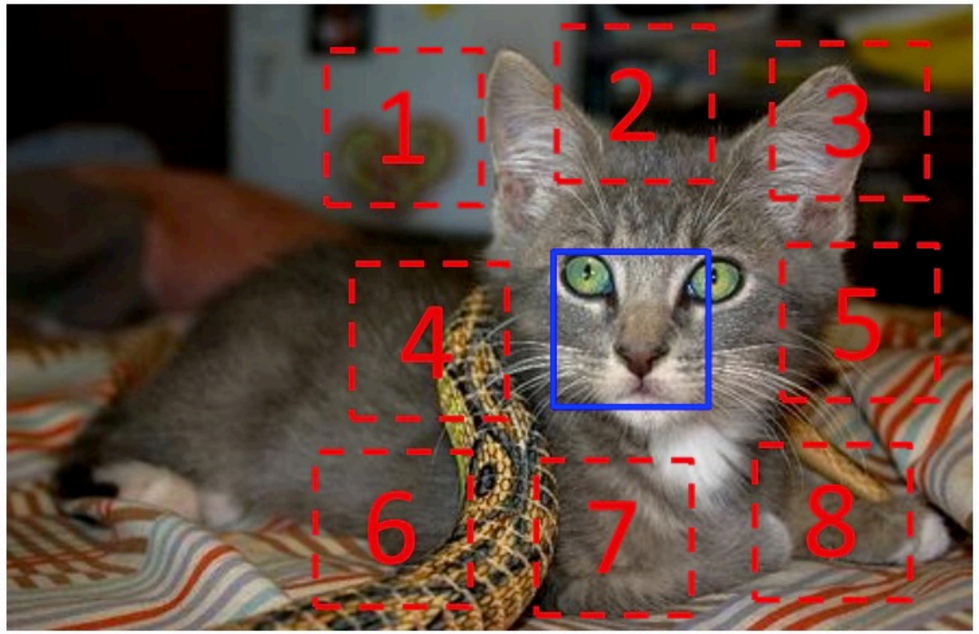


Noroozi et al. ICCV 2017

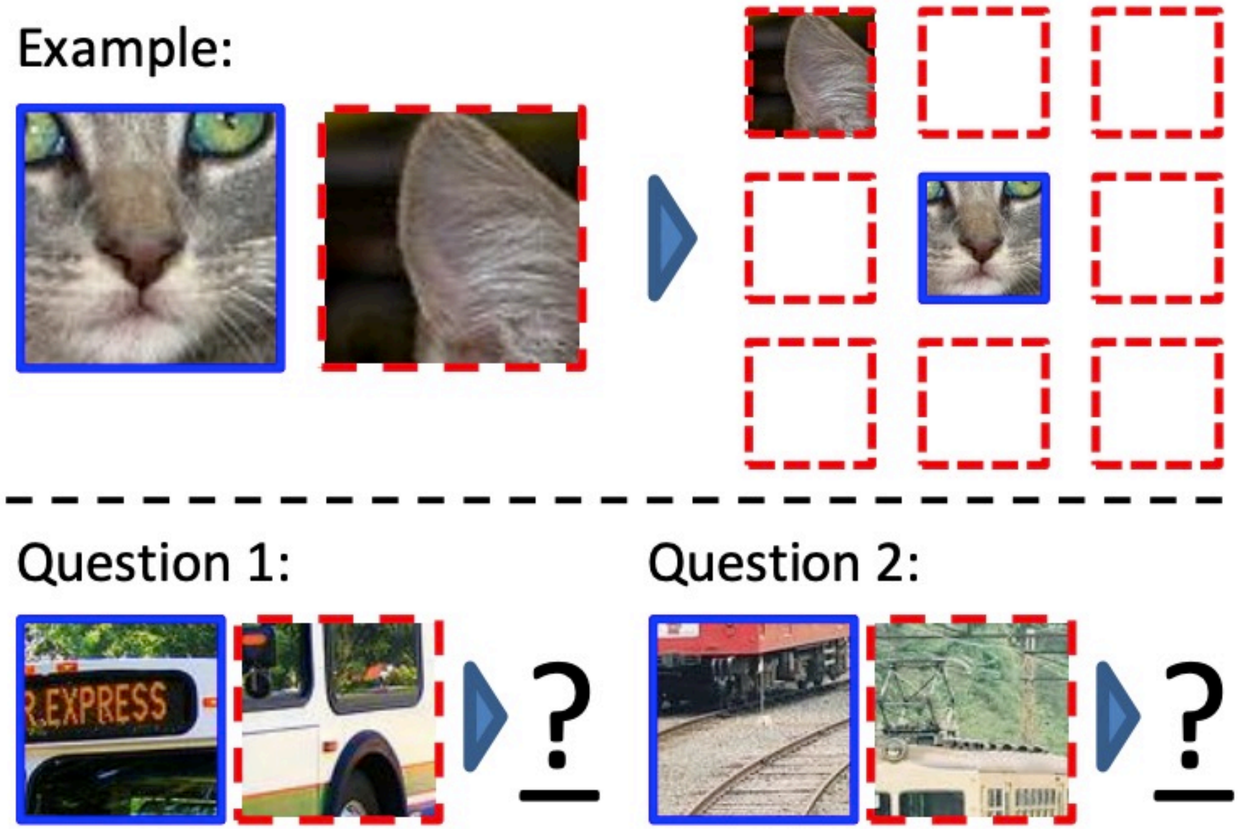


Gidaris et al. ICLR 2018

Self-Supervised Learning with Context Prediction

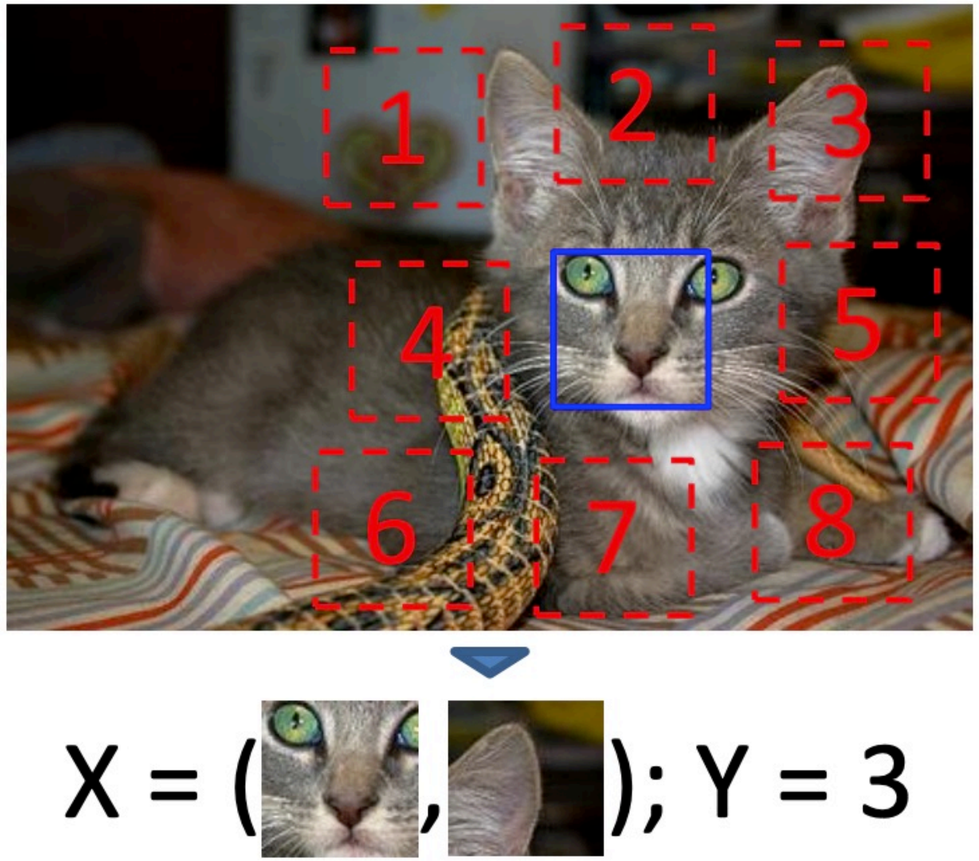


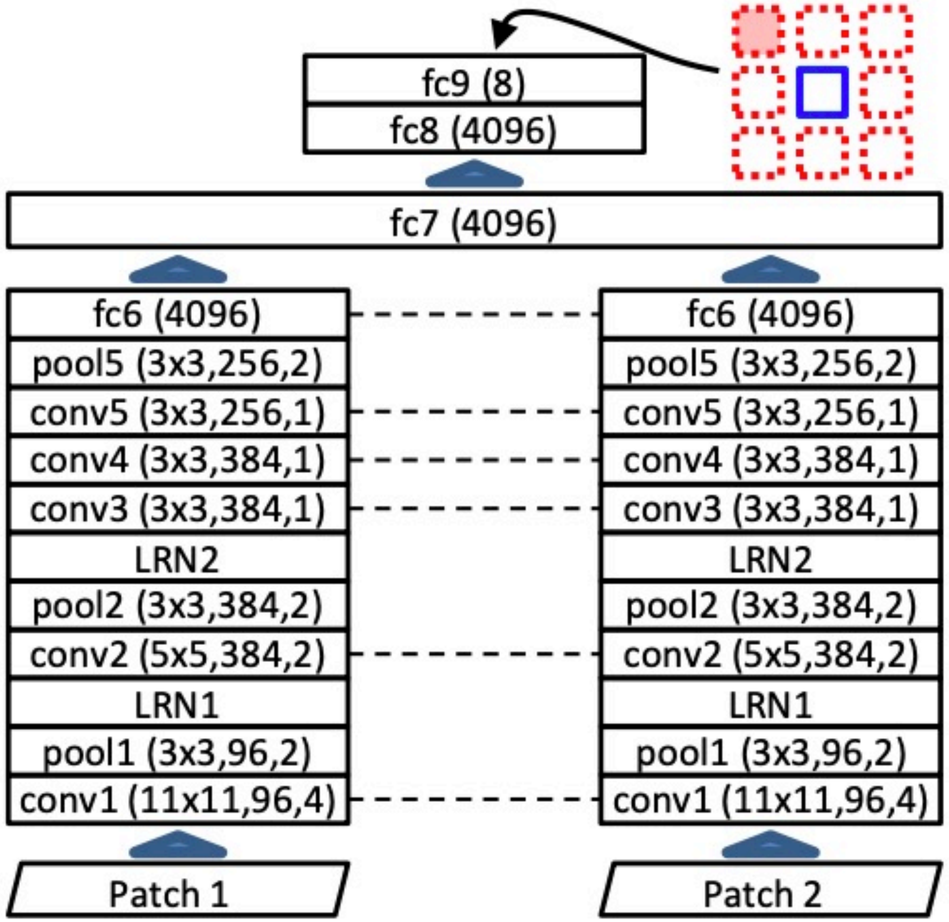
X = (,); Y = 3



[Doersch et al. 2015]

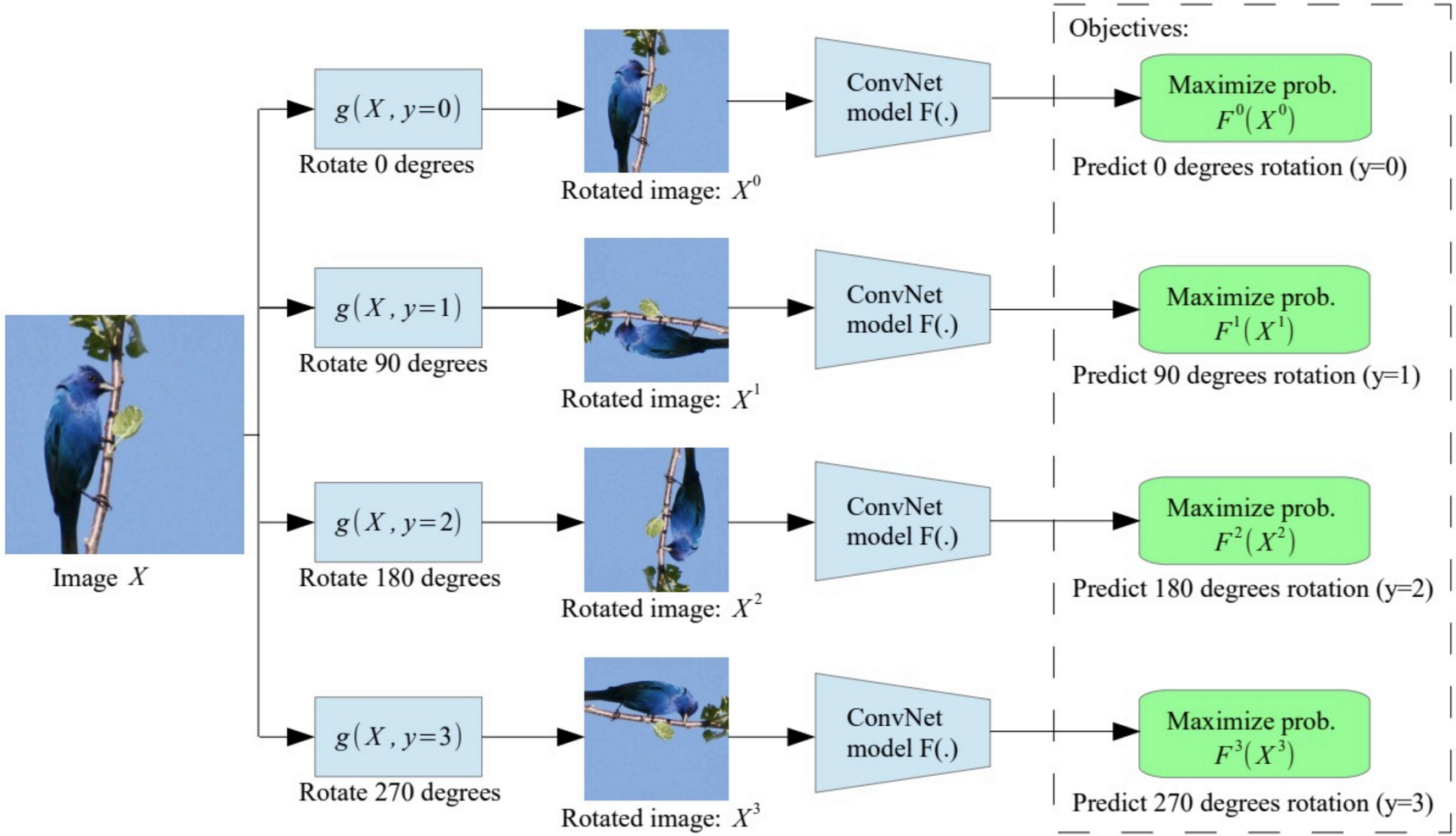
Self-Supervised Learning with Context Prediction





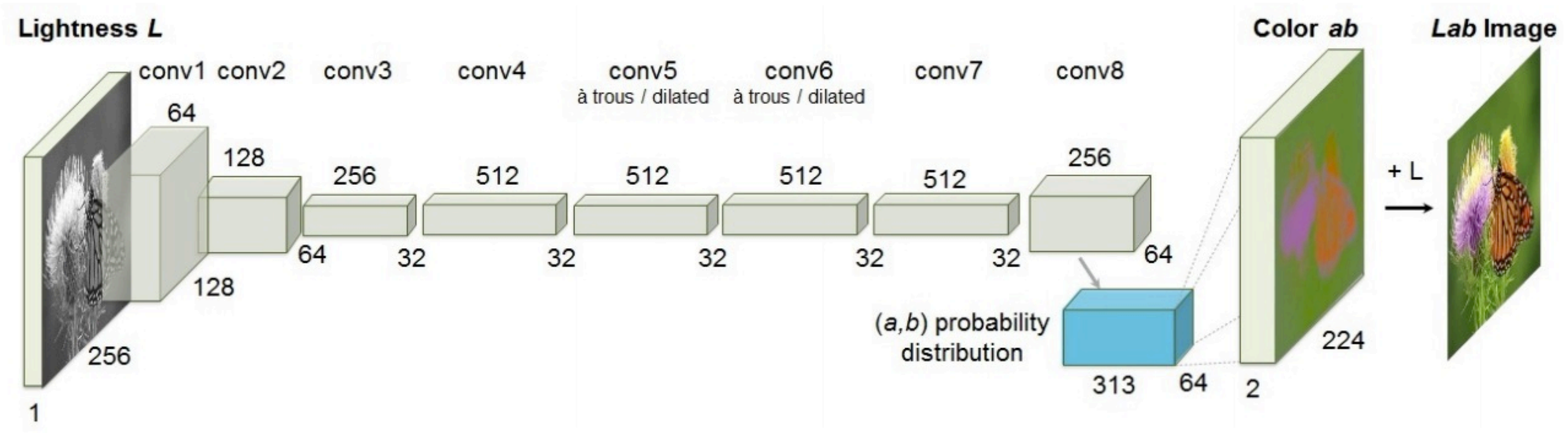
[Doersch et al. 2015]

Self-Supervised Learning with Rotation Prediction



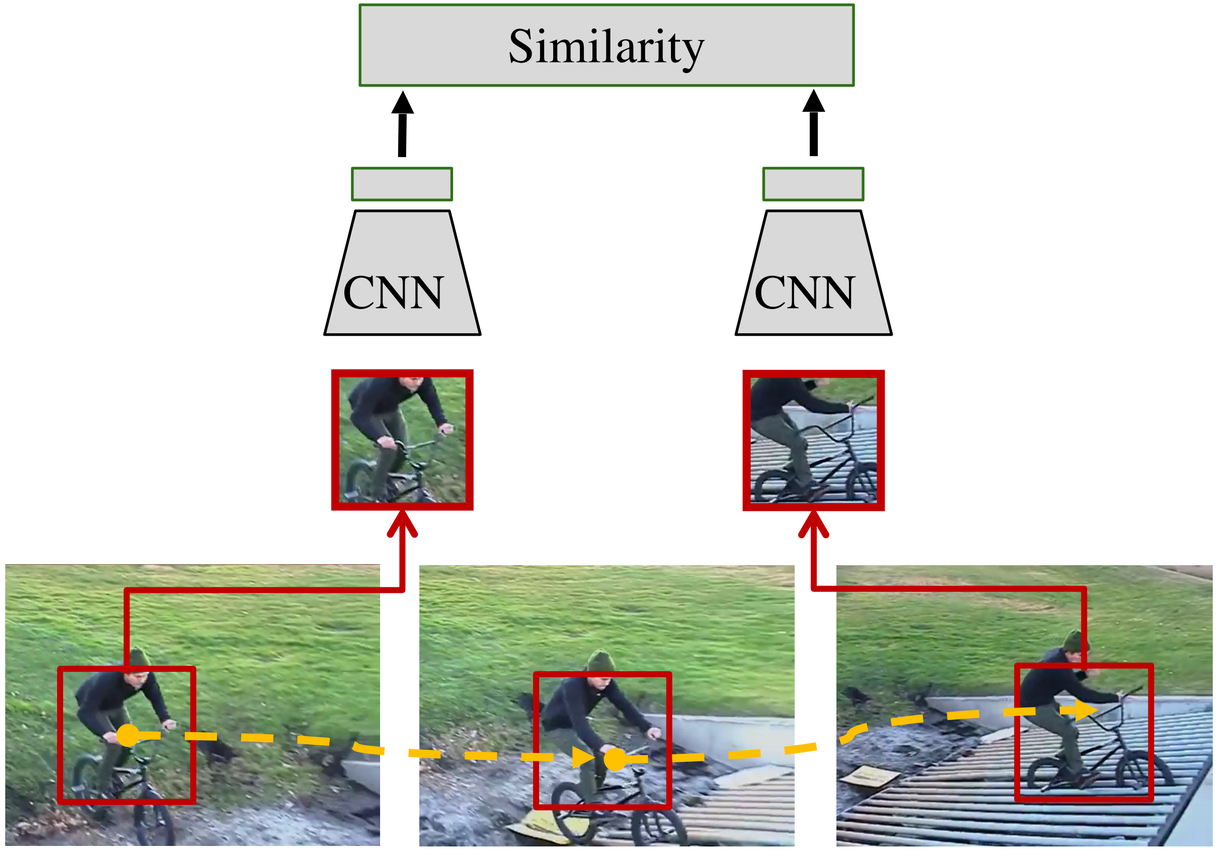
[Gidaris et al. 2018]

Self-Supervised Learning with Image Colorization



[Zhang et al. 2016]

Self-Supervised Learning with Tracking



Tracking → Similarity

[Wang et al. 2015]

How to Evaluate the Representation

- Linear classification protocol
  - Freeze the features (trained neural network)
  - Train an extra supervised linear classifier (a fully-connected layer followed by softmax)

- Transfer feature to downstream tasks by fine-tuning the whole network
  - Object detection
  - Image segmentation

Self-Supervised Learning with Rotation Prediction

Table 5: **Task Generalization: ImageNet top-1 classification with linear layers**. We compare our unsupervised feature learning approach with other unsupervised approaches by training logistic regression classifiers on top of the feature maps of each layer to perform the 1000-way ImageNet classification task, as proposed by Zhang et al. (2016a). All weights are frozen and feature maps are spatially resized (with adaptive max pooling) so as to have around 9000 elements. All approaches use AlexNet variants and were pre-trained on ImageNet without labels except the ImageNet labels and Random entries.

Method	Conv1	Conv2	Conv3	Conv4	Conv5
ImageNet labels	19.3	36.3	44.2	48.3	50.5
Random	11.6	17.1	16.9	16.3	14.1
Random rescaled Krähenbühl et al. (2015)	17.5	23.0	24.5	23.2	20.6
Context (Doersch et al., 2015)	16.2	23.3	30.2	31.7	29.6
Context Encoders (Pathak et al., 2016b)	14.1	20.7	21.0	19.8	15.5
Colorization (Zhang et al., 2016a)	12.5	24.5	30.4	31.5	30.3
Jigsaw Puzzles (Noroozi & Favaro, 2016)	18.2	28.8	34.0	33.9	27.1
BIGAN (Donahue et al., 2016)	17.7	24.5	31.0	29.9	28.0
Split-Brain (Zhang et al., 2016b)	17.7	29.3	35.4	35.2	32.8
Counting (Noroozi et al., 2017)	18.0	30.6	34.3	32.5	25.7
(Ours) RotNet	**18.8**	**31.7**	**38.7**	**38.2**	**36.5**

Self-Supervised Learning with Rotation Prediction

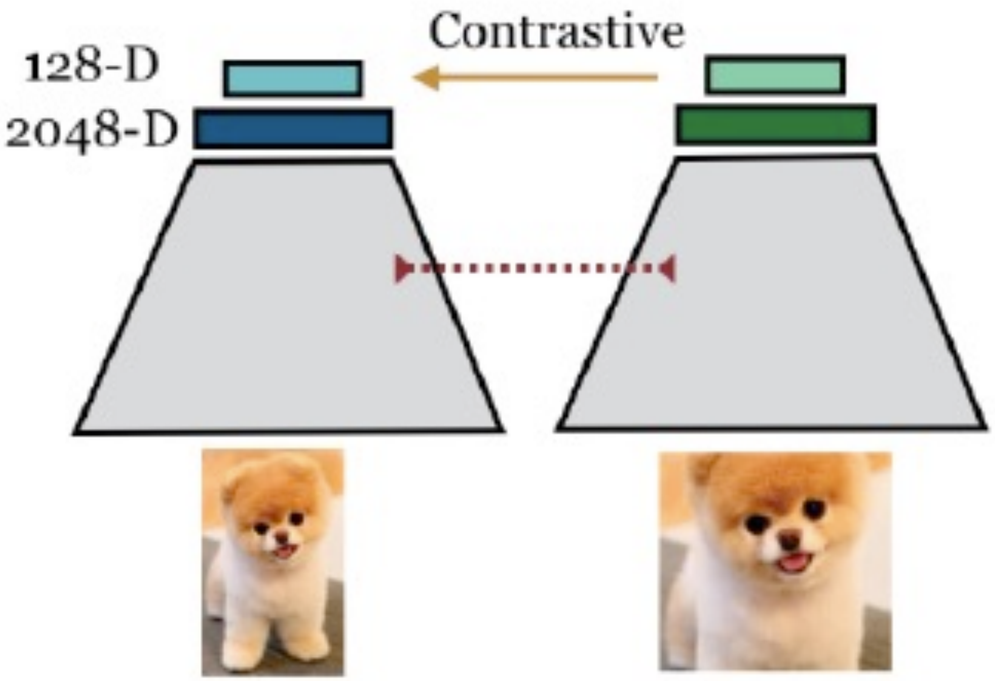
Table 7: **Task & Dataset Generalization: PASCAL VOC 2007 classification and detection results, and PASCAL VOC 2012 segmentation results.** We used the publicly available testing frameworks of Krähenbühl et al. (2015) for classification, of Girshick (2015) for detection, and of Long et al. (2015) for segmentation. For classification, we either fix the features before conv5 (column *fc6-8*) or we fine-tune the whole model (column *all*). For detection we use multi-scale training and single scale testing. All approaches use AlexNet variants and were pre-trained on ImageNet without labels except the ImageNet labels and Random entries. After unsupervised training, we absorb the batch normalization units on the linear layers and we use the weight rescaling technique proposed by Krähenbühl et al. (2015) (which is common among the unsupervised methods). As customary, we report the mean average precision (mAP) on the classification and detection tasks, and the mean intersection over union (mIoU) on the segmentation task.

	Classification (%mAP)		Detection (%mAP)	Segmentation (%mIoU)
Trained layers	fc6-8	all	all	all
ImageNet labels	78.9	79.9	56.8	48.0
Random		53.3	43.4	19.8
Random rescaled Krähenbühl et al. (2015)	39.2	56.6	45.6	32.6
Egomotion (Agrawal et al., 2015)	31.0	54.2	43.9	
Context Encoders (Pathak et al., 2016b)	34.6	56.5	44.5	29.7
Tracking (Wang & Gupta, 2015)	55.6	63.1	47.4	
Context (Doersch et al., 2015)	55.1	65.3	51.1	
Colorization (Zhang et al., 2016a)	61.5	65.6	46.9	35.6
BIGAN (Donahue et al., 2016)	52.3	60.1	46.9	34.9
Jigsaw Puzzles (Noroozi & Favaro, 2016)	-	67.6	53.2	37.6
NAT (Bojanowski & Joulin, 2017)	56.7	65.3	49.4	
Split-Brain (Zhang et al., 2016b)	63.0	67.1	46.7	36.0
ColorProxy (Larsson et al., 2017)		65.9		38.4
Counting (Noroozi et al., 2017)	-	67.7	51.4	36.6
(Ours) RotNet	**70.87**	**72.97**	**54.4**	**39.1**

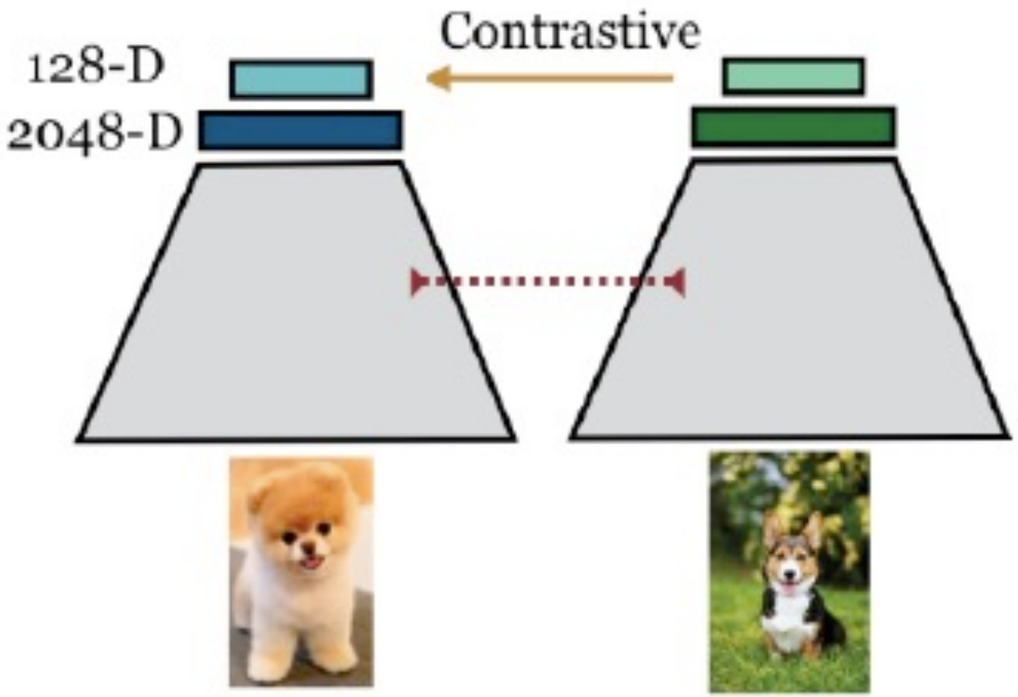
Contrastive Learning

Contrastive Learning

Similar



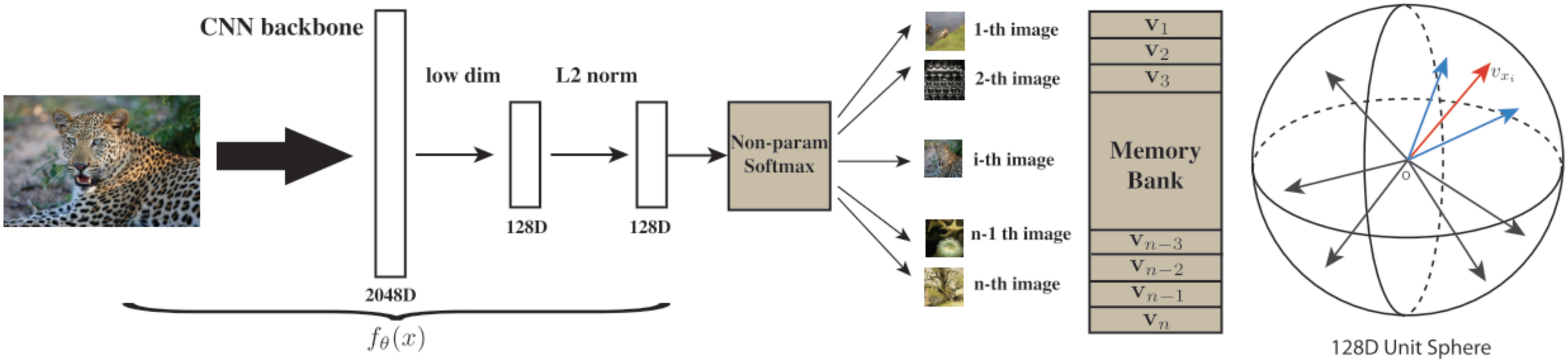
Not Similar



Where do the negative examples come from?

Store Negative Examples in Memory Bank

- The first work using contrastive learning for unsupervised representation learning
- Take about 600MB disk to store embeddings of ImageNet dataset.



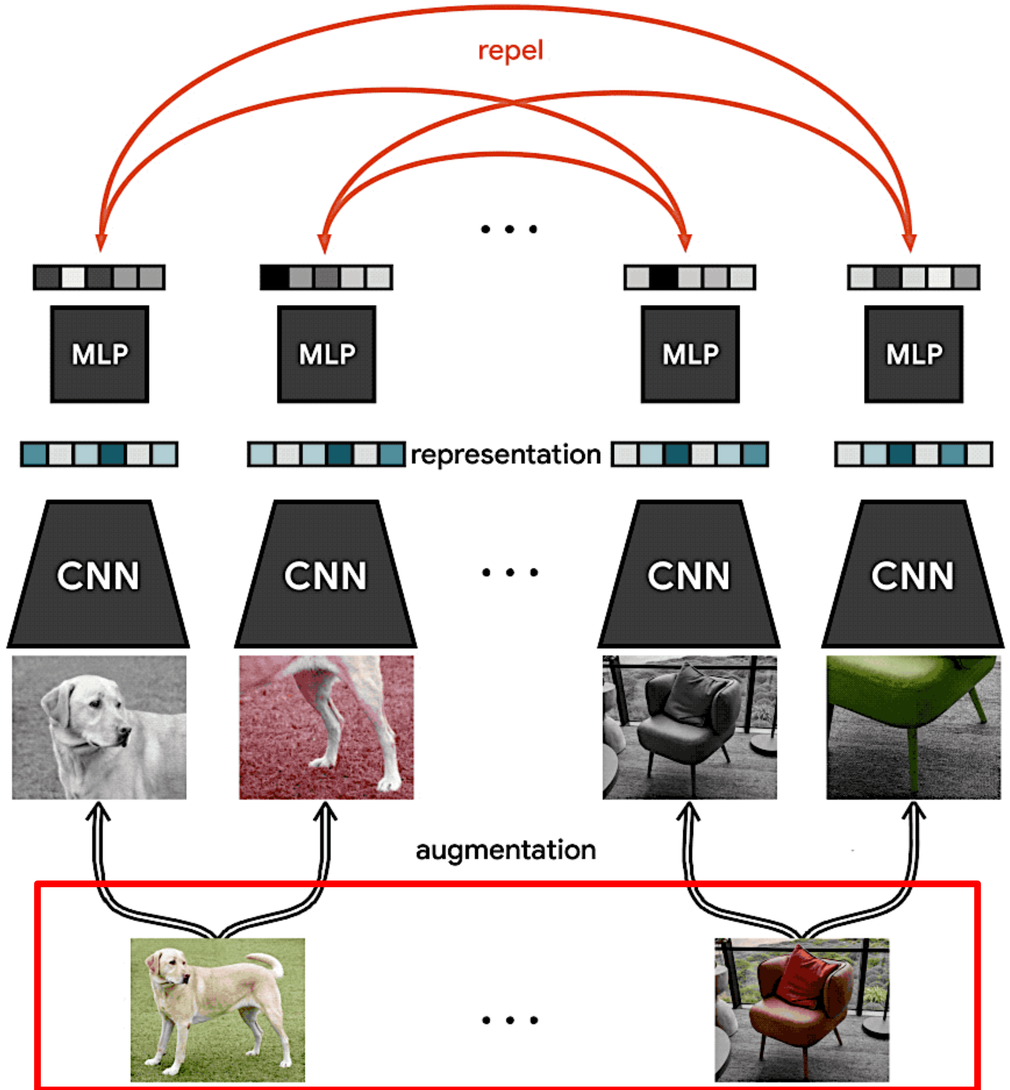
Wu et al. Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination. CVPR 2018.

Without Memory bank, we still have negative examples within batches.

Use Intra-Batch Negative Examples

The improved design choice in SimCLR :

- Strong Data Argumentation
- Non-Linear Transformation
- Larger Batch & More steps



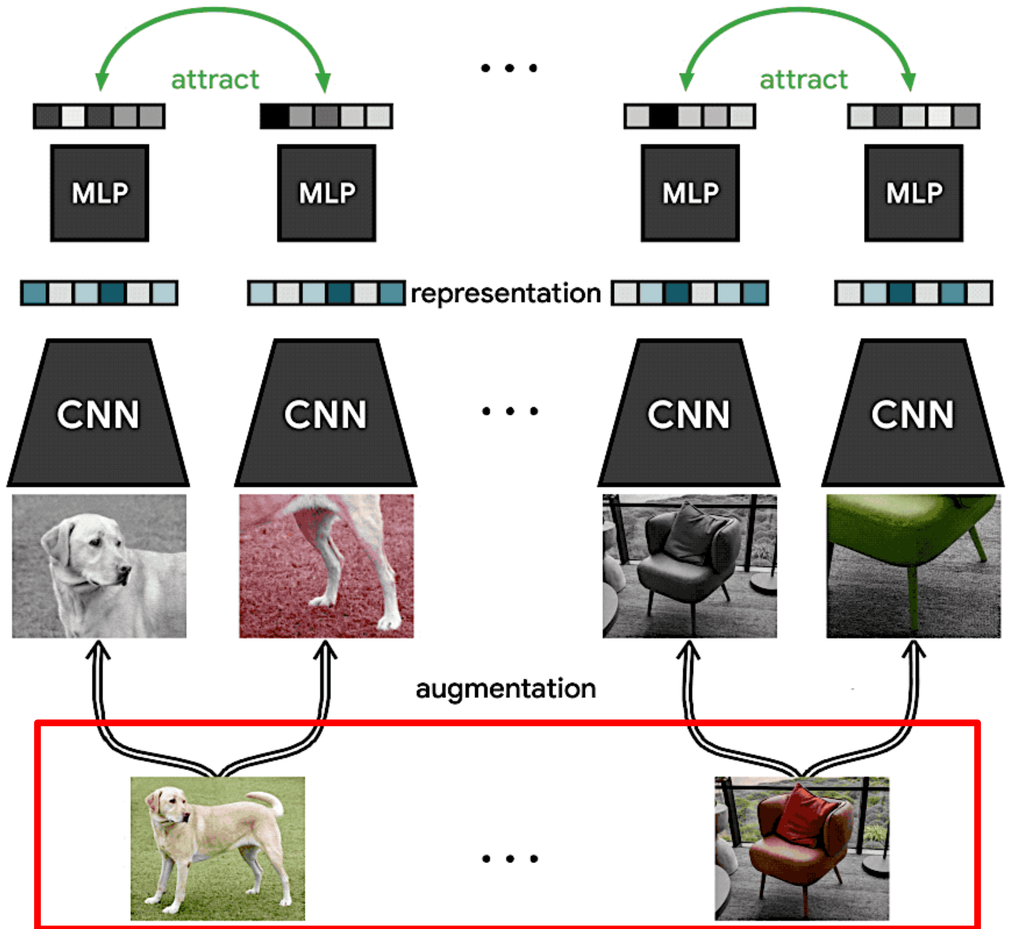
Images in the same batch

Chen et al. 2020

Use Intra-Batch Negative Examples

The improved design choice in SimCLR :

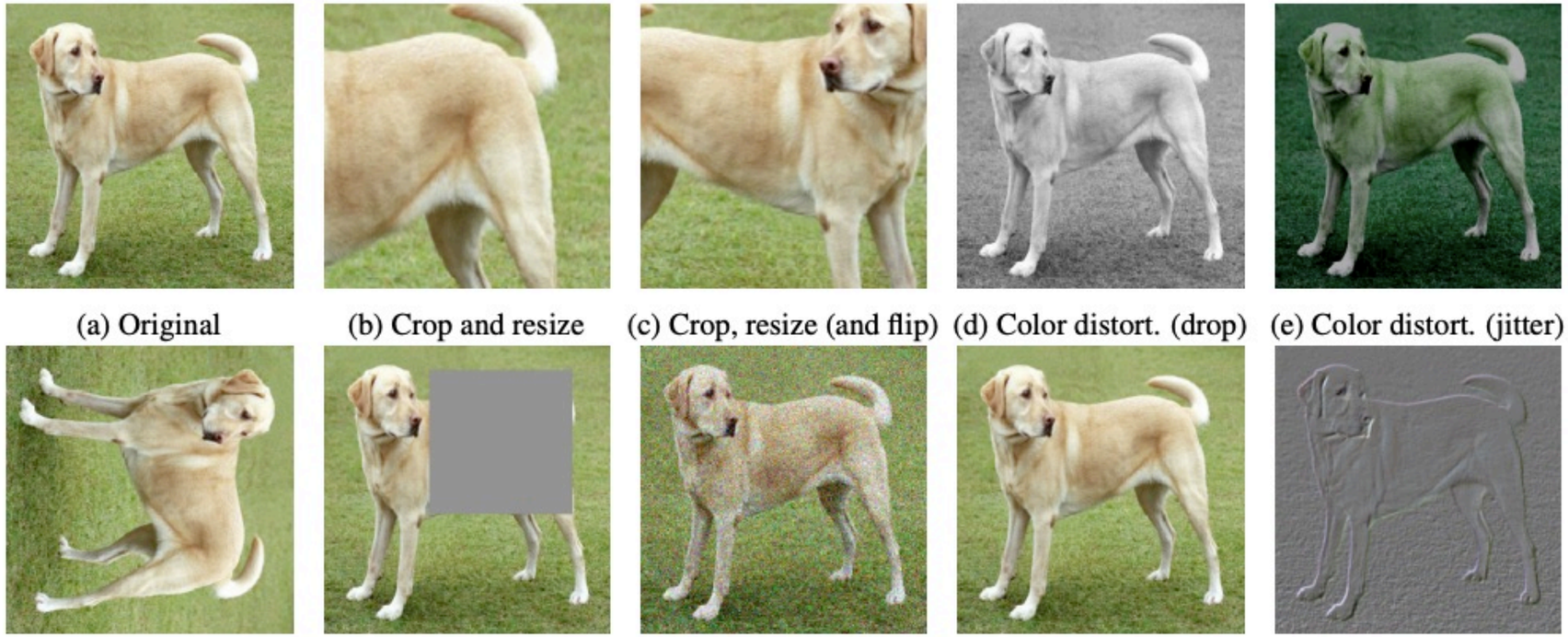
- Strong Data Argumentation
- Non-Linear Transformation
- Larger Batch & More steps



Images in the same batch

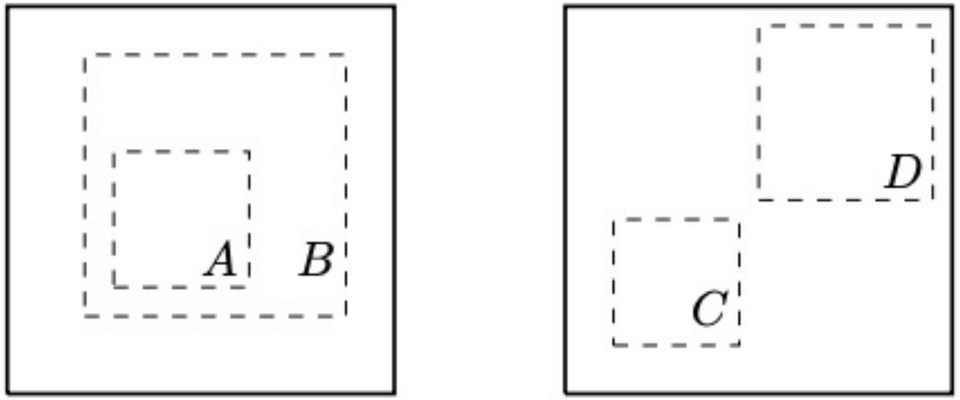
Chen et al. 2020

SimCLR



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering



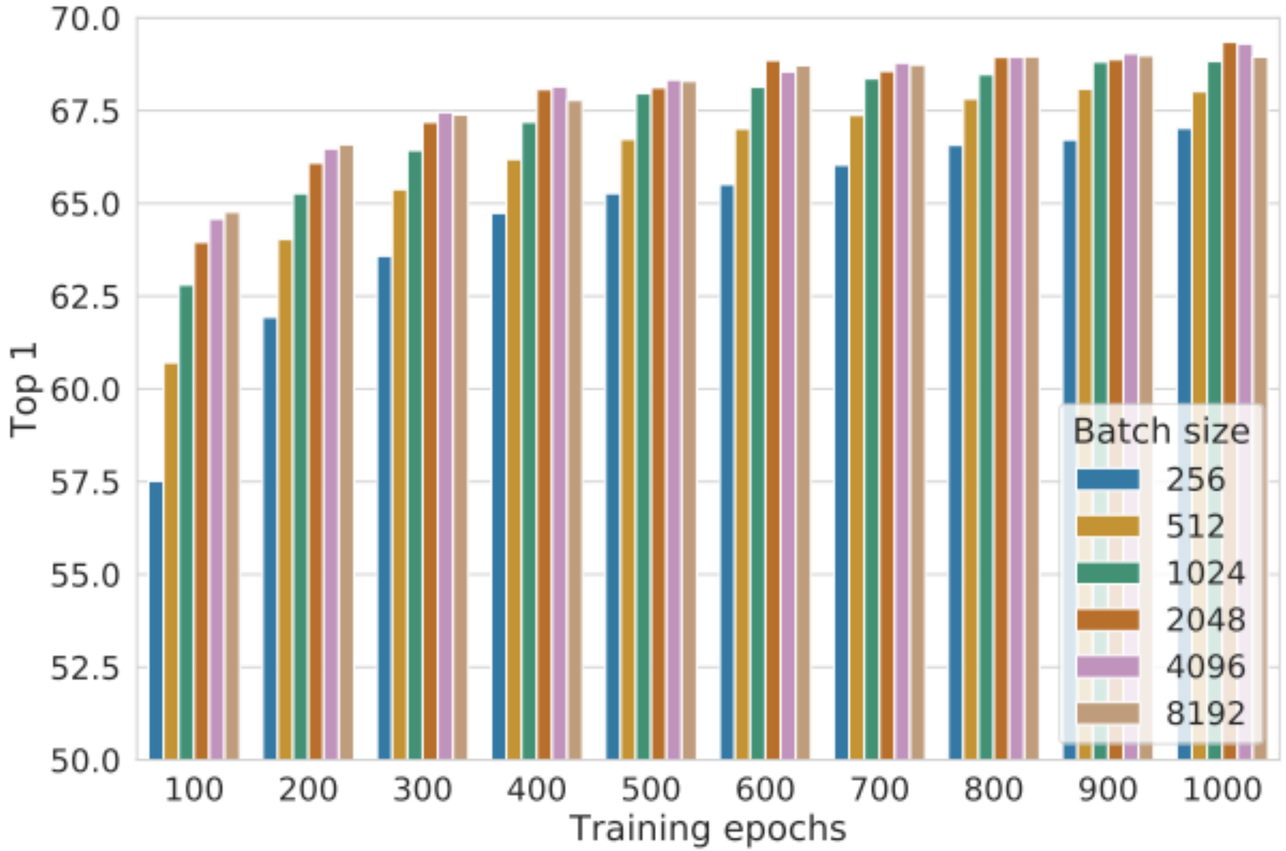
(a) Global and local views. (b) Adjacent views.

Require large number of negatives

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

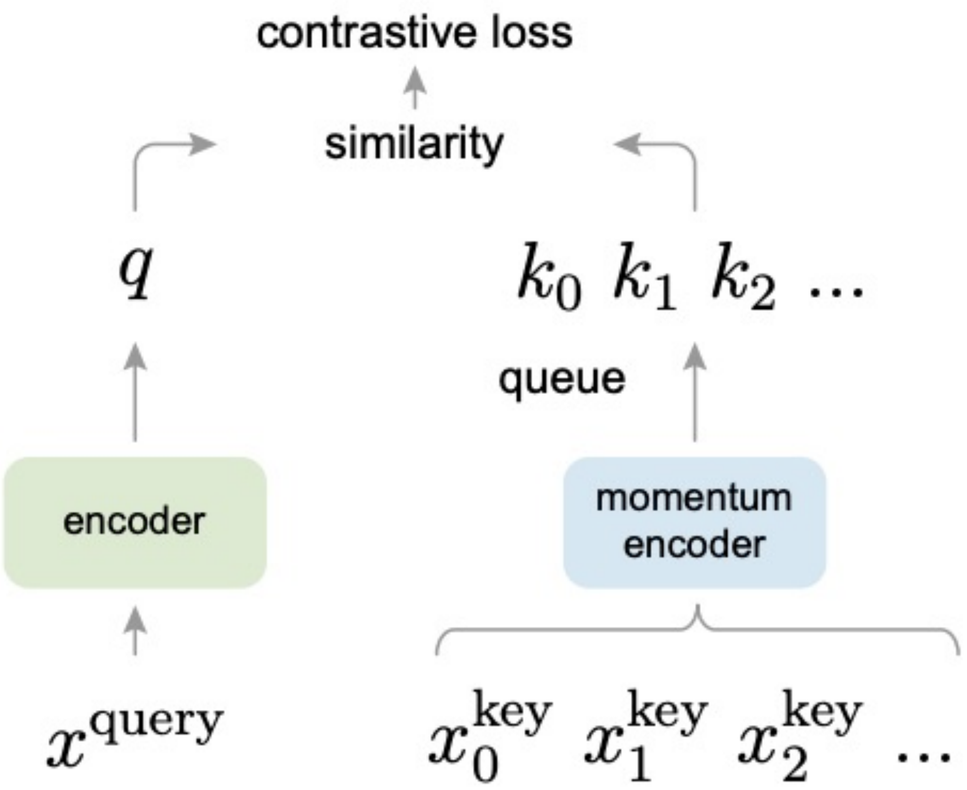
Larger Batch & More Steps

- Larger batch
  - More negative sample pairs in the same batch
- Training longer
  - The gaps between different batch sizes decrease or disappear.



Chen et al. 2020

MoCo



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

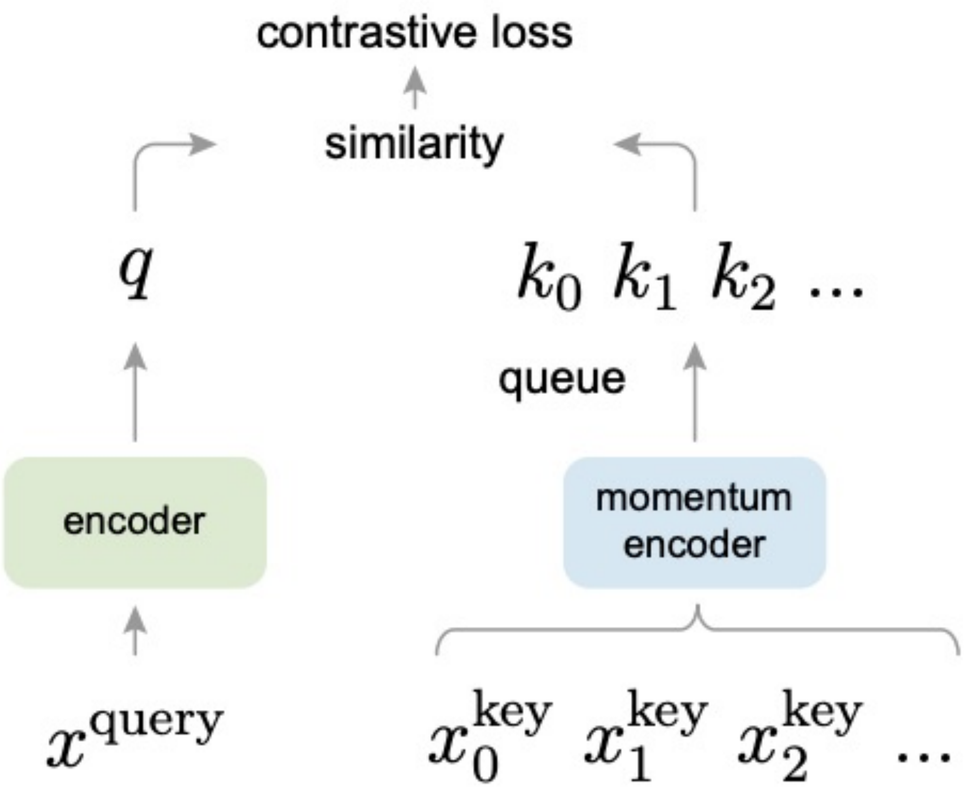
k+ represents the positive paired sample

ki represents one of the K negative samples

K = 60,000

He et al. Momentum Contrast for Unsupervised Visual Representation Learning. 2020.

MoCo



$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1-m)\theta_{\mathrm{q}}$$

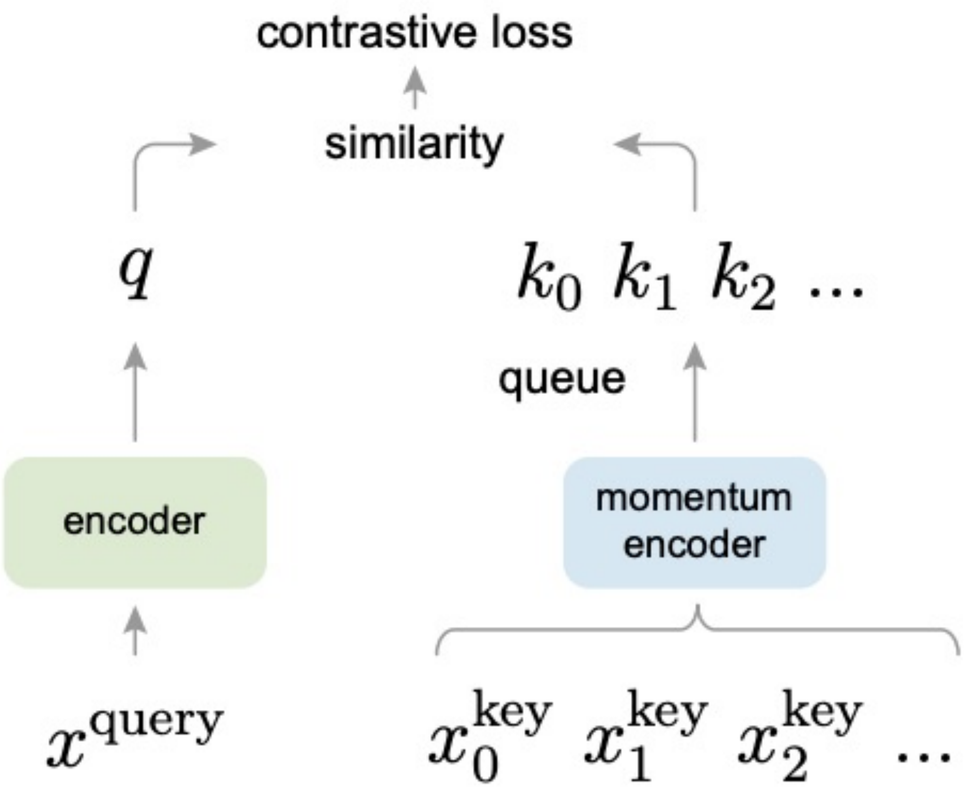
Momentum encoder is a moving average of the encoder

$m = 0.999$

Momentum encoder does not receive gradients from the loss.

He et al. Momentum Contrast for Unsupervised Visual Representation Learning. 2020.

MoCo



Since the momentum encoder changes very slowly. We can maintain a queue to store the negative features.

A queue has K=60,000 examples, each example has 512 dimensions.

Suppose the batch size for each iteration is 256. We will extract the image features and add the 256 features to the queue, and pop out the oldest 256 examples.

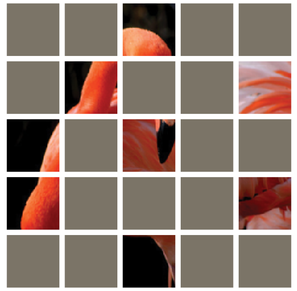
He et al. Momentum Contrast for Unsupervised Visual Representation Learning. 2020.

MoCo

case	unsup. pre-train MLP	aug+	cos	epochs	ImageNet acc.	VOC detection AP_{50}	AP	AP_{75}
supervised					76.5	81.3	53.5	58.8
MoCo v1				200	60.6	81.5	55.9	62.6
(a)	✓			200	66.2	82.0	56.4	62.6
(b)		✓		200	63.4	82.2	56.8	63.2
(c)	✓	✓		200	67.3	**82.5**	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	**800**	**71.1**	**82.5**	**57.4**	**64.0**

Masked Autoencoder

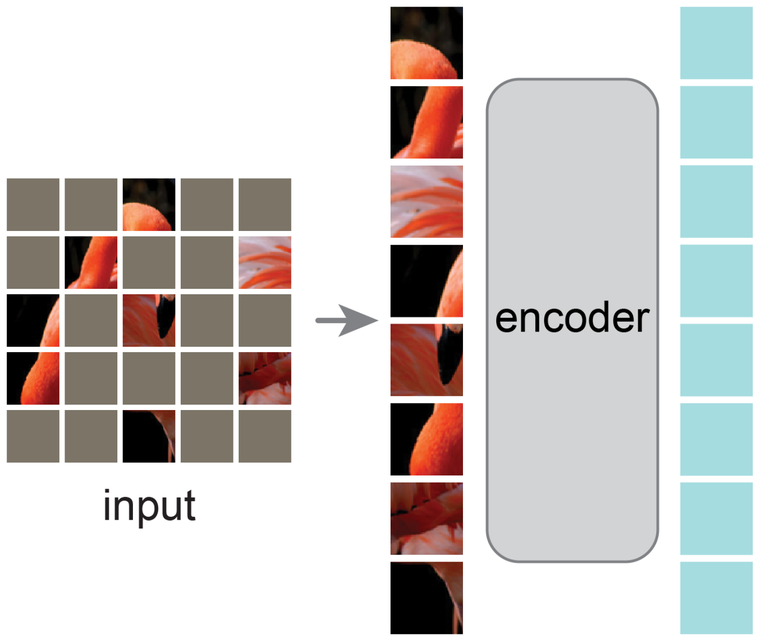
Masked Autoencoder



random masking

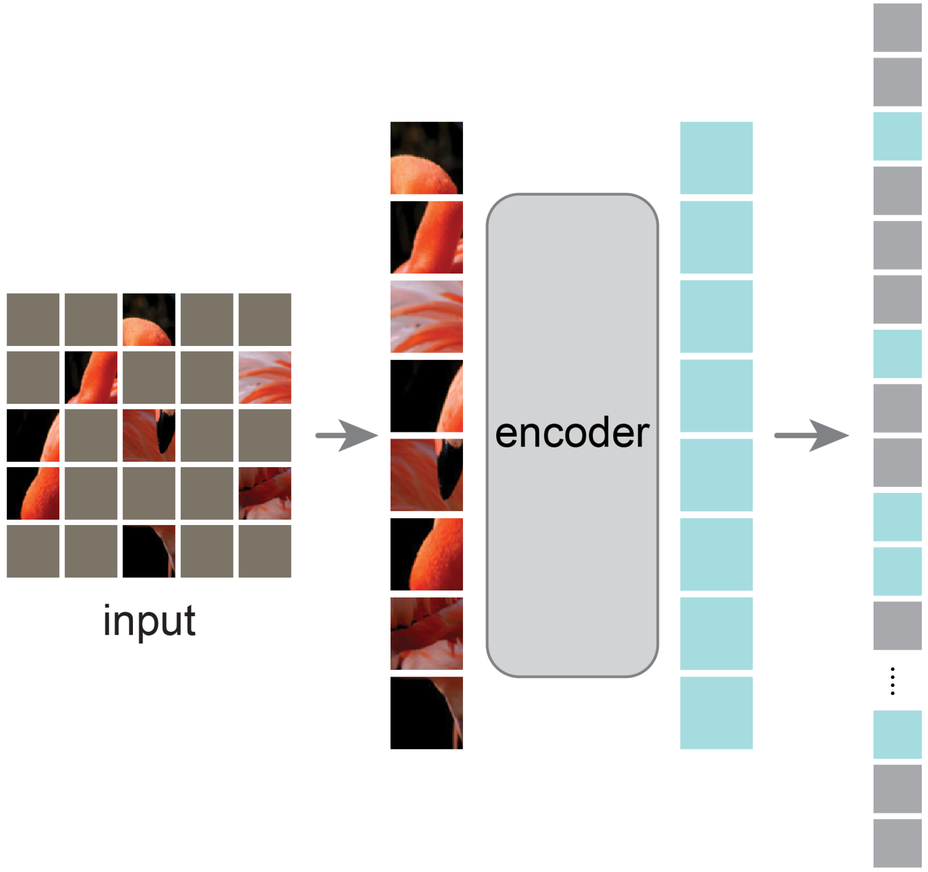
Slides credits: Kaiming He

Masked Autoencoder



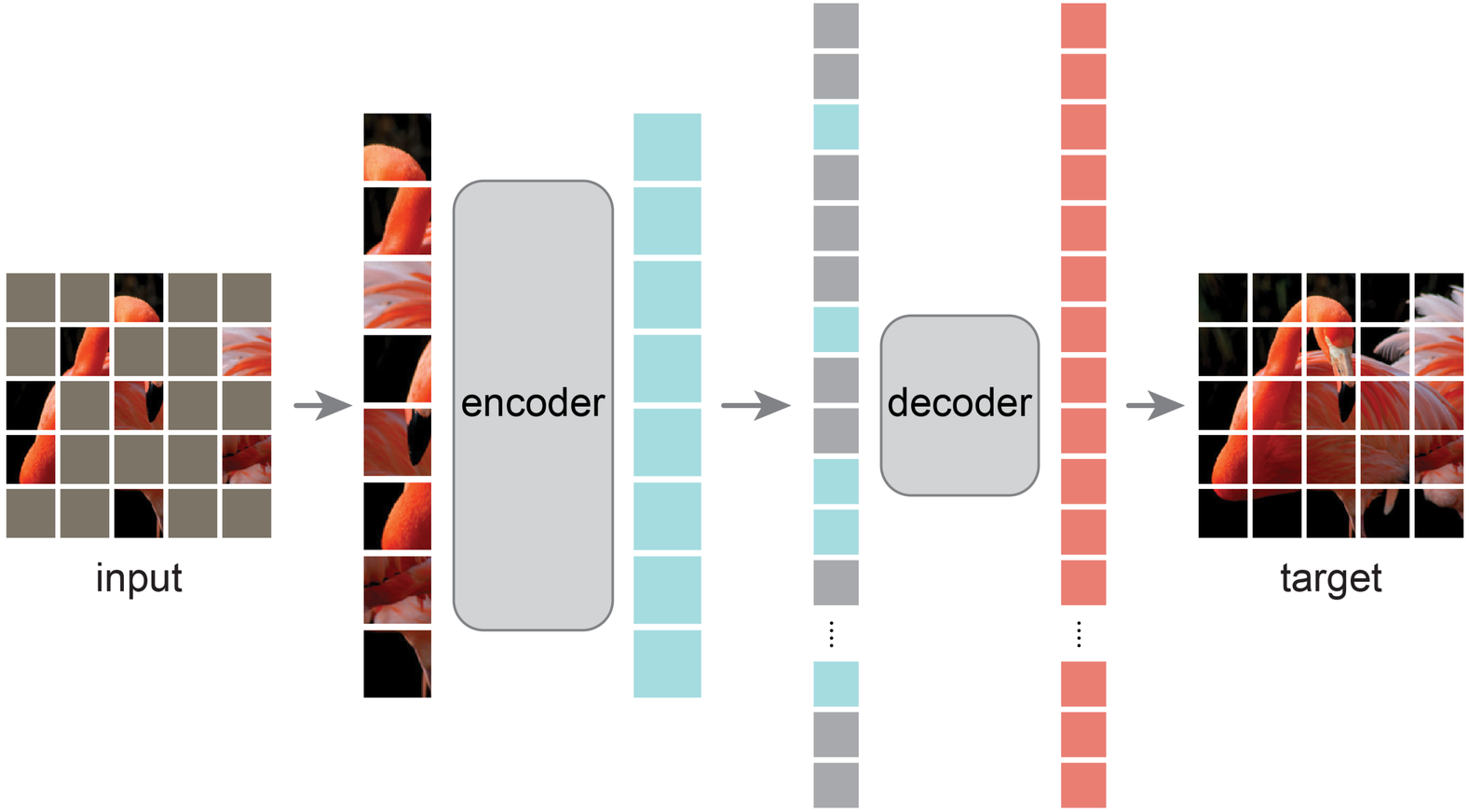
encode visible patches

Masked Autoencoder



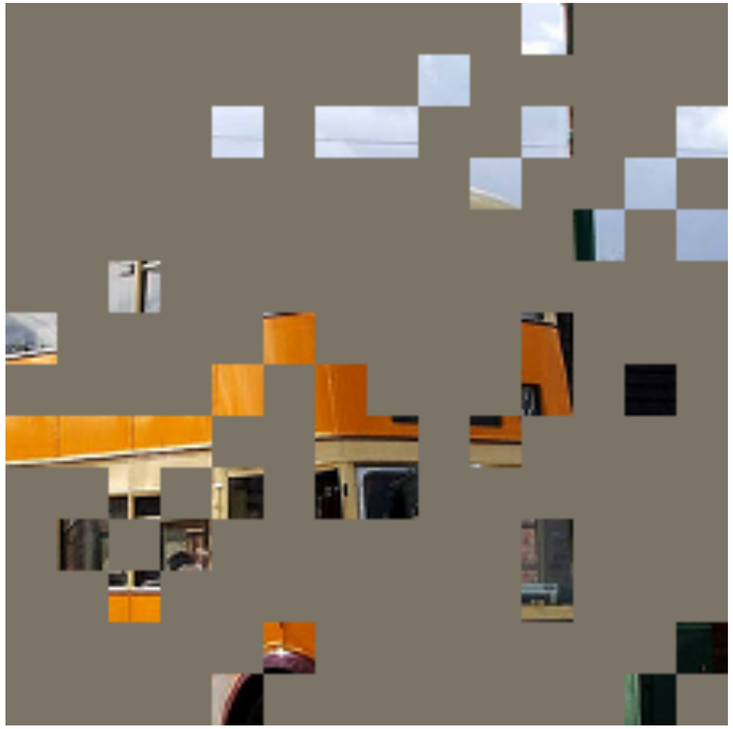
add mask tokens

Masked Autoencoder



reconstruct

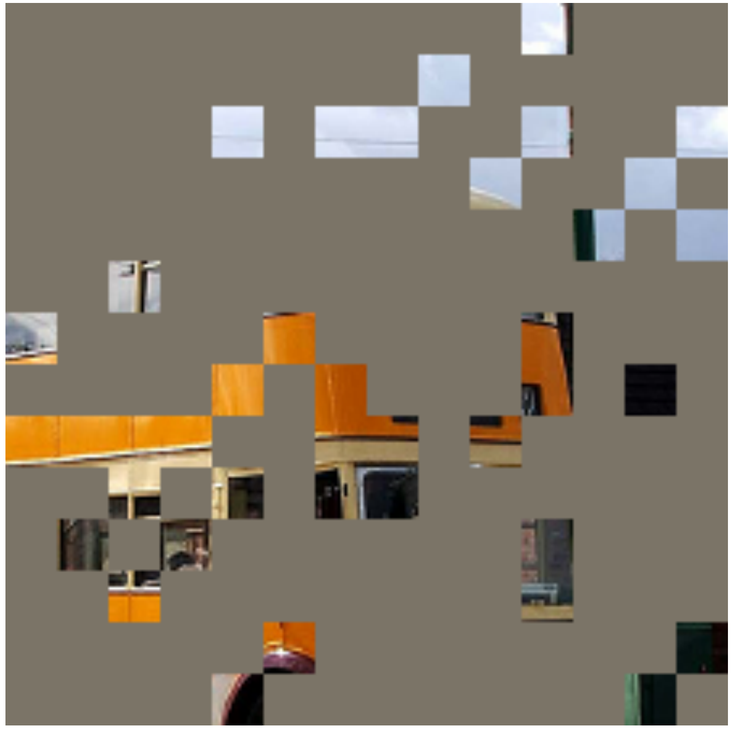
Example Reconstructions



mask 80%

?

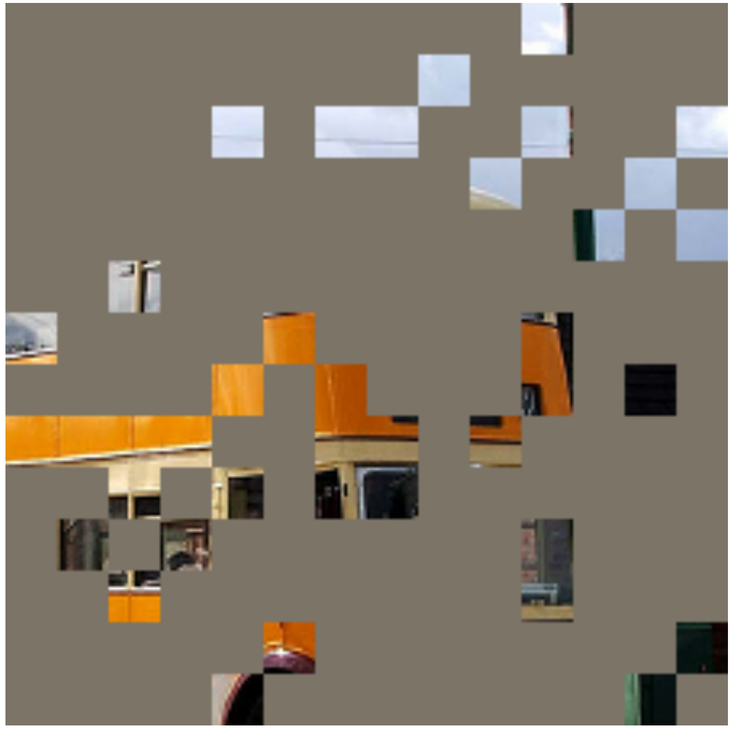
Example Reconstructions



mask 80%

reconstruction

Example Reconstructions



mask 80%

reconstruction

ground-truth

Example Reconstructions

mask 80%

?

Example Reconstructions

mask 80%

reconstruction

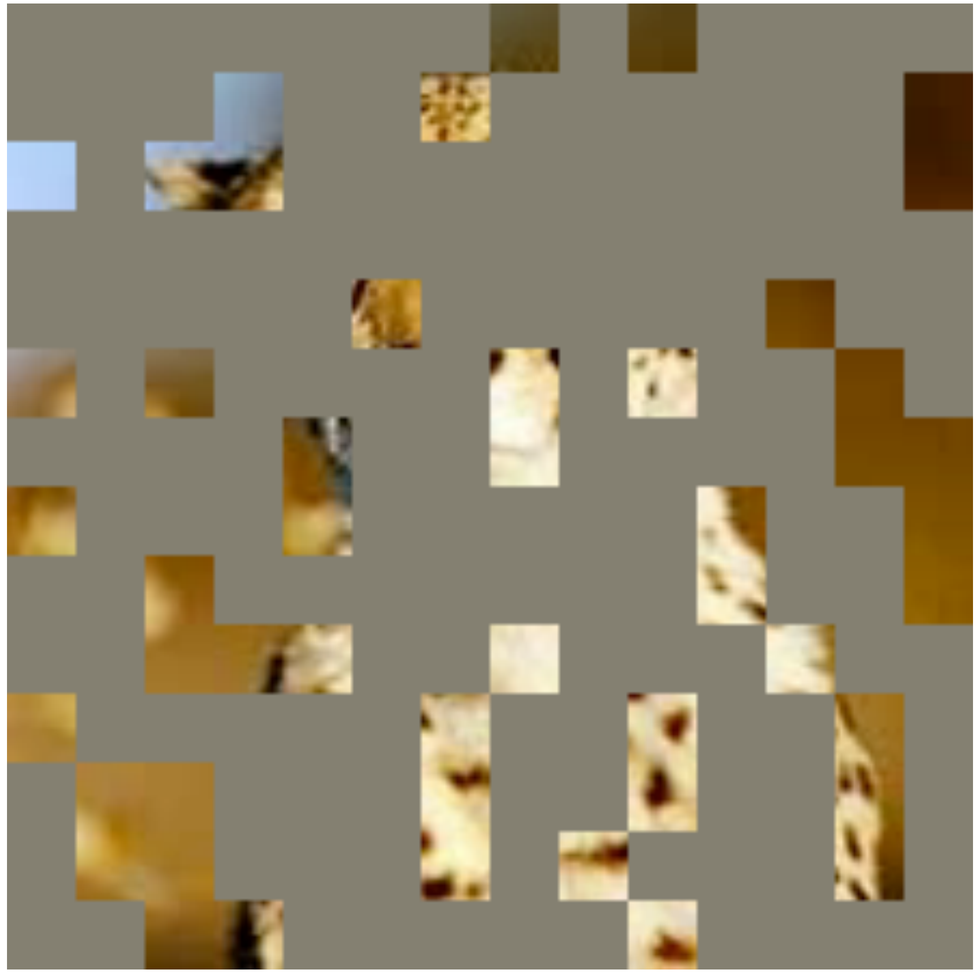
Example Reconstructions

mask 80%

reconstruction

ground-truth

reconstruction vs. # epochs

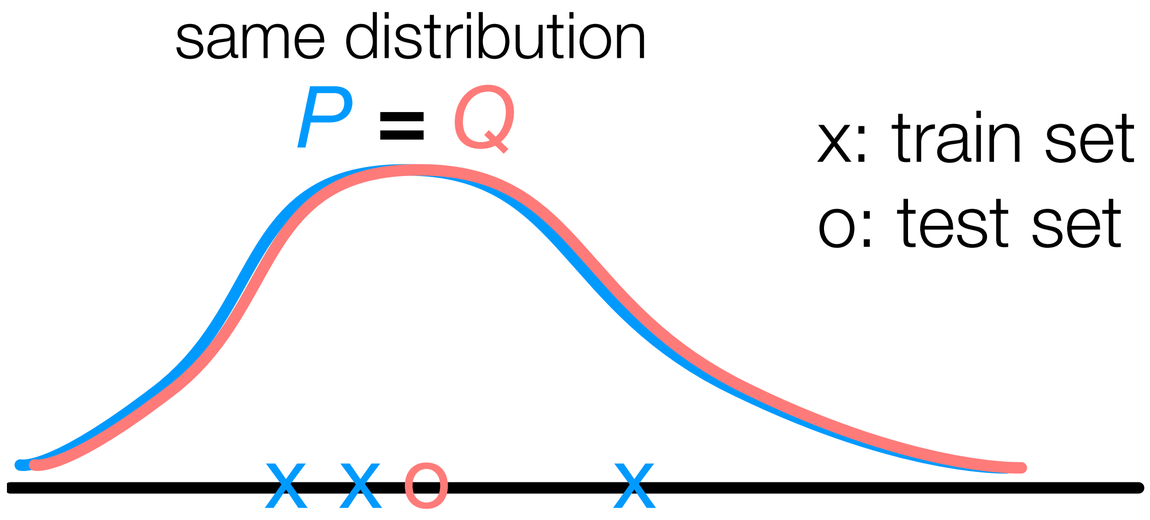


reconstruction vs. # epochs

reconstruction vs. # epochs

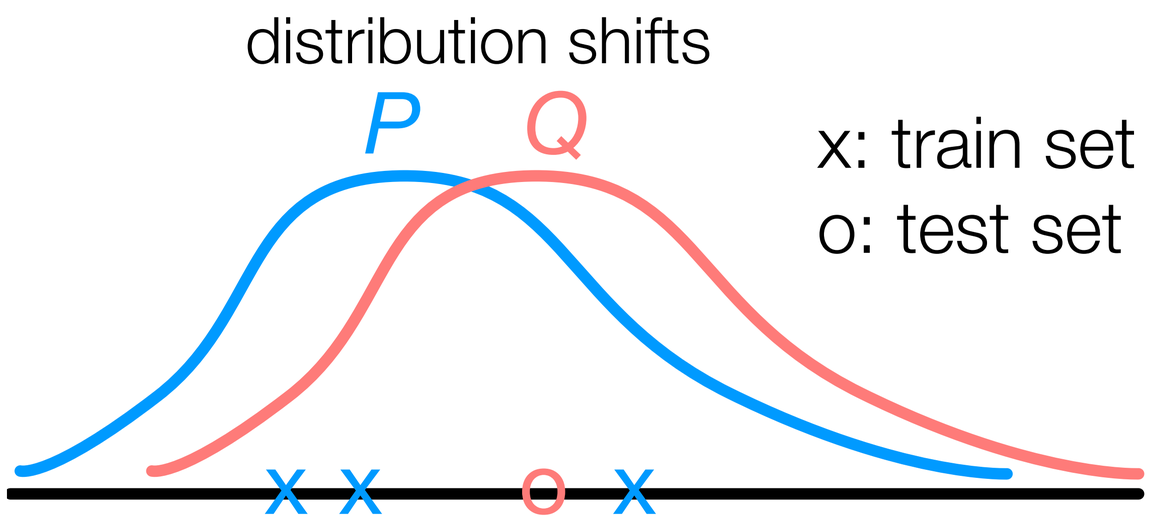
reconstruction vs. # epochs

A general framework using Self-Supervised Learning to improve supervised task in test time



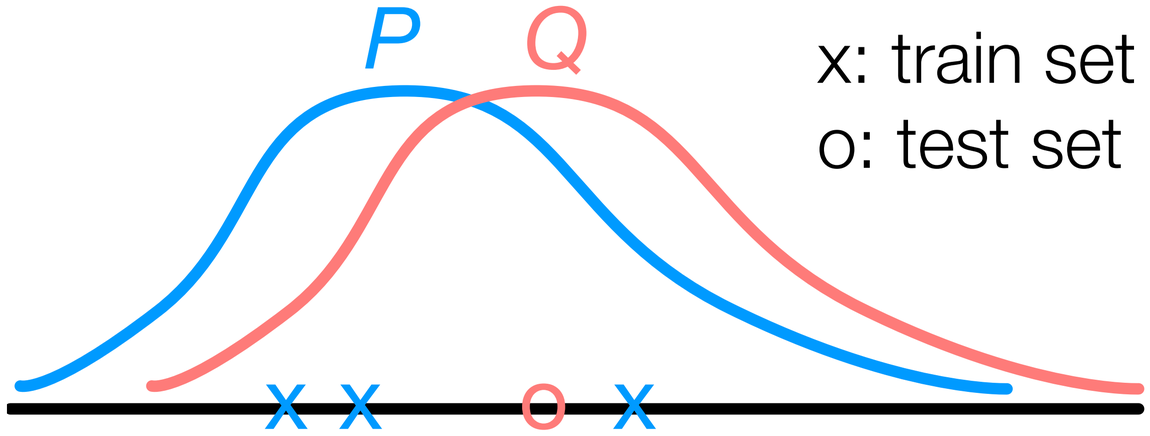
- **In theory**: same distribution for training and testing

Sun et al. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. ICML 2020.

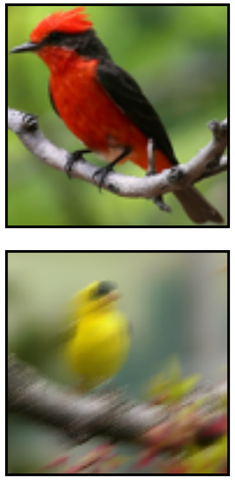


- **In theory**: same distribution for training and testing
- **In the real word**: distribution shifts are everywhere

distribution shifts



- **In theory**: same distribution for training and testing
- **In the real word**: distribution shifts are everywhere



Hendrycks and Dietterich, 2018

Recht, Roelofs, Schmidt and Shankar, 2019

Test-Time Training (TTT)

$$\text{standard test error} = \mathbb{E}_Q[\ell(x, y);\ \theta]$$

- Does not anticipate the test distribution
- The test sample x gives us a hint about Q

Test-Time Training (TTT)

$$\text{standard test error} = \mathbb{E}_Q[\ell(x, y);\ \theta]$$

$$\text{our test error} = \mathbb{E}_Q[\ell(x, y);\ \theta(x)]$$

- Does not anticipate the test distribution
- The test sample x gives us a hint about Q
- No fixed model, but adapt at test time

Test-Time Training (TTT)

$$\text{standard test error} = \mathbb{E}_Q[\ell(x, y);\ \theta]$$

$$\text{our test error} = \mathbb{E}_Q[\ell(x, y);\ \theta(x)]$$

- Does not anticipate the test distribution
- The test sample x gives us a hint about Q
- No fixed model, but adapt at test time
- One sample learning problem
- No label? Self-supervision!

Rotation prediction as self-supervision

(Gidaris et al. 2018)

x

- Create labels from unlabeled input

Unsupervised Representation Learning by Predicting Image Rotations
Gidaris, Singh and Komodakis, 2018

Rotation prediction as self-supervision

(Gidaris et al. 2018)

x	y_{s}
	0º
	90º
	180º
	270º

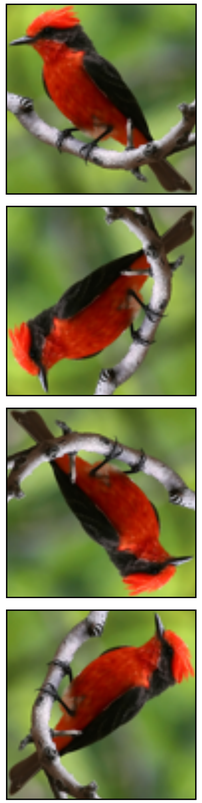
- Create labels from unlabeled input
- Rotate input image by multiples of 90º

Unsupervised Representation Learning by Predicting Image Rotations
Gidaris, Singh and Komodakis, 2018

Rotation prediction as self-supervision

(Gidaris et al. 2018)

x



CNN

θ

y_{s}

0°

90°

180°

270°

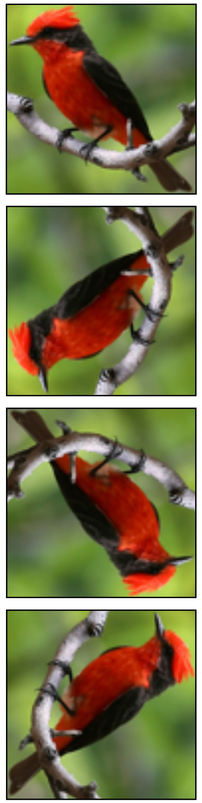
- Create labels from unlabeled input
- Rotate input image by multiples of 90°
- Produce a four-way classification problem

Unsupervised Representation Learning by Predicting Image Rotations
Gidaris, Singh and Komodakis, 2018

Rotation prediction as self-supervision

(Gidaris et al. 2018)

x



θ_{e} θ_{s}

y_{s}

0°

90°

180°

270°

- Create labels from unlabeled input
- Rotate input image by multiples of 90°
- Produce a four-way classification problem
- Usually a pre-training step

Unsupervised Representation Learning by Predicting Image Rotations
Gidaris, Singh and Komodakis, 2018

Rotation prediction as self-supervision

(Gidaris et al. 2018)

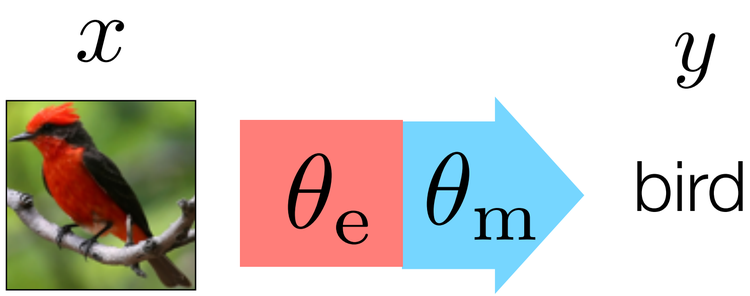
θ_{e}

- Create labels from unlabeled input
- Rotate input image by multiples of 90º
- Produce a four-way classification problem
- Usually a pre-training step
  - After training, take feature extractor

Unsupervised Representation Learning by Predicting Image Rotations
Gidaris, Singh and Komodakis, 2018

Rotation prediction as self-supervision

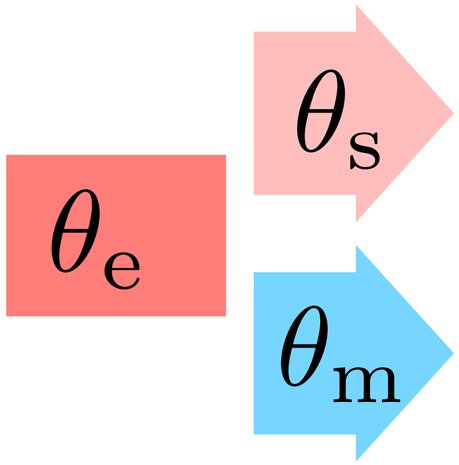
(Gidaris et al. 2018)



- Create labels from unlabeled input
- Rotate input image by multiples of 90°
- Produce a four-way classification problem
- Usually a pre-training step
  - After training, take feature extractor
  - Use it for a downstream main task

Unsupervised Representation Learning by Predicting Image Rotations
Gidaris, Singh and Komodakis, 2018

Algorithm for TTT

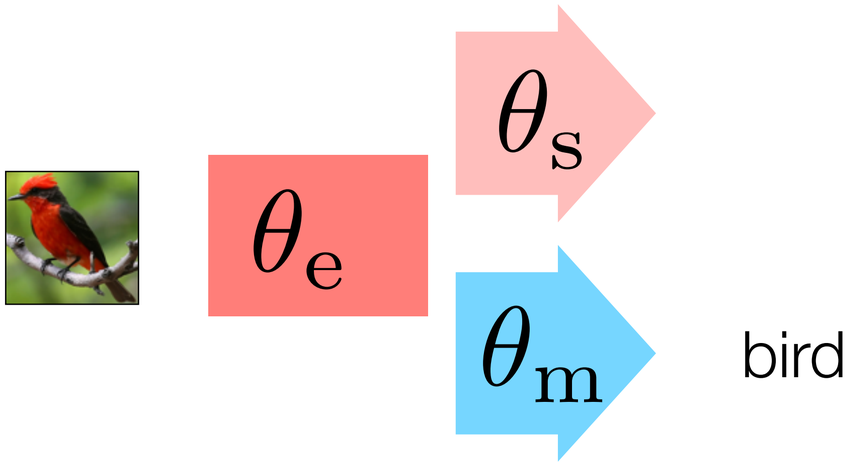


network
architecture

Sun et al. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. ICML 2020.

Algorithm for TTT

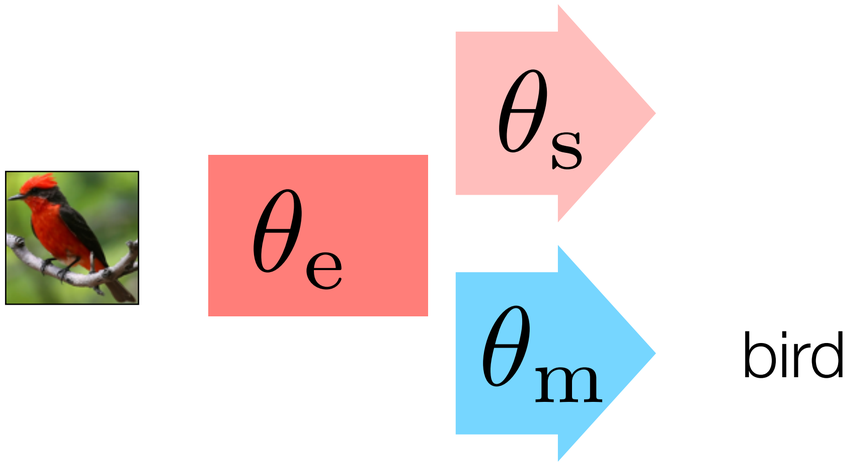
training



Algorithm for TTT

training

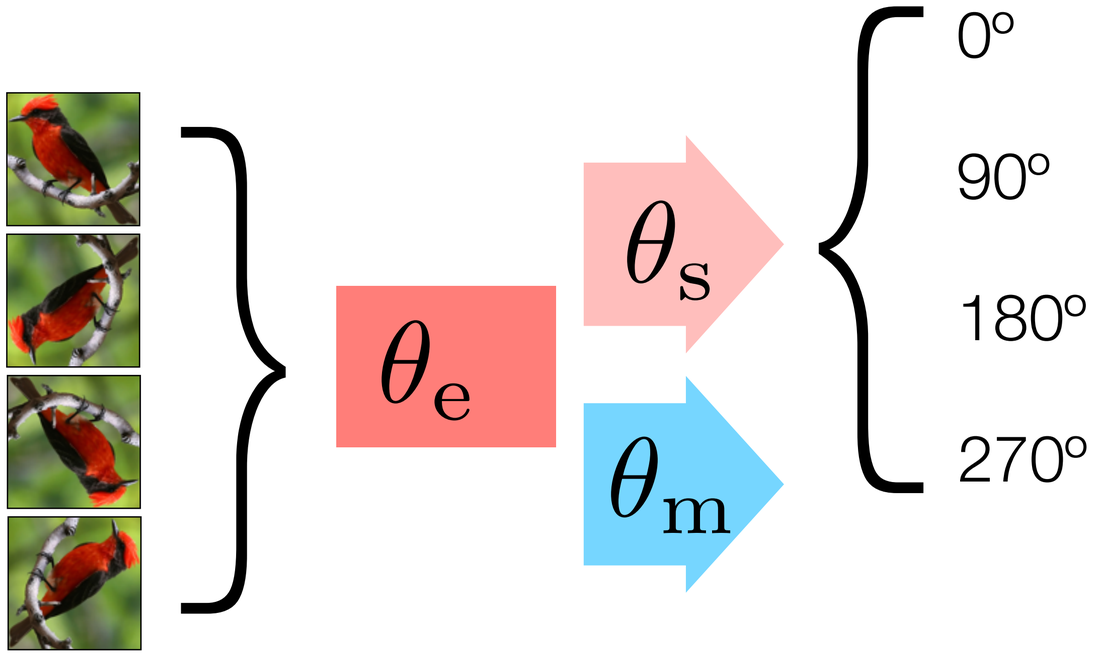
$\ell_{\mathrm{m}}(x, y; \theta_{\mathrm{e}}, \theta_{\mathrm{m}})$



Algorithm for TTT

training

$\ell_{\mathrm{m}}(x, y; \theta_{\mathrm{e}}, \theta_{\mathrm{m}})$

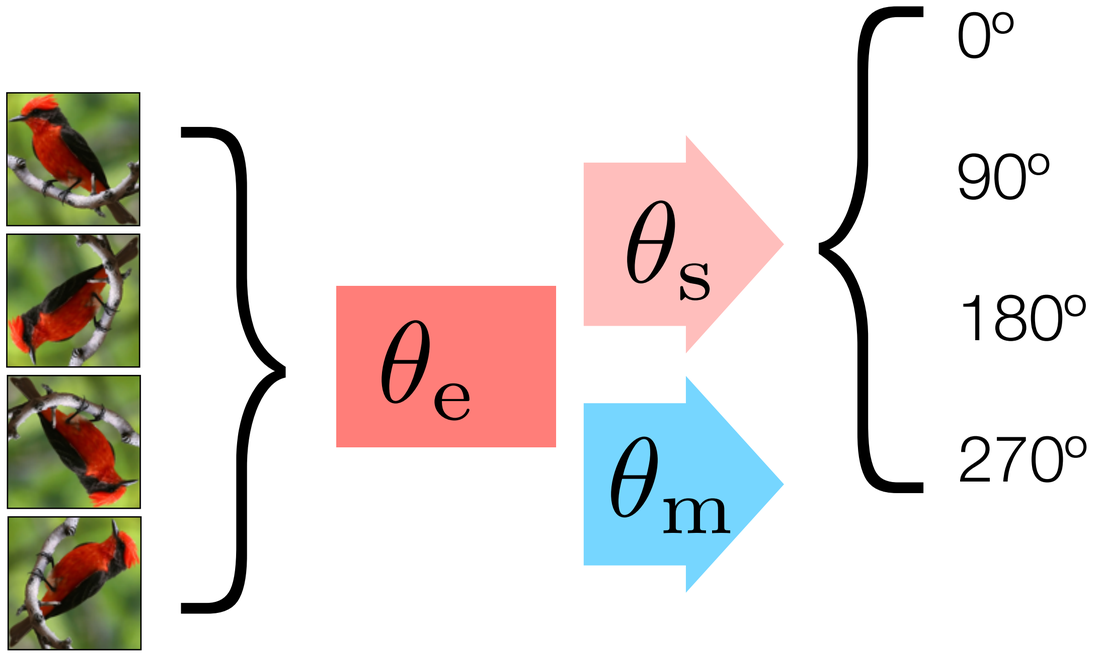


Algorithm for TTT

training

$$\ell_{\mathrm{m}}(x, y; \theta_{\mathrm{e}}, \theta_{\mathrm{m}})$$

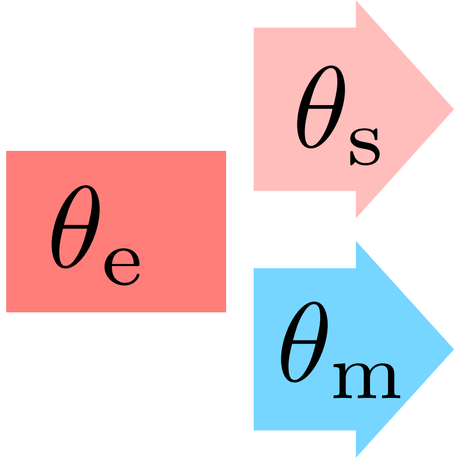
$$+\,\ell_s(x, y_{\mathrm{s}}; \theta_e, \theta_s)$$



Algorithm for TTT

training

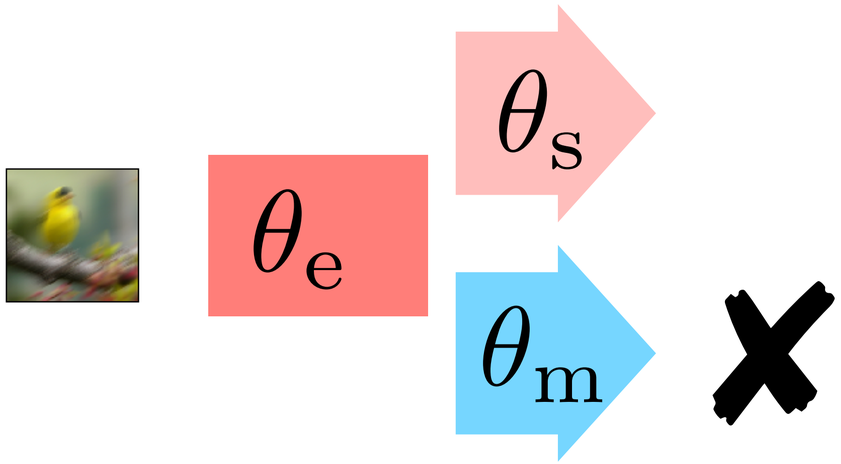
$$\min_{\theta_{\mathrm{e}},\theta_{\mathrm{s}},\theta_{\mathrm{m}}} \mathbb{E}_P \left[\begin{array}{l} \ell_{\mathrm{m}}(x, y; \theta_{\mathrm{e}}, \theta_{\mathrm{m}}) \\ + \ell_s(x, y_{\mathrm{s}}; \theta_e, \theta_s) \end{array} \right]$$



Algorithm for TTT

$$\min_{\theta_\mathrm{e},\theta_\mathrm{s},\theta_\mathrm{m}} \mathbb{E}_P \quad \ell_\mathrm{m}(x, y; \theta_\mathrm{e}, \theta_\mathrm{m}) \quad \ell_s(x, y_\mathrm{s}; \theta_e, \theta_s)$$

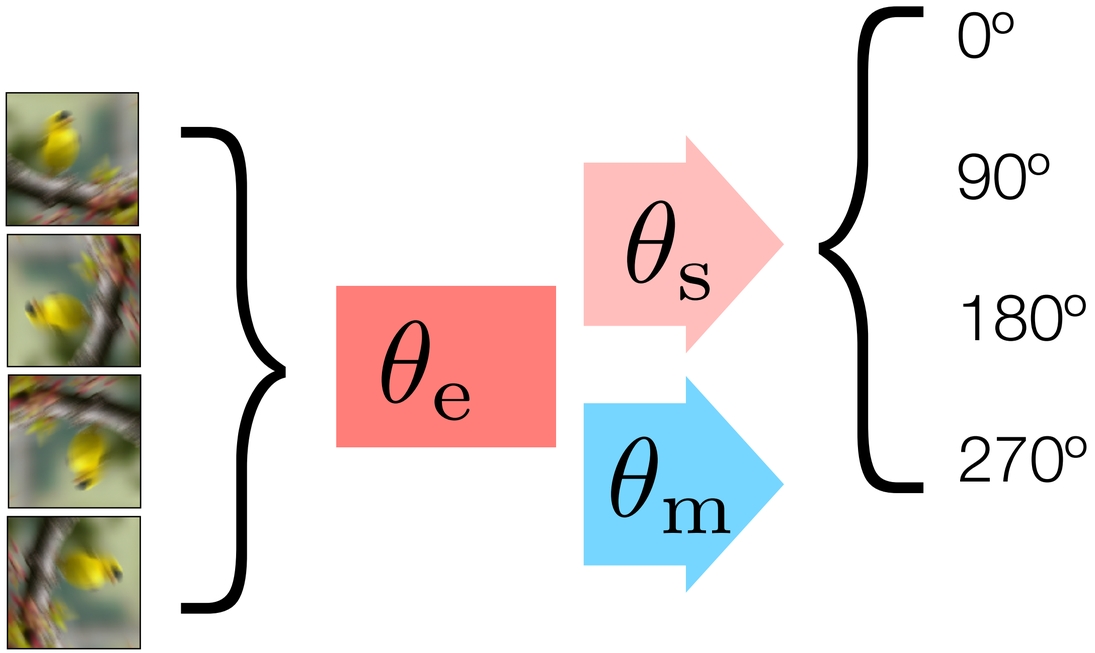
testing



Algorithm for TTT

$$\min_{\theta_\mathrm{e},\theta_\mathrm{s},\theta_\mathrm{m}} \mathbb{E}_P \quad \ell_\mathrm{m}(x, y; \theta_\mathrm{e}, \theta_\mathrm{m}) \quad \ell_s(x, y_\mathrm{s}; \theta_e, \theta_s)$$

testing



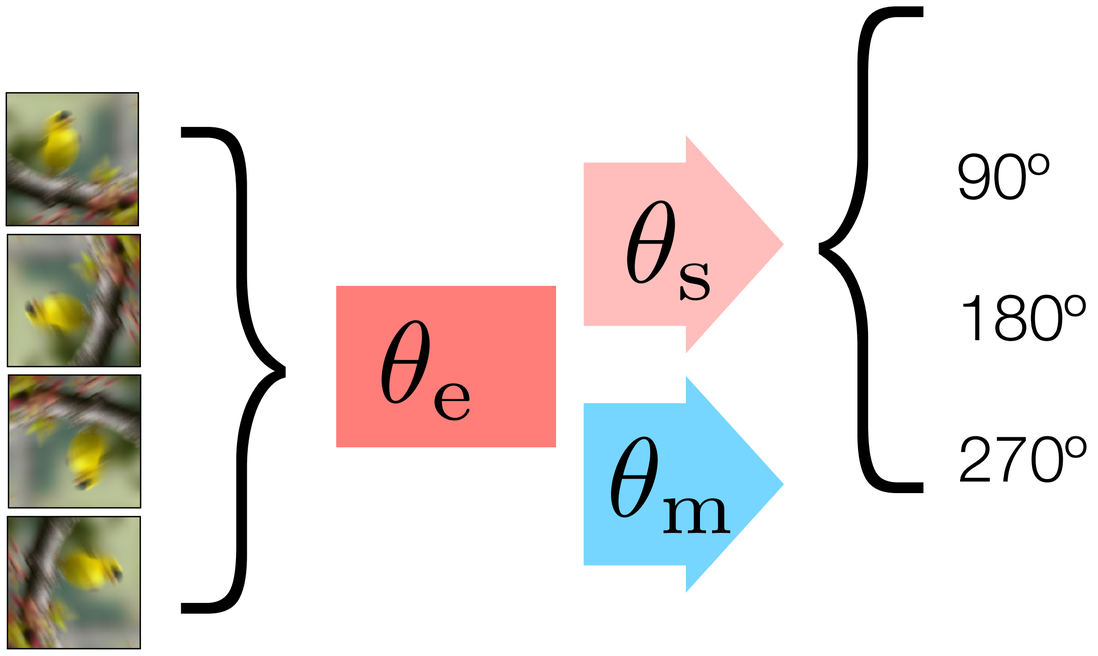
Algorithm for TTT

$$\min_{\theta_e, \theta_s, \theta_m} \mathbb{E}_P \quad \ell_m(x, y; \theta_e, \theta_m)$$

$$\ell_s(x, y_s; \theta_e, \theta_s)$$

testing

$$\min_{\theta_e, \theta_s} \left[\ell_s(x, y_s; \theta_e, \theta_s) \right]$$



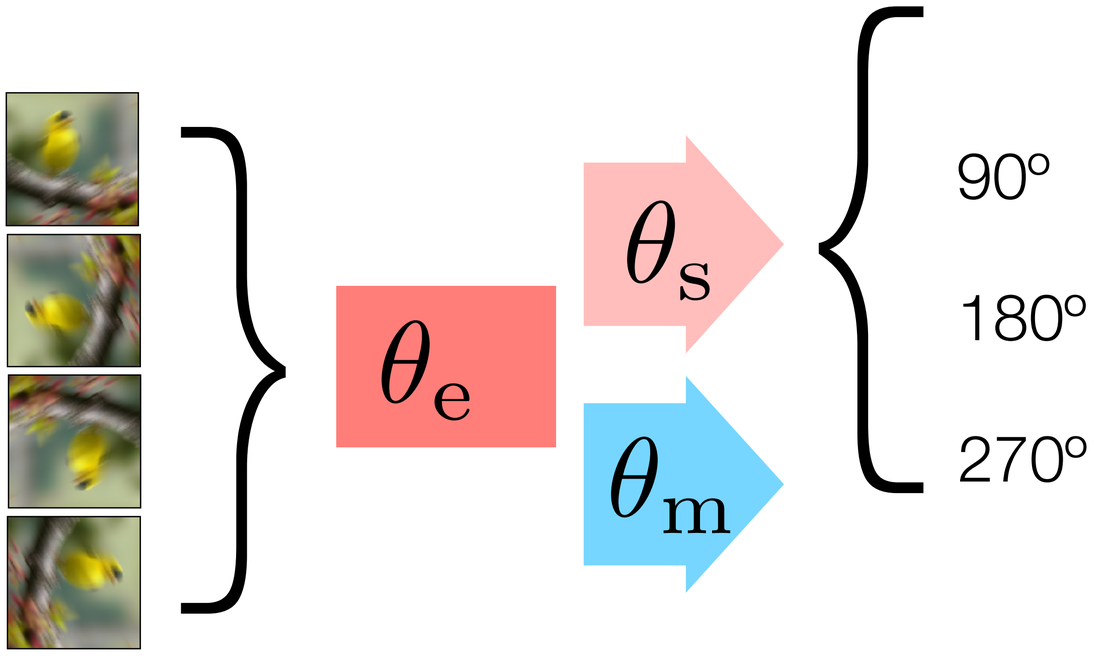
Algorithm for TTT

$$\min_{\theta_e, \theta_s, \theta_m} \mathbb{E}_P \quad \ell_m(x, y; \theta_e, \theta_m)$$

$$\ell_s(x, y_s; \theta_e, \theta_s)$$

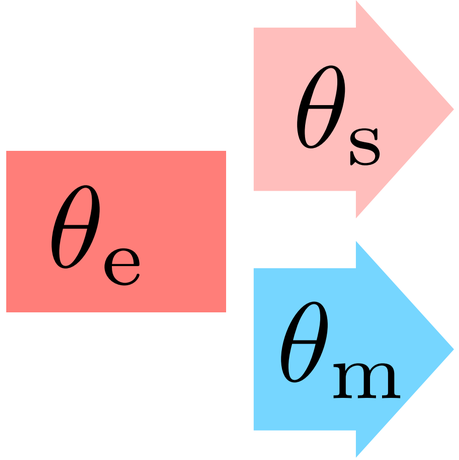
testing

$$\min_{\theta_e, \theta_s} \mathbb{E}_Q \left[\ell_s(x, y_s; \theta_e, \theta_s) \right]$$



Algorithm for TTT

$$\min_{\theta_\mathrm{e},\theta_\mathrm{s},\theta_\mathrm{m}} \mathbb{E}_P \quad \begin{matrix} \ell_\mathrm{m}(x, y; \theta_\mathrm{e}, \theta_\mathrm{m}) \\ \ell_s(x, y_\mathrm{s}; \theta_e, \theta_s) \end{matrix}$$



testing

$$\min_{\theta_\mathrm{e},\theta_\mathrm{s}} \mathbb{E}_Q \left[\ell_s(x, y_\mathrm{s}; \theta_e, \theta_s) \right]$$

$\rightarrow \theta(x)$: make prediction on x

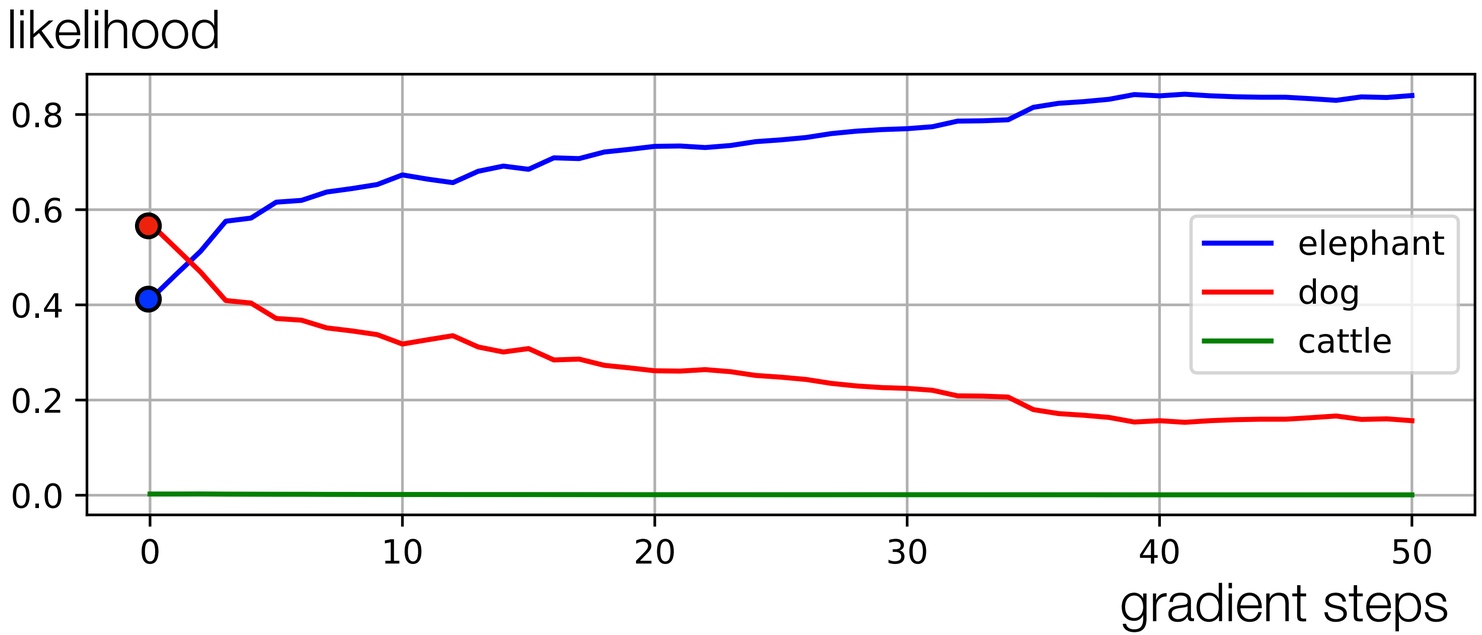
Algorithm for TTT

$$\min_{\theta_e, \theta_s, \theta_m} \mathbb{E}_P \quad \begin{matrix} \ell_m(x, y; \theta_e, \theta_m) \\ \ell_s(x, y_s; \theta_e, \theta_s) \end{matrix}$$

testing

$$\min_{\theta_e, \theta_s} \mathbb{E}_Q \left[\ell_s(x, y_s; \theta_e, \theta_s) \right]$$

$\rightarrow \theta(x)$: make prediction on x



Algorithm for TTT

$$\min_{\theta_\mathrm{e},\theta_\mathrm{s},\theta_\mathrm{m}} \mathbb{E}_P \quad \begin{matrix} \ell_\mathrm{m}(x, y; \theta_\mathrm{e}, \theta_\mathrm{m}) \\ \ell_s(x, y_\mathrm{s}; \theta_e, \theta_s) \end{matrix}$$

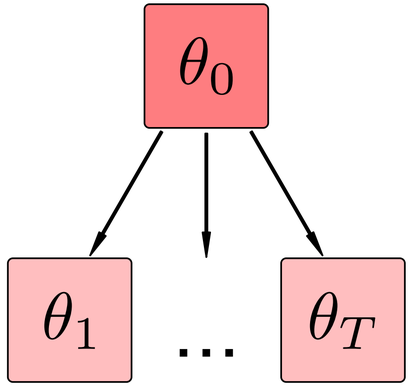
testing

$$\min_{\theta_\mathrm{e},\theta_\mathrm{s}} \mathbb{E}_Q \left[\ell_s(x, y_\mathrm{s}; \theta_e, \theta_s) \right]$$

$\rightarrow \theta(x)$: make prediction on x

multiple test samples $x_1, \ldots, x_T$

θ_0: parameters after joint training



Algorithm for TTT

$$\min_{\theta_e, \theta_s, \theta_m} \mathbb{E}_P \quad \ell_m(x, y; \theta_e, \theta_m) \quad \ell_s(x, y_s; \theta_e, \theta_s)$$

testing

$$\min_{\theta_e, \theta_s} \mathbb{E}_Q \left[\ell_s(x, y_s; \theta_e, \theta_s) \right]$$

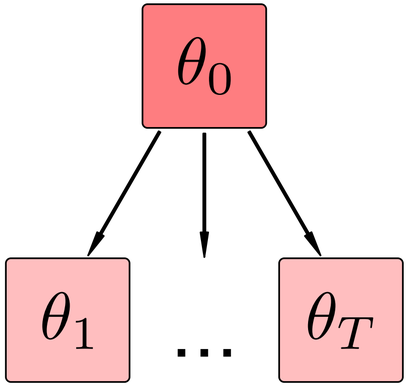
$\rightarrow \theta(x)$: make prediction on x

multiple test samples $x_1, \ldots, x_T$

θ_0: parameters after joint training

standard version

no assumption on the test samples



Algorithm for TTT

$$\min_{\theta_e, \theta_s, \theta_m} \mathbb{E}_P \quad \begin{matrix} \ell_m(x, y; \theta_e, \theta_m) \\ \ell_s(x, y_s; \theta_e, \theta_s) \end{matrix}$$

testing

$$\min_{\theta_e, \theta_s} \mathbb{E}_Q \left[\ell_s(x, y_s; \theta_e, \theta_s) \right]$$

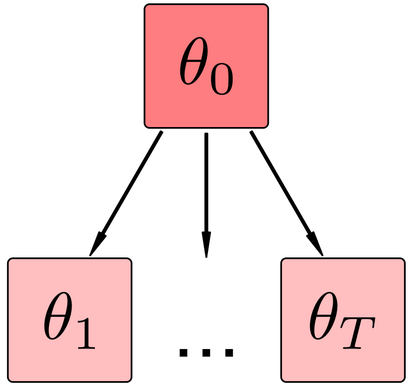
$\rightarrow \theta(x)$: make prediction on x

multiple test samples $x_1, \ldots, x_T$

θ_0: parameters after joint training

standard version

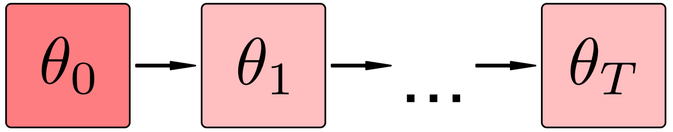
no assumption on the test samples



online version

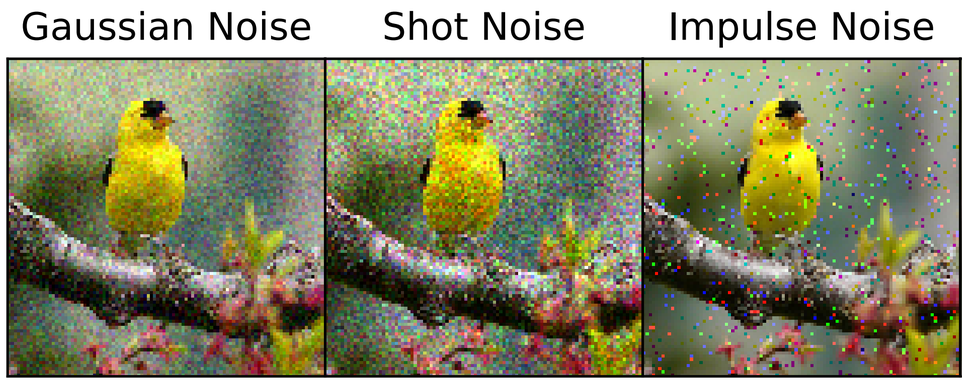
$x_1, \ldots, x_T$ come from the same Q

or smoothly changing $Q_1, \ldots, Q_T$



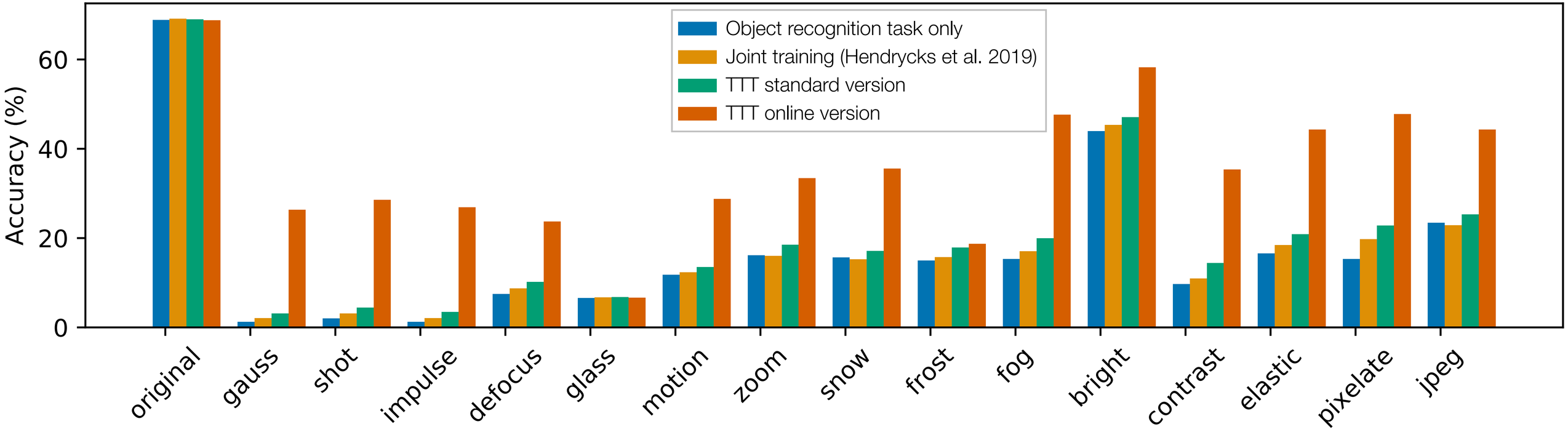
Object recognition with corruptions

- 15 corruptions
- ImageNet: 1000 classes
- No knowledge of the corruptions during training



Benchmarking Neural Network Robustness to Common Corruptions and Perturbations
Hendrycks and Dietterich, 2018

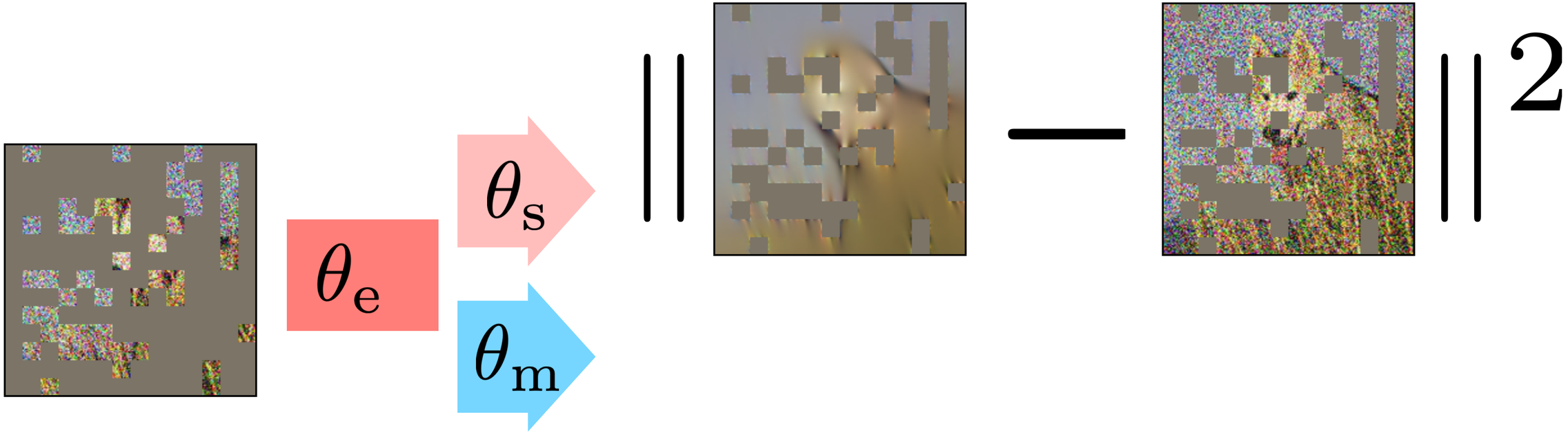
Results on ImageNet-C



Joint training reported here is our improved implementation of their method. Please see our paper for clarification, and their paper for their original results.

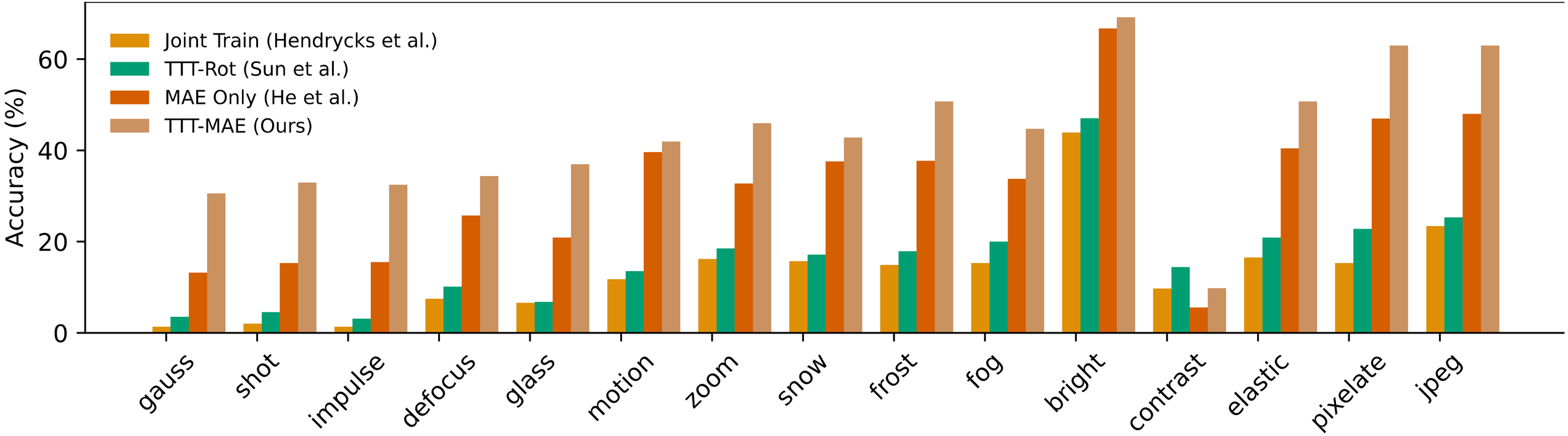
Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty
Hendrycks, Mazeika, Kadavath and Song, 2019

TTT with Masked Autoencoders (MAE)



Test-Time Training with Masked Autoencoding, *NeurIPS 2022*
Yossi Gandelsman*, Yu Sun*, Xinlei Chen, Alexei Efros
*: Equal contribution

TTT-MAE on ImageNet-C

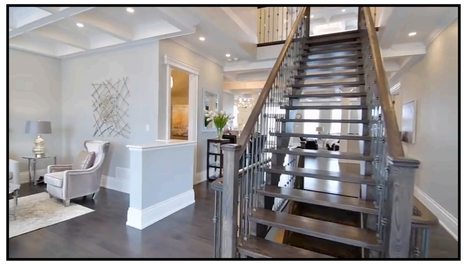


Test-Time Training with Masked Autoencoding, *NeurIPS 2022*

Yossi Gandelsman*, Yu Sun*, Xinlei Chen, Alexei Efros

*: Equal contribution

TTT-MAE



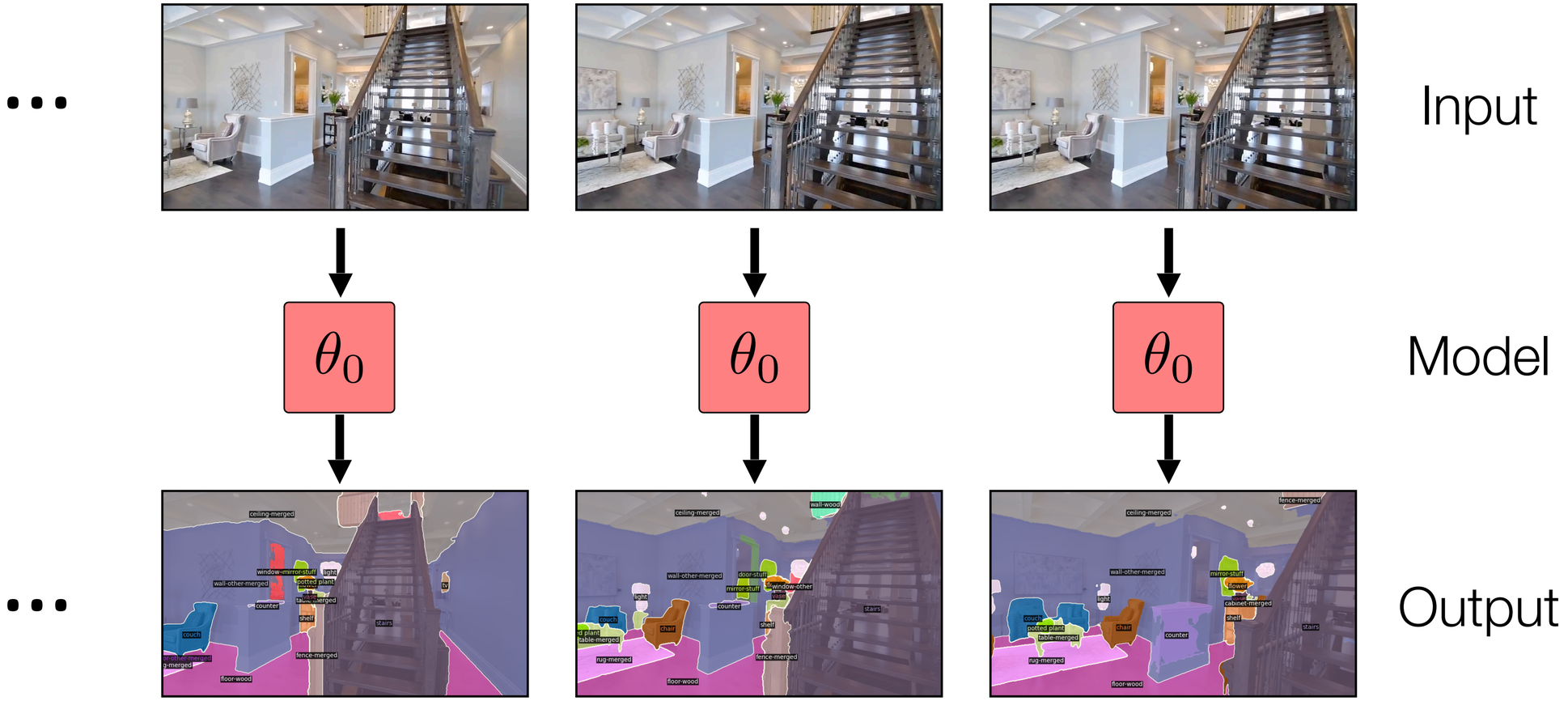
Input

θ_0

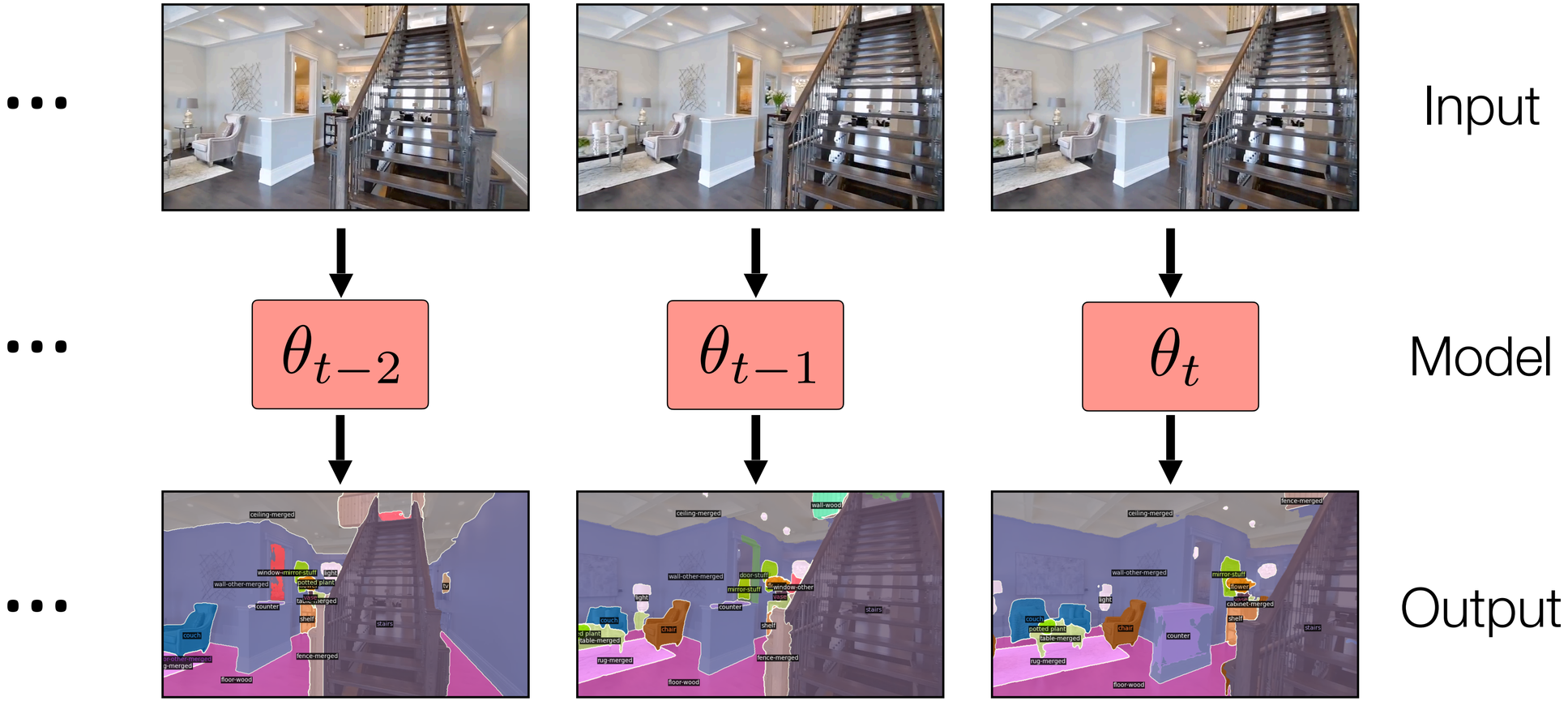
Model

Output

TTT-MAE



Test-Time Training on Video Streams

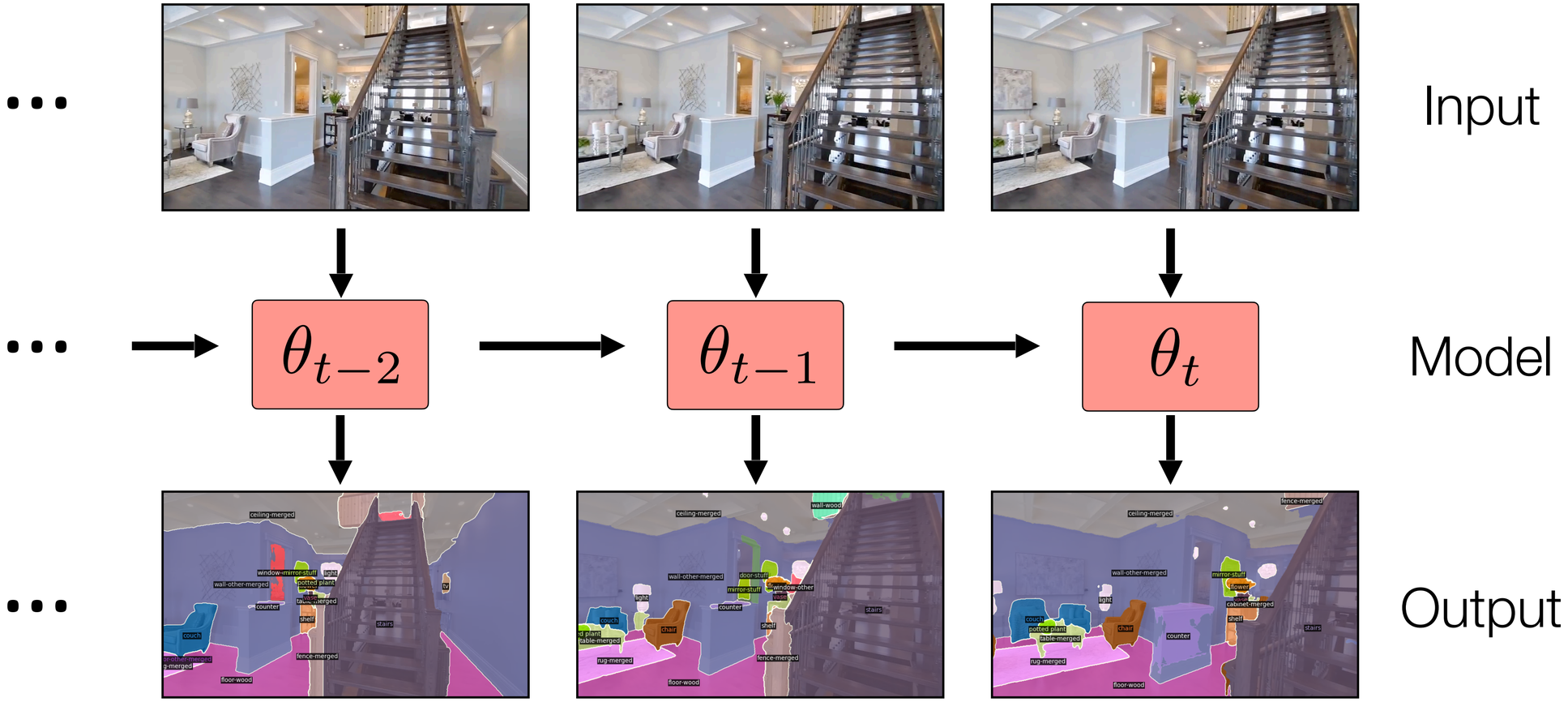


Test-Time Training on Video Streams

Renhao Wang*, Yu Sun*, Yossi Gandelsman, Xinlei Chen, Alexei A. Efros, **Xiaolong Wang**

*: Equal contribution

Test-Time Training on Video Streams



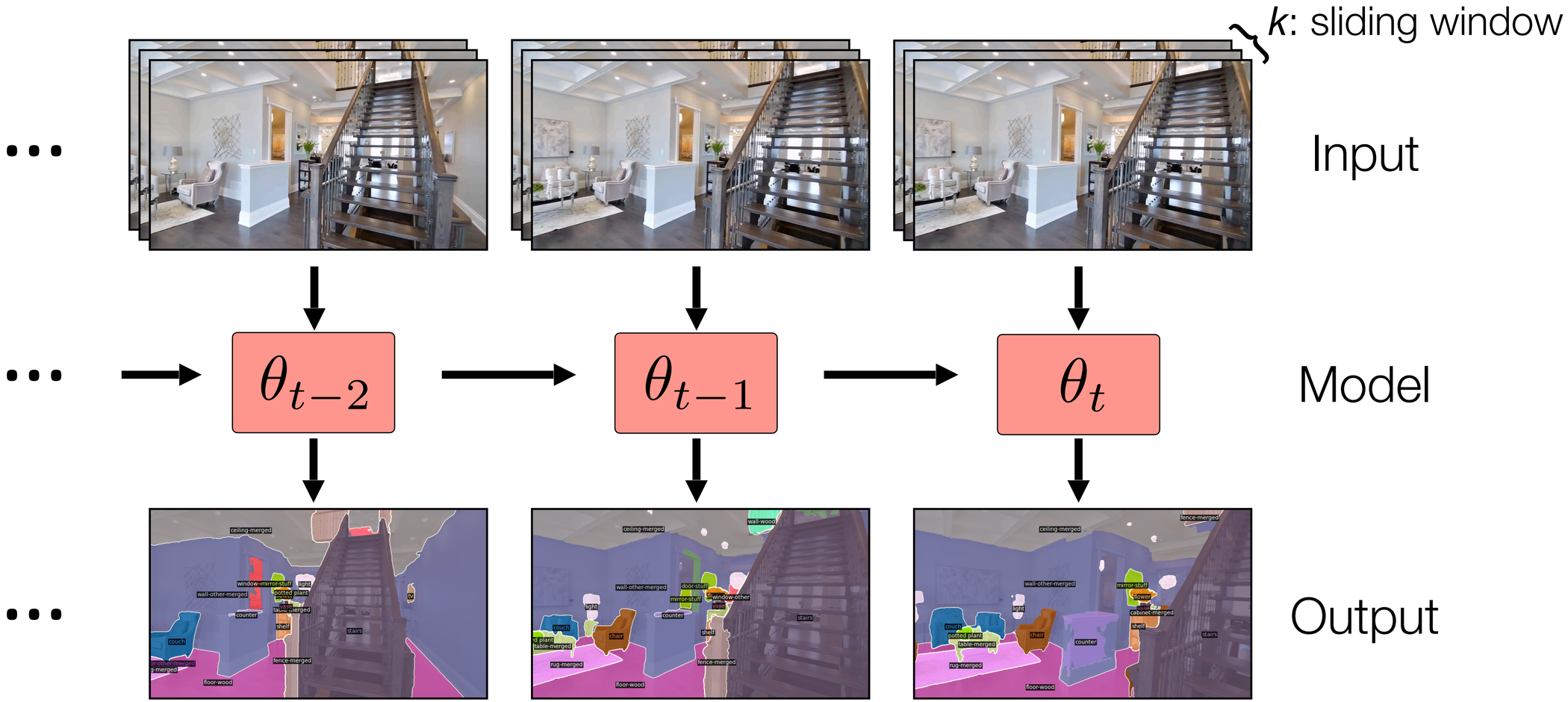
Test-Time Training on Video Streams

Renhao Wang*, Yu Sun*, Yossi Gandelsman, Xinlei Chen, Alexei A. Efros, **Xiaolong Wang**

*: Equal contribution

Test-Time Training on Video Streams

$k \stackrel{?}{=} t = K$

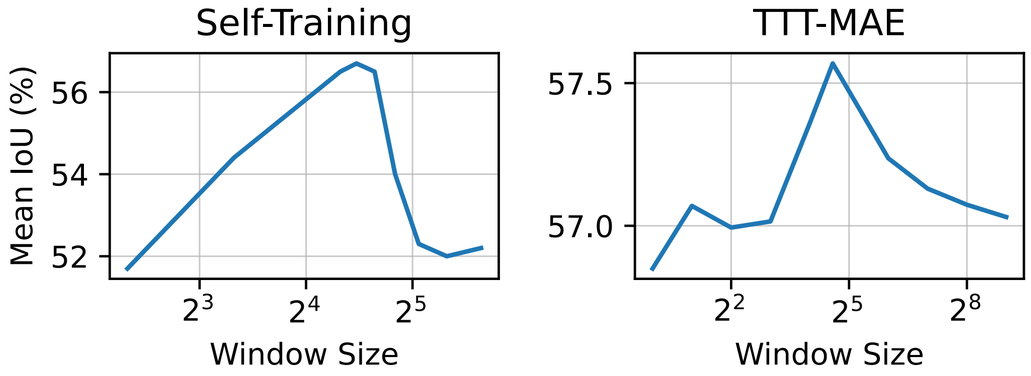


Test-Time Training on Video Streams

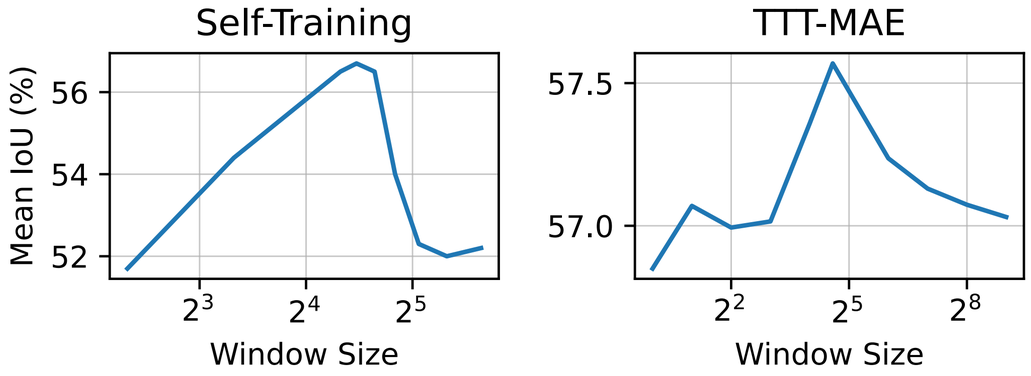
Renhao Wang*, Yu Sun*, Yossi Gandelsman, Xinlei Chen, Alexei A. Efros, **Xiaolong Wang**

*: Equal contribution

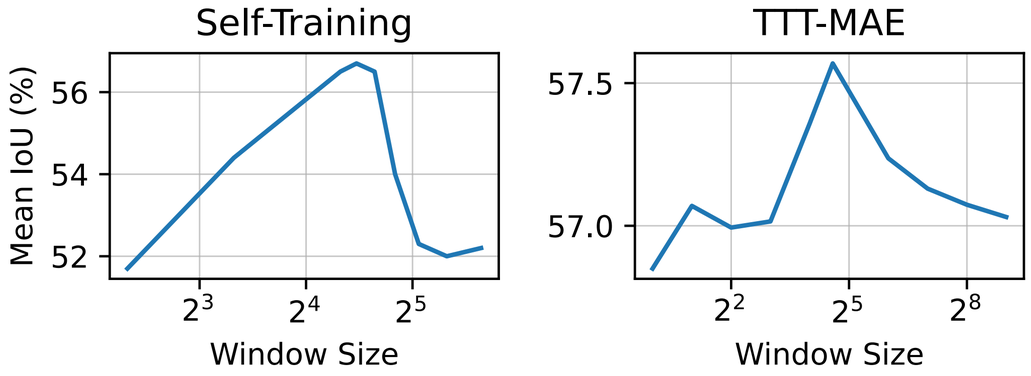
Forgetting can be beneficial

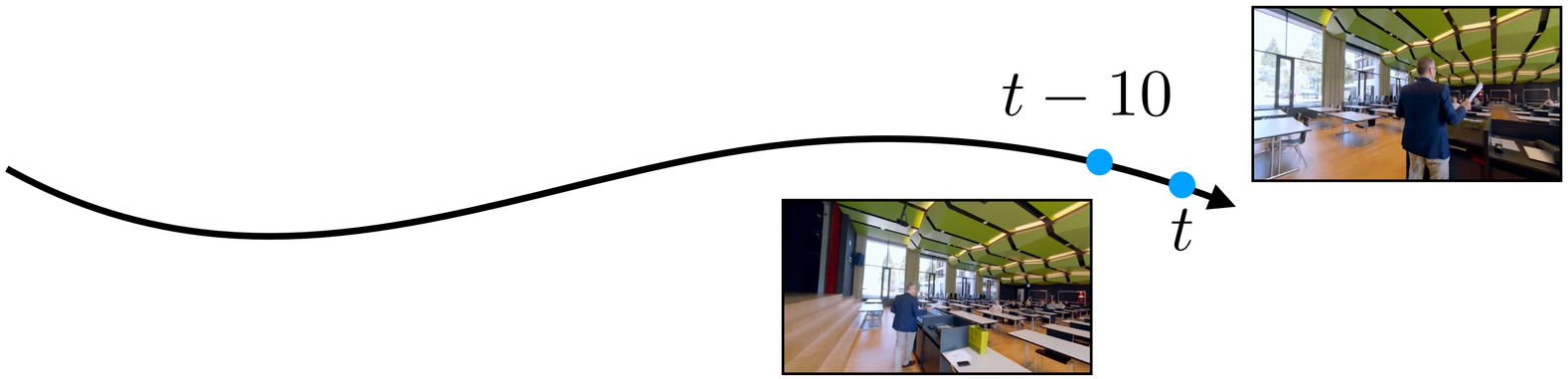


Forgetting can be beneficial

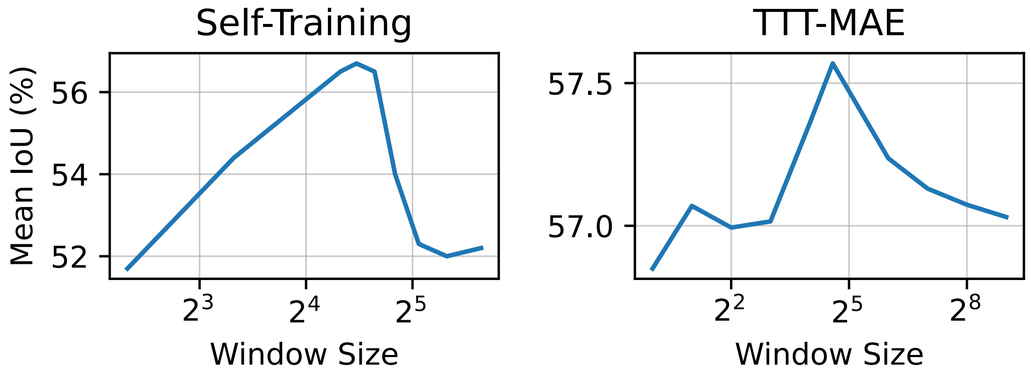


Forgetting can be beneficial

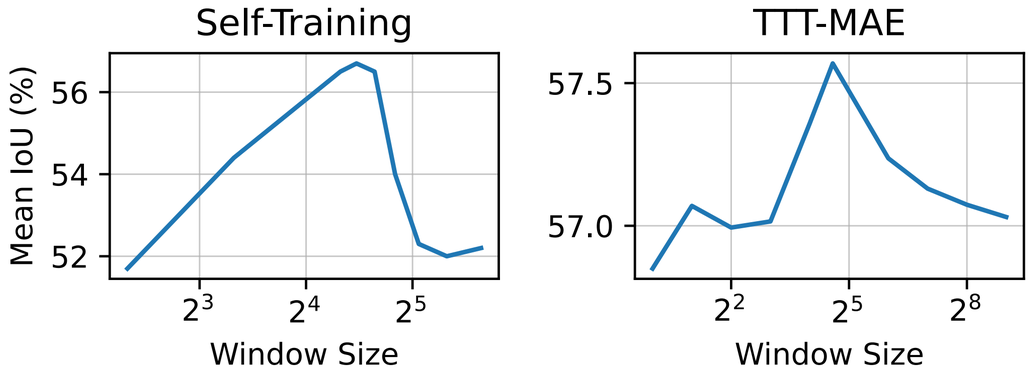




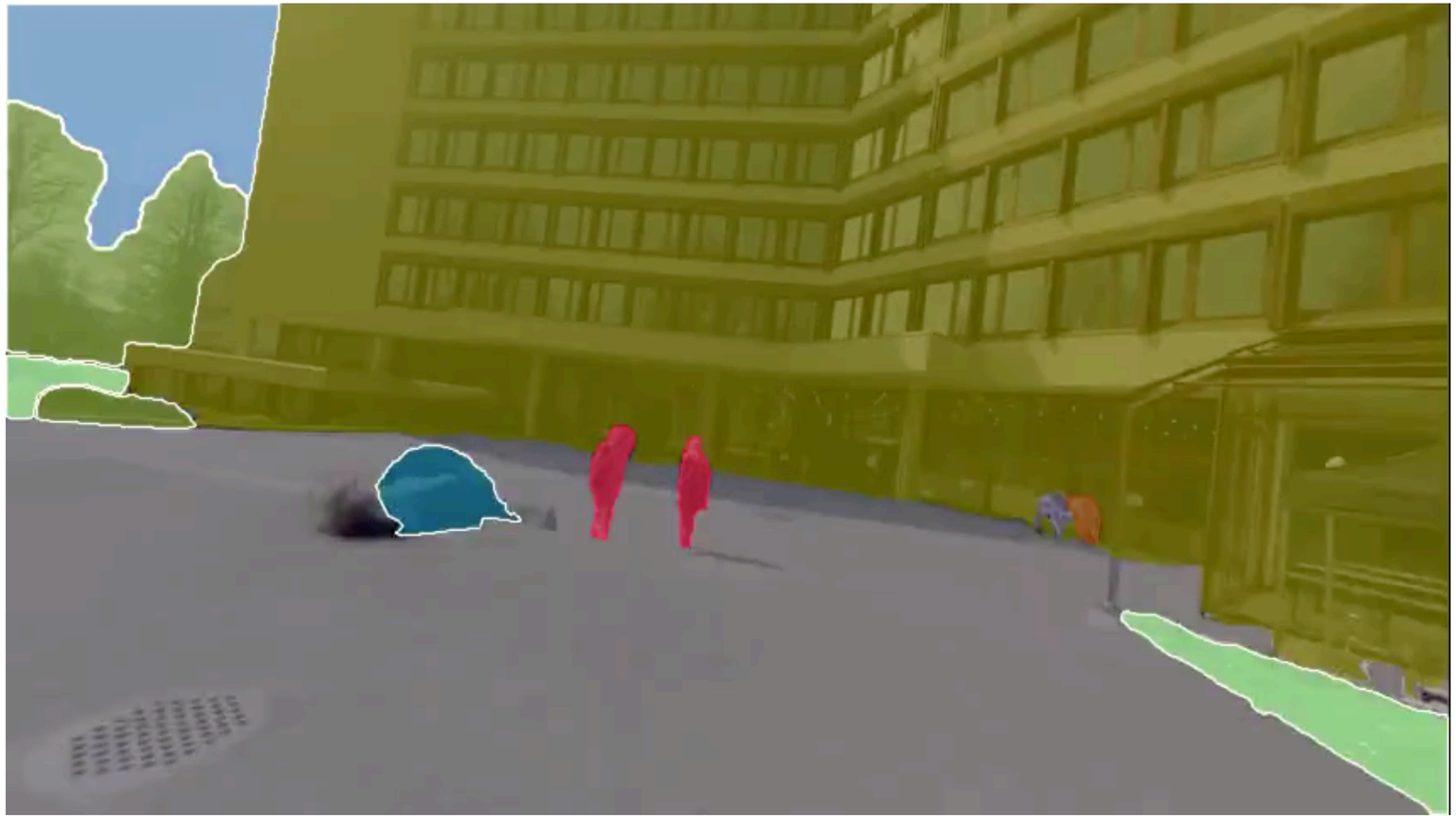
Forgetting can be beneficial



Forgetting can be beneficial



Results on COCO-Videos



Dataset	Len.	Frames	Rate	Cls.
CityScapes-VPS [32]	1.8	3000	17	19
DAVIS [49]	3.5	3455	30	-
YouTube-VOS [76]	4.5	123,467	30	94
KITTI-STEP [72]	40	8,008	10	19
COCO Videos (Ours)	309	30,925	10	134