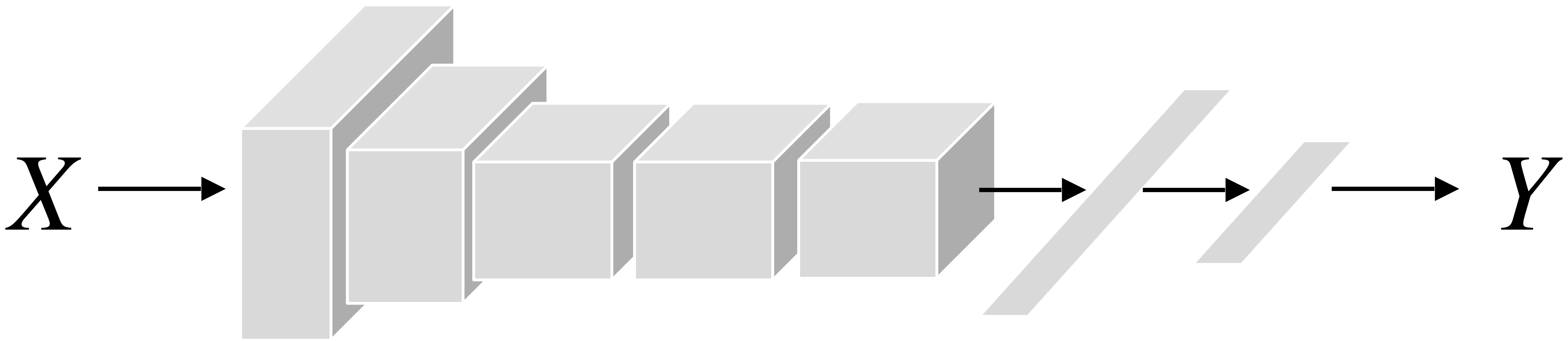
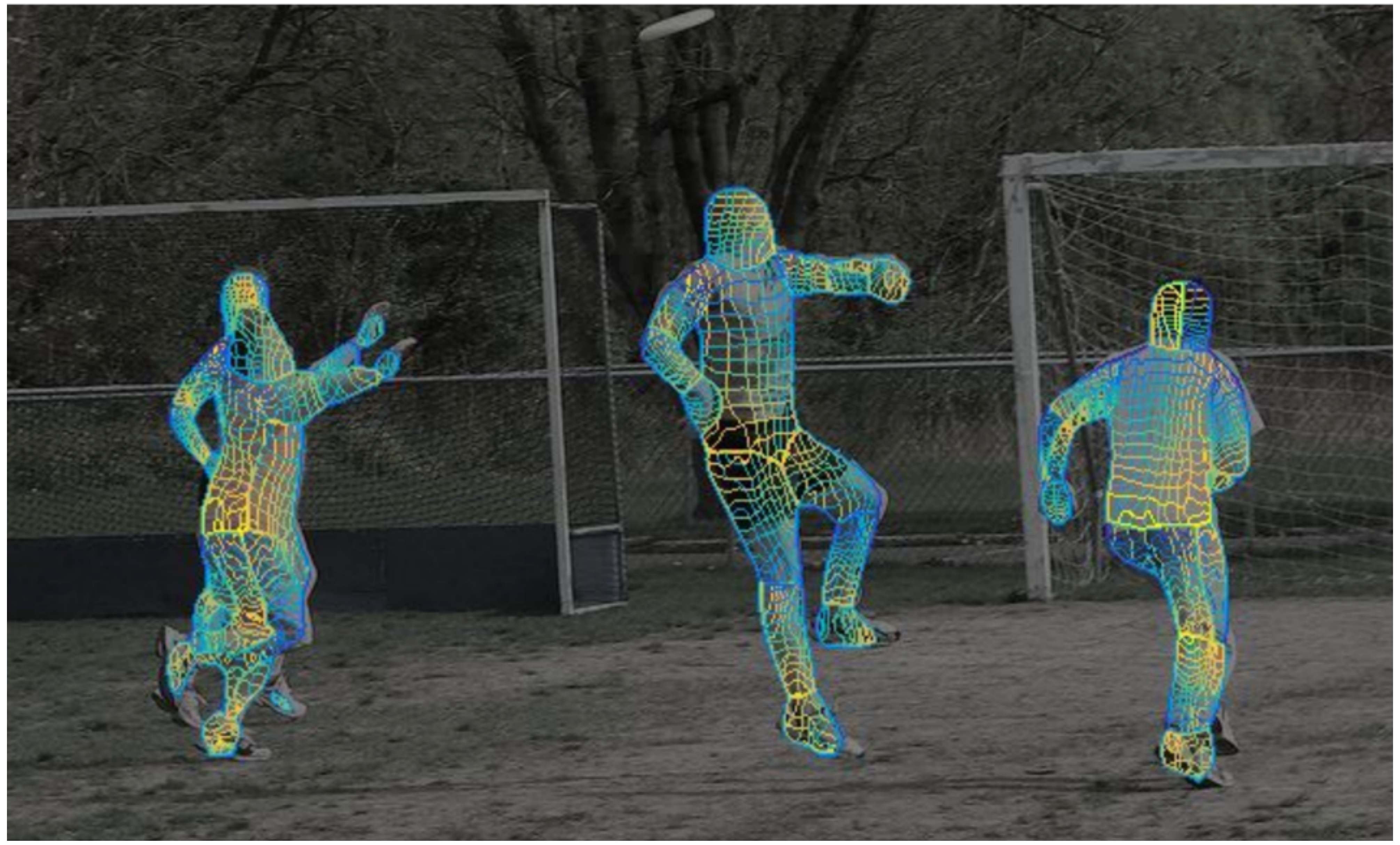
Self-Supervised Visual Representation Learning

Xiaolong Wang

Deep Learning





He et al. Mask R-CNN. ICCV 2017.

Güler et al. DensePose: Dense Human Pose Estimation In The Wild. CVPR 2018.

The Key is The Supervision

People have labeled

IMAGENET

1.2M images

ACTIVITYNET

300K videos

Data uploaded on the web

800M images everyday
300 hours of video every minute

Challenge in Generalization

Performance drop

Image Dog

Video Dog

Self-Supervised Learning

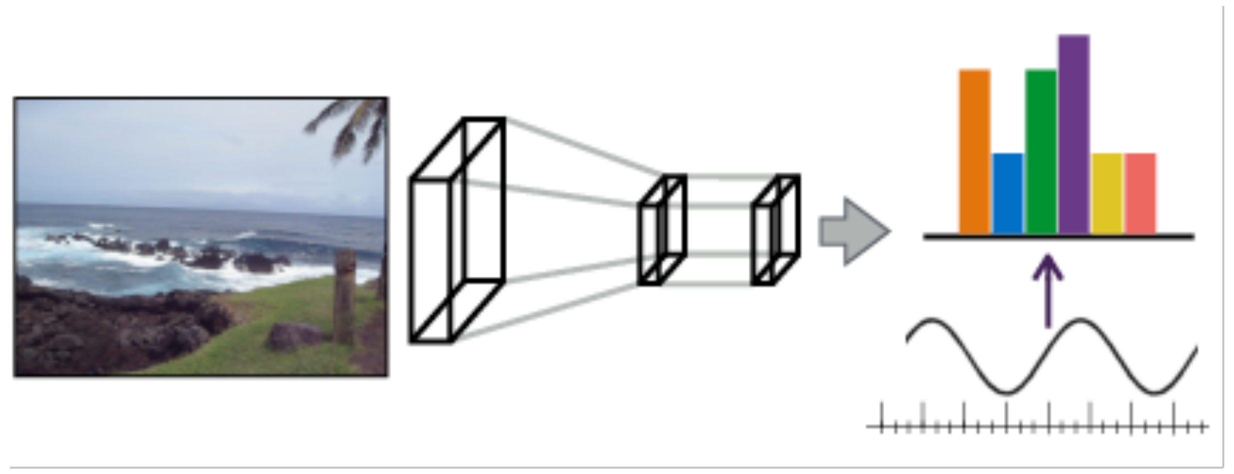
- Designing pretext tasks for general representation
  - Transfer the learned representation to downstream tasks via fine-tuning

- Utilize self-supervision during Test Time
  - Adapting supervised task, RL task for out-of-distribution generalization

Pretext Tasks + Fine-tuning

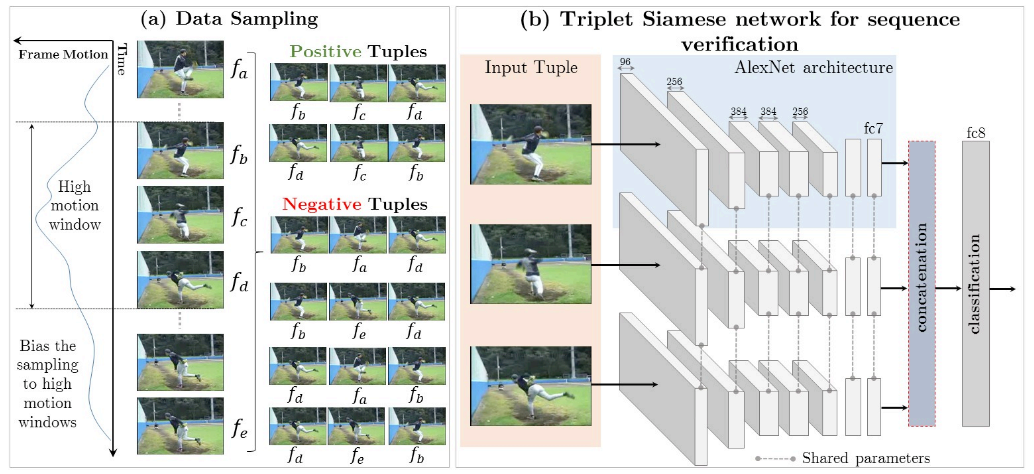
Pretext Task

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

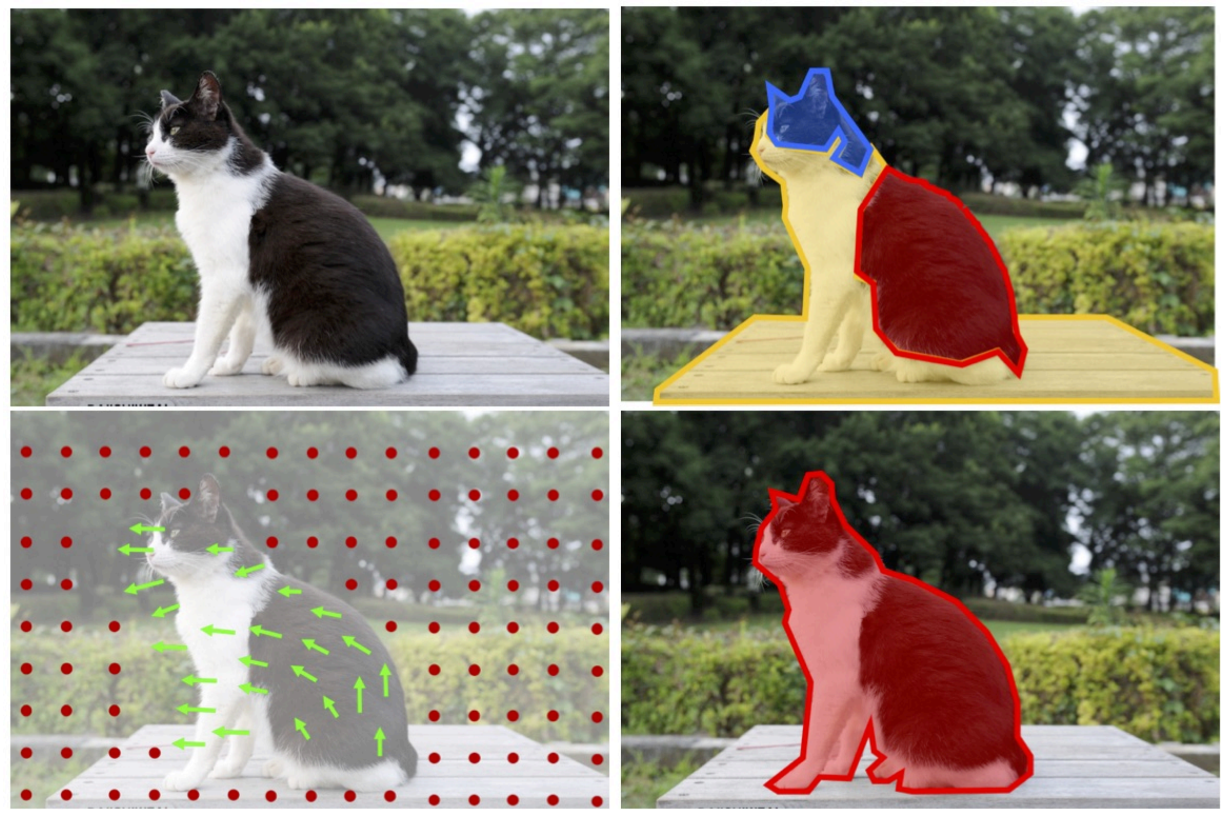


Owens et al. ECCV 2016

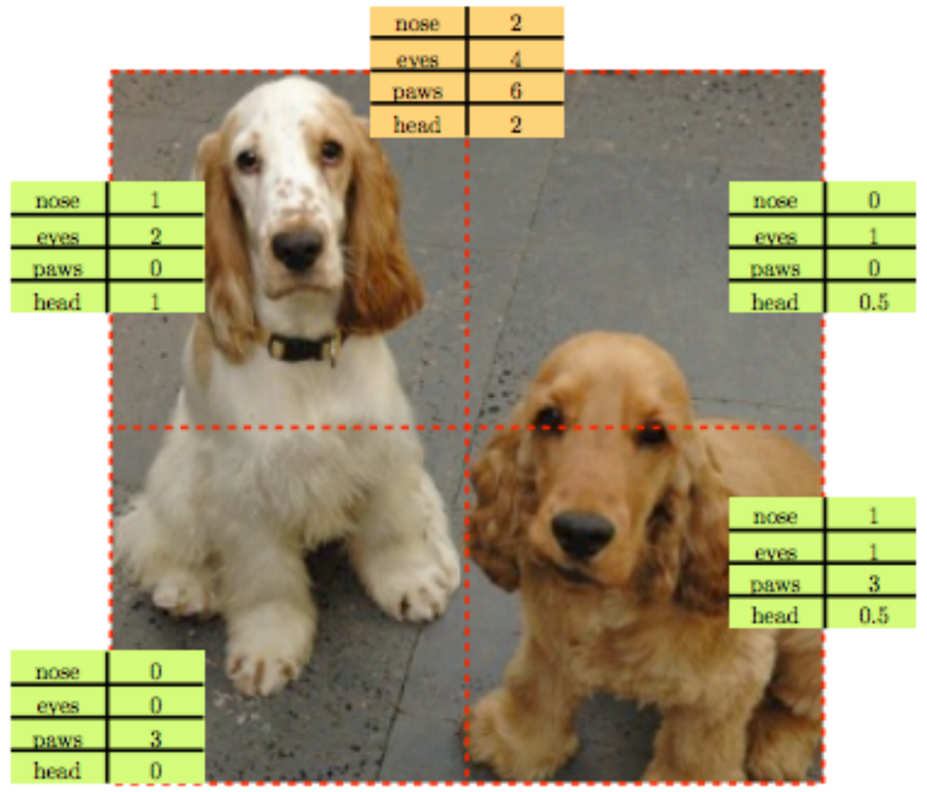
Zhang et al. ECCV 2016



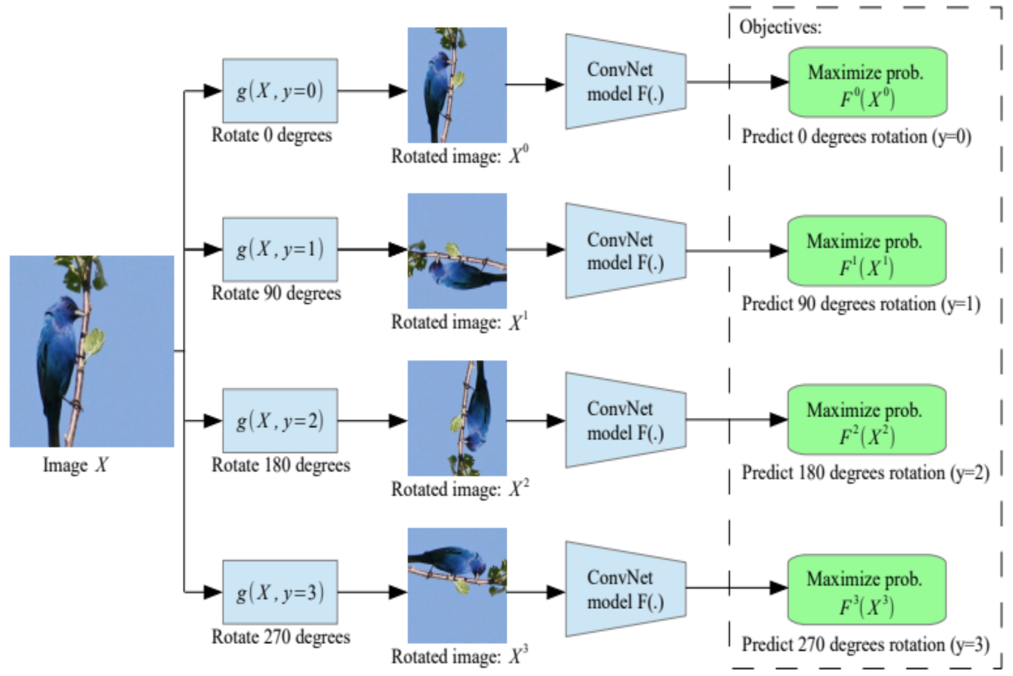
Misra et al. ECCV 2016



Pathak et al. CVPR 2017

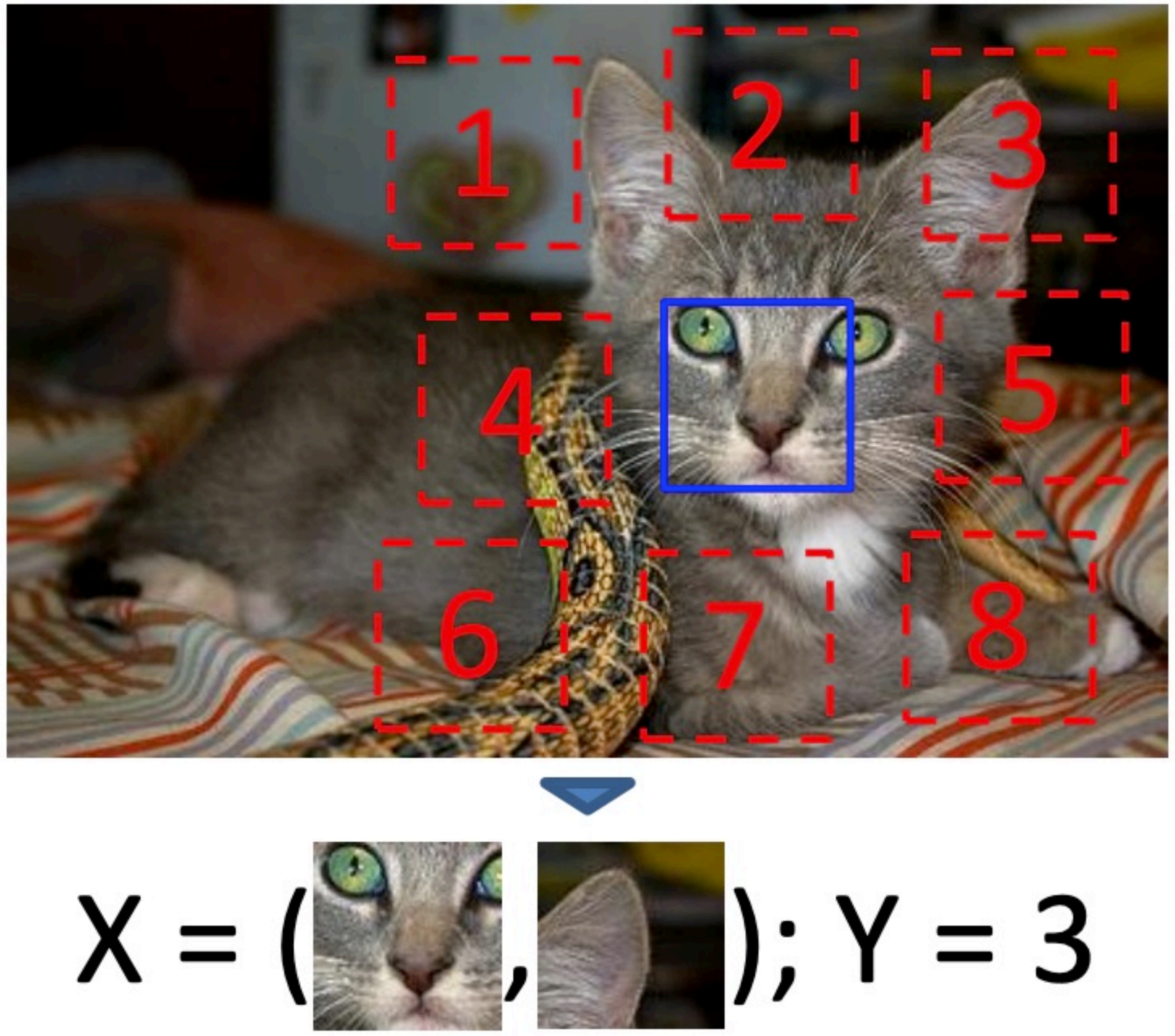


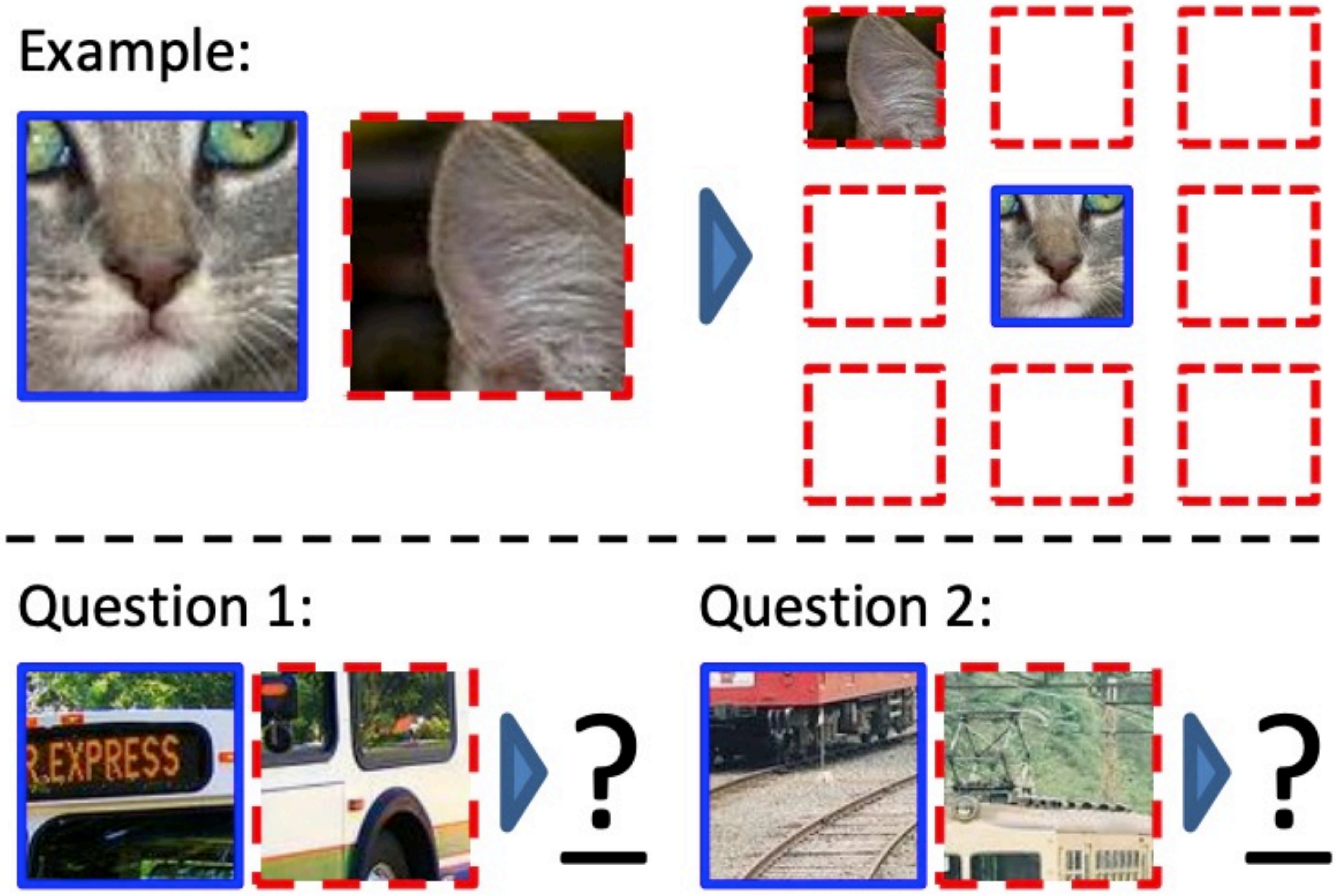
Noroozi et al. ICCV 2017



Gidaris et al. ICLR 2018

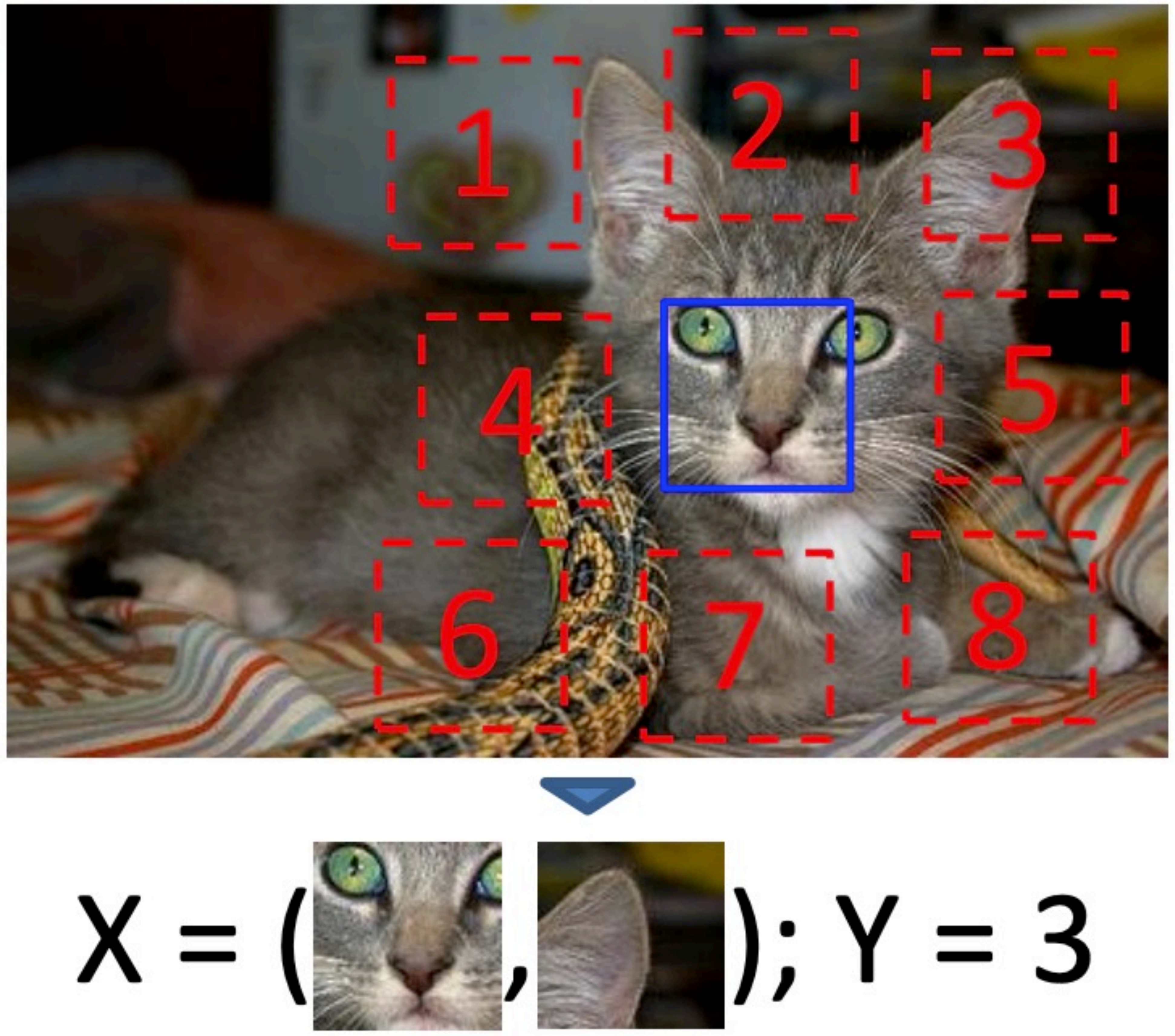
Self-Supervised Learning with Context Prediction

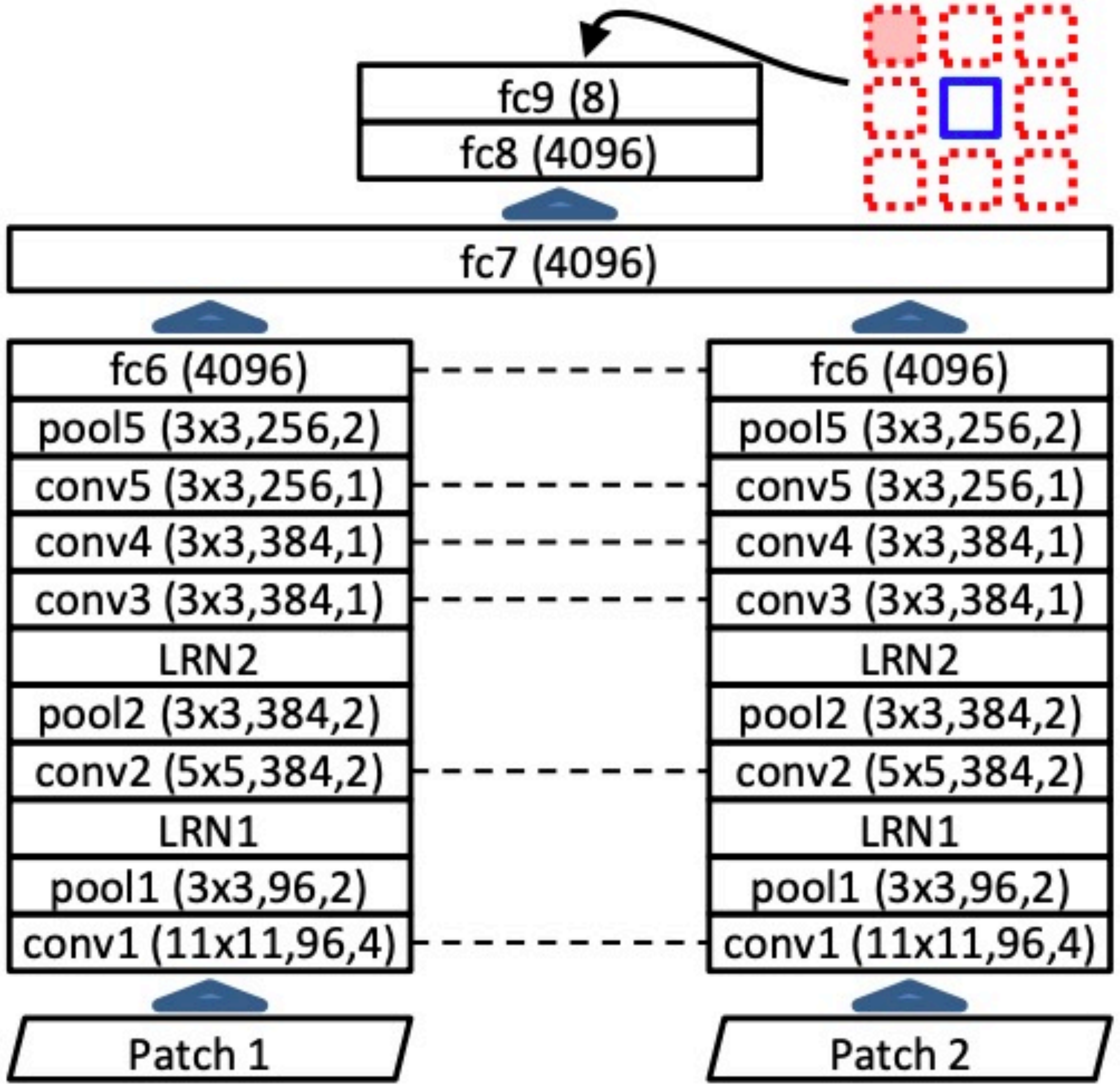




[Doersch et al. 2015]

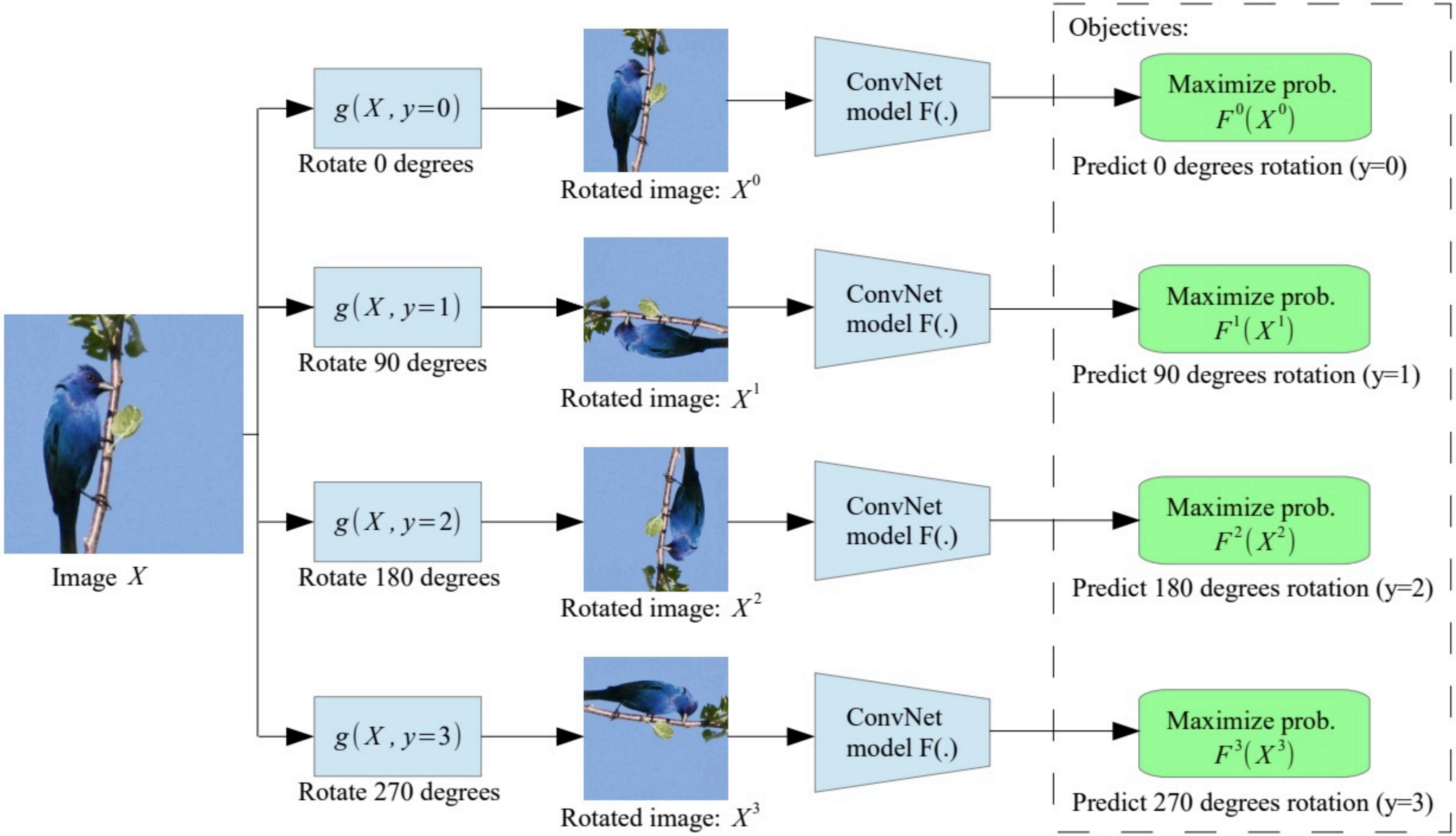
Self-Supervised Learning with Context Prediction





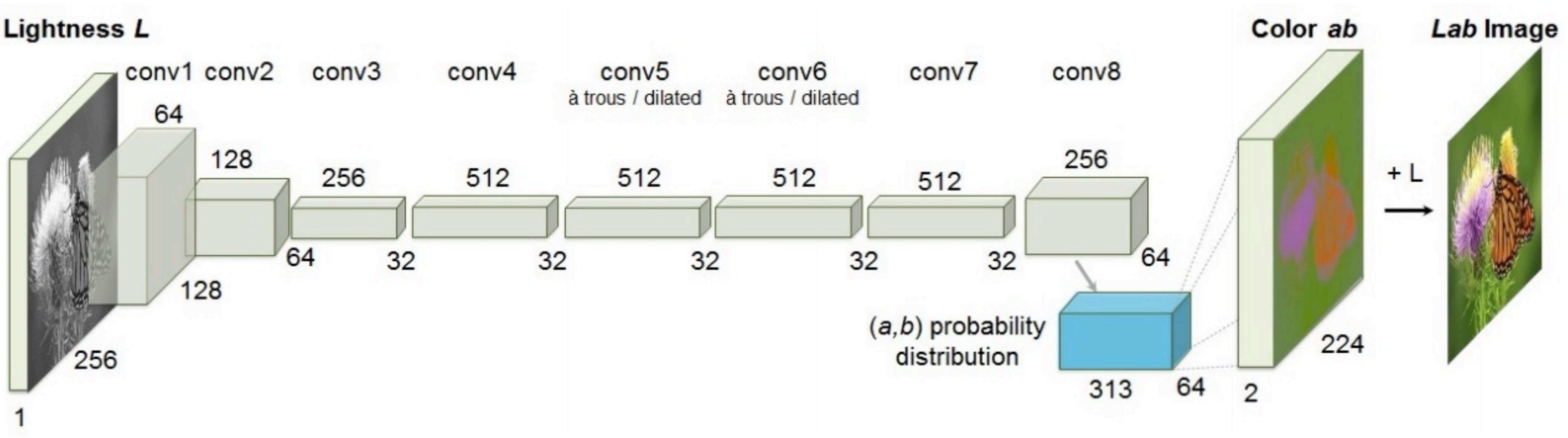
[Doersch et al. 2015]

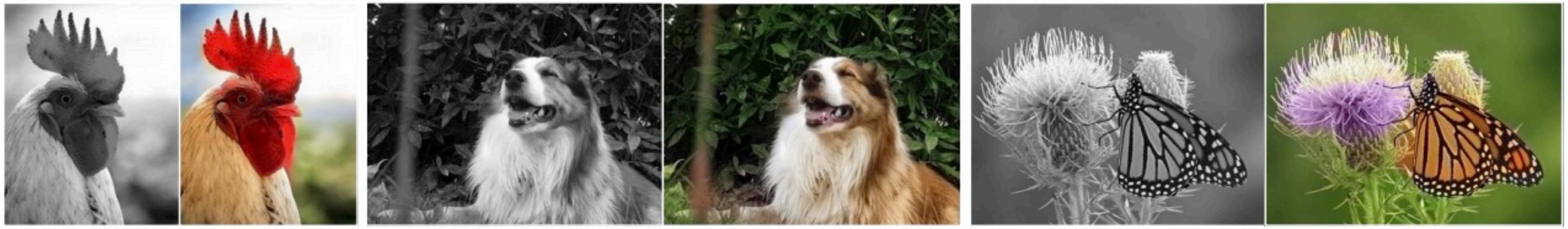
Self-Supervised Learning with Rotation Prediction



[Gidaris et al. 2018]

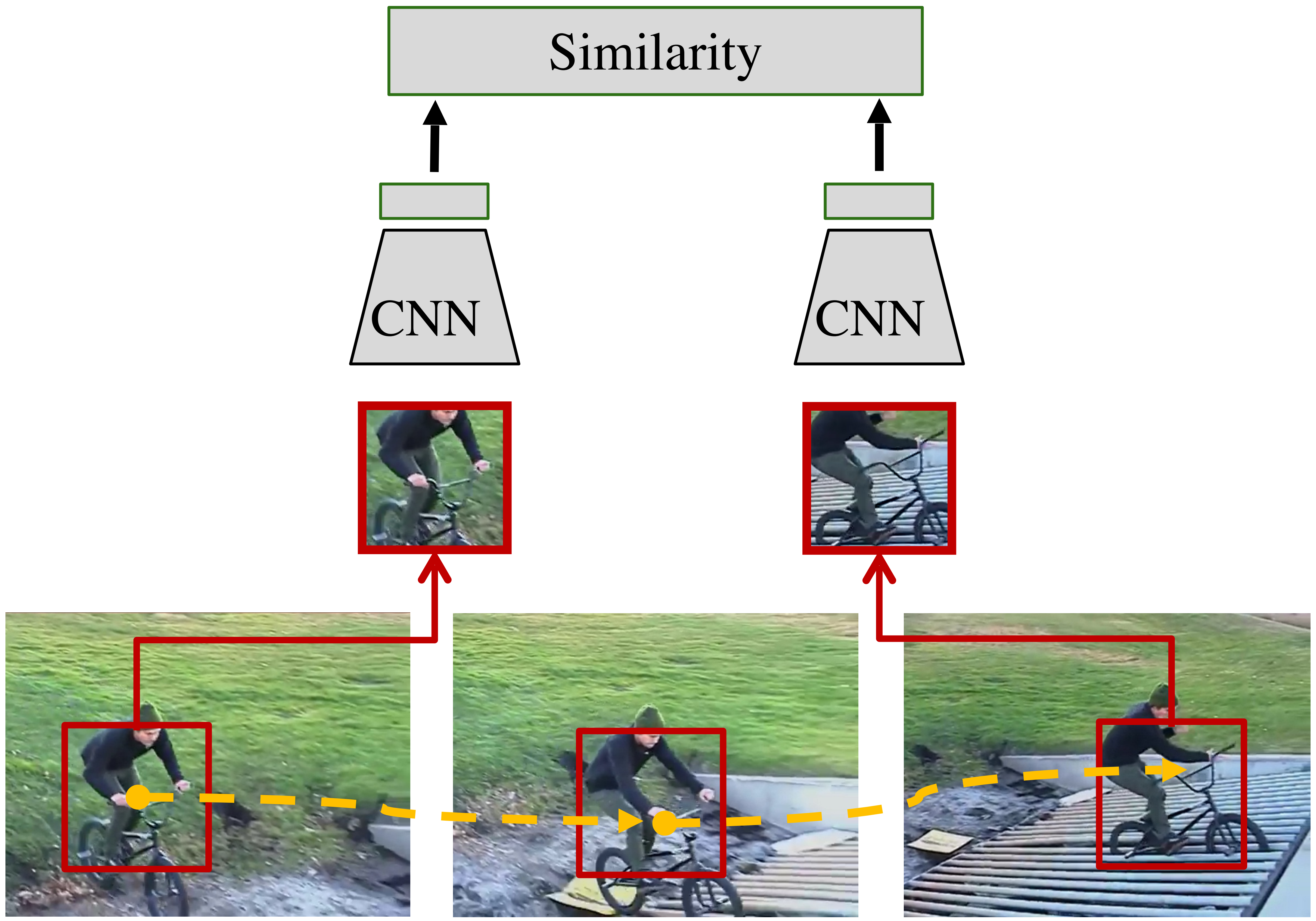
Self-Supervised Learning with Image Colorization





[Zhang et al. 2016]

Self-Supervised Learning with Tracking

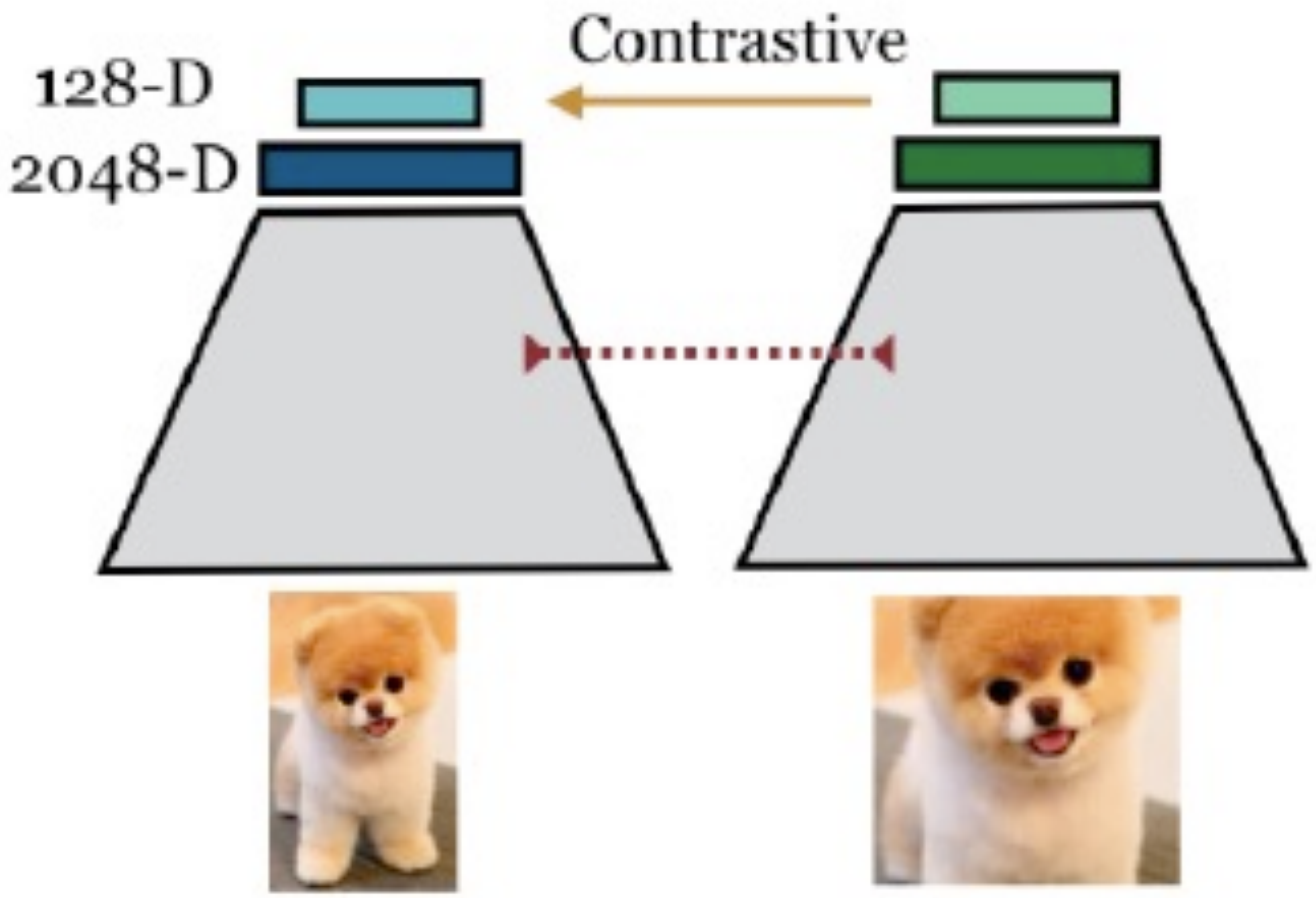


Tracking → Similarity

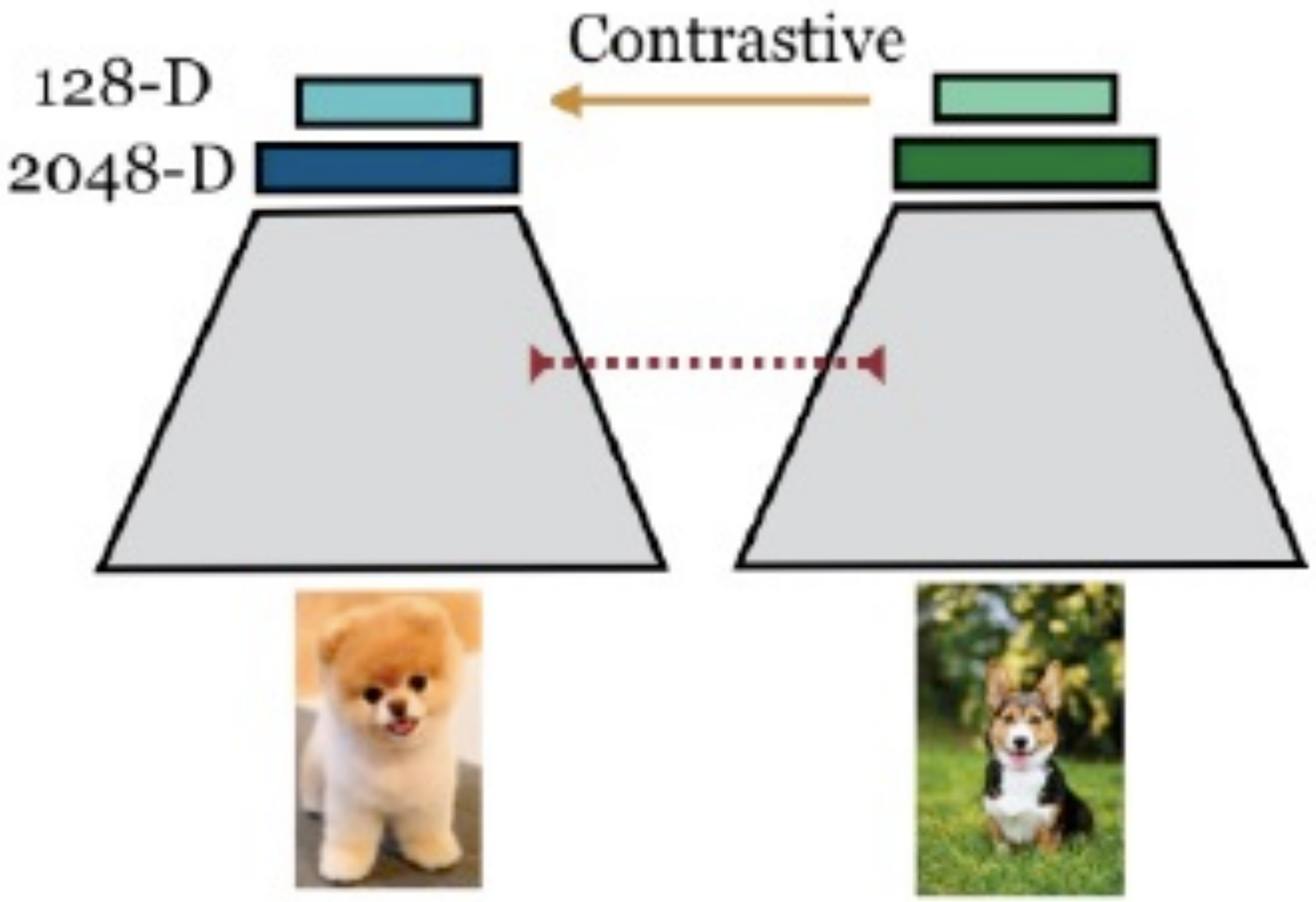
[Wang et al. 2015]

Contrastive Learning

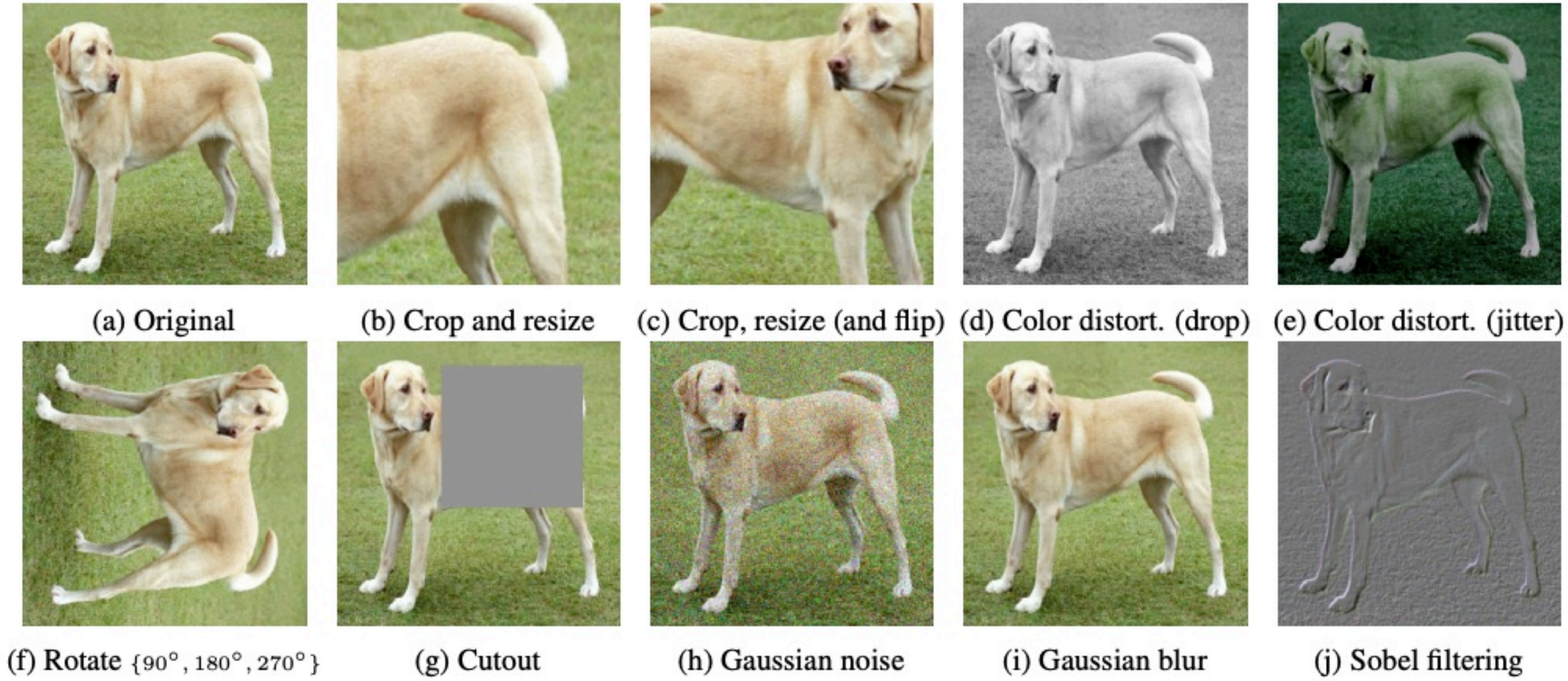
Similar

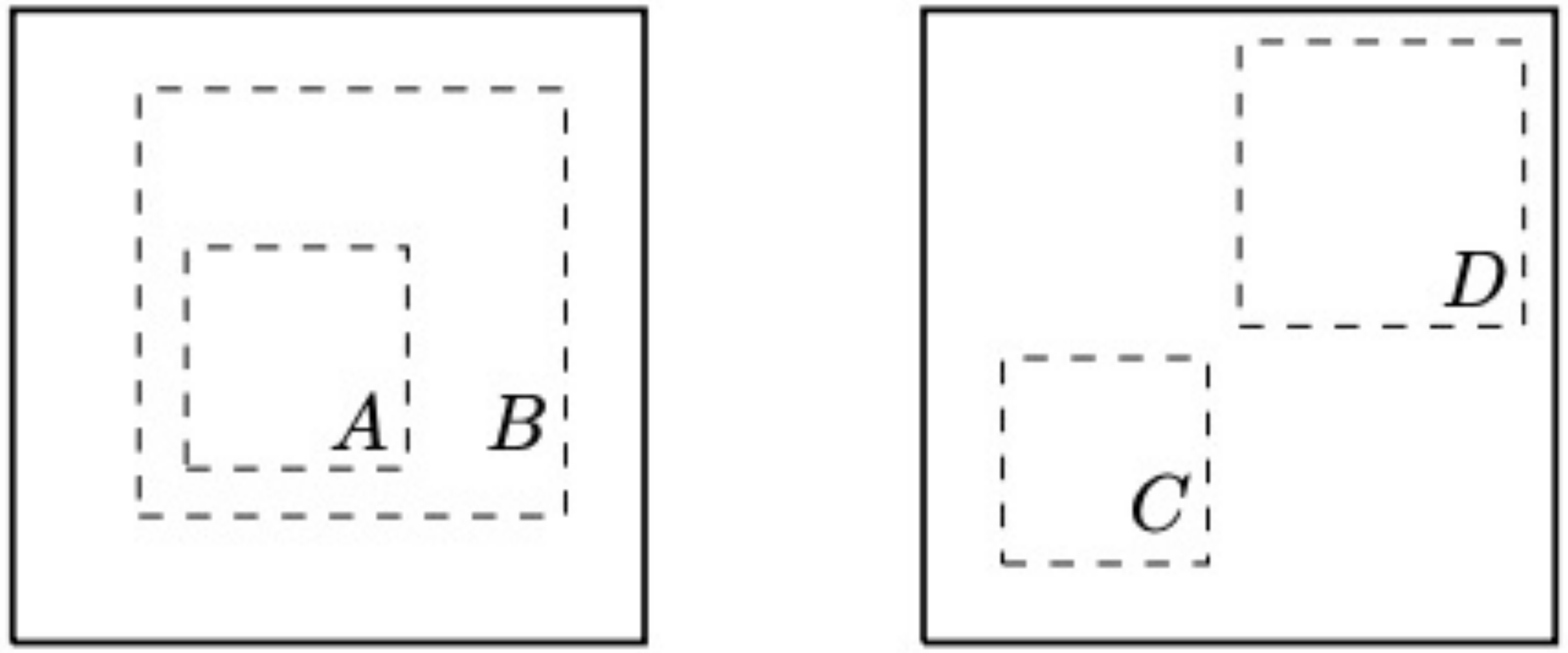


Not Similar



SimCLR



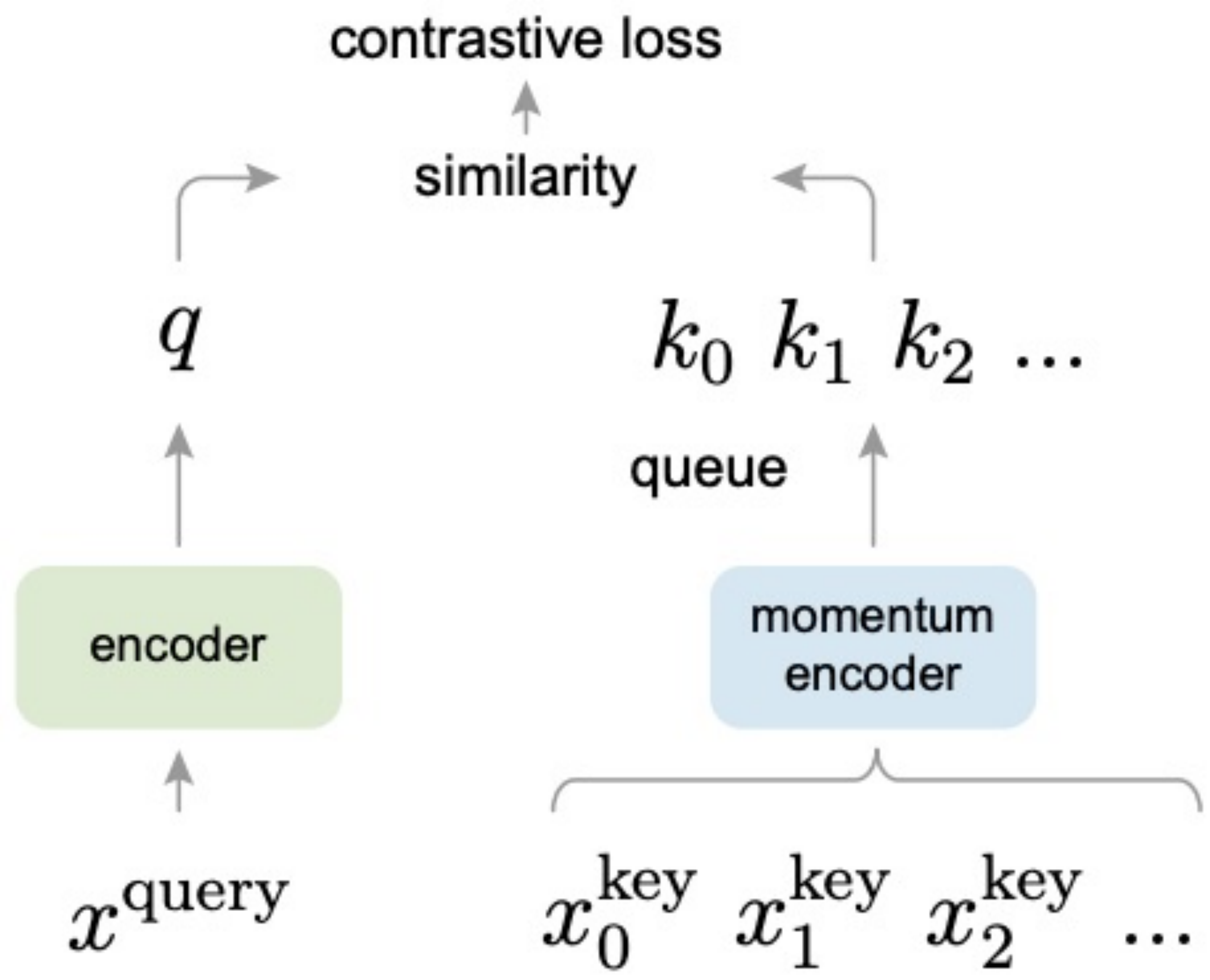


(a) Global and local views. (b) Adjacent views.

Require large number of negatives

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

MoCo



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

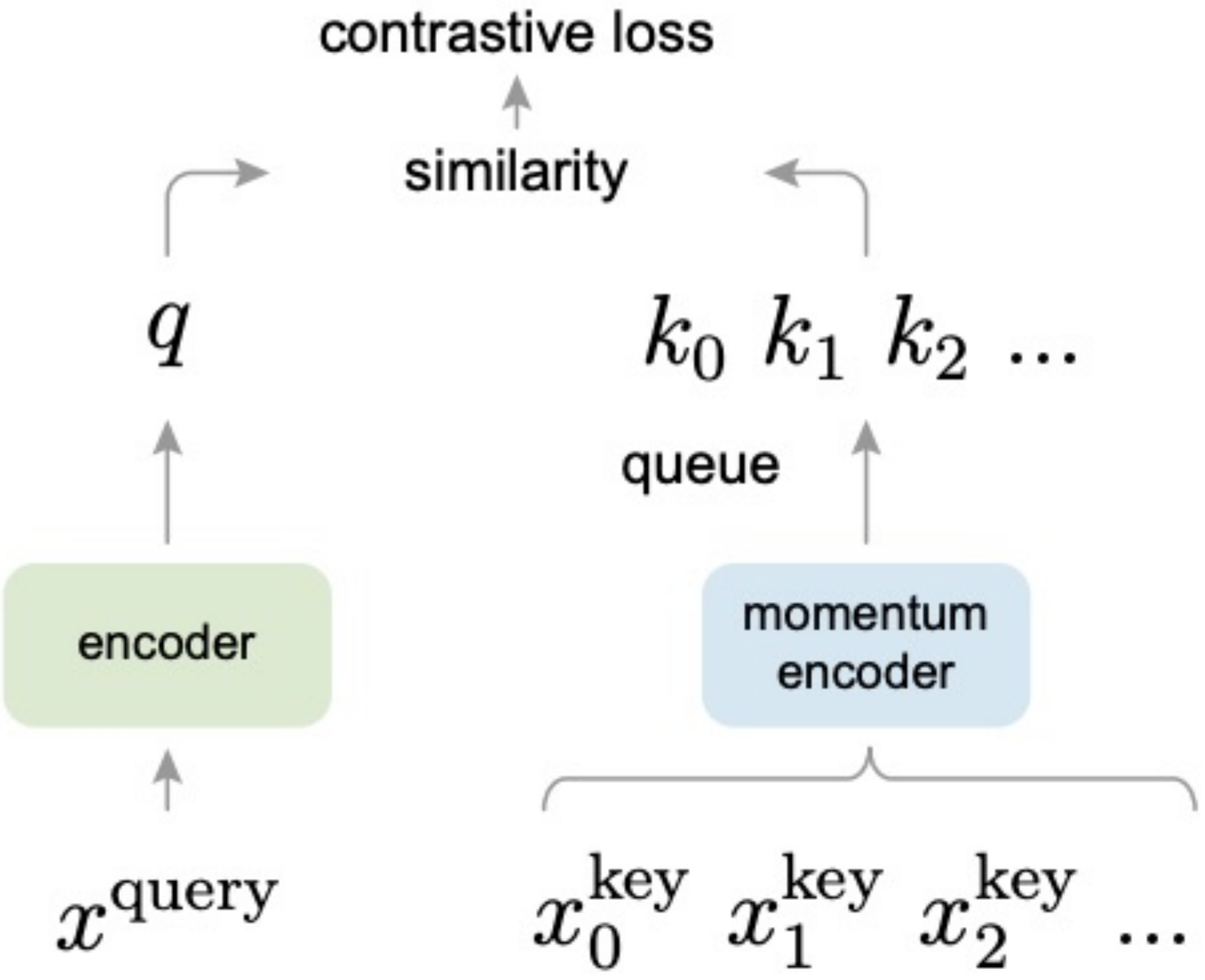
k+ represents the positive paired sample

ki represents one of the K negative samples

K = 60,000

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

MoCo



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

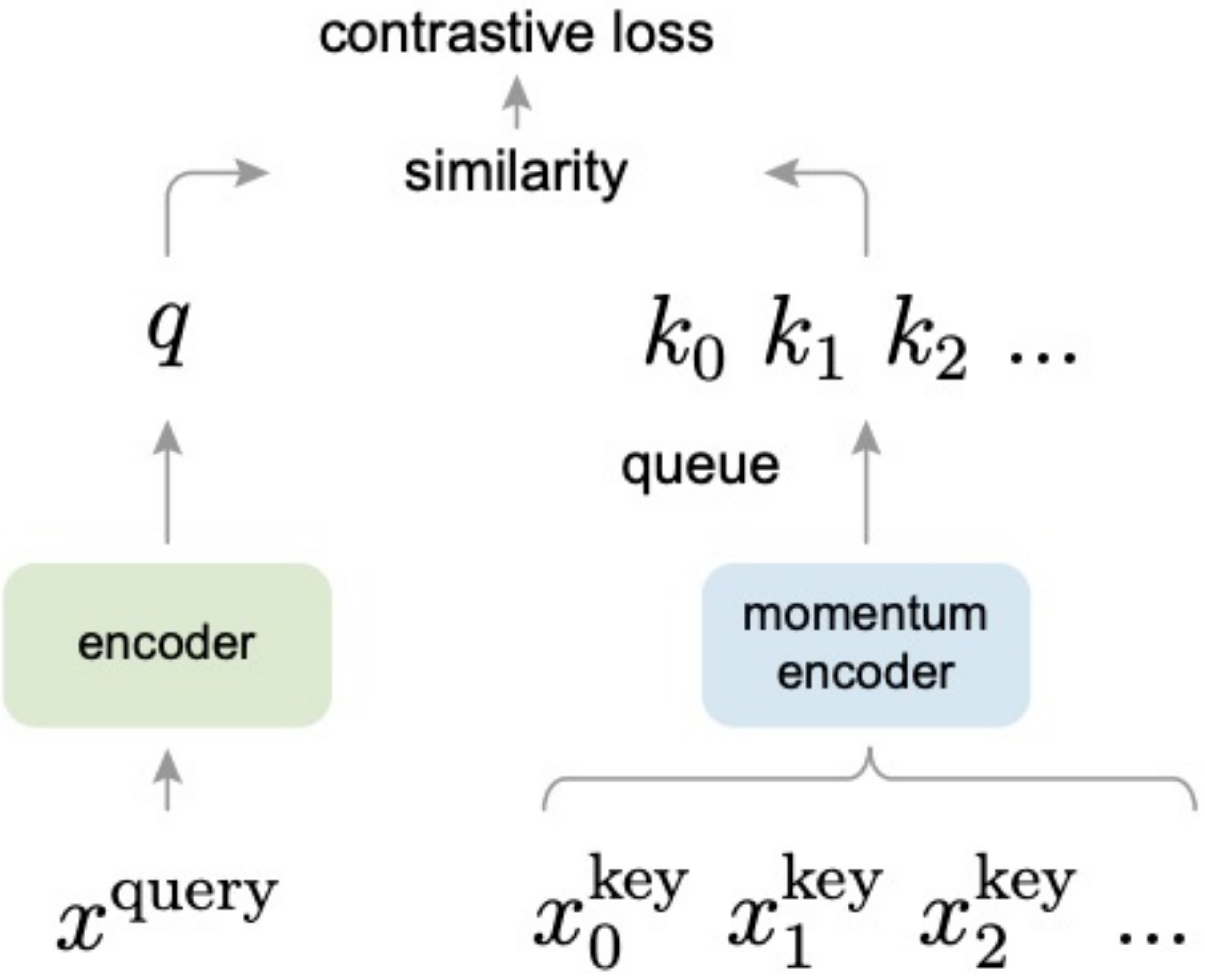
Momentum encoder is a moving average of the encoder

$m = 0.999$

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.

How to Evaluate the Representation

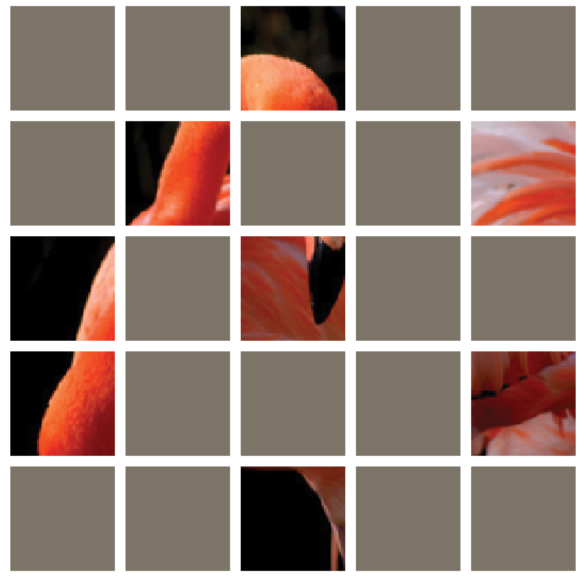
- Linear classification protocol
  - Freeze the features (trained neural network)
  - Train an extra supervised linear classifier (a fully-connected layer followed by softmax)

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

MoCo

case	unsup. pre-train				ImageNet	VOC detection		
	MLP	aug+	cos	epochs	acc.	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**

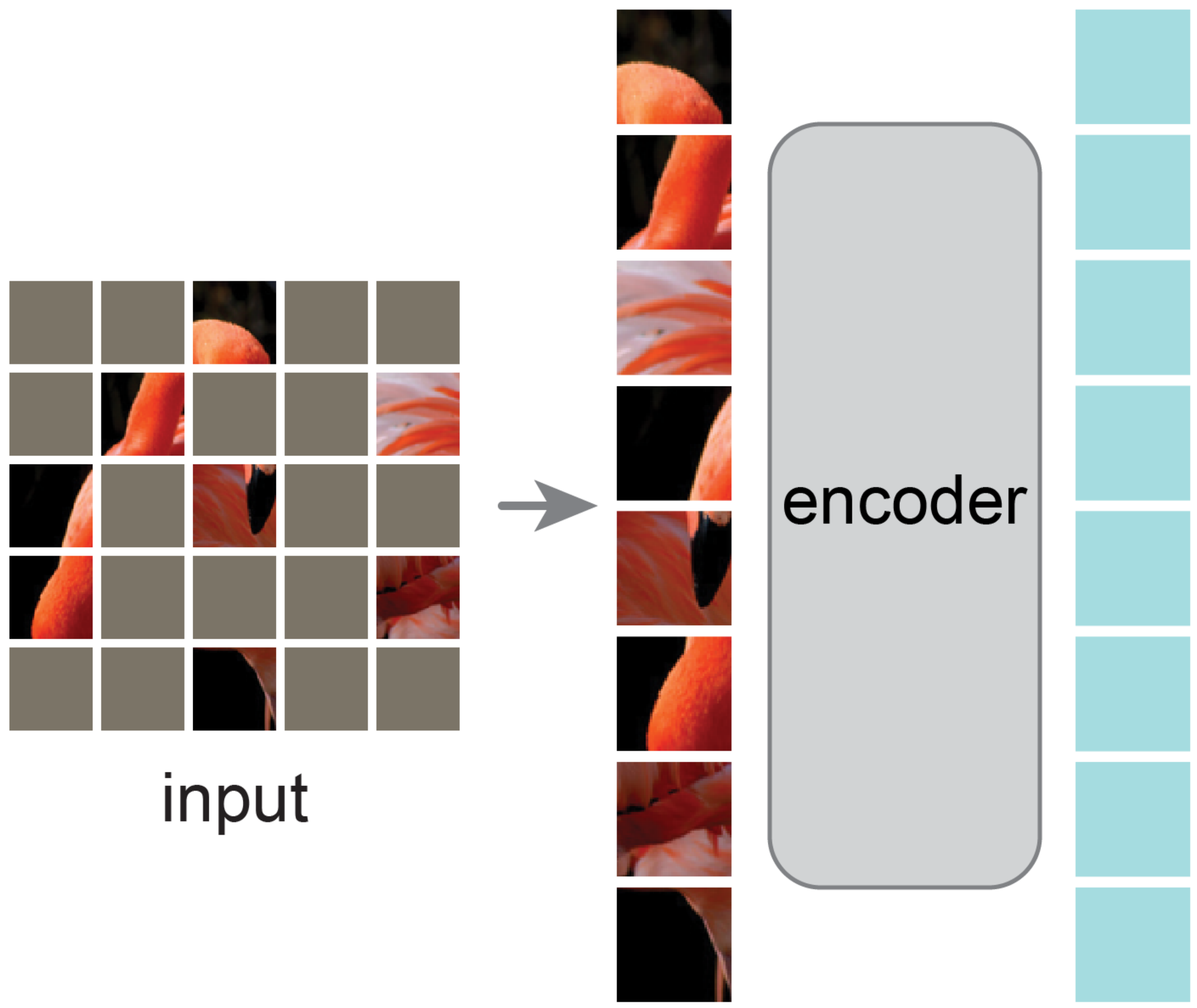
Masked Autoencoder



random masking

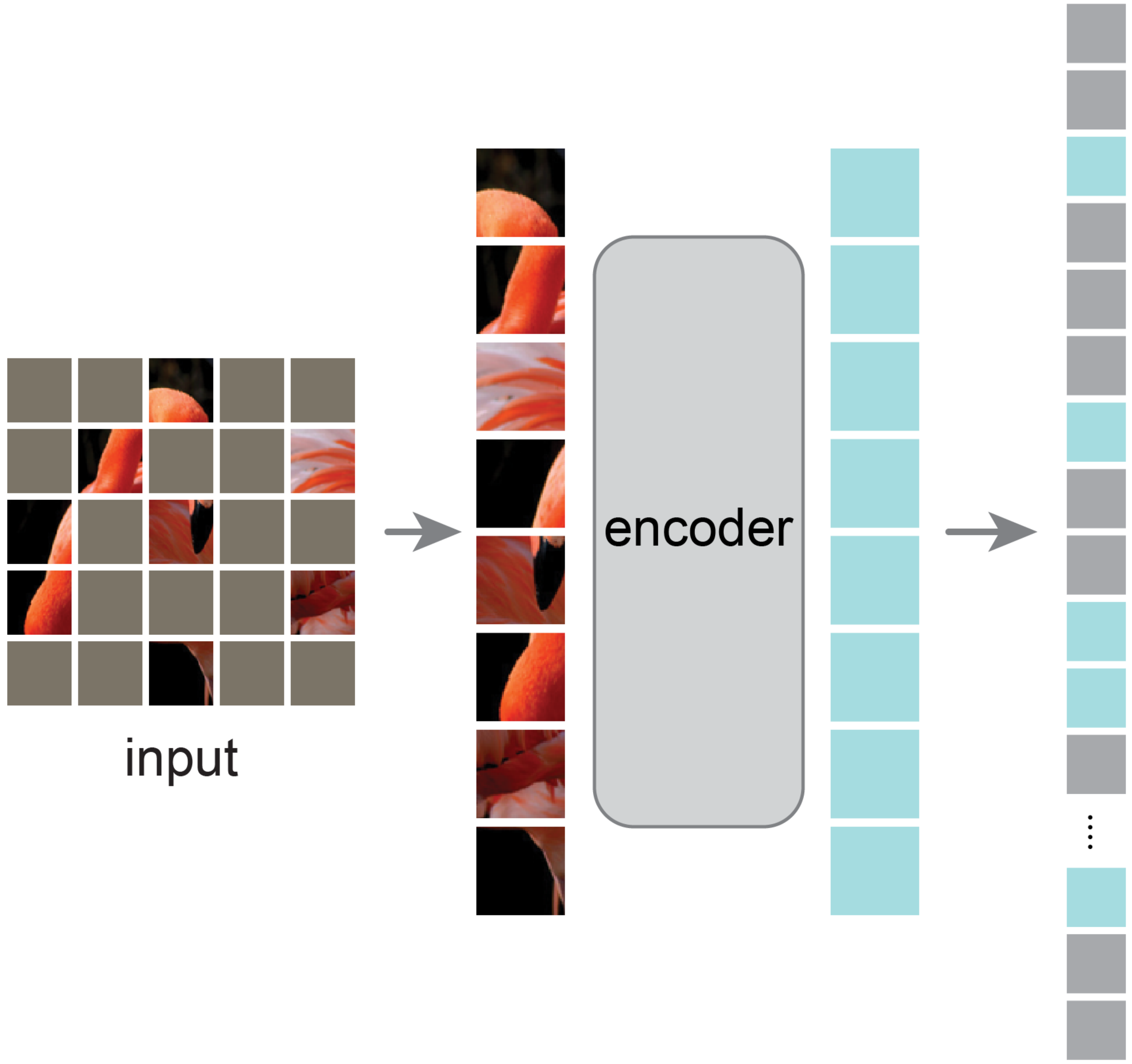
Slides credits: Kaiming He

Masked Autoencoder



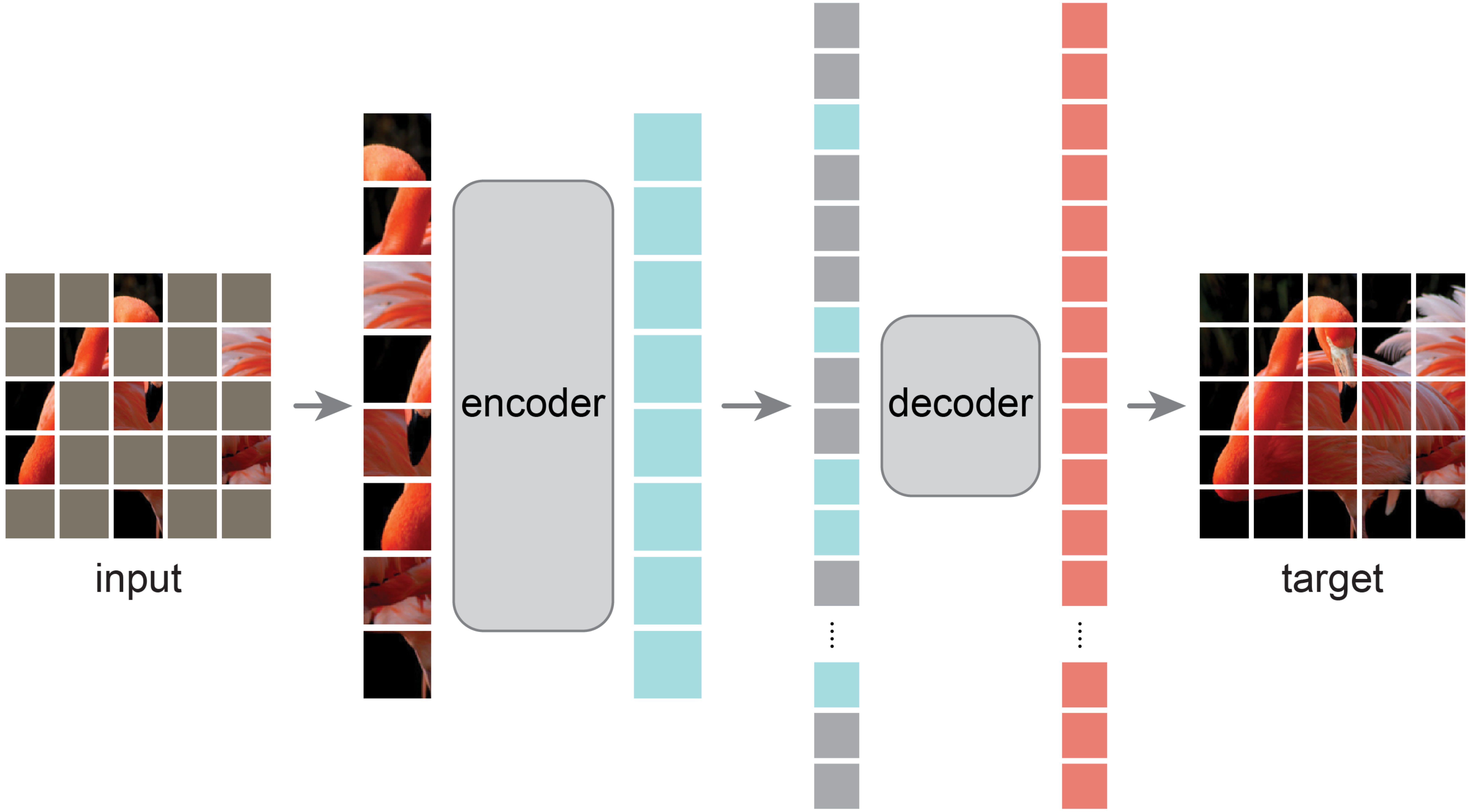
encode visible patches

Masked Autoencoder



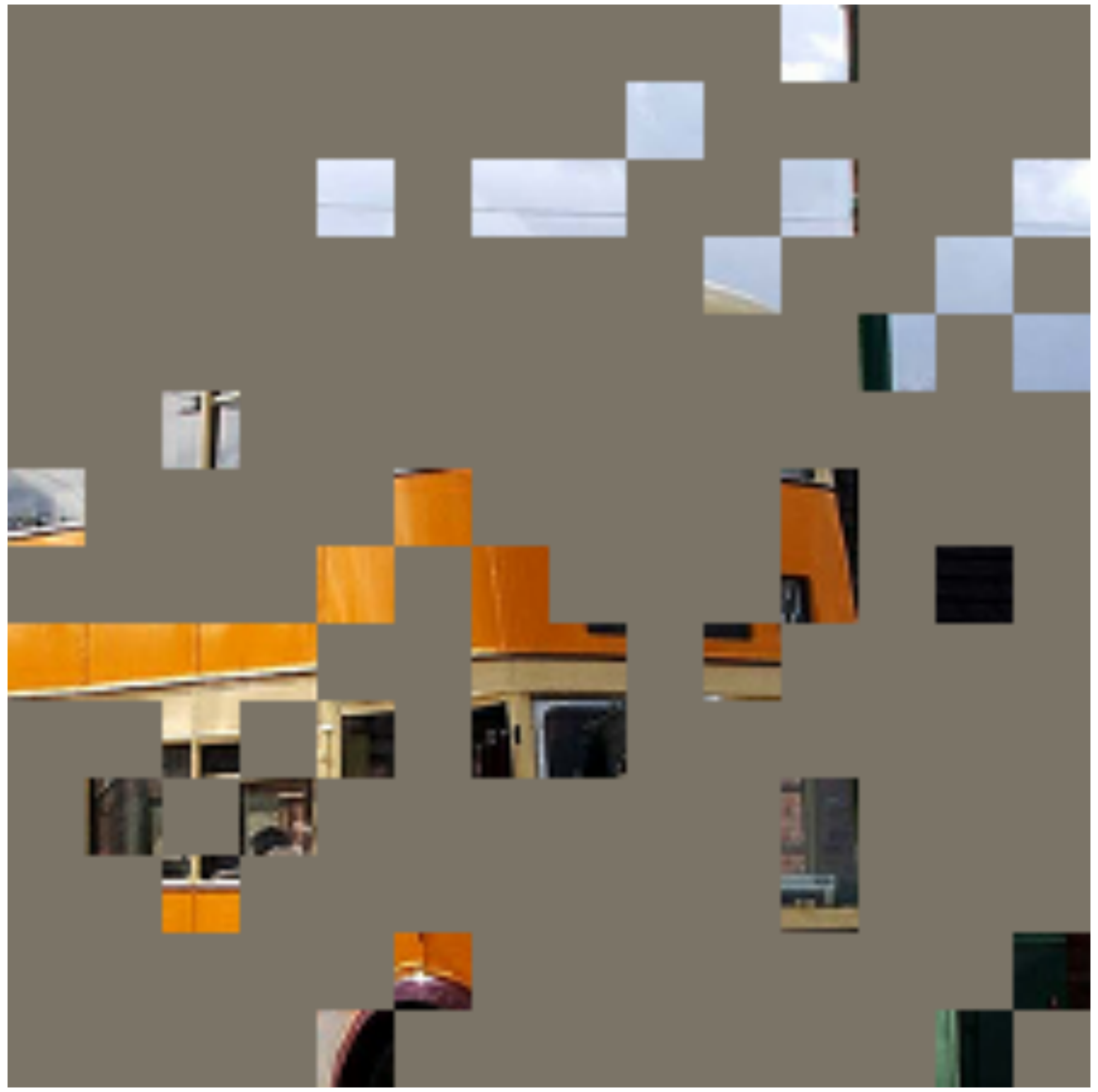
add mask tokens

Masked Autoencoder

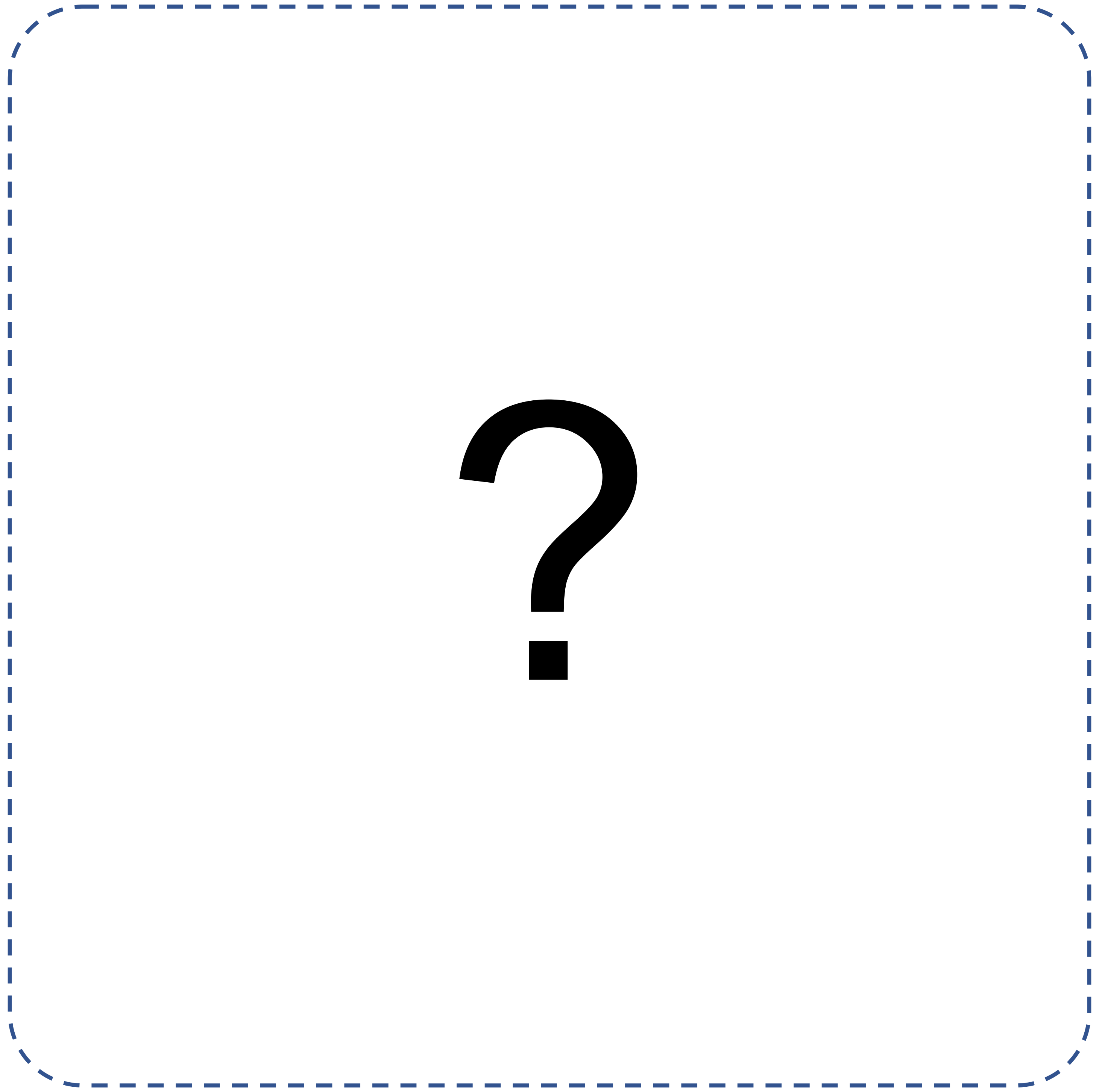


reconstruct

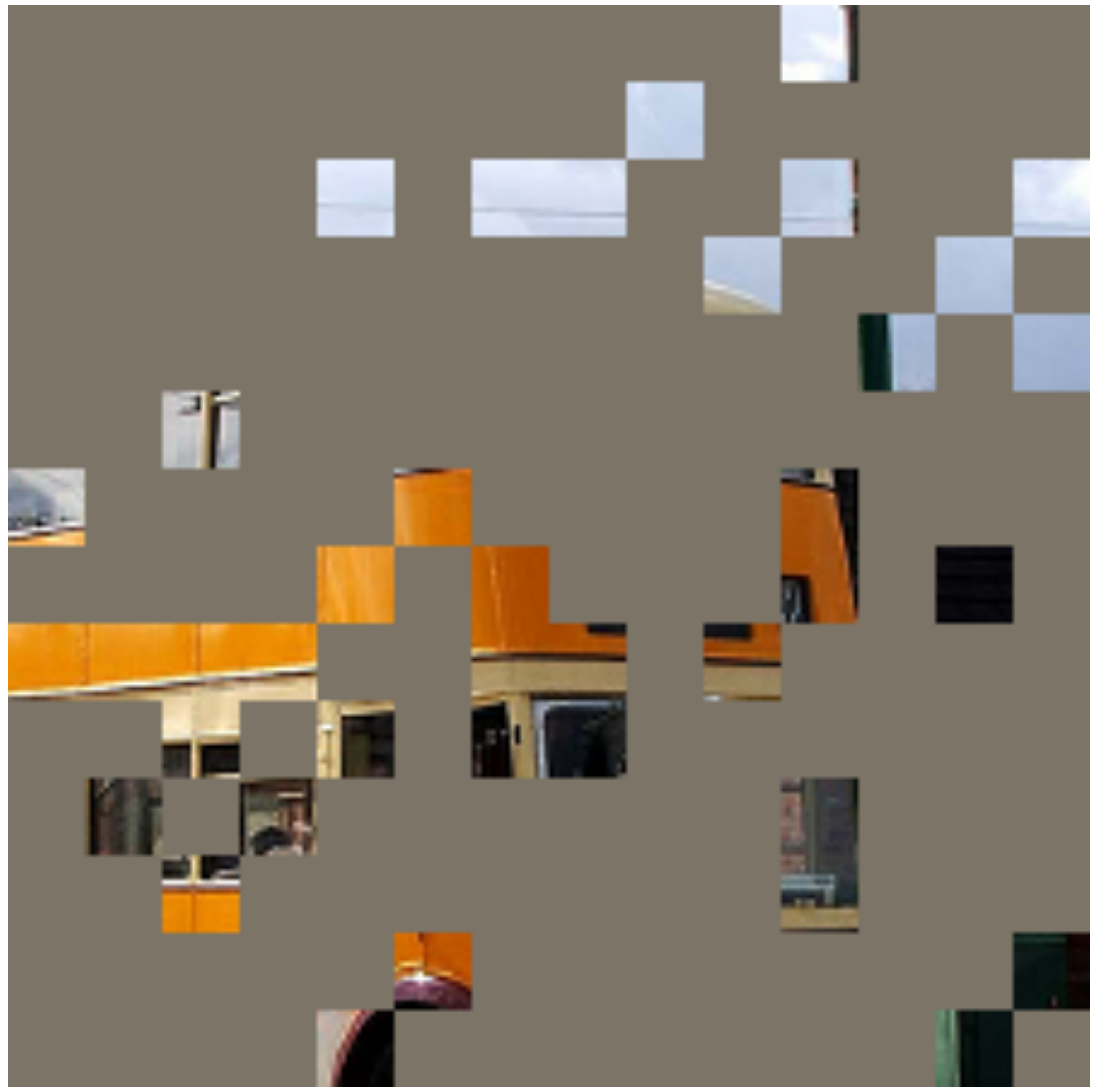
Example Reconstructions



mask 80%



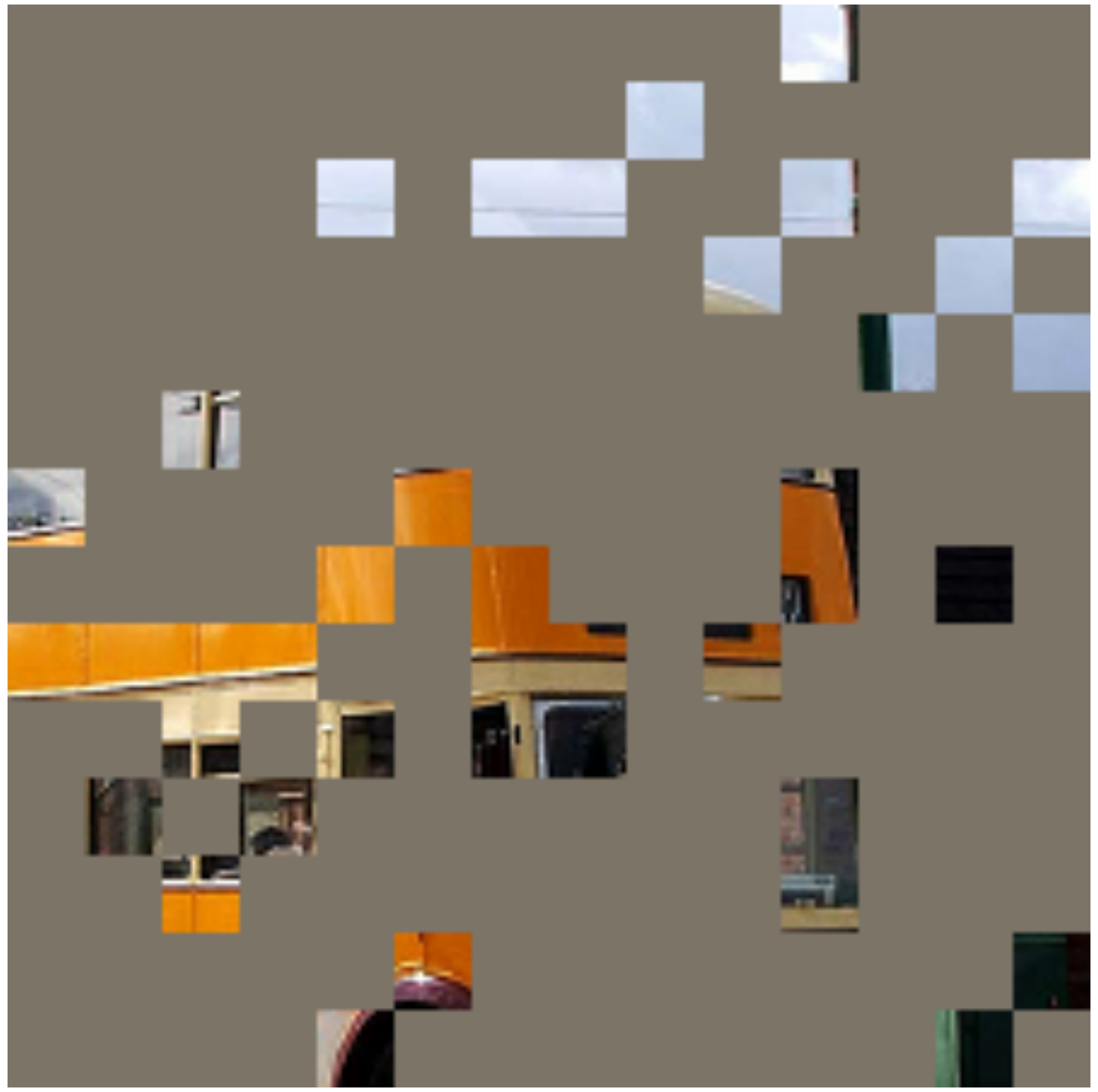
Example Reconstructions



mask 80%

reconstruction

Example Reconstructions



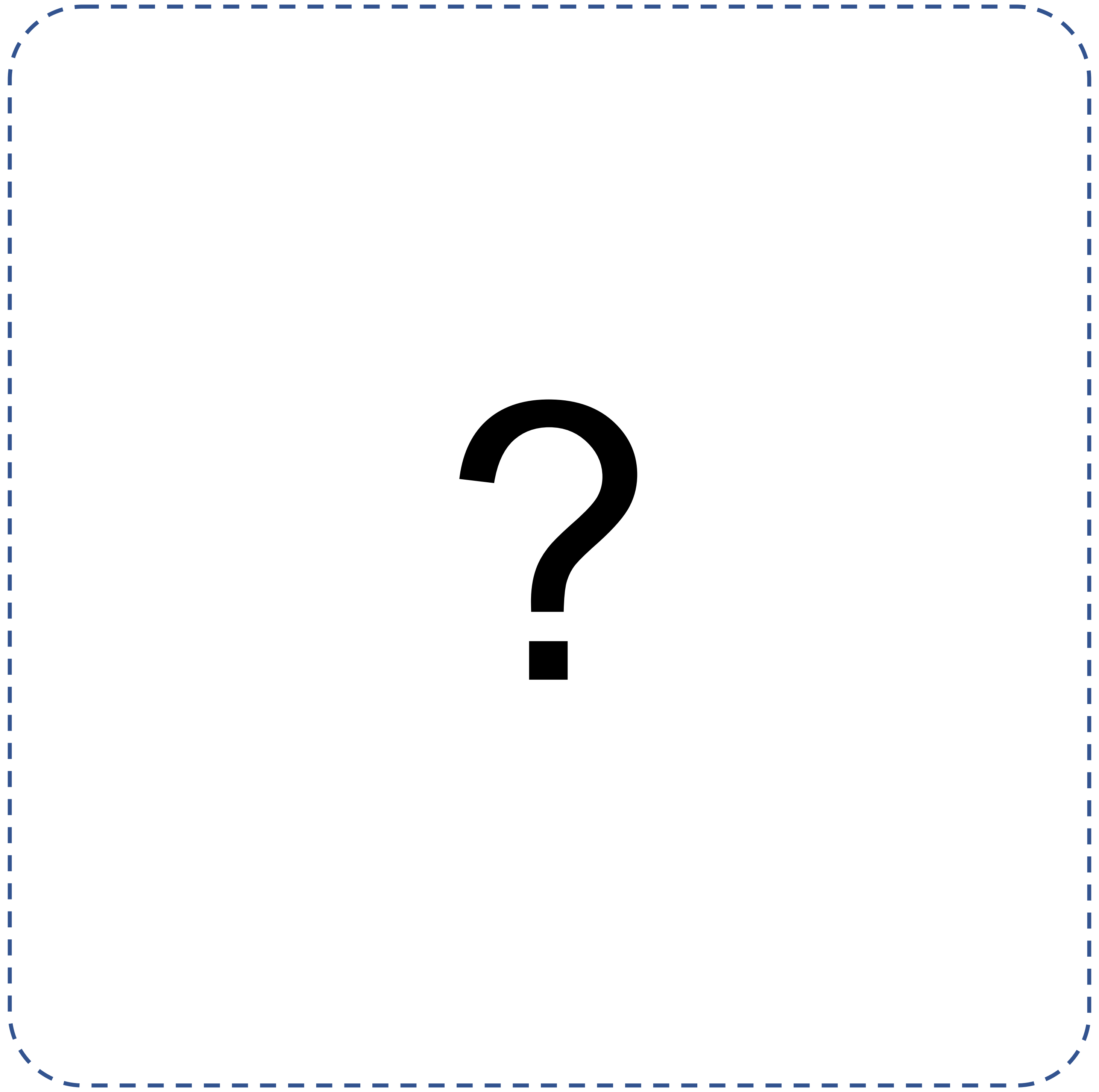
mask 80%

reconstruction

ground-truth

Example Reconstructions

mask 80%



Example Reconstructions

mask 80%

reconstruction

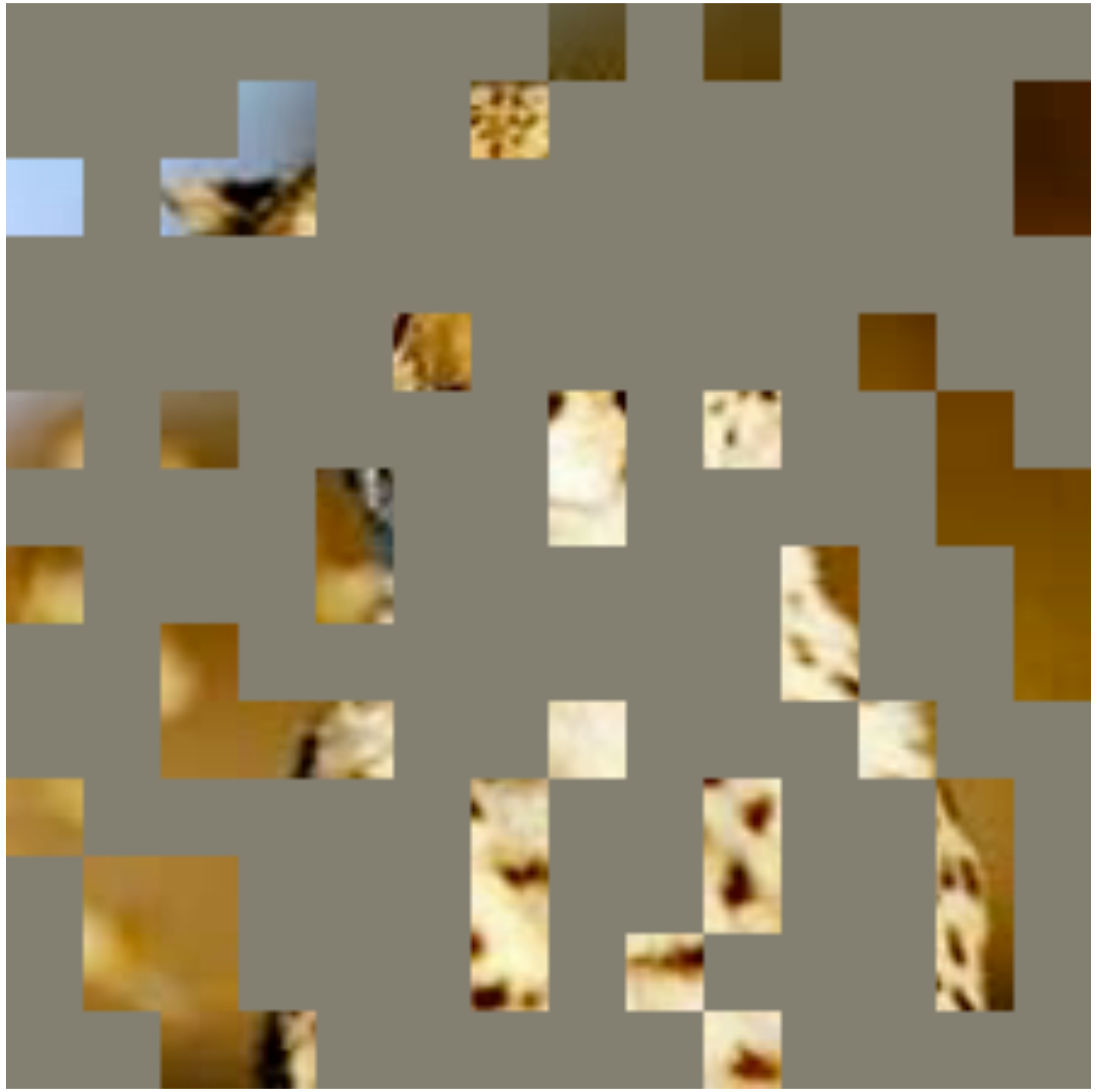
Example Reconstructions

mask 80%

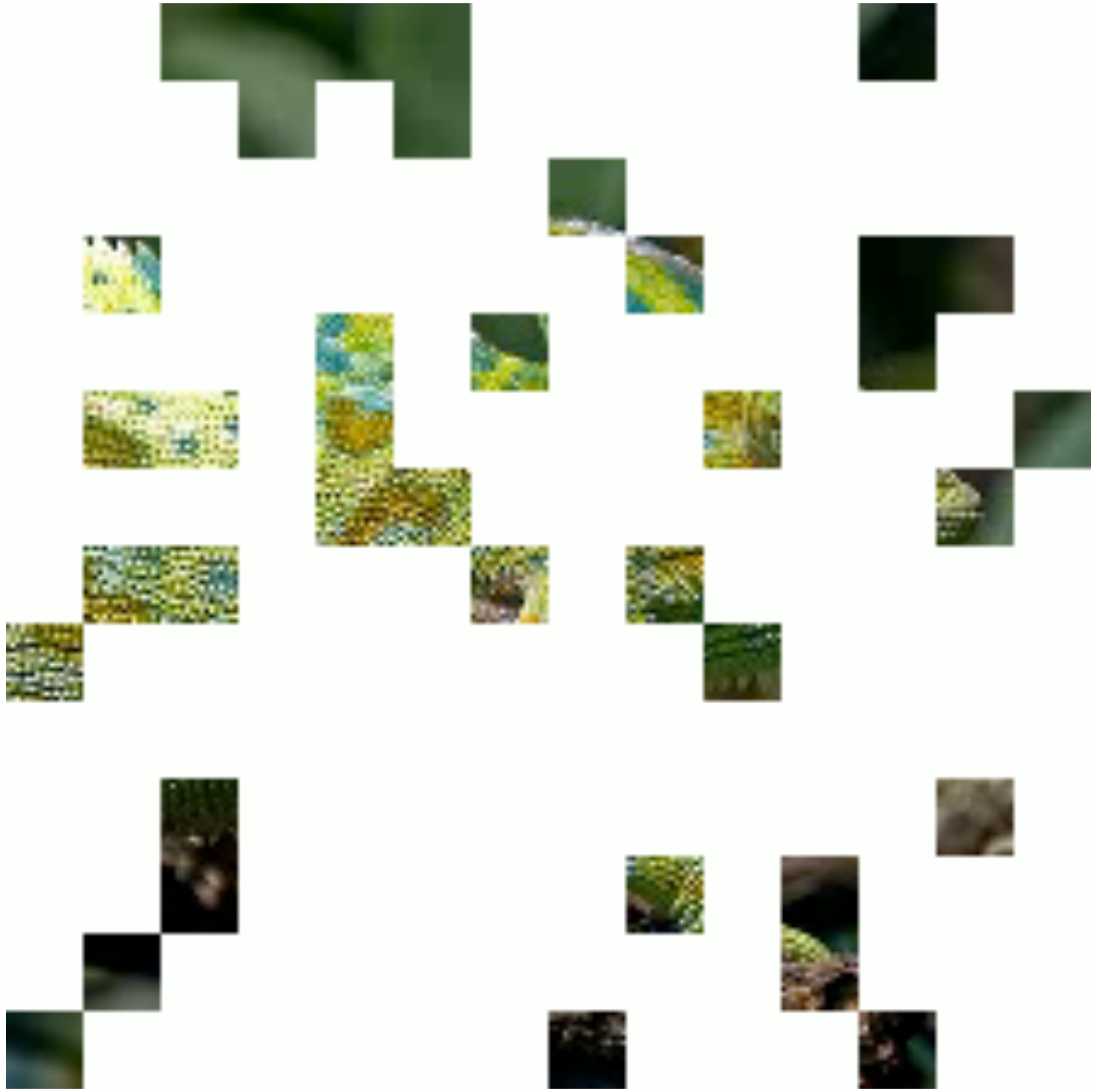
reconstruction

ground-truth

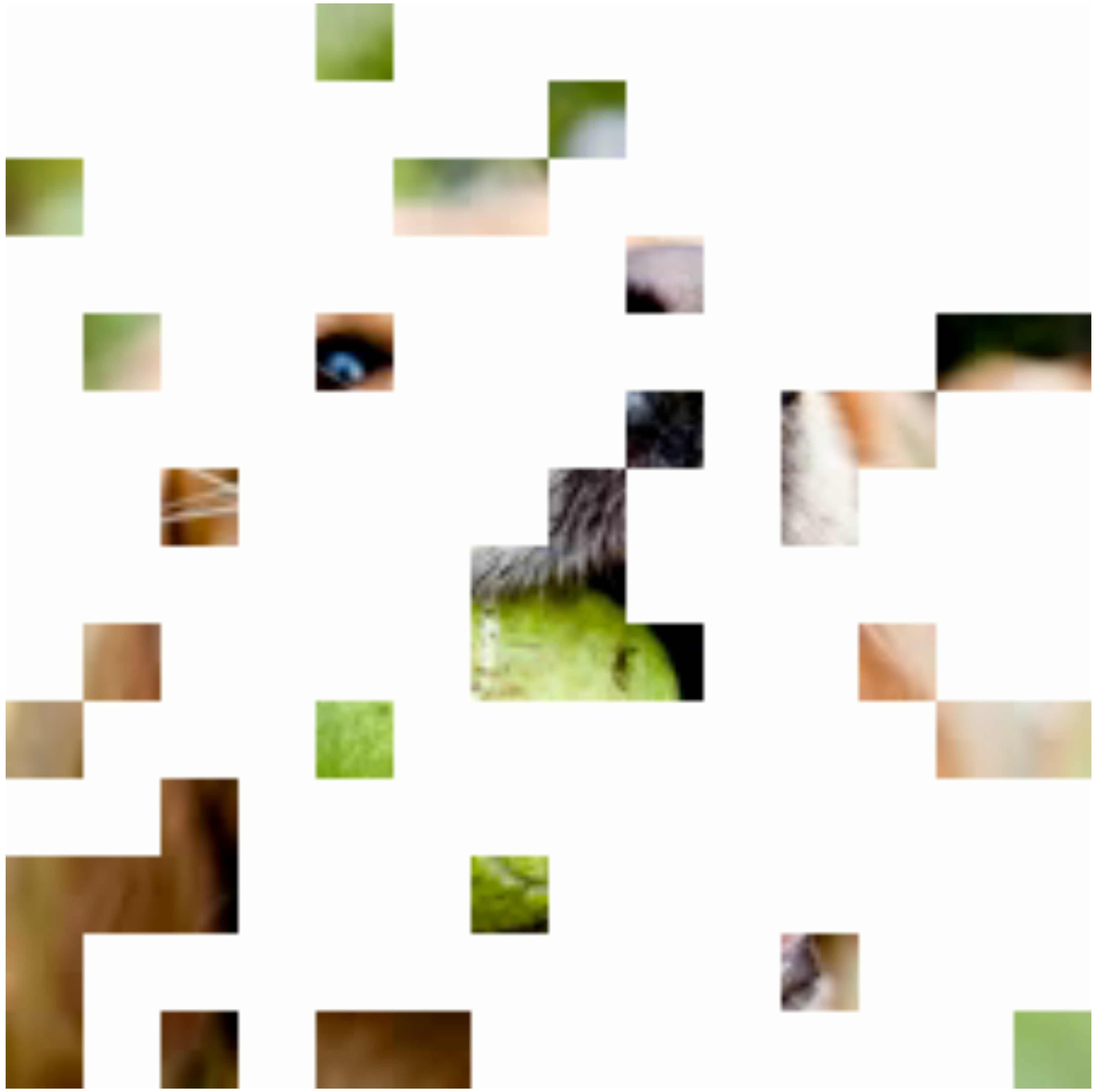
reconstruction vs. # epochs



reconstruction vs. # epochs

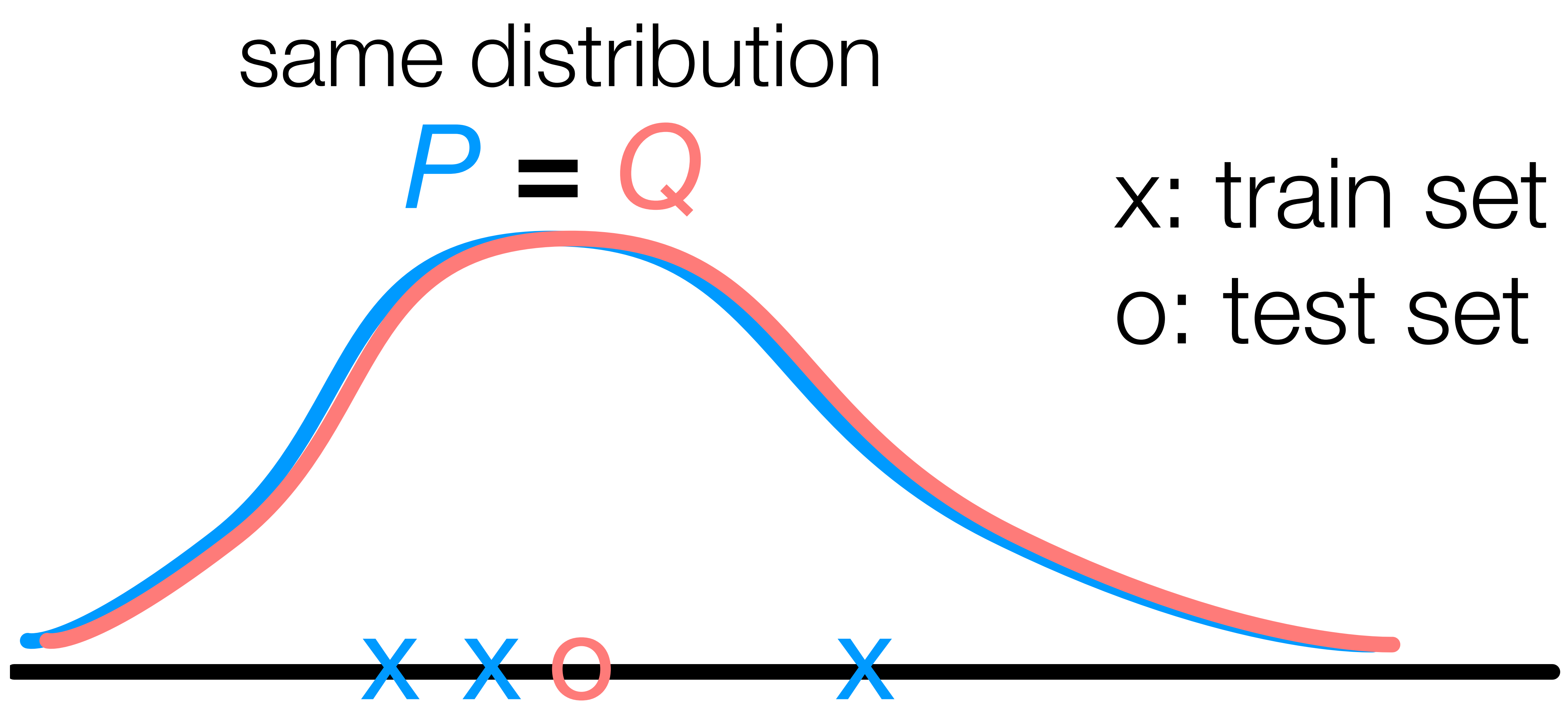


reconstruction vs. # epochs



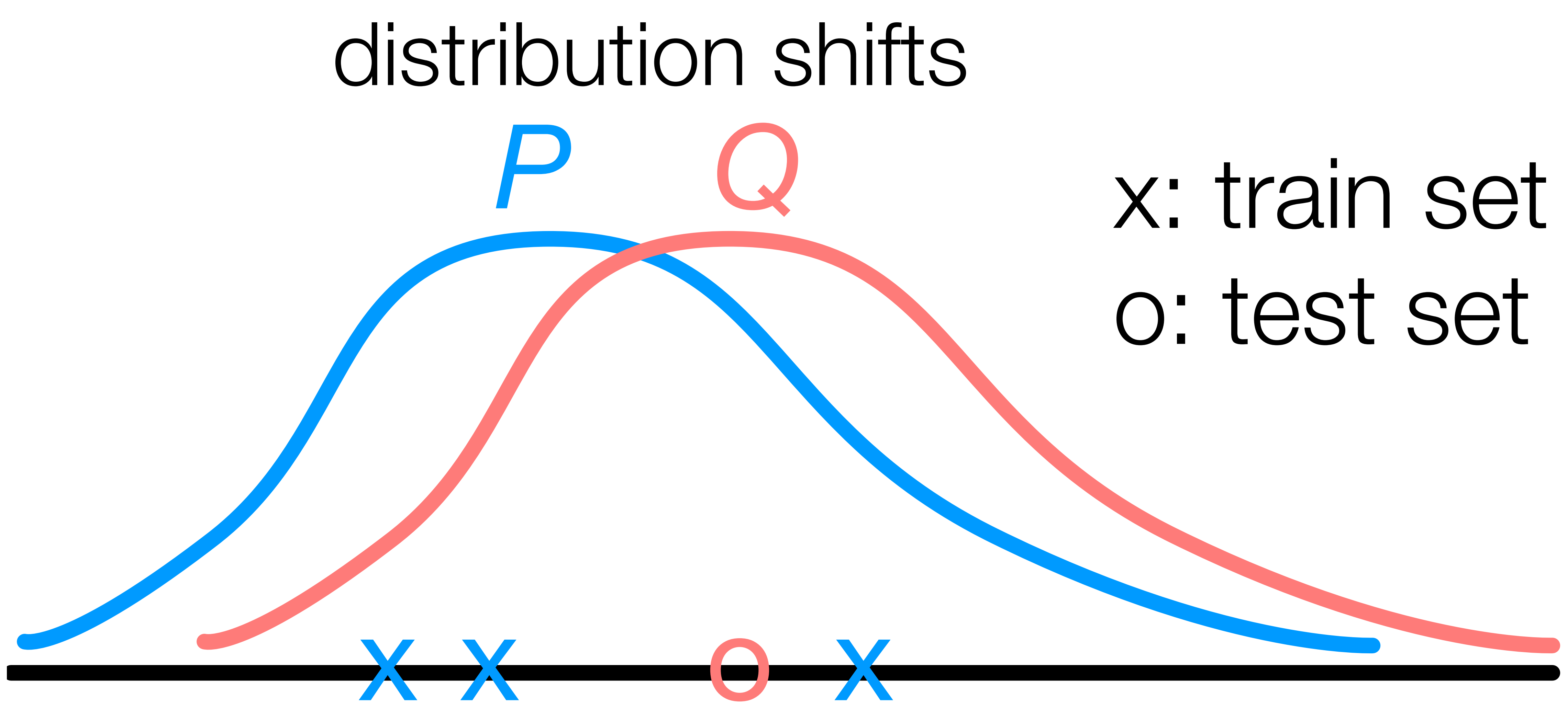
reconstruction vs. # epochs

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



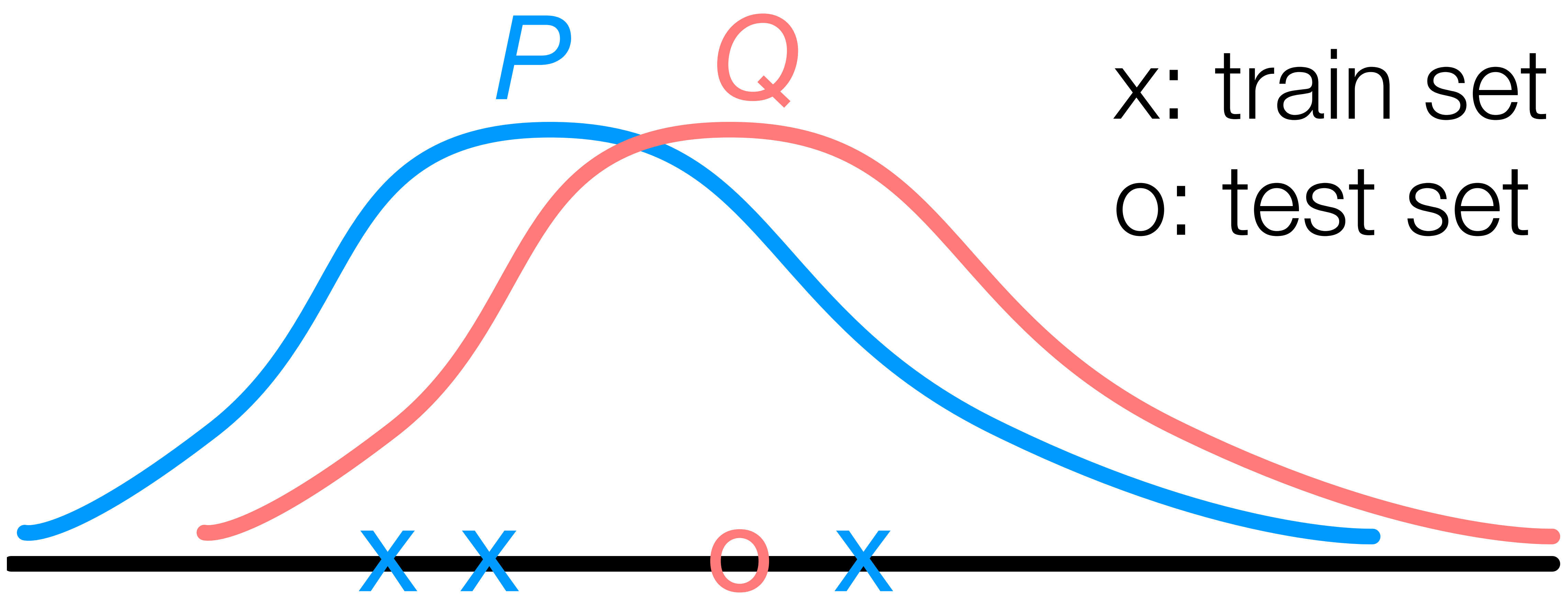
- **In theory**: same distribution for training and testing

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

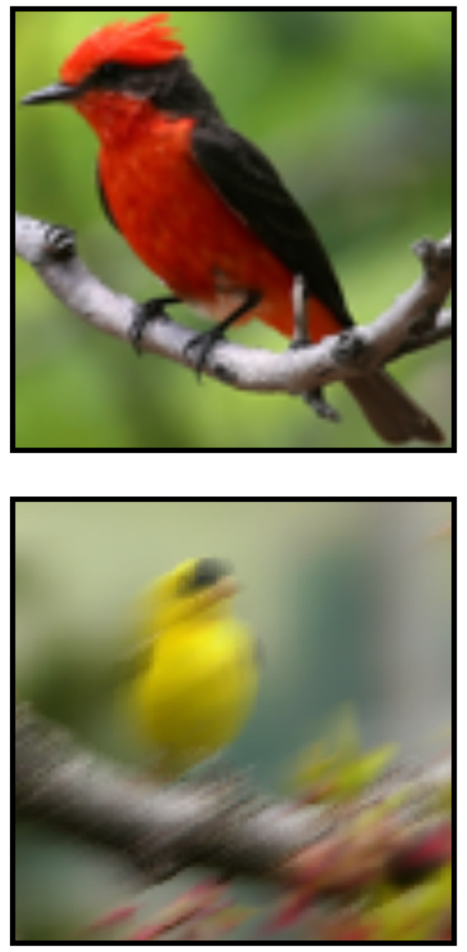


- **In theory**: same distribution for training and testing
- **In the real word**: distribution shifts are everywhere

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}


0°


90°


180°

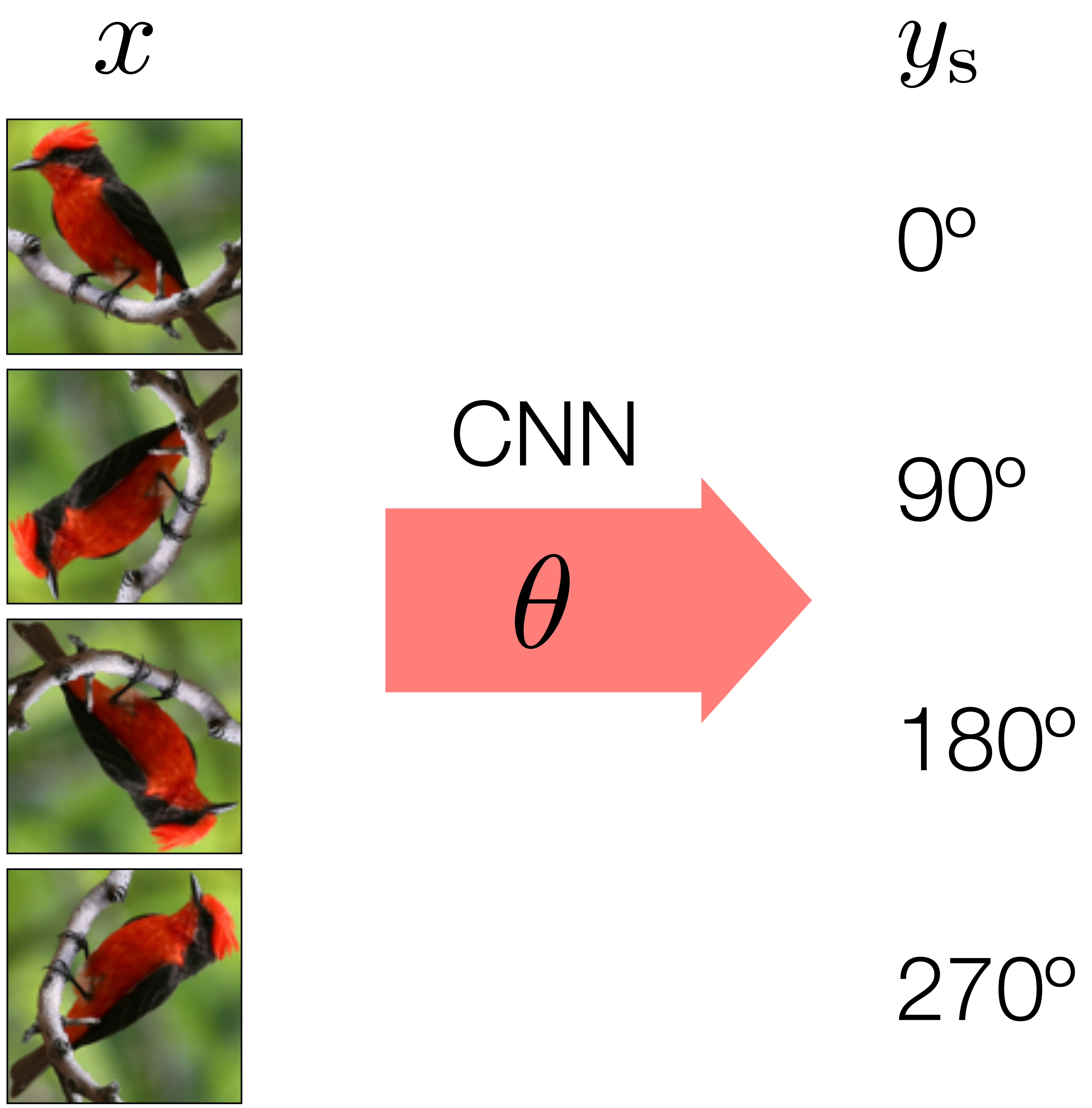

270°

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

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

Rotation prediction as self-supervision

(Gidaris et al. 2018)



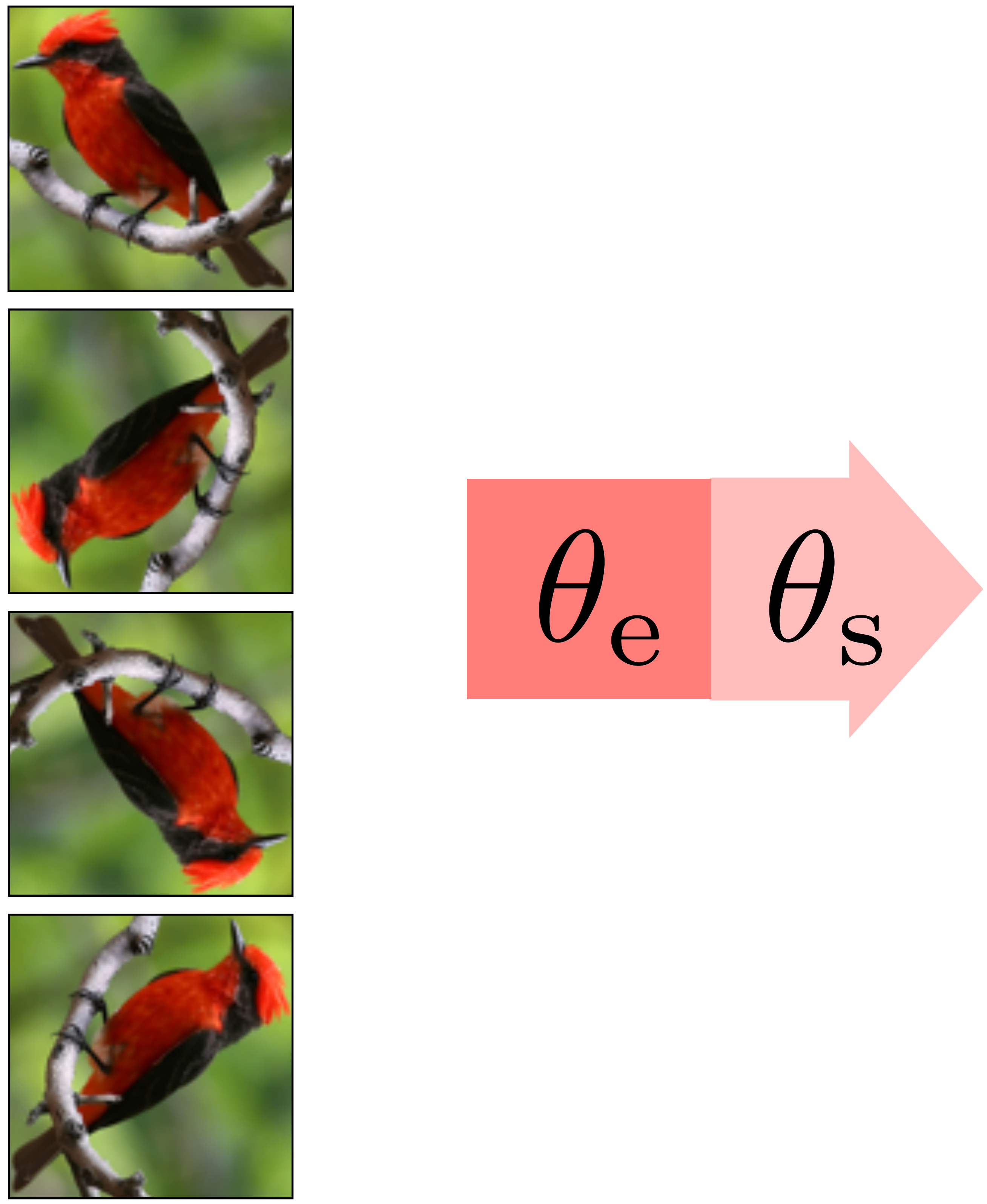
- Create labels from unlabeled input
- Rotate input image by multiples of 90º
- Produce a four-way classification problem

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

Rotation prediction as self-supervision

(Gidaris et al. 2018)

x 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)

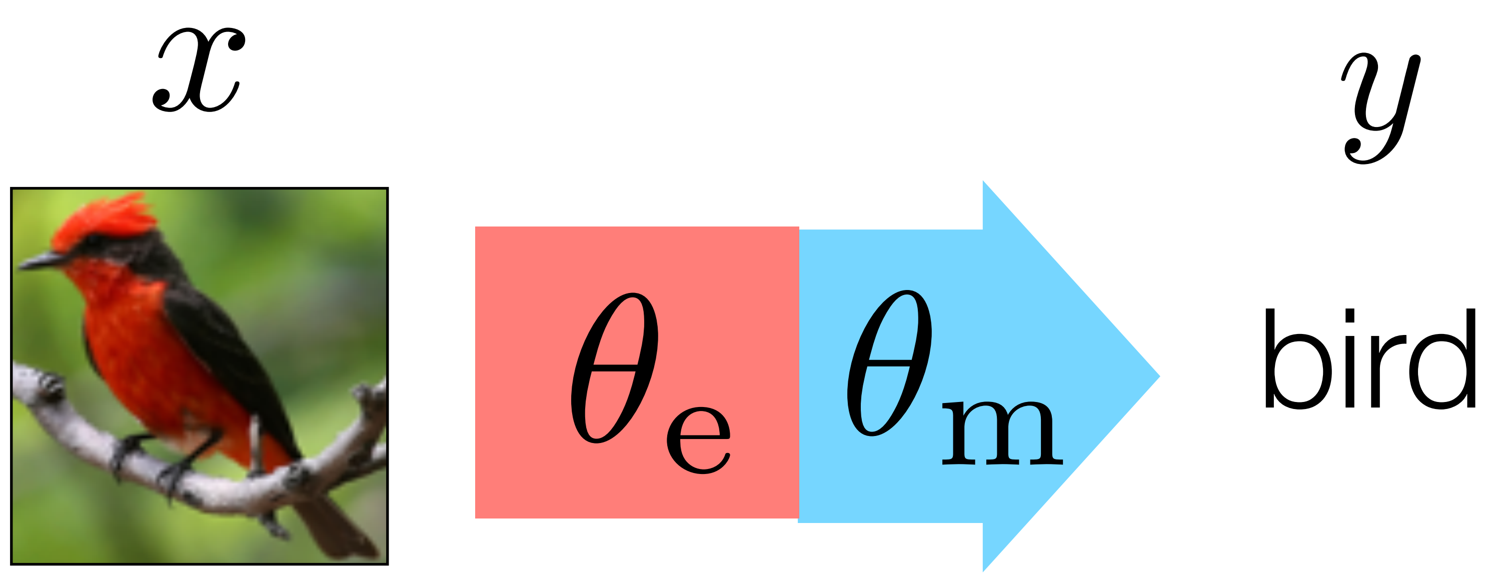
θ_{e}

- 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

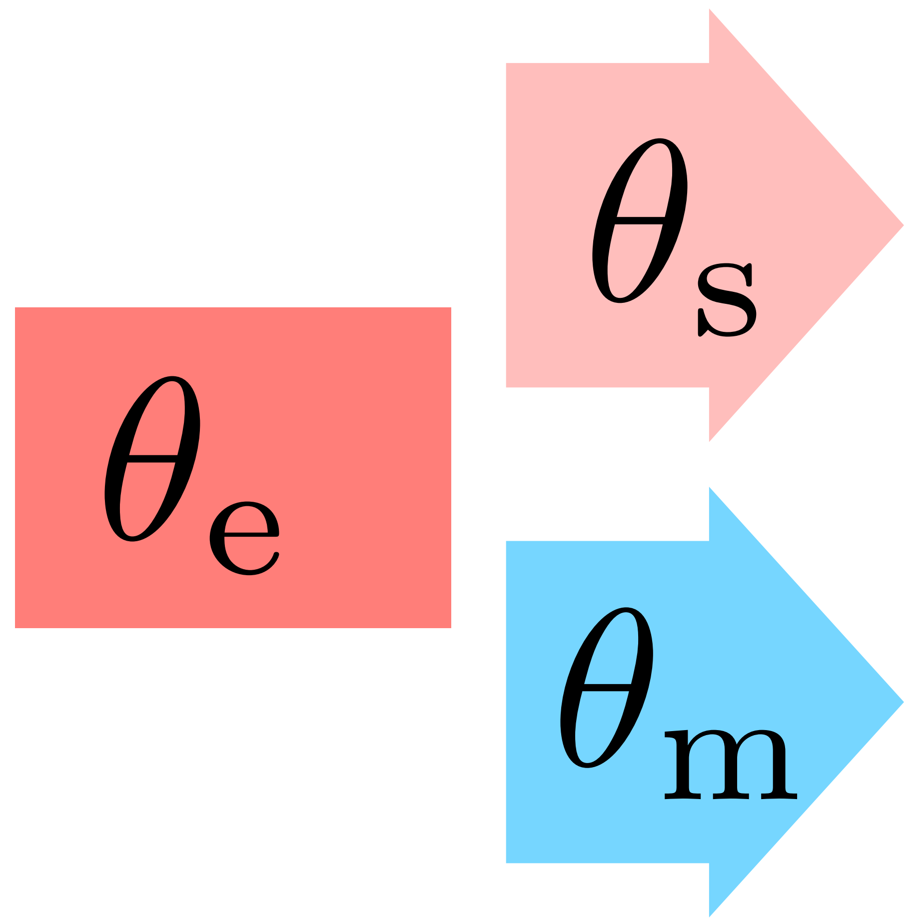
(Gidaris et al. 2018)



- 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

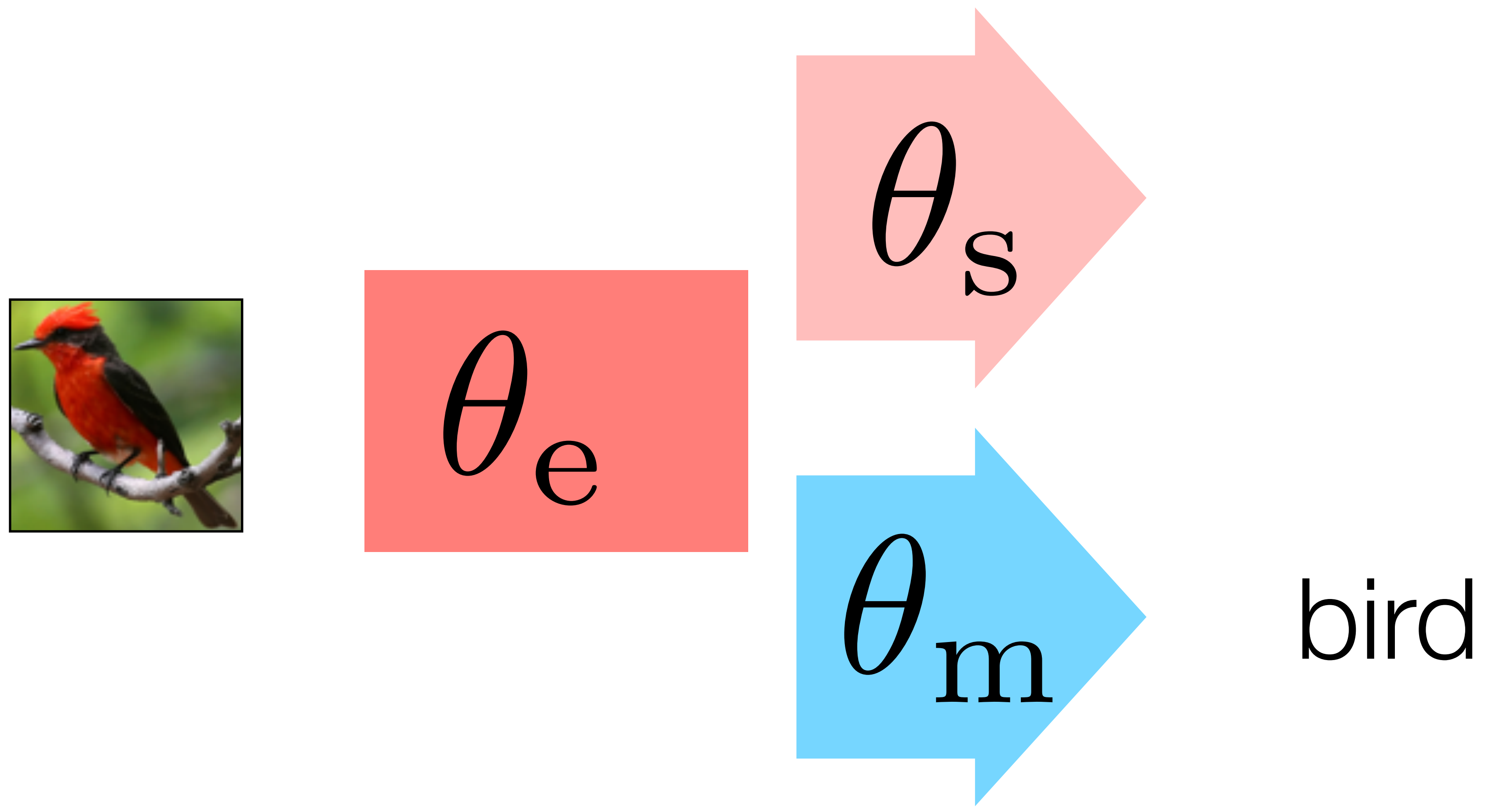


network
architecture

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

Algorithm for TTT

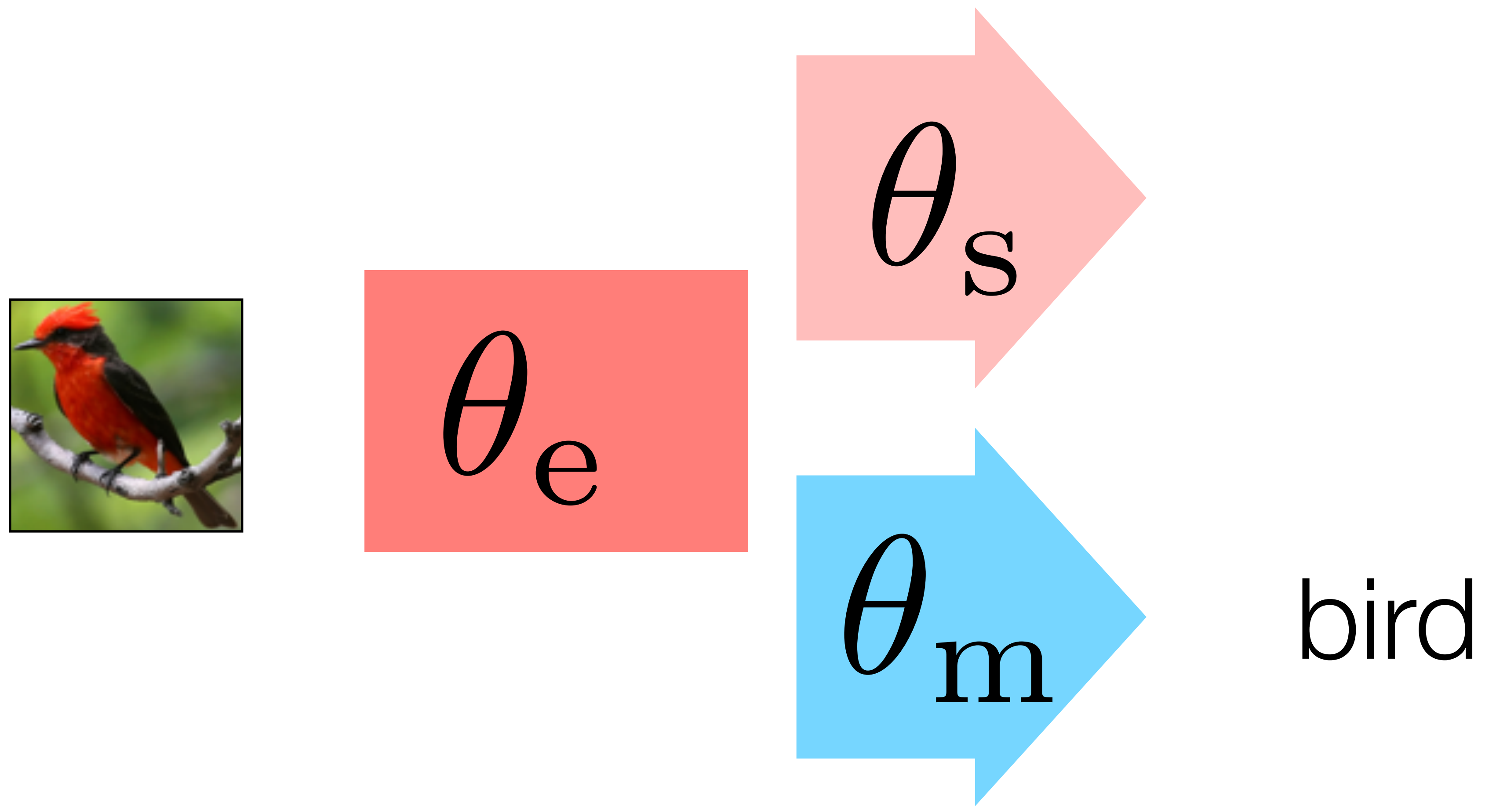
training



Algorithm for TTT

training

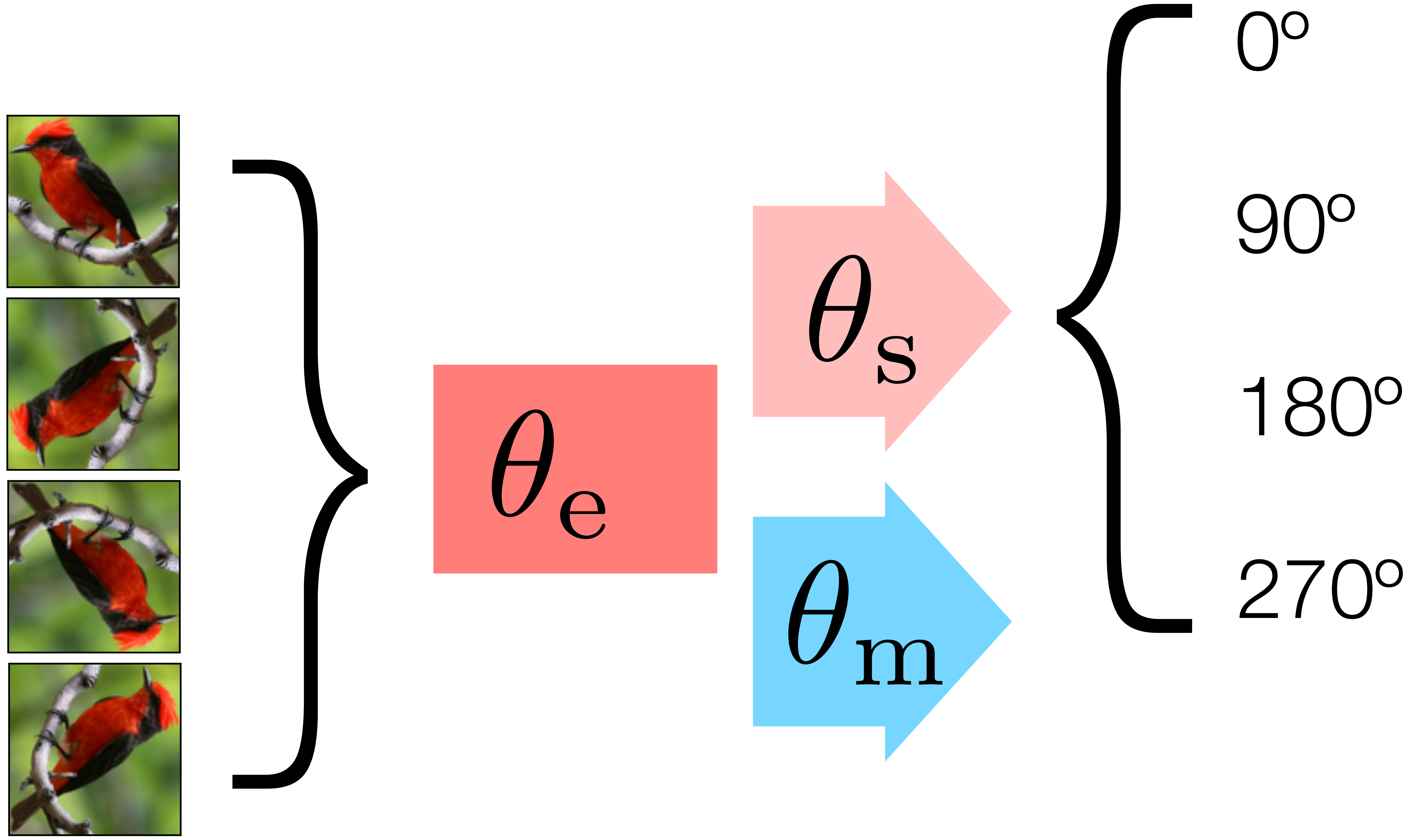
$\ell_{\mathrm{m}}(x, y; \theta_{\mathrm{e}}, \theta_{\mathrm{m}})$



Algorithm for TTT

training

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

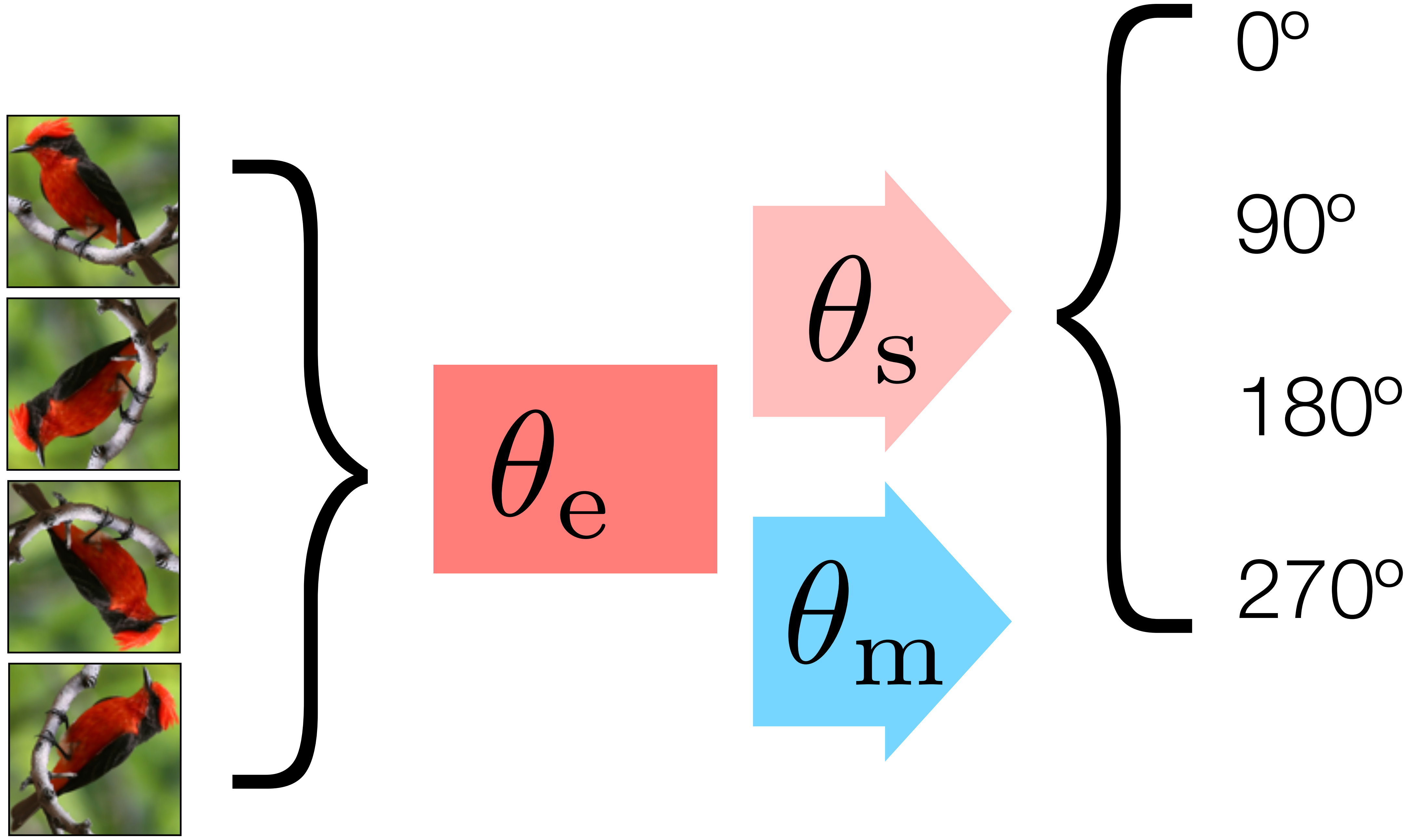


Algorithm for TTT

training

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

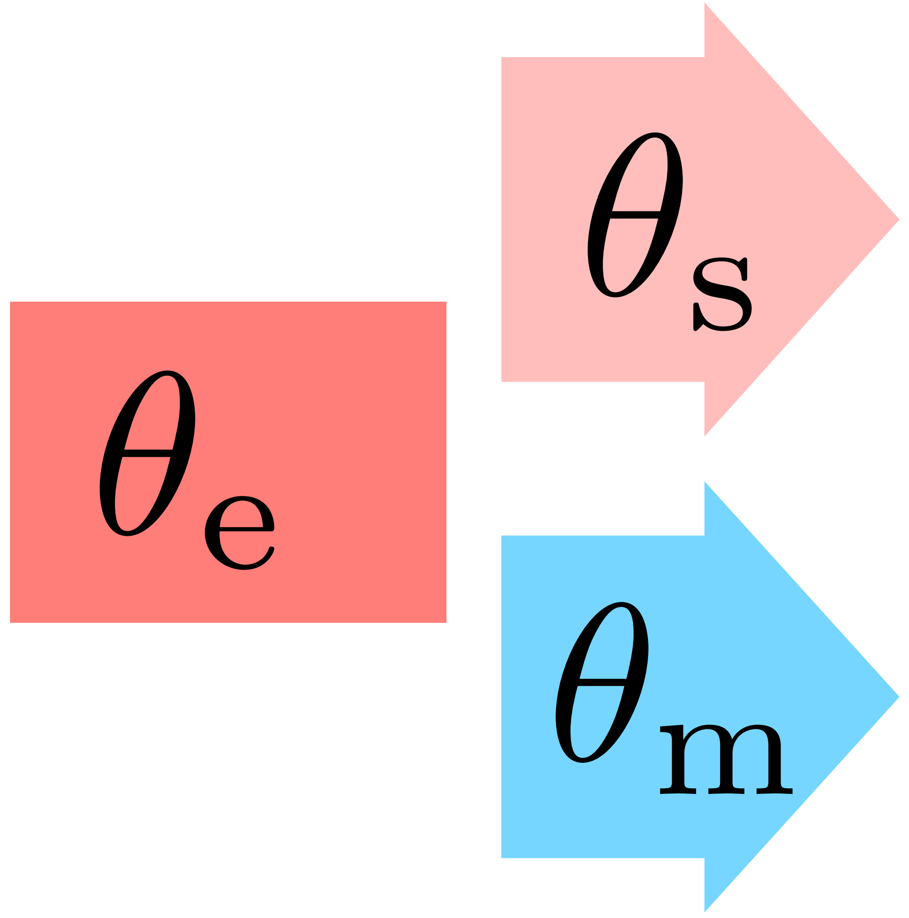
$+ \ell_s(x, y_{\mathrm{s}}; \theta_e, \theta_s)$



Algorithm for TTT

training

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

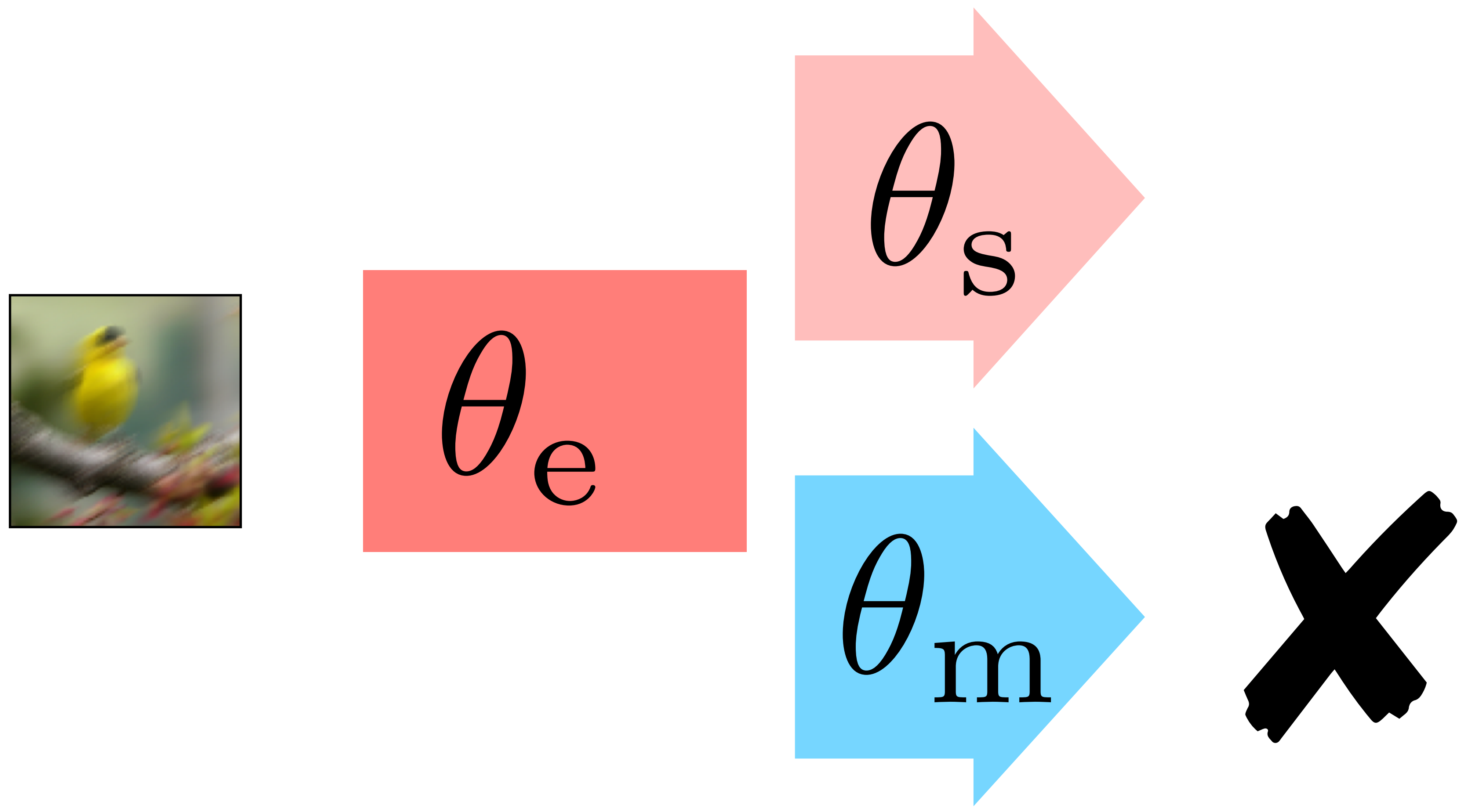


Algorithm for TTT

training

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

testing

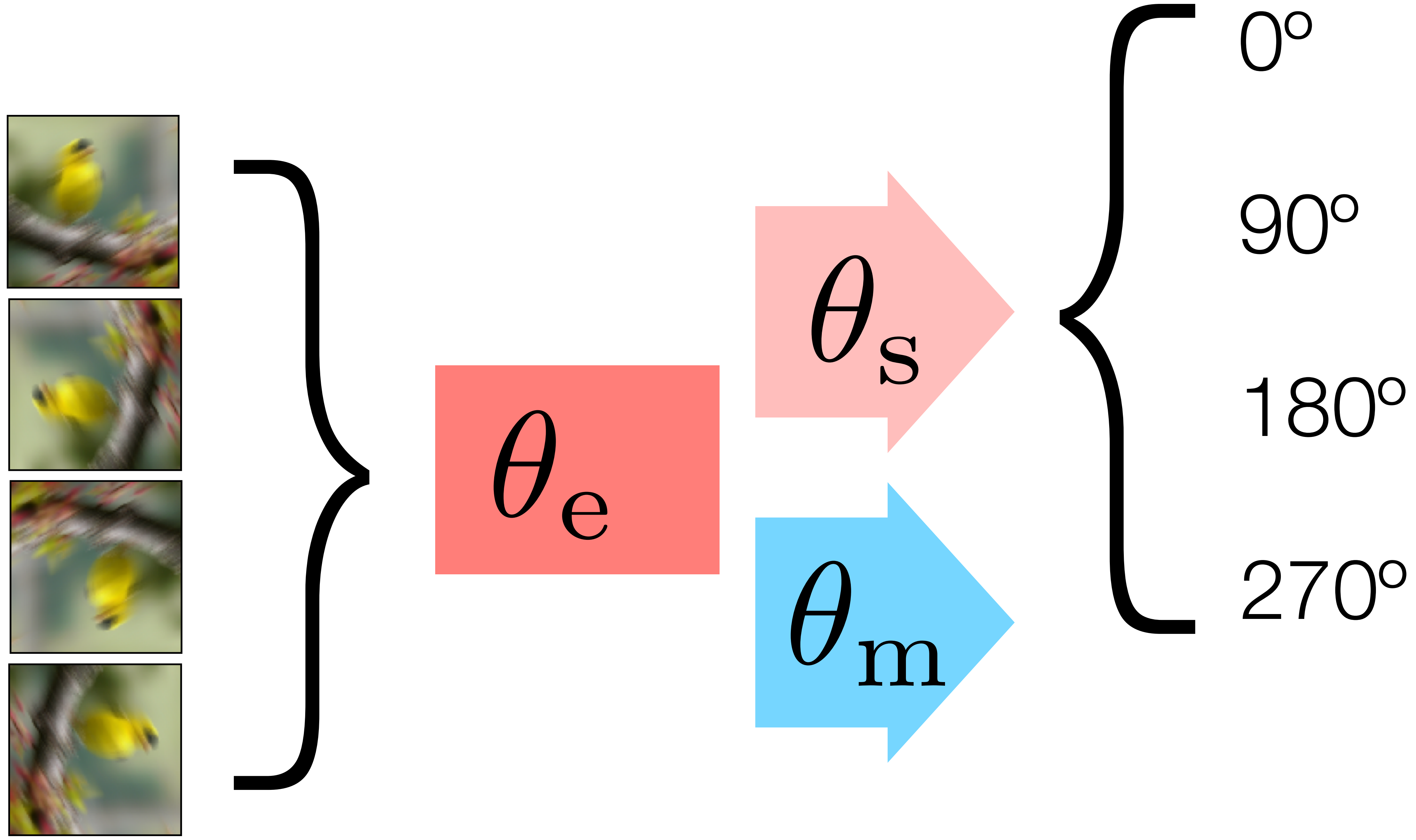


Algorithm for TTT

training

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

testing



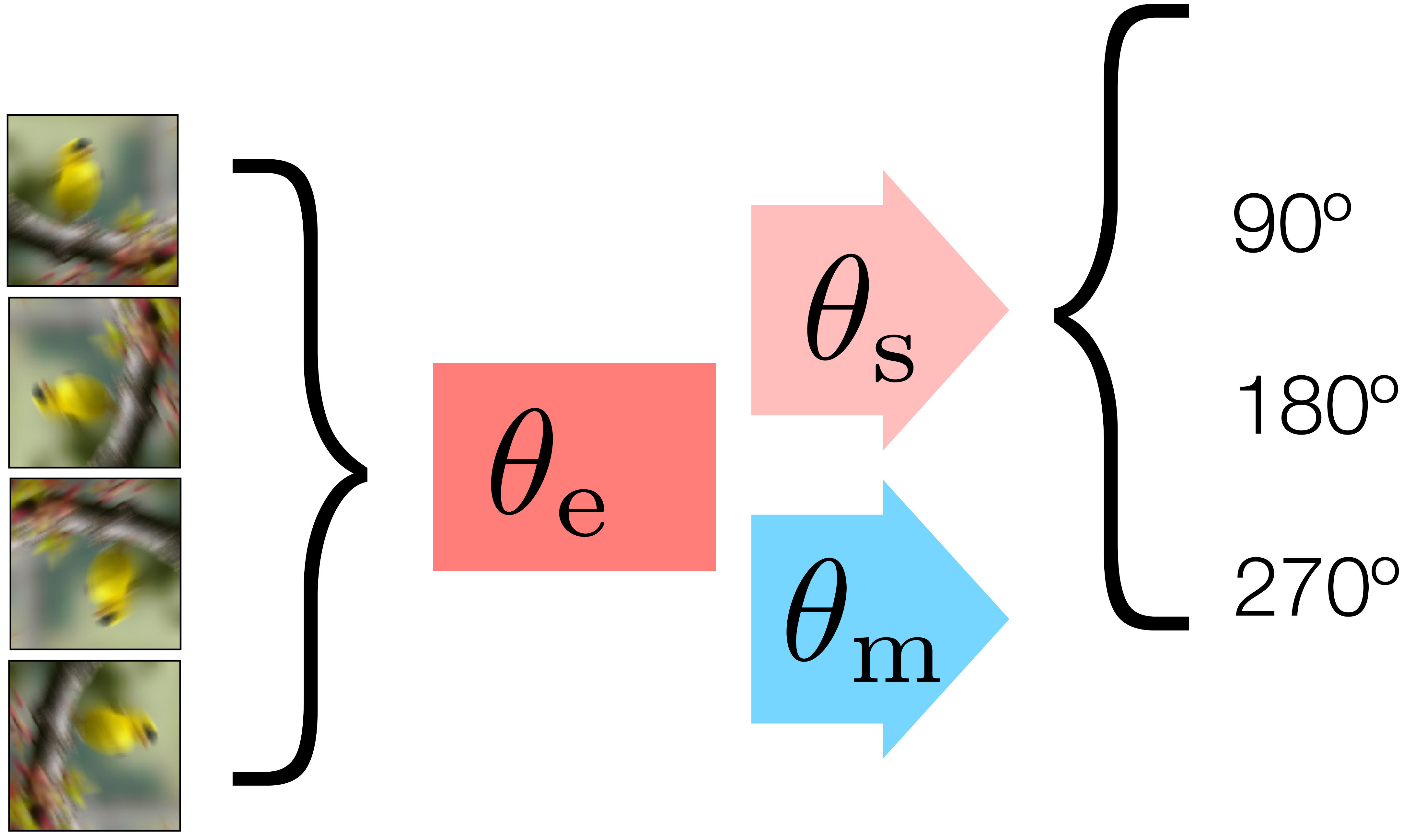
Algorithm for TTT

training

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

testing

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



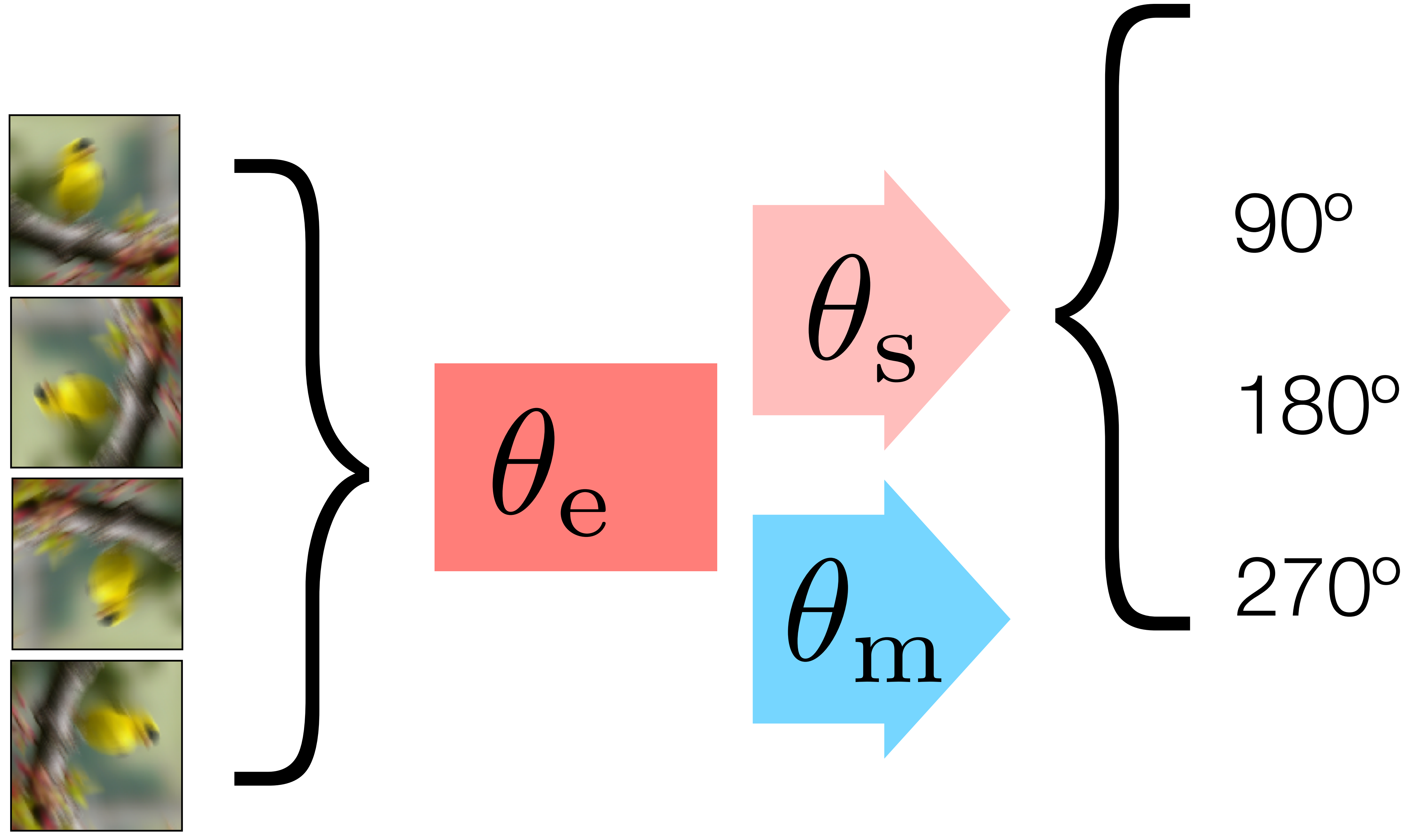
Algorithm for TTT

training

$$\min_{\theta_e, \theta_s, \theta_m} \mathbb{E}_P \left[\begin{array}{l} \ell_m(x, y; \theta_e, \theta_m) \\ + \ell_s(x, y_s; \theta_e, \theta_s) \end{array} \right]$$

testing

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



Algorithm for TTT

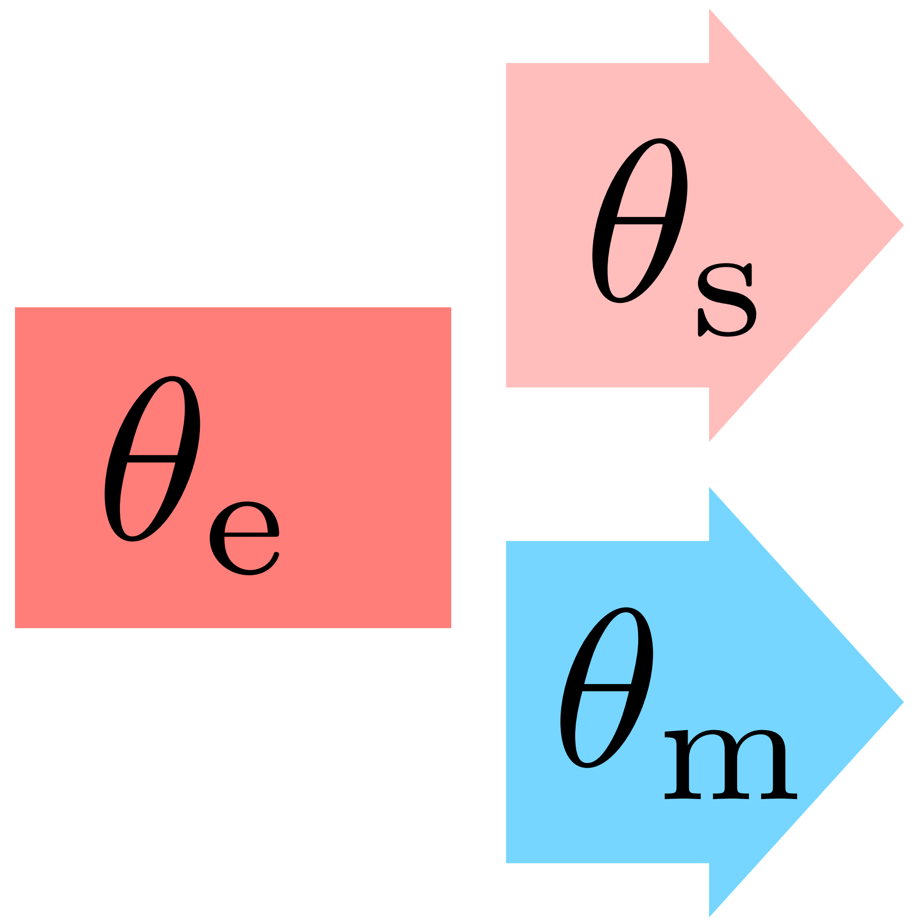
training

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

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

training

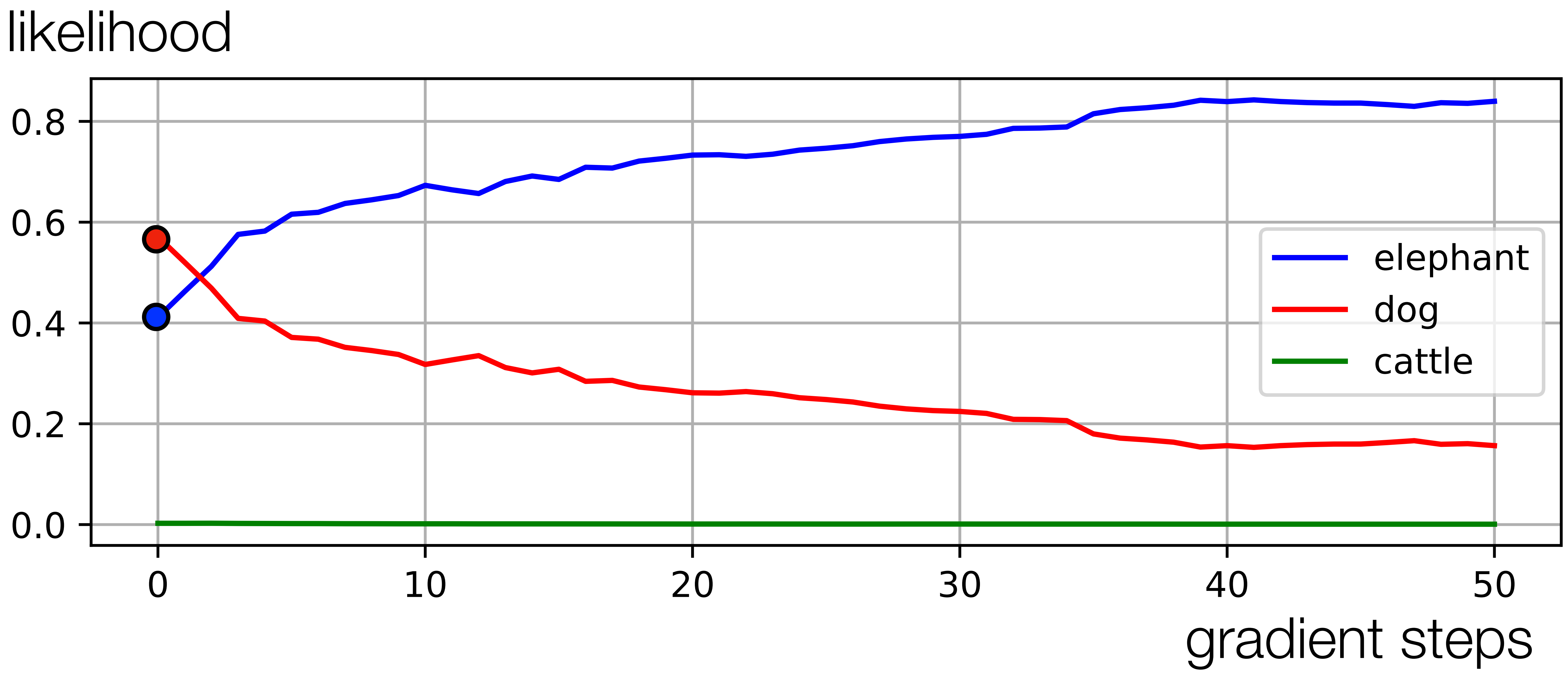
$$\min_{\theta_e, \theta_s, \theta_m} \mathbb{E}_P \left[\begin{array}{l} \ell_m(x, y; \theta_e, \theta_m) \\ + \ell_s(x, y_s; \theta_e, \theta_s) \end{array} \right]$$

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

elephant



Algorithm for TTT

training

$$\min_{\theta_e, \theta_s, \theta_m} \mathbb{E}_P \left[\begin{array}{l} \ell_m(x, y; \theta_e, \theta_m) \\ + \ell_s(x, y_s; \theta_e, \theta_s) \end{array} \right]$$

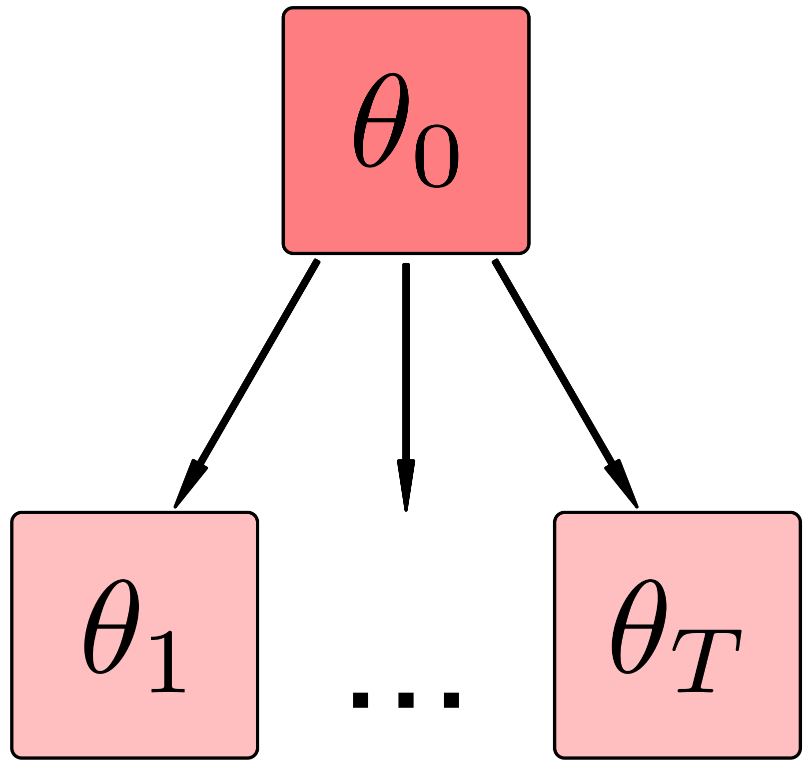
testing

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

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

multiple test samples $x_1, ..., x_T$

θ_0: parameters after joint training



Algorithm for TTT

training

$$\min_{\theta_e, \theta_s, \theta_m} \mathbb{E}_P \left[\begin{array}{l} \ell_m(x, y; \theta_e, \theta_m) \\ + \ell_s(x, y_s; \theta_e, \theta_s) \end{array} \right]$$

testing

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

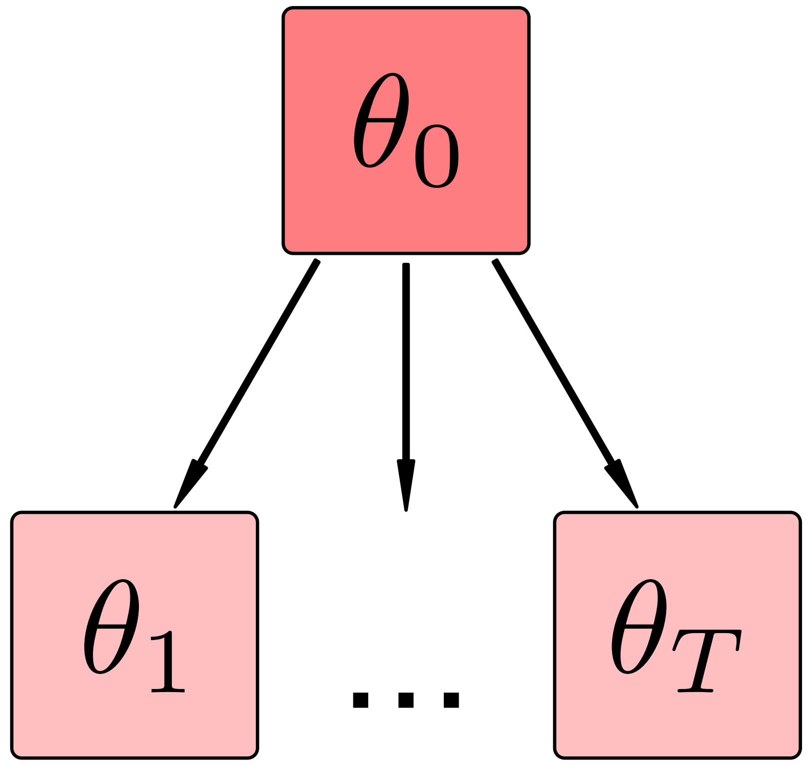
$\rightarrow \theta(x)$: make prediction on x

multiple test samples $x_1, ..., x_T$

θ_0: parameters after joint training

standard version

no assumption on the test samples



Algorithm for TTT

training

$$\min_{\theta_e,\theta_s,\theta_m} \mathbb{E}_P \left[\begin{array}{l} \ell_m(x, y; \theta_e, \theta_m) \\ + \ell_s(x, y_s; \theta_e, \theta_s) \end{array} \right]$$

testing

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

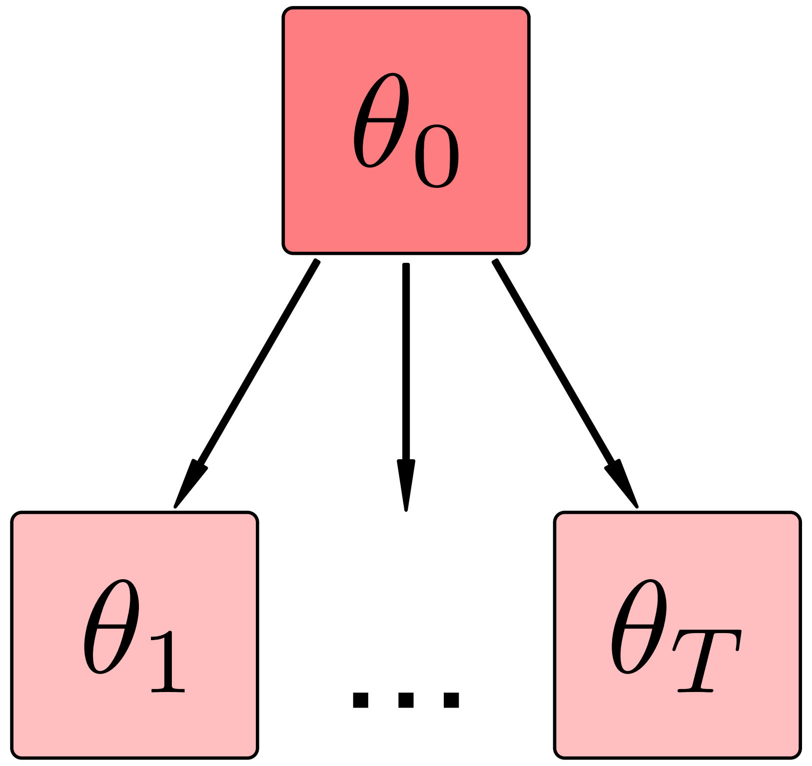
$\rightarrow \theta(x)$: make prediction on x

multiple test samples $x_1, ..., x_T$

θ_0: parameters after joint training

standard version

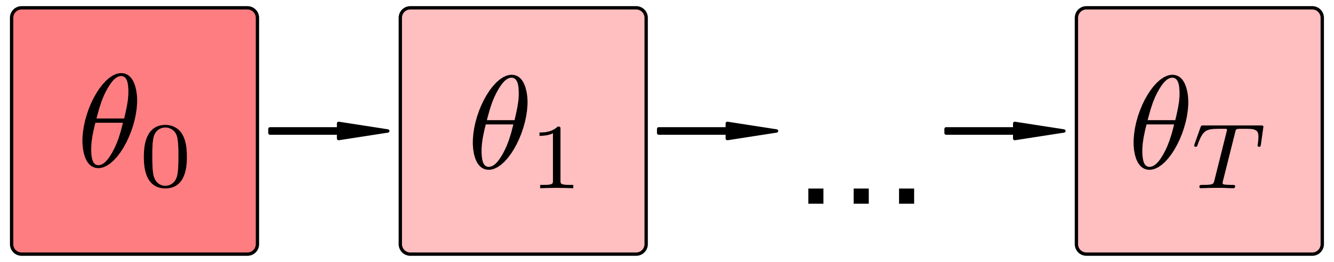
no assumption on the test samples



online version

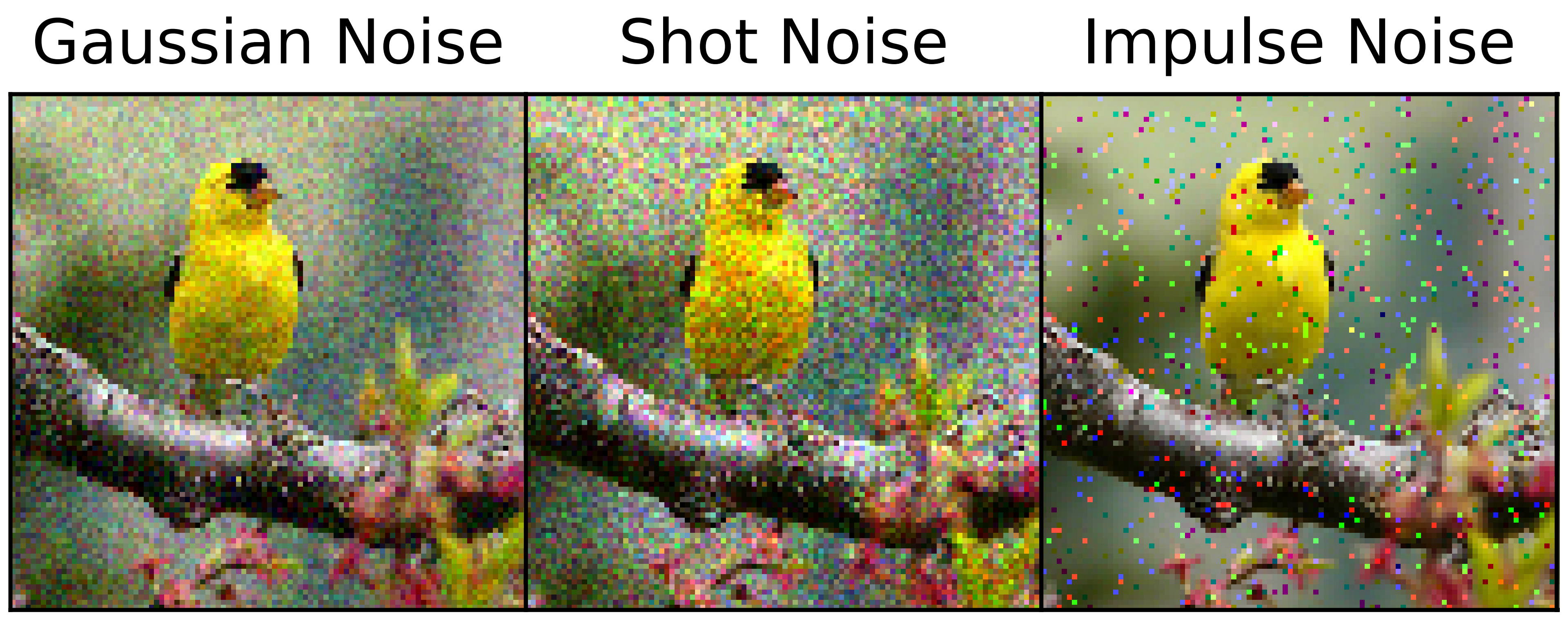
$x_1, ..., x_T$ come from the same Q

or smoothly changing $Q_1, ..., Q_T$



Object recognition with corruptions

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



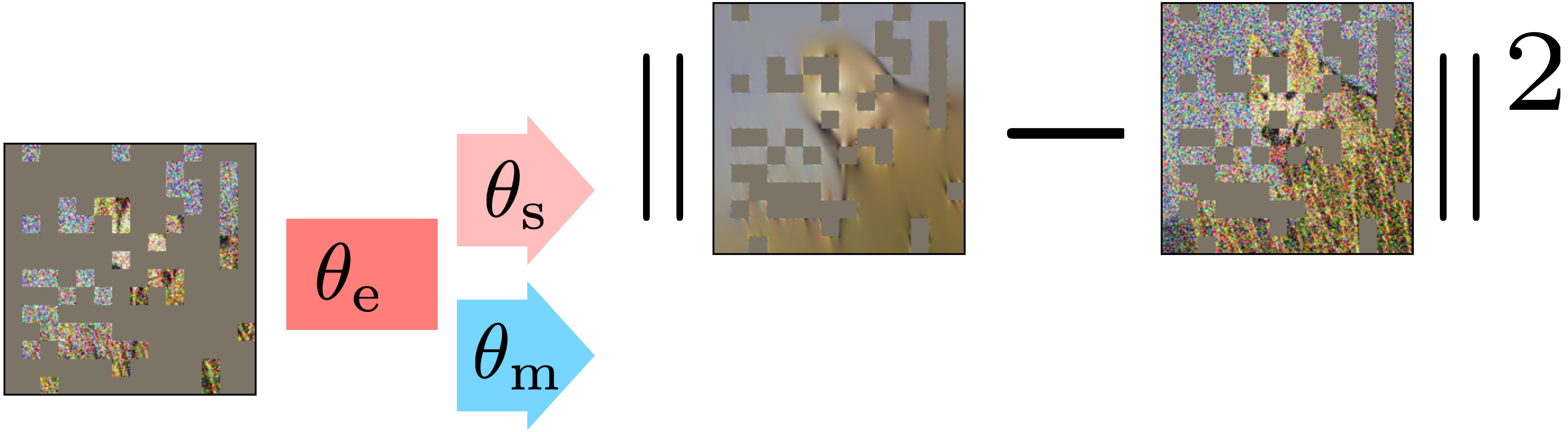
Benchmarking Neural Network Robustness to Common Corruptions and Perturbations
Hendrycks and Dietterich, 2018

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

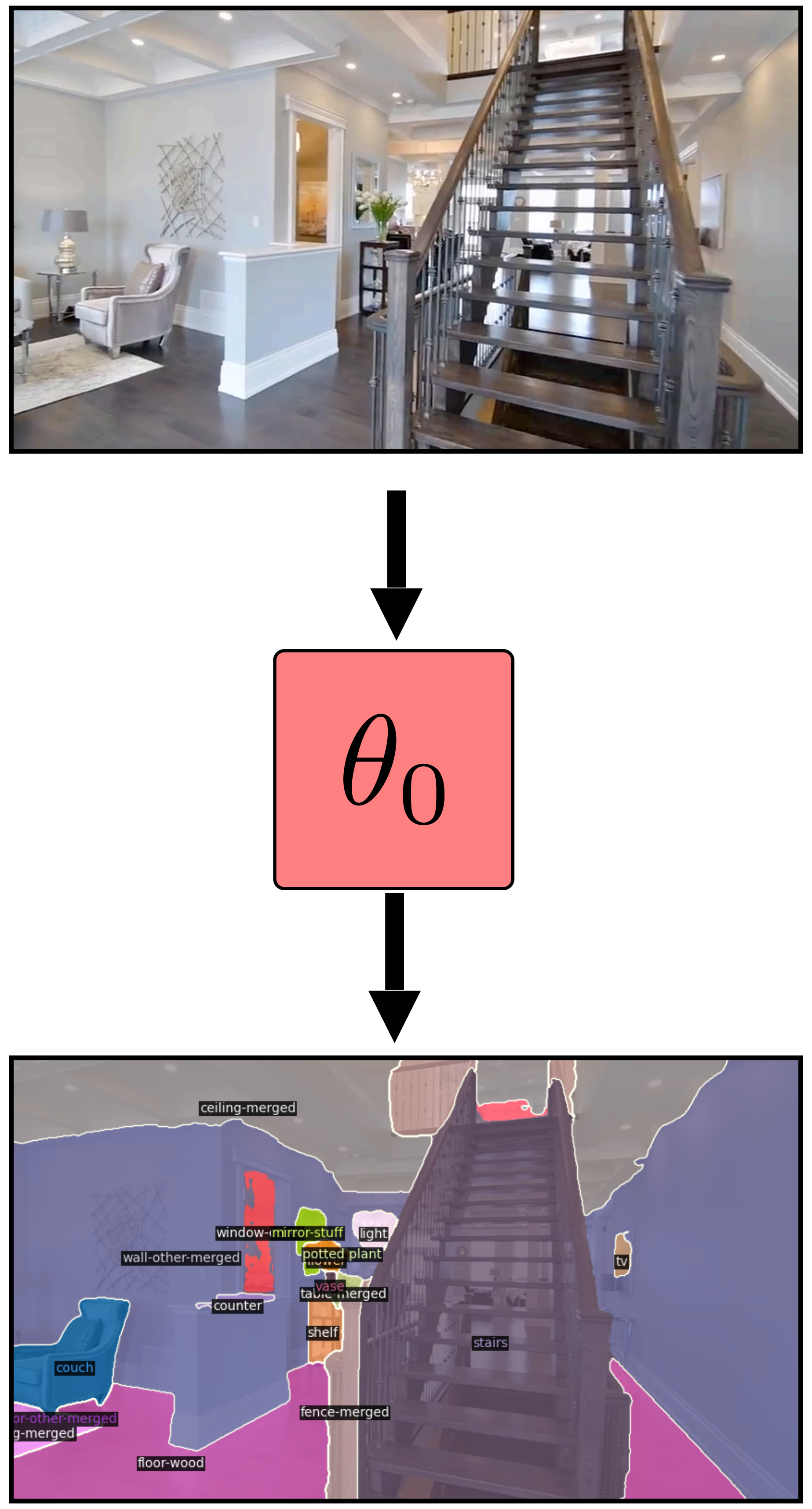
TTT-MAE on ImageNet-C

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

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

*: Equal contribution

TTT-MAE

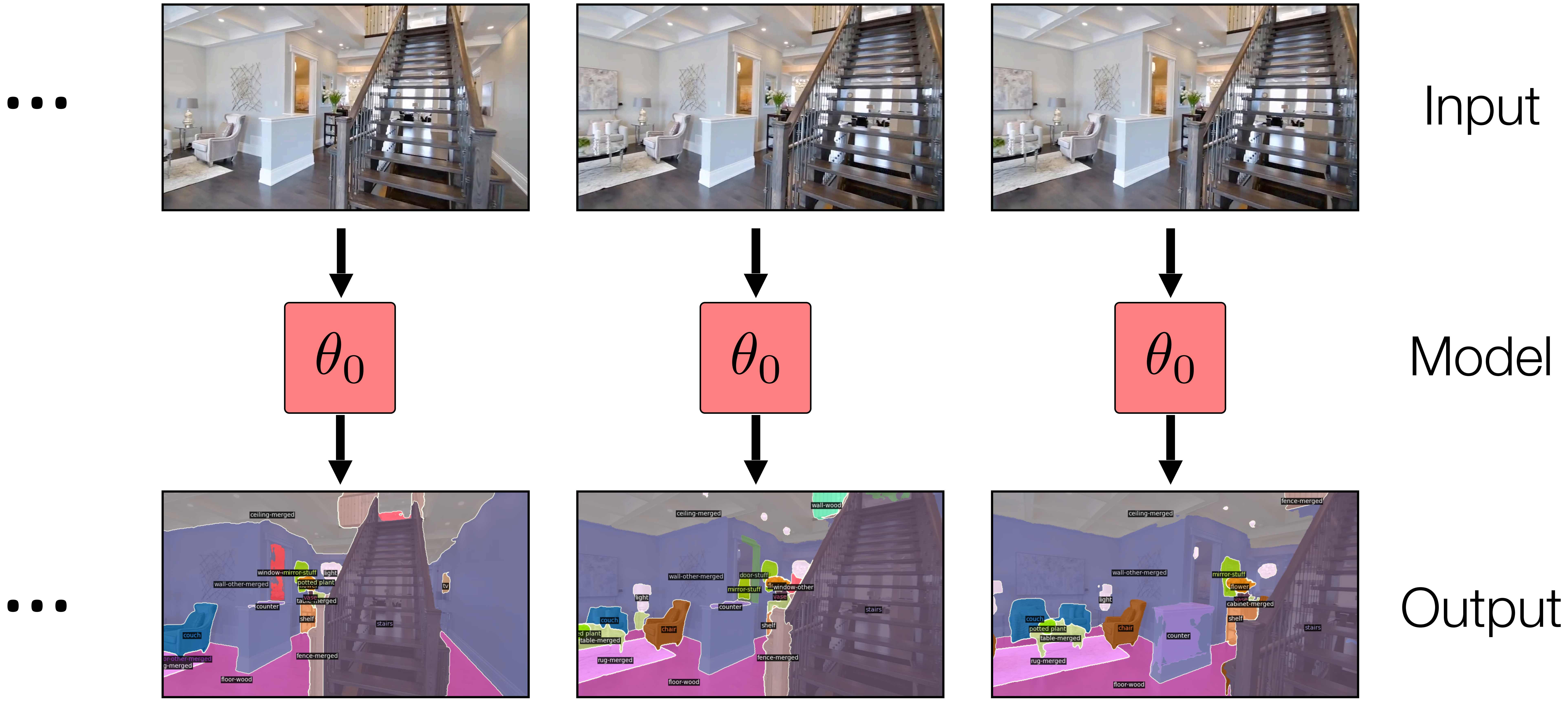


Input

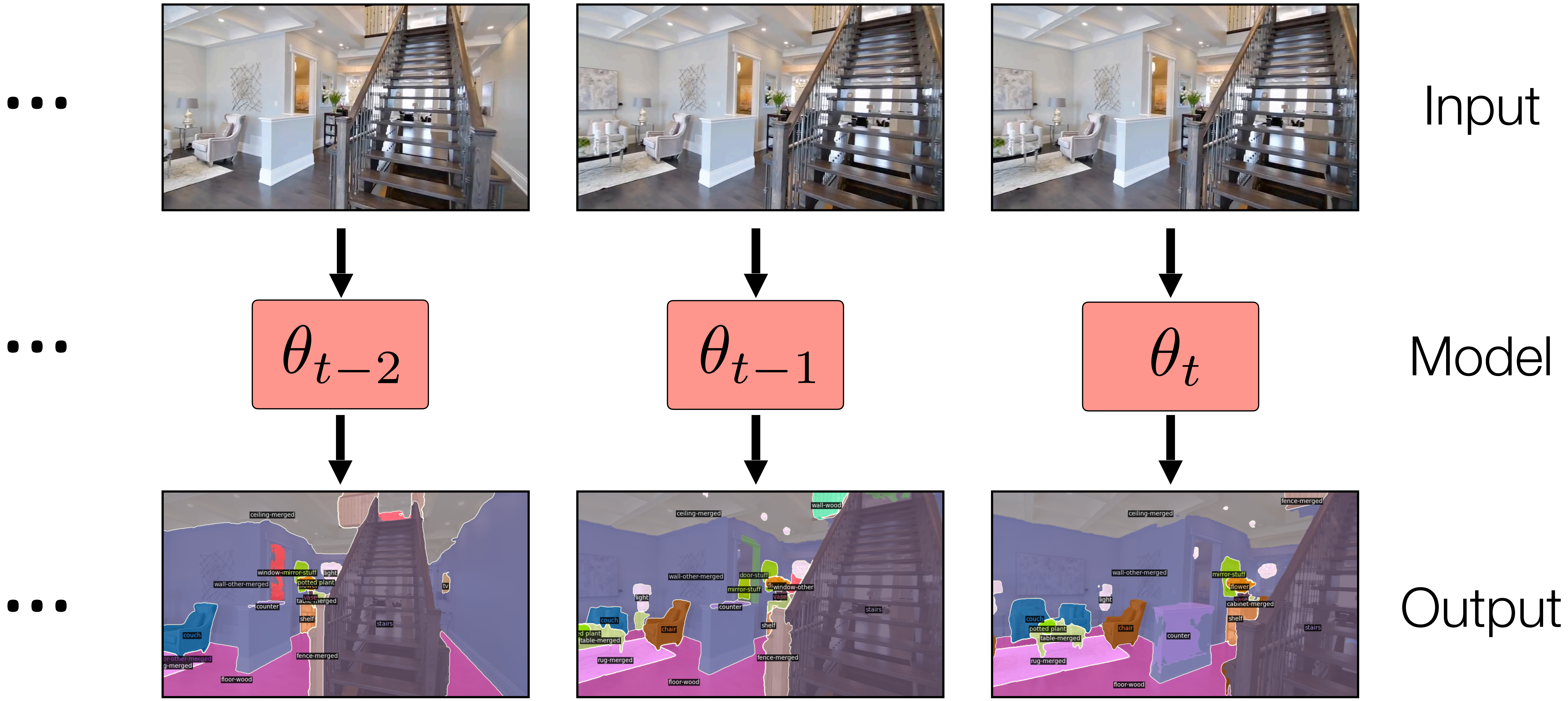
Model

Output

TTT-MAE

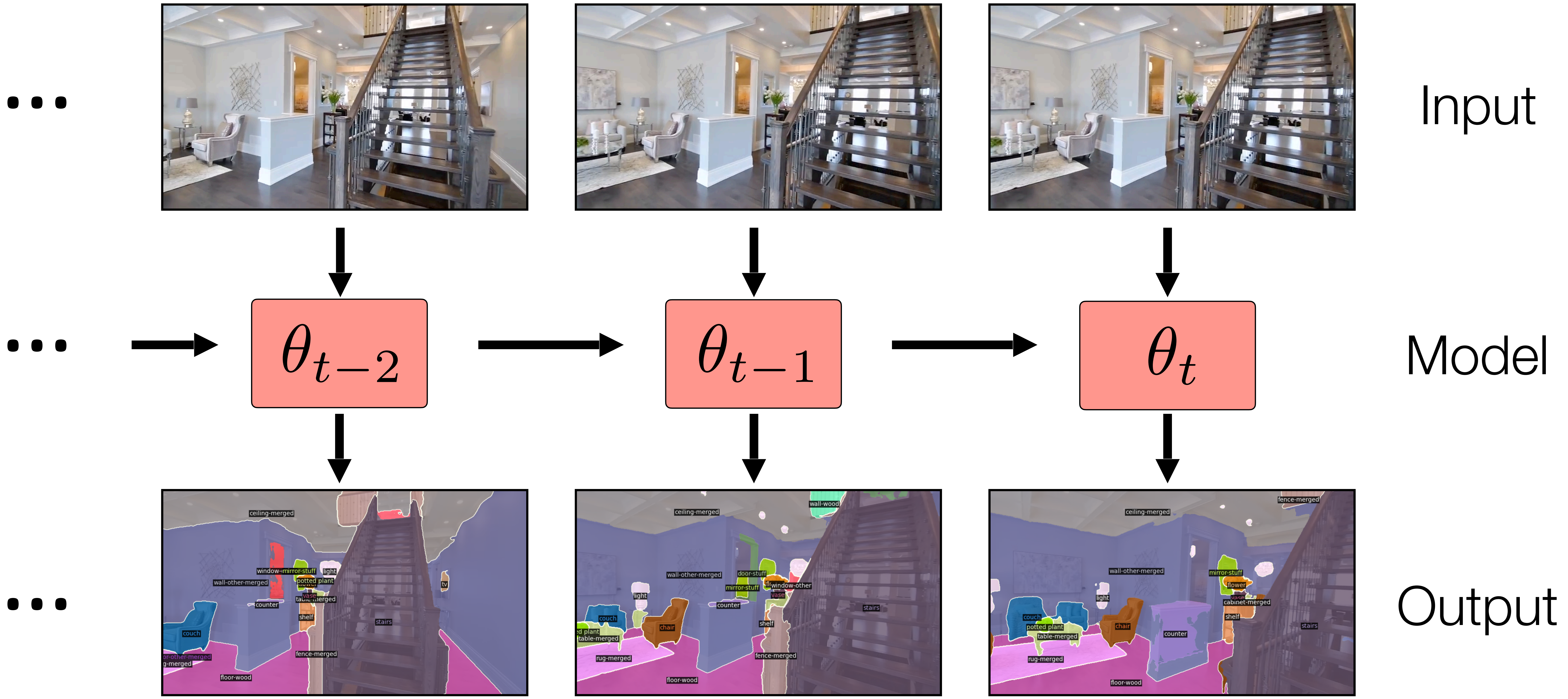


Test-Time Training on Video Streams



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

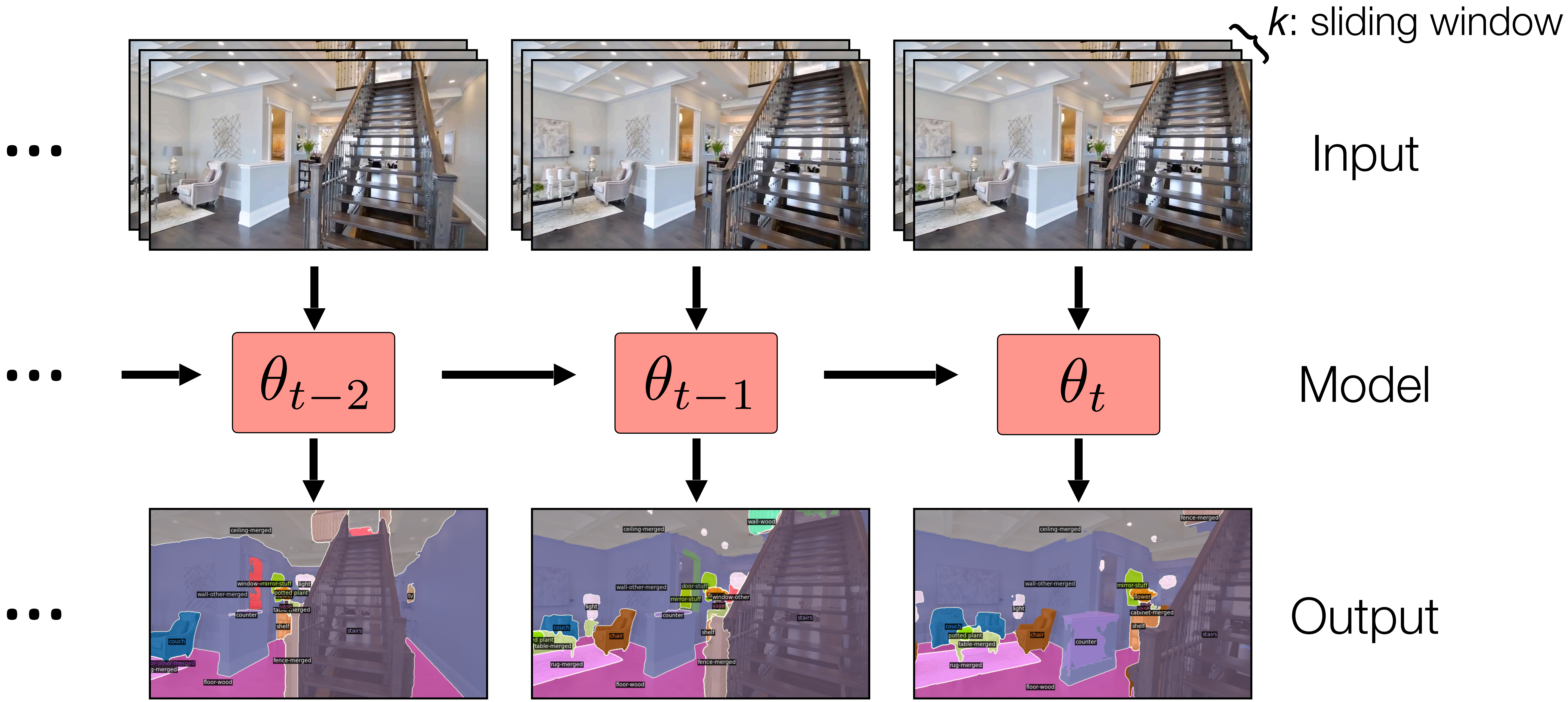


Test-Time Training on Video Streams

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

Test-Time Training on Video Streams

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

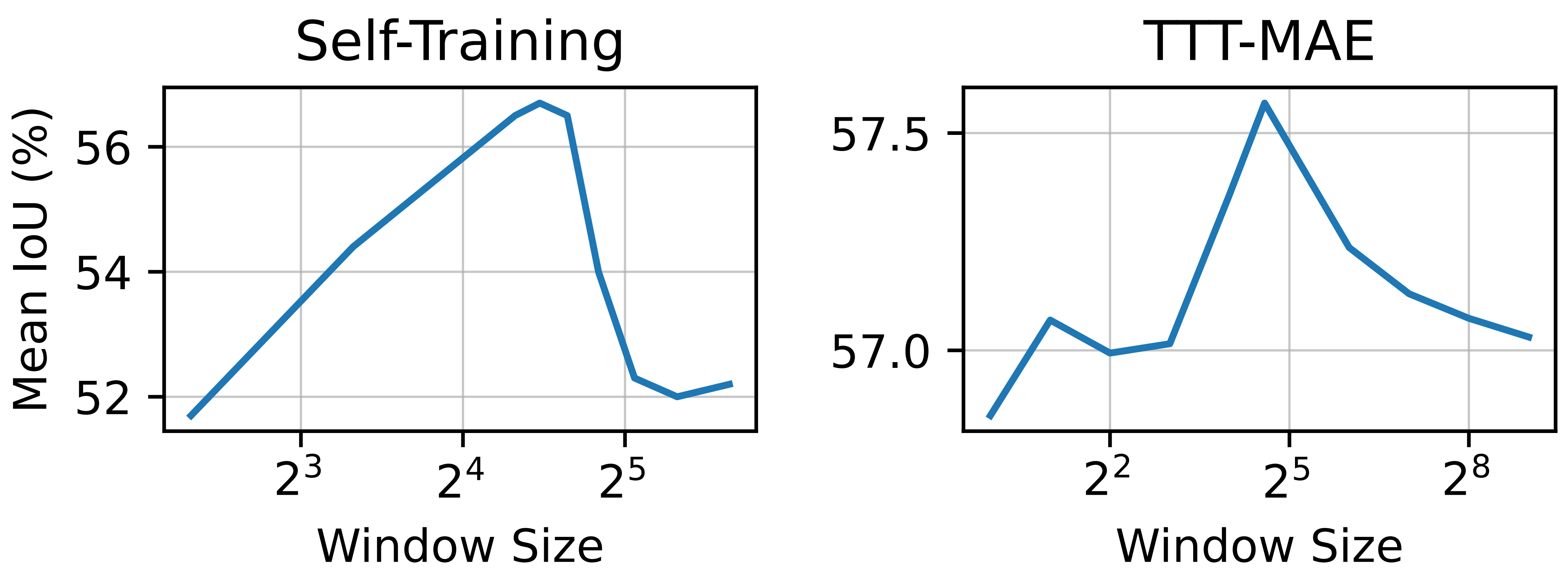


Test-Time Training on Video Streams

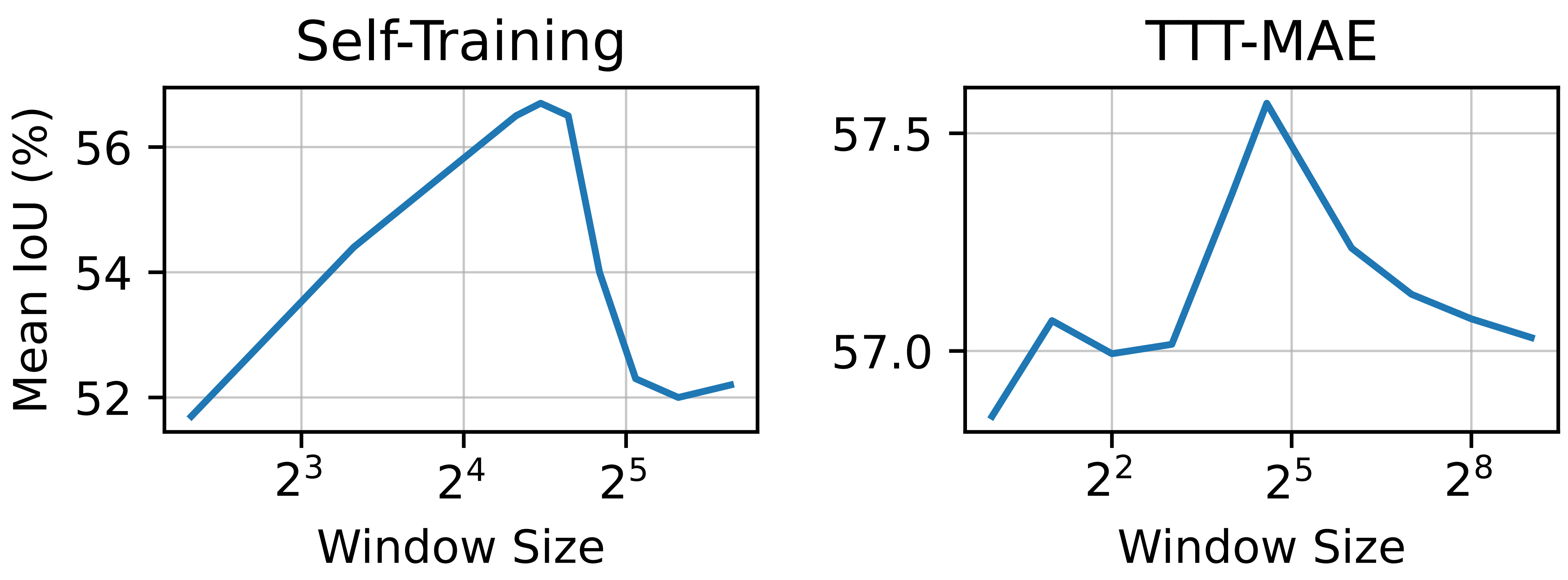
Renhao Wang*, Yu Sun*, Yossi Gandelsman, Xinlei Chen, Alexei A. Efros, **Xiaolong Wang**

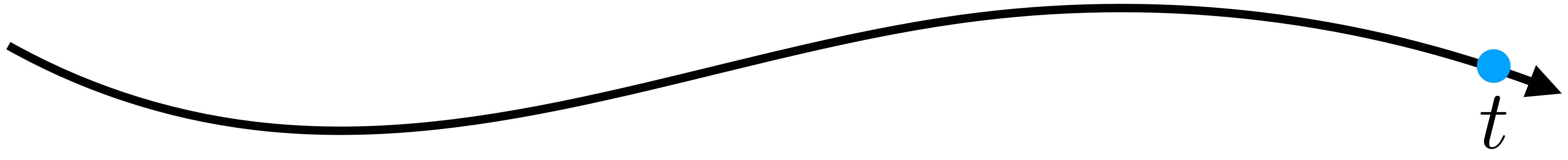
*: Equal contribution

Forgetting can be beneficial

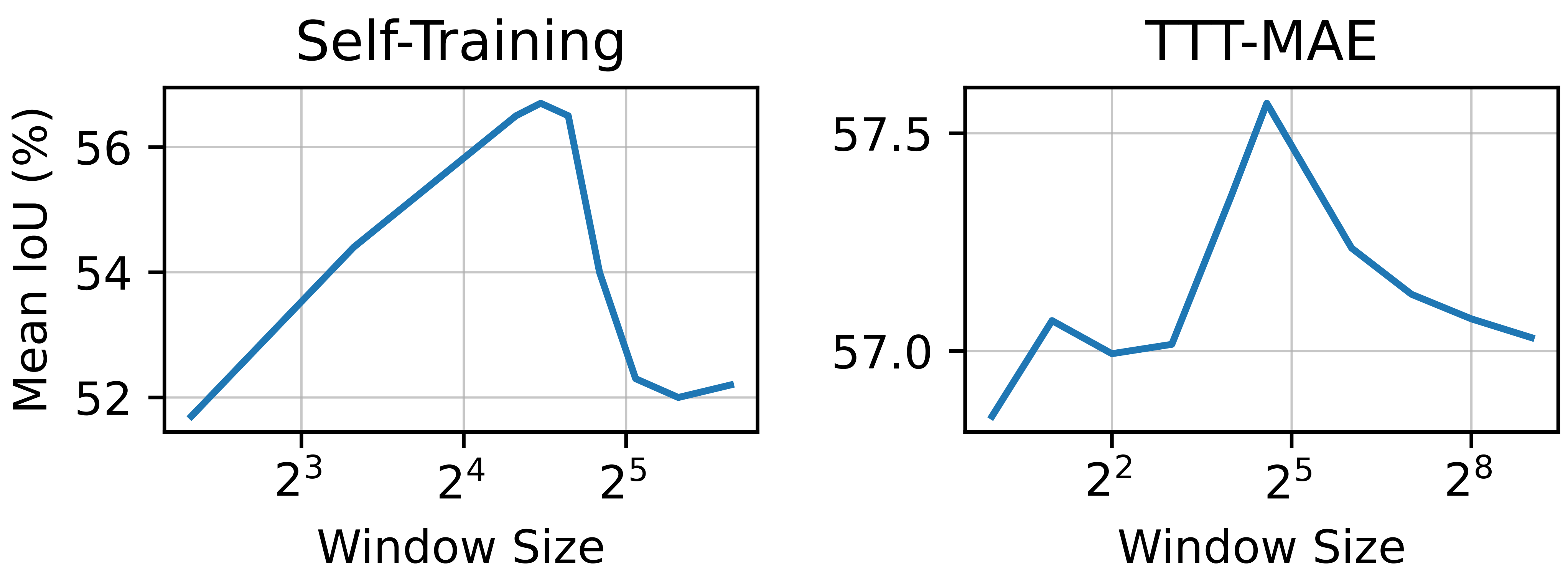


Forgetting can be beneficial

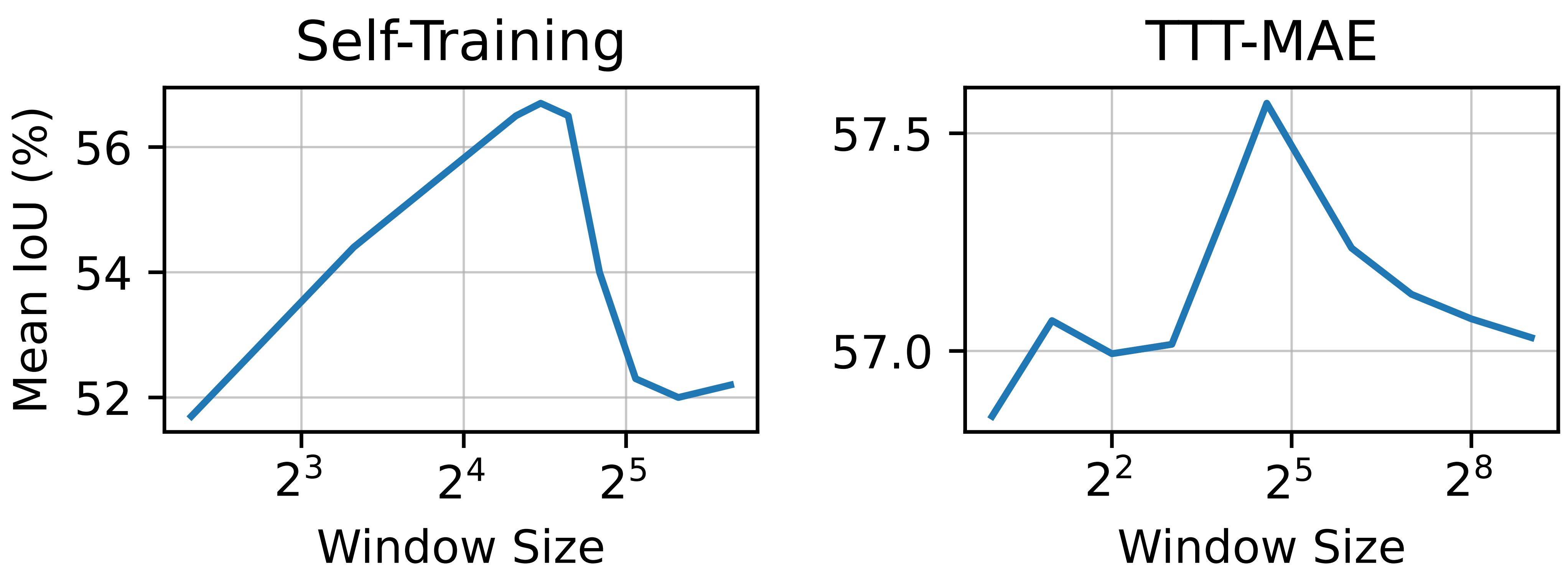




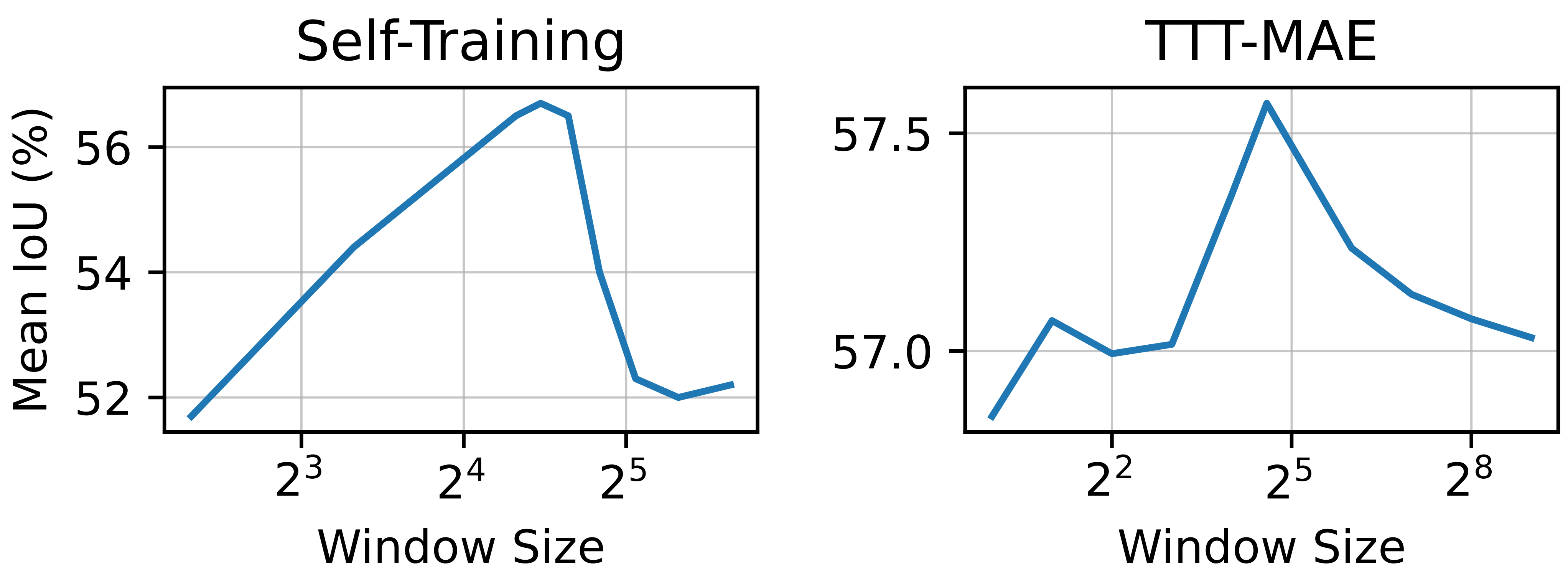
Forgetting can be beneficial



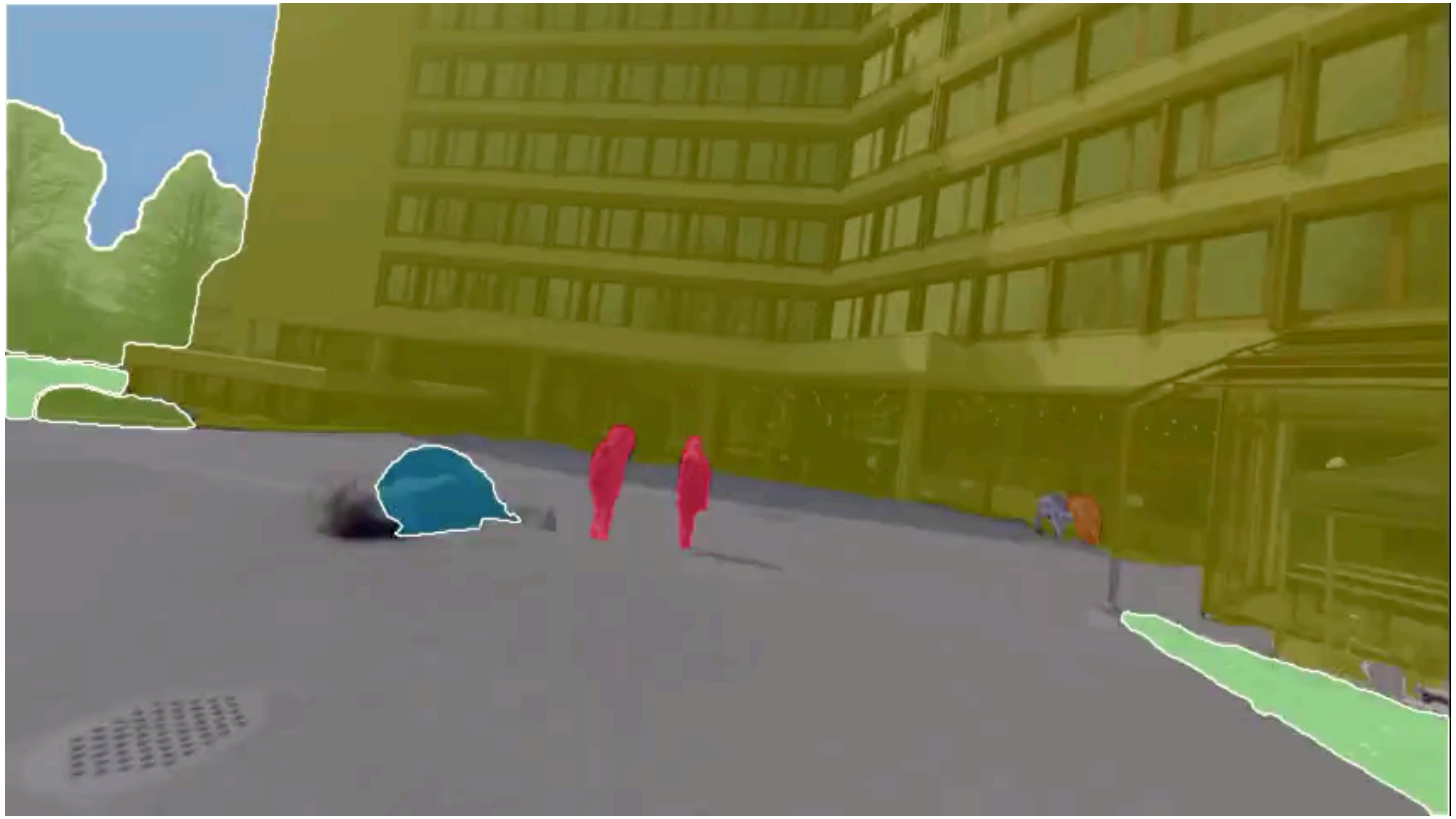
Forgetting can be beneficial



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