北京大学 PEKING UNIVERSITY

Rotating Features for Object Discovery

Sindy Löwe, Phillip Lippe, Francesco Locatello, Max Welling

NeurIPS 2023 Oral

PRESENTER: MINGHAO LIU

2023/12/3

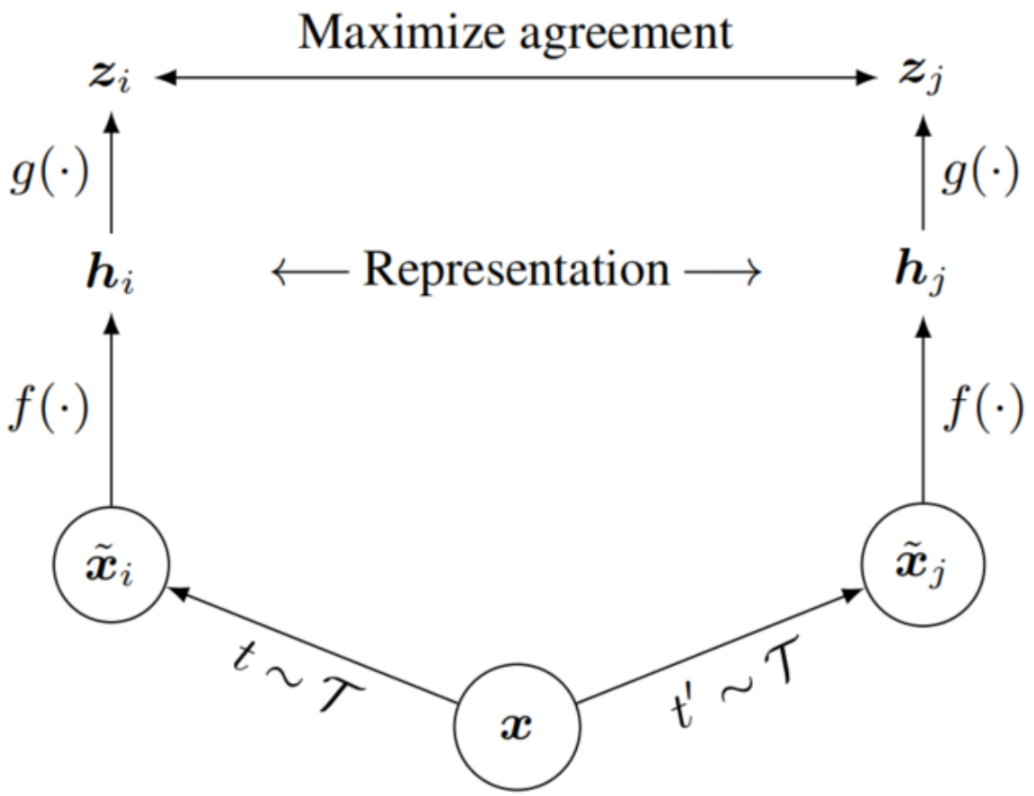
Outline

Background

A Simple Framework for Contrastive Learning of Visual Representations

SimCLR (ICML 2020)

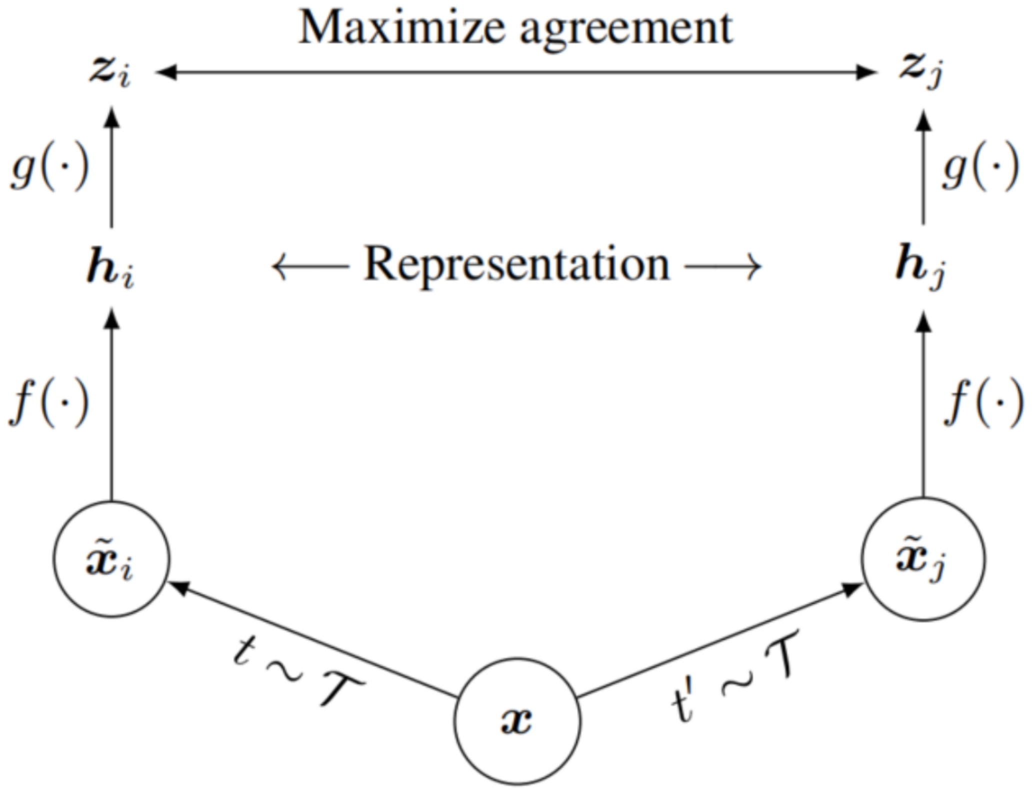
Ting Chen [1] Simon Kornblith [1] Mohammad Norouzi [1] Geoffrey Hinton [1]



Background

- **SimCLR**

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \quad (1)$$



Background

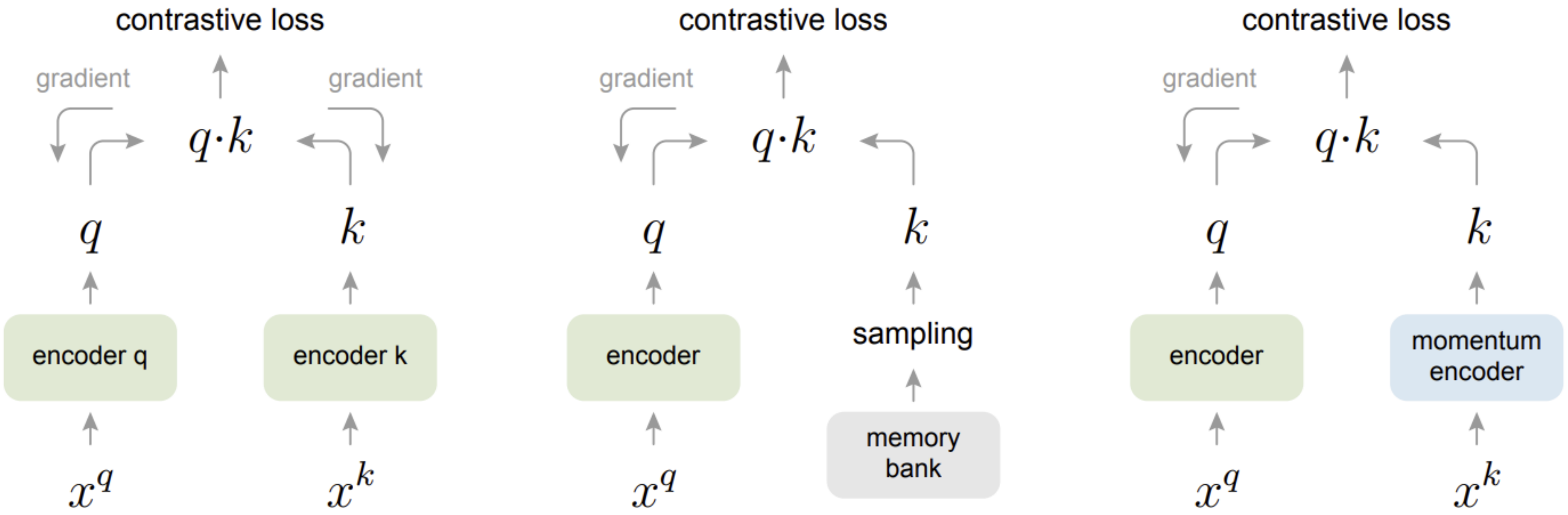
Momentum Contrast for Unsupervised Visual Representation Learning

Kaiming He Haoqi Fan Yuxin Wu Saining Xie Ross Girshick

Facebook AI Research (FAIR)

■ MoCo (CVPR 2020)

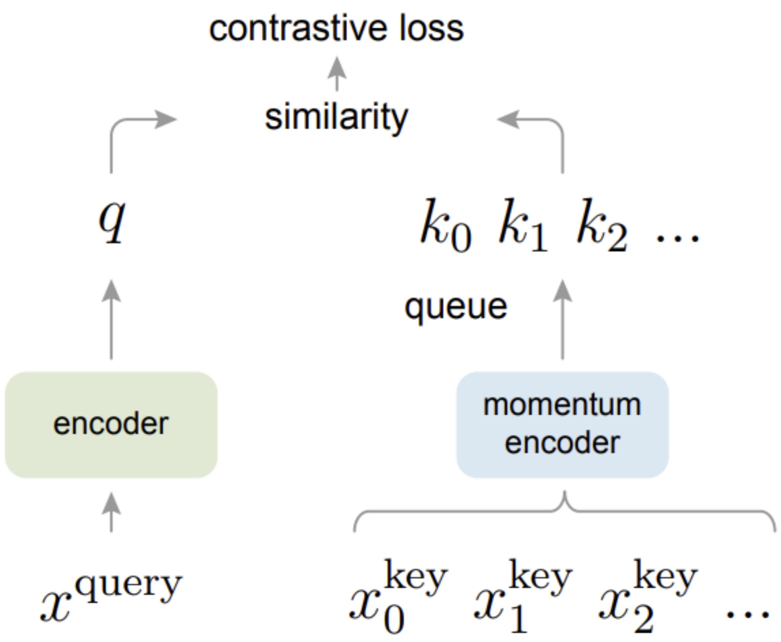
$$L_q = -log \frac{exp\left(q \cdot k_+ / \tau\right)}{\sum_{i=0}^{K} exp\left(q \cdot k_i / \tau\right)}$$



(a) end-to-end (b) memory bank (c) MoCo

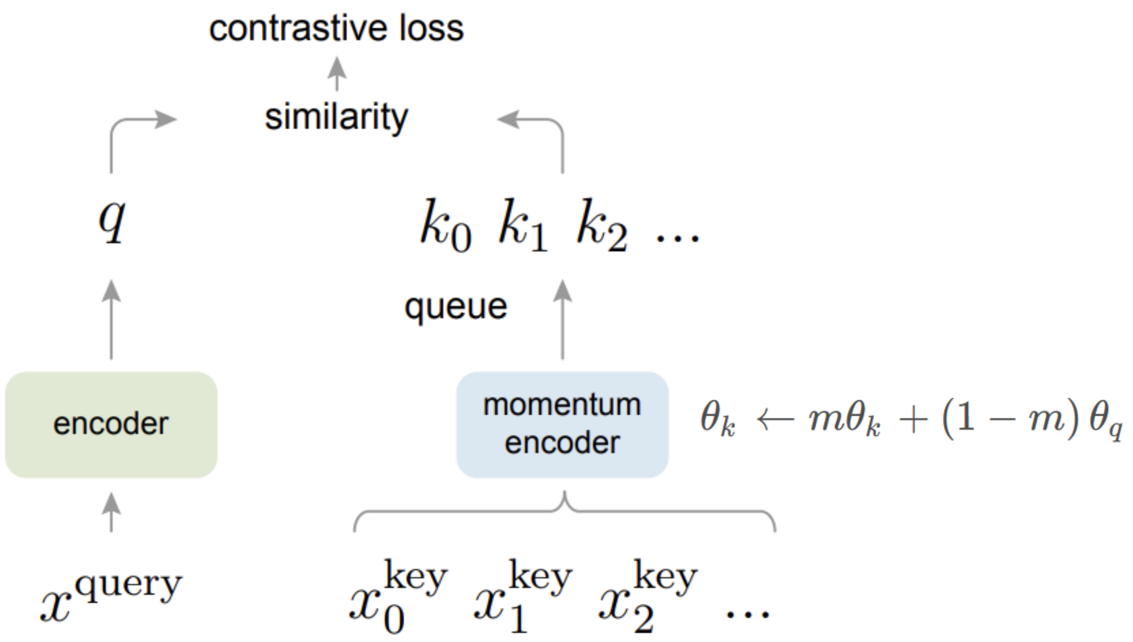
Background

- **MoCo**



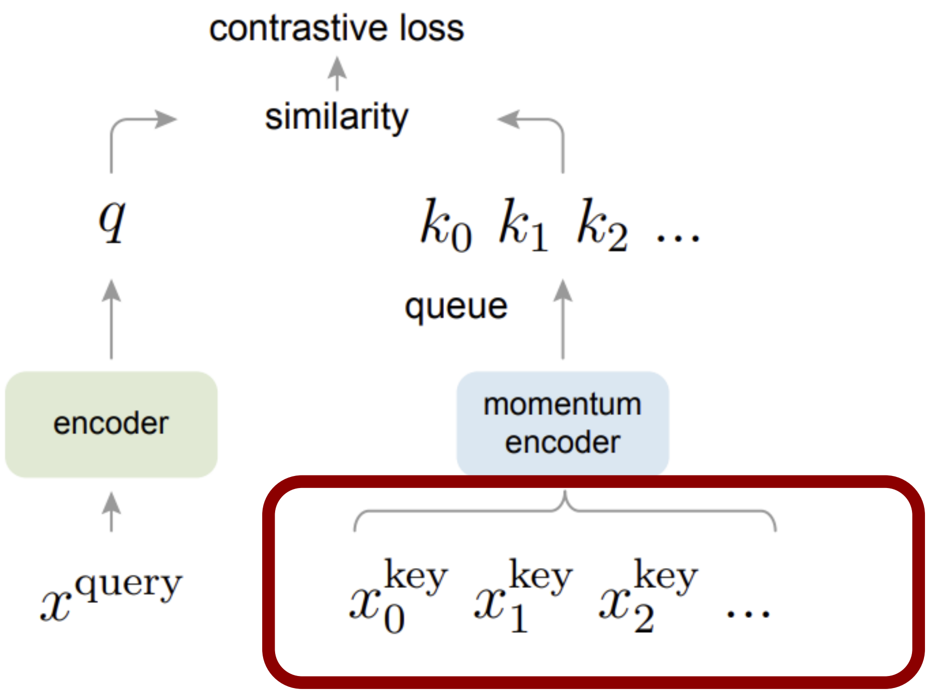
Background

- **MoCo**



Background

- MoCo



Background

- **MoCo v3**

Algorithm 1 MoCo v3: PyTorch-like Pseudocode

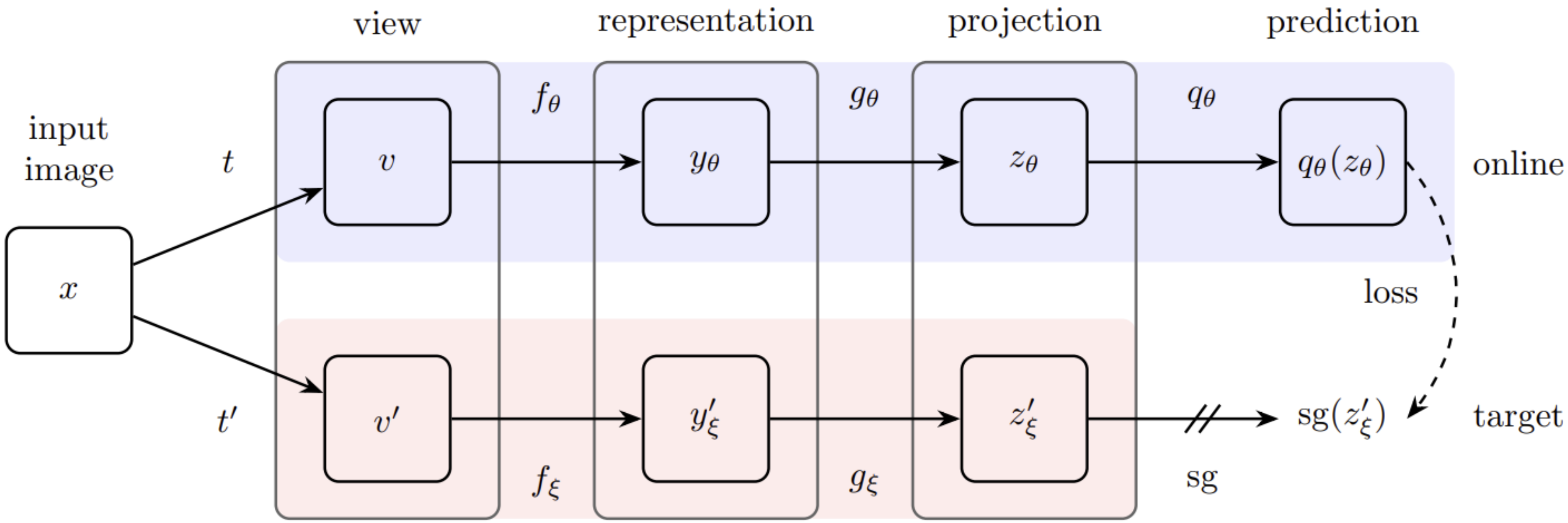
```
# f_q: encoder: backbone + proj mlp + pred mlp
# f_k: momentum encoder: backbone + proj mlp
# m: momentum coefficient
# tau: temperature

for x in loader: # load a minibatch x with N samples
    x1, x2 = aug(x), aug(x) # augmentation
    q1, q2 = f_q(x1), f_q(x2) # queries: [N, C] each
    k1, k2 = f_k(x1), f_k(x2) # keys: [N, C] each

    loss = ctr(q1, k2) + ctr(q2, k1) # symmetrized
    loss.backward()

    update(f_q) # optimizer update: f_q
    f_k = m*f_k + (1-m)*f_q # momentum update: f_k

# contrastive loss
def ctr(q, k):
    logits = mm(q, k.t()) # [N, N] pairs
    labels = range(N) # positives are in diagonal
    loss = CrossEntropyLoss(logits/tau, labels)
    return 2 * tau * loss
```

Background

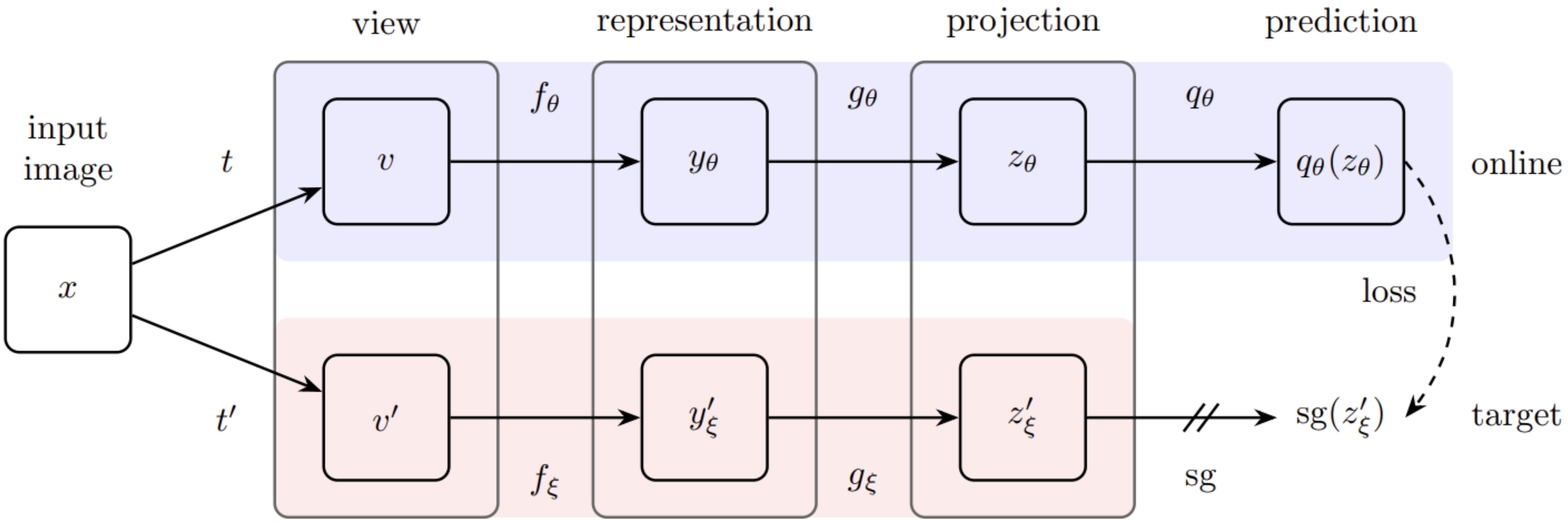
- BYOL (NeurIPS 2020)

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



BYOL


view
representation
projection
prediction
input
image
x
t
t'
v
v'
f_θ
f_ξ
y_θ
y'_ξ
g_θ
g_ξ
z_θ
z'_ξ
q_θ
$q_\theta(z_\theta)$
sg
$\mathrm{sg}(z'_\xi)$
online
loss
target

BYOL

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

Inputs :

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

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

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

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

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

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

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

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

  end

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

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

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

end

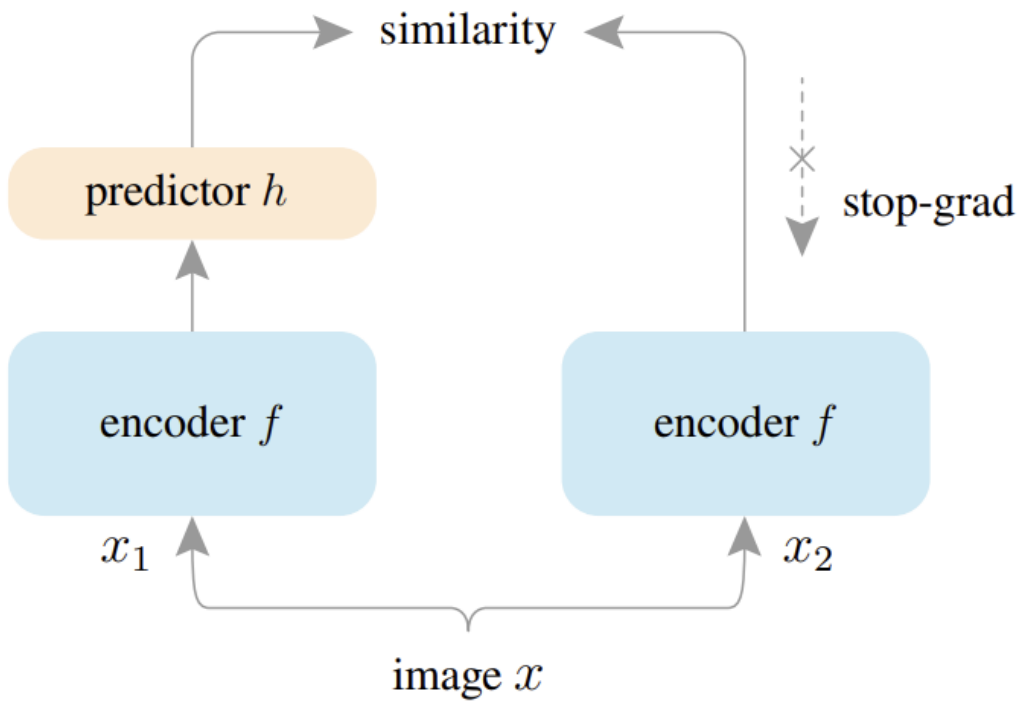
Output : encoder f_θ

Background

■ SimSiam

Exploring Simple Siamese Representation Learning

Xinlei Chen   Kaiming He
Facebook AI Research (FAIR)

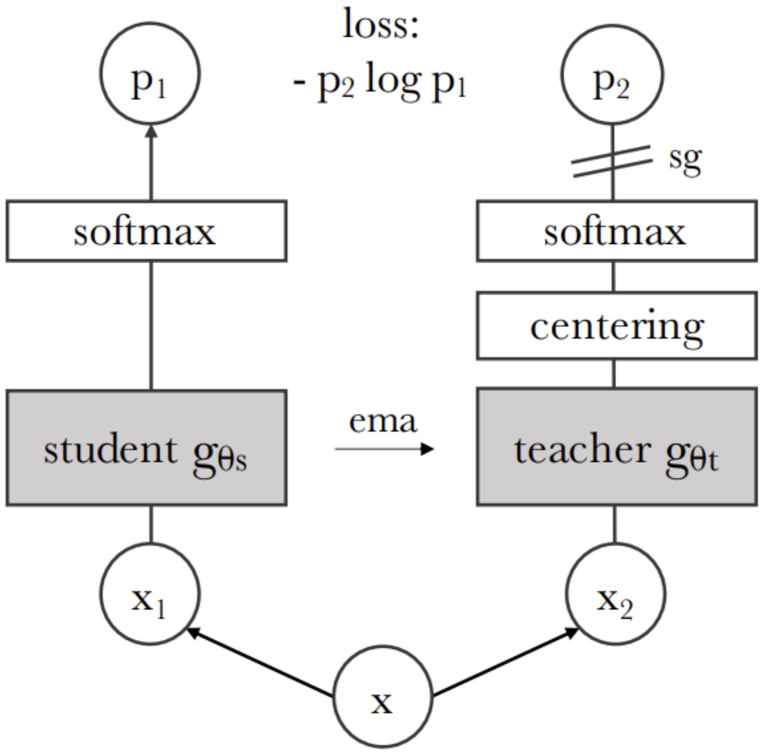


Algorithm 1 SimSiam Pseudocode, PyTorch-like

```
# 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()
```

Background

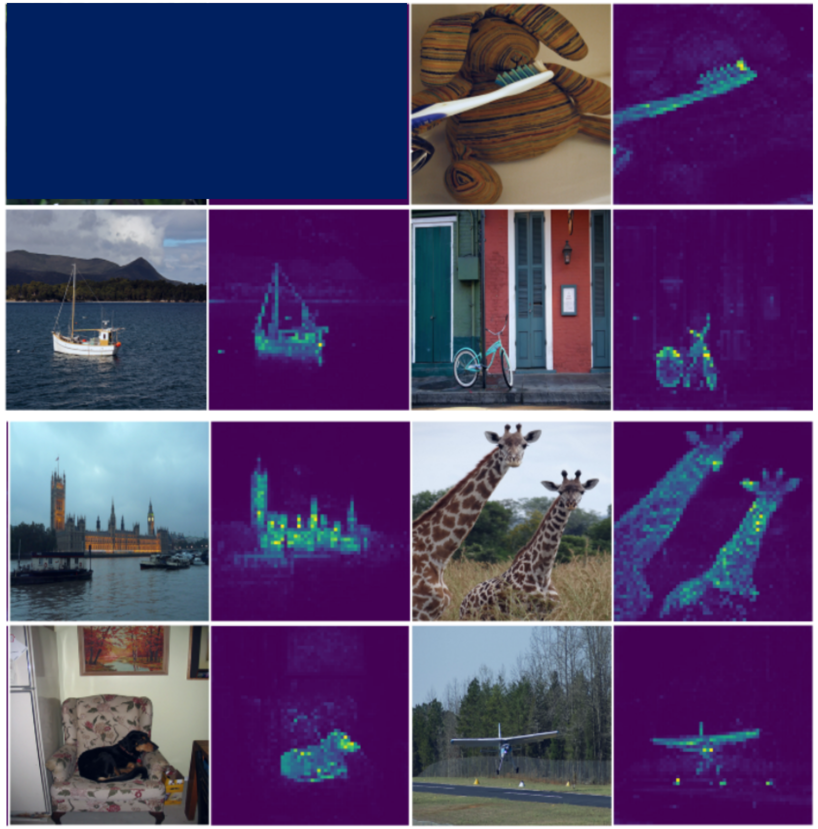
- **DINO(CVPR 2021)**



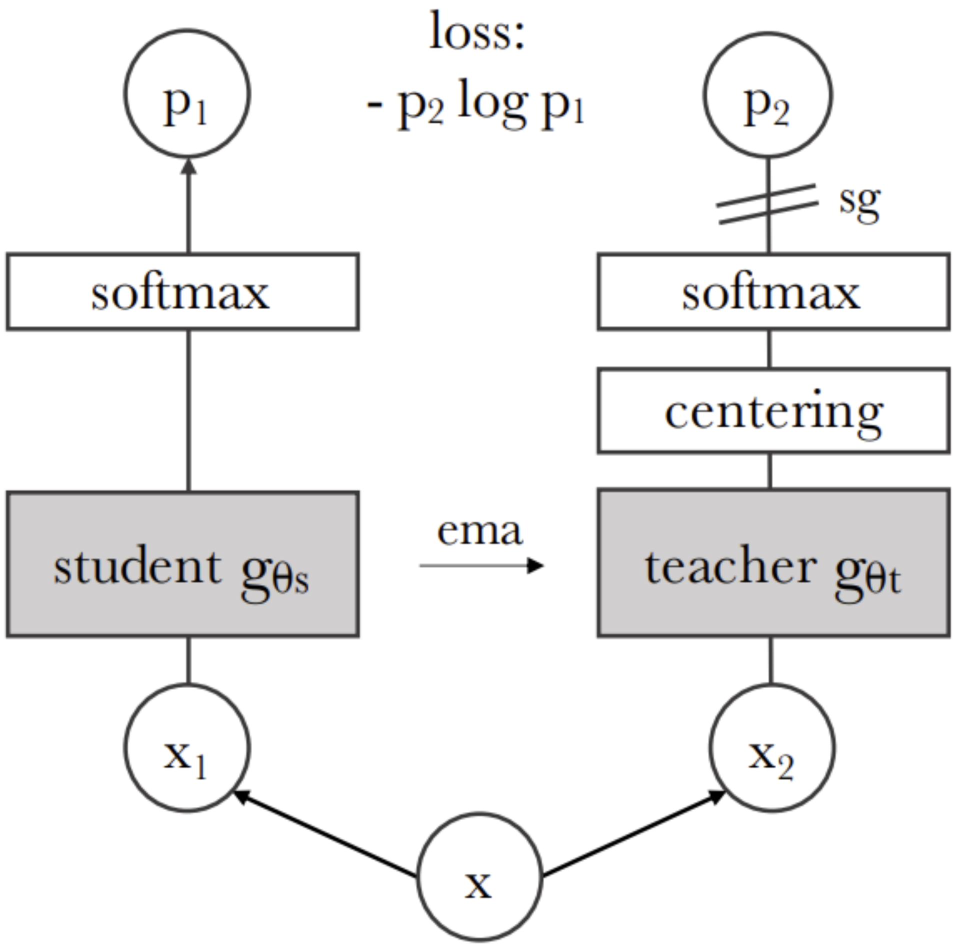
Emerging Properties in Self-Supervised Vision Transformers

Mathilde Caron[1,2] Hugo Touvron[1,3] Ishan Misra[1] Hervé Jegou[1]
Julien Mairal[2] Piotr Bojanowski[1] Armand Joulin[1]

[1] Facebook AI Research [2] Inria* [3] Sorbonne University



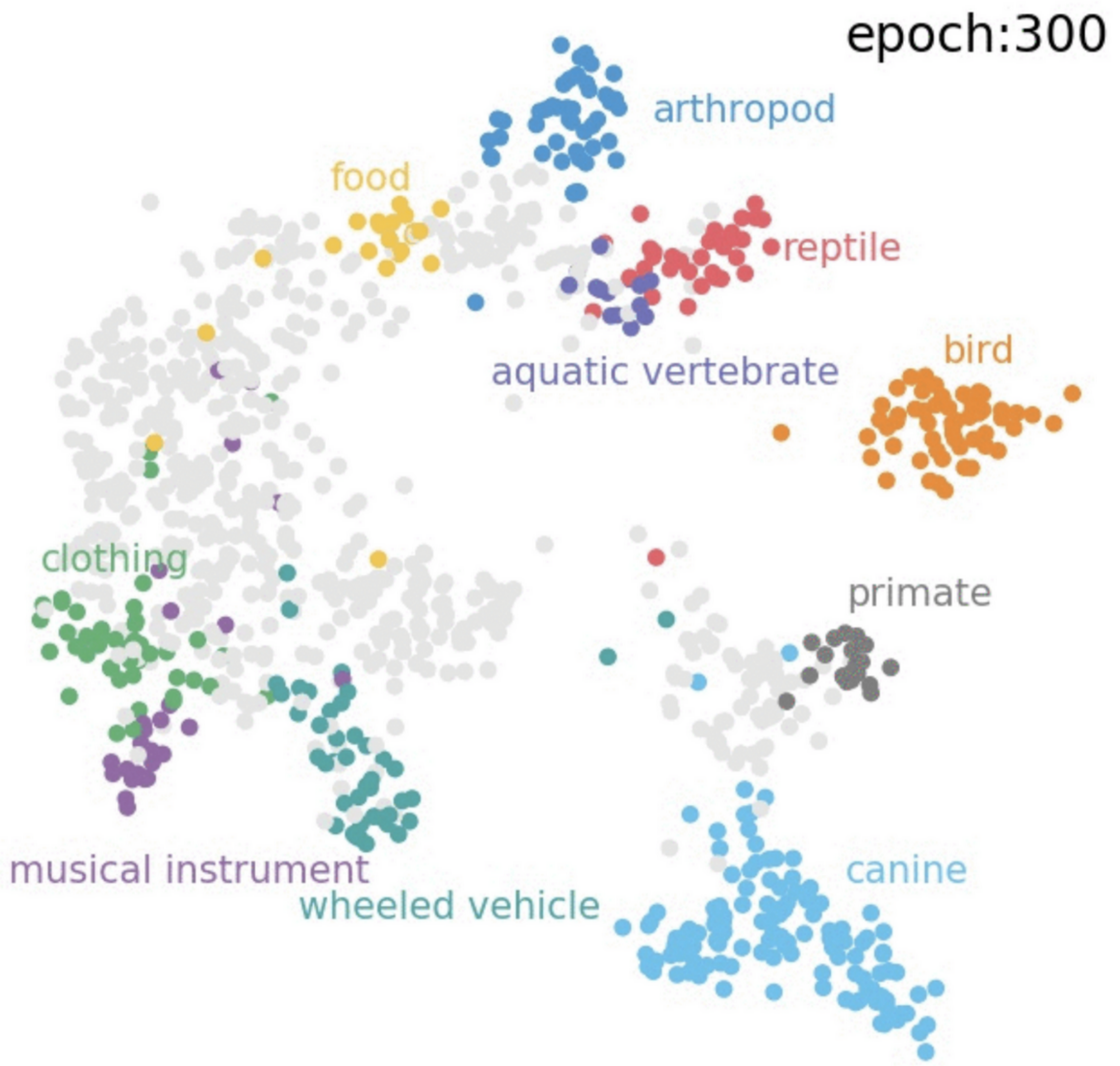
DINO



Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

DINO


epoch:300
arthropod
food
reptile
aquatic vertebrate
bird
clothing
primate
musical instrument
wheeled vehicle
canine

Background

- Object Discovery

Background

Unsupervised Conditional Slot Attention for Object Centric Learning

Avinash Kori [†] Francesco Locatello [*] Francesca Toni [†] Ben Glocker [†]

[†] Department of Computing, Imperial College London
a.kori21@imperial.ac.uk

- Object-centric representations
- Decompose scenes in terms of abstract building blocks

Background

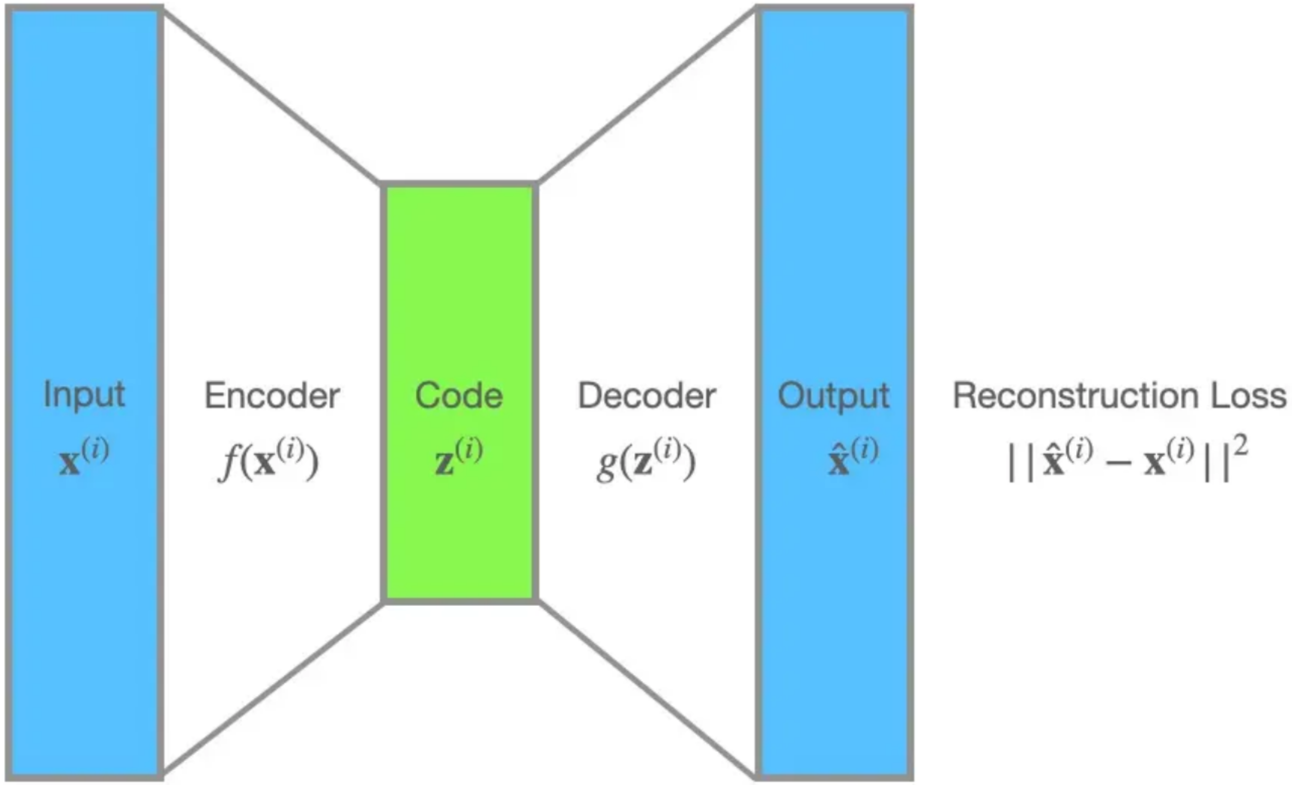
- **MONet**
- **Slot-Based Methods**
- **Multi Objects**
- **Components VAE**
- **Recurrent attention network**

MONet: Unsupervised Scene Decomposition and Representation

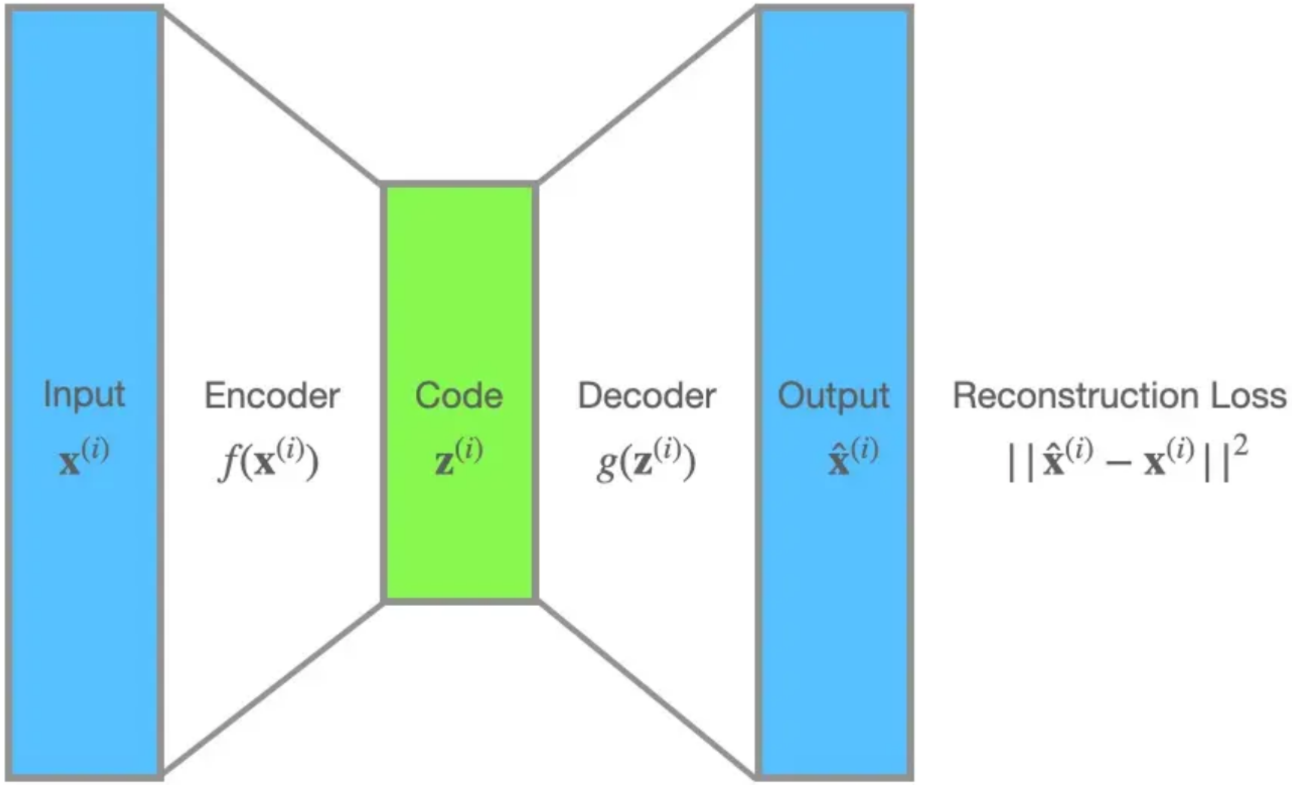
Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, Alexander Lerchner

DeepMind
London, United Kingdom

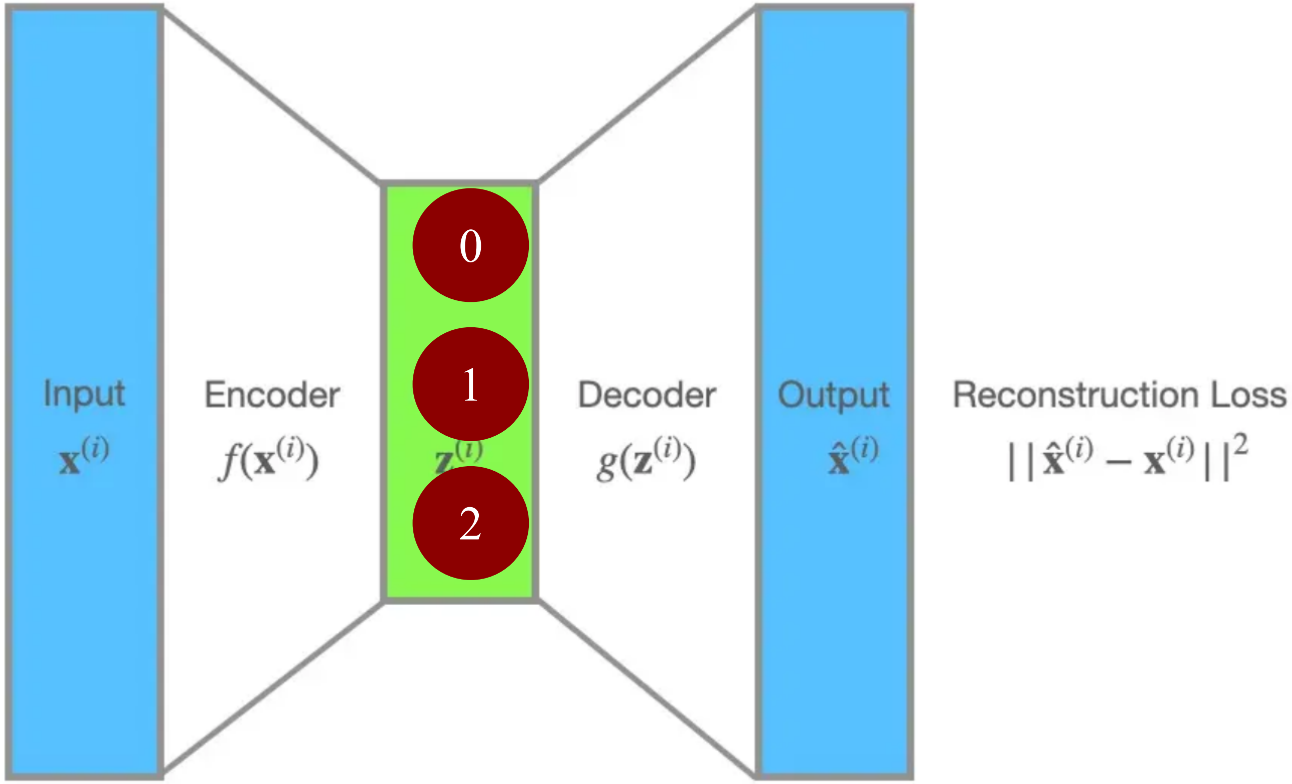
VAE


Input
$\mathbf{x}^{(i)}$
Encoder
$f(\mathbf{x}^{(i)})$
Code
$\mathbf{z}^{(i)}$
Decoder
$g(\mathbf{z}^{(i)})$
Output
$\hat{\mathbf{x}}^{(i)}$
Reconstruction Loss
$||\hat{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}||^2$

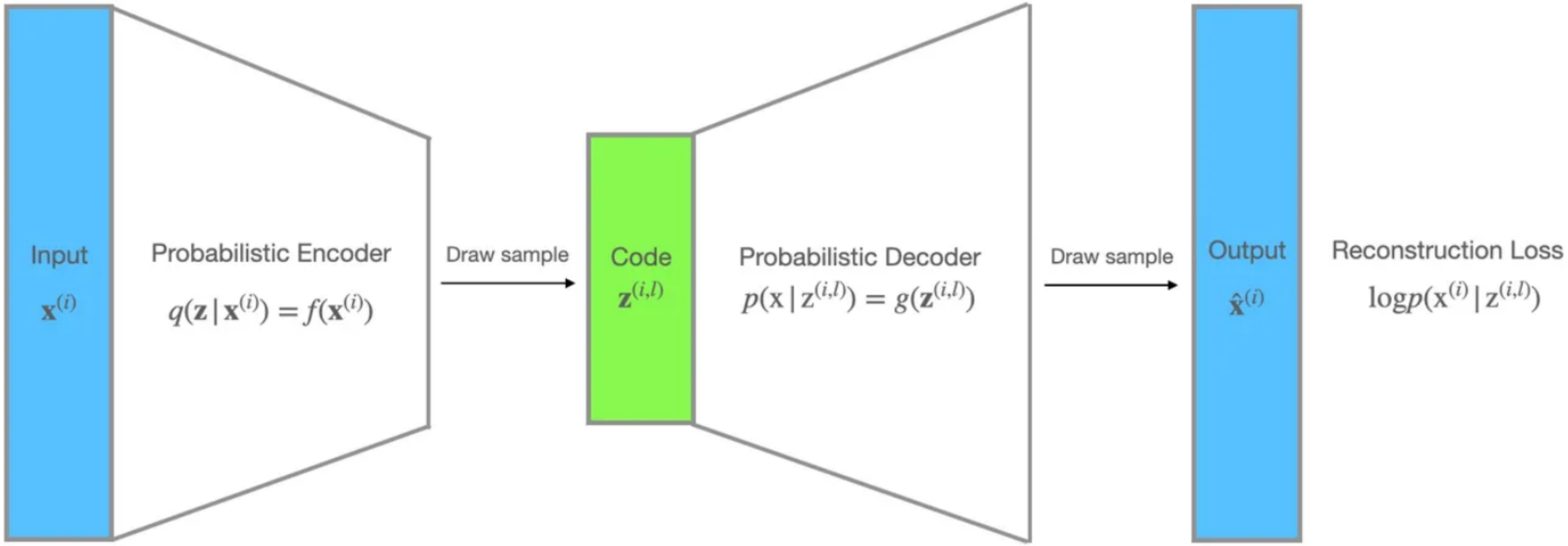
VAE


Input
$\mathbf{x}^{(i)}$
Encoder
$f(\mathbf{x}^{(i)})$
Code
$\mathbf{z}^{(i)}$
Decoder
$g(\mathbf{z}^{(i)})$
Output
$\hat{\mathbf{x}}^{(i)}$
Reconstruction Loss
$||\hat{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}||^2$

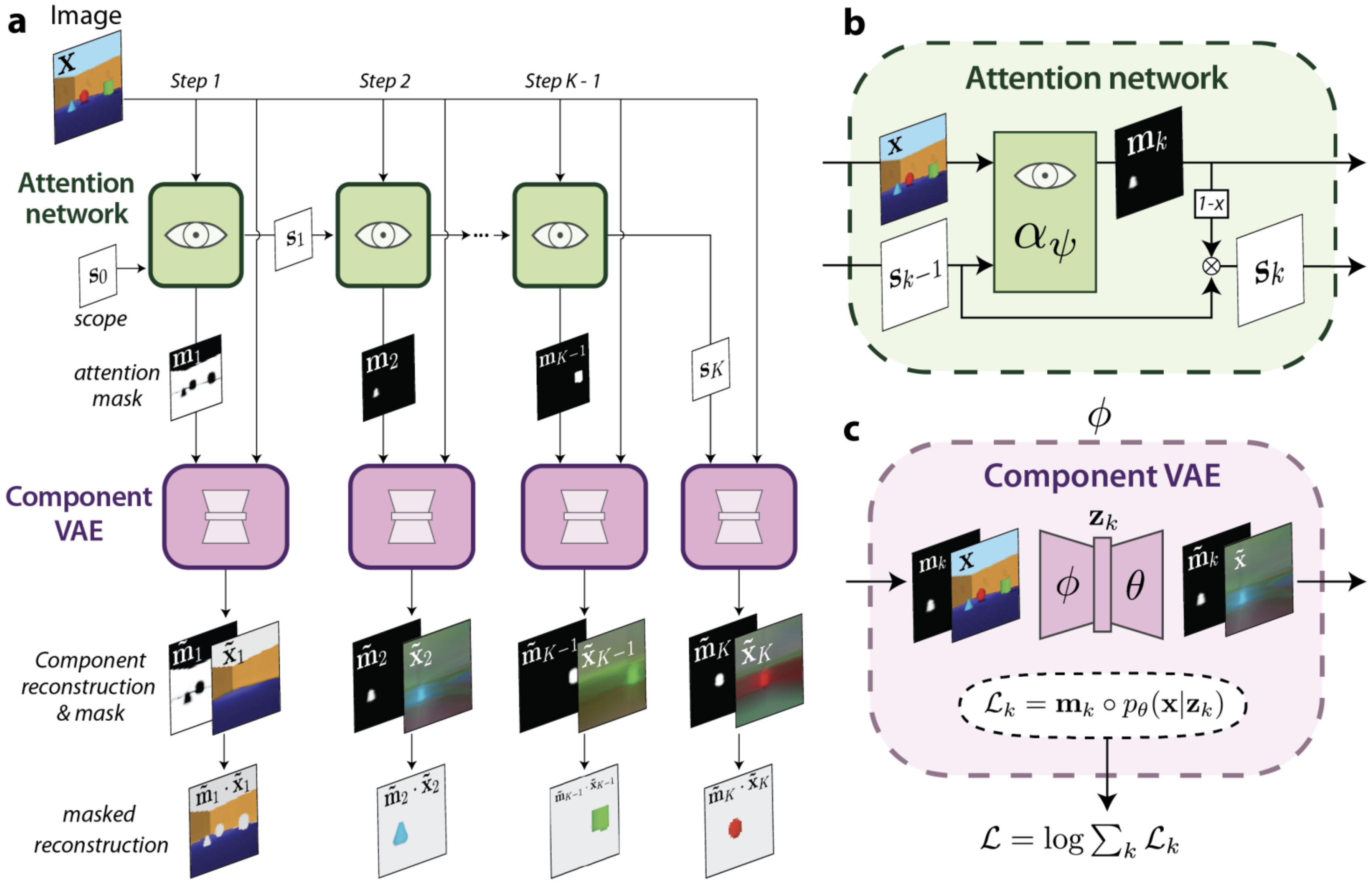
VAE


0
1
2
Input
$\mathbf{x}^{(i)}$
Encoder
$f(\mathbf{x}^{(i)})$
$\mathbf{z}^{(i)}$
Decoder
$g(\mathbf{z}^{(i)})$
Output
$\hat{\mathbf{x}}^{(i)}$
Reconstruction Loss
$||\hat{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}||^2$

VAE

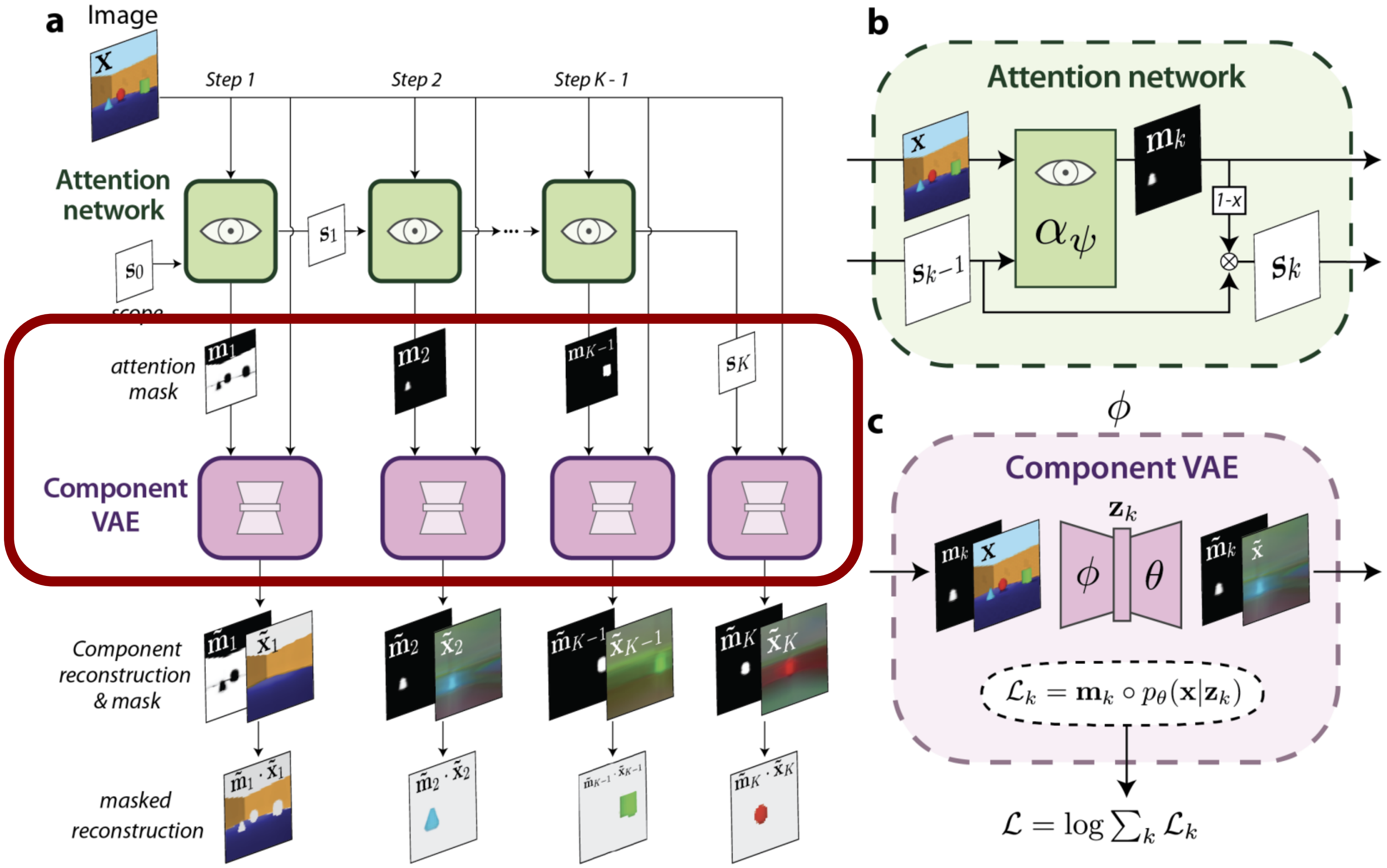

Input
$\mathbf{x}^{(i)}$
Probabilistic Encoder
$q(\mathbf{z}|\mathbf{x}^{(i)}) = f(\mathbf{x}^{(i)})$
Draw sample
Code
$\mathbf{z}^{(i,l)}$
Probabilistic Decoder
$p(\mathrm{x}|\mathrm{z}^{(i,l)}) = g(\mathbf{z}^{(i,l)})$
Draw sample
Output
$\hat{\mathbf{x}}^{(i)}$
Reconstruction Loss
$\log p(\mathrm{x}^{(i)}|\mathrm{z}^{(i,l)})$

MONet

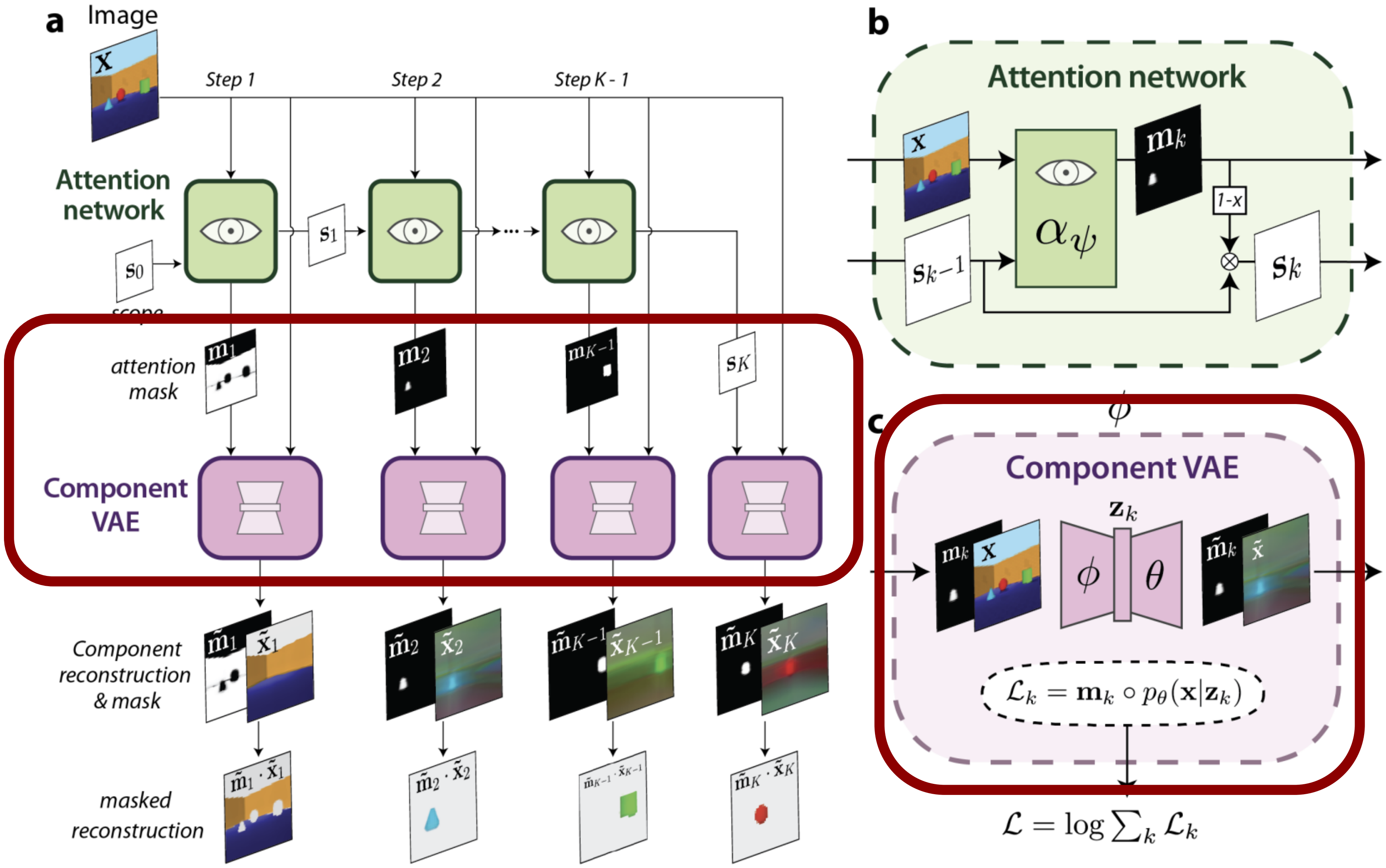


a
Image
x
Step 1
Step 2
Step K - 1
Attention network
s0
scope
s1
attention mask
m1
m2
mK-1
sK
Component VAE
Component reconstruction & mask
masked reconstruction
b
Attention network
x
mk
1-x
sk-1
sk
c
Component VAE
zk
mk
x
ϕ
θ
$\mathcal{L}_k = \mathbf{m}_k \circ p_\theta(\mathbf{x}|\mathbf{z}_k)$
$\mathcal{L} = \log \sum_k \mathcal{L}_k$

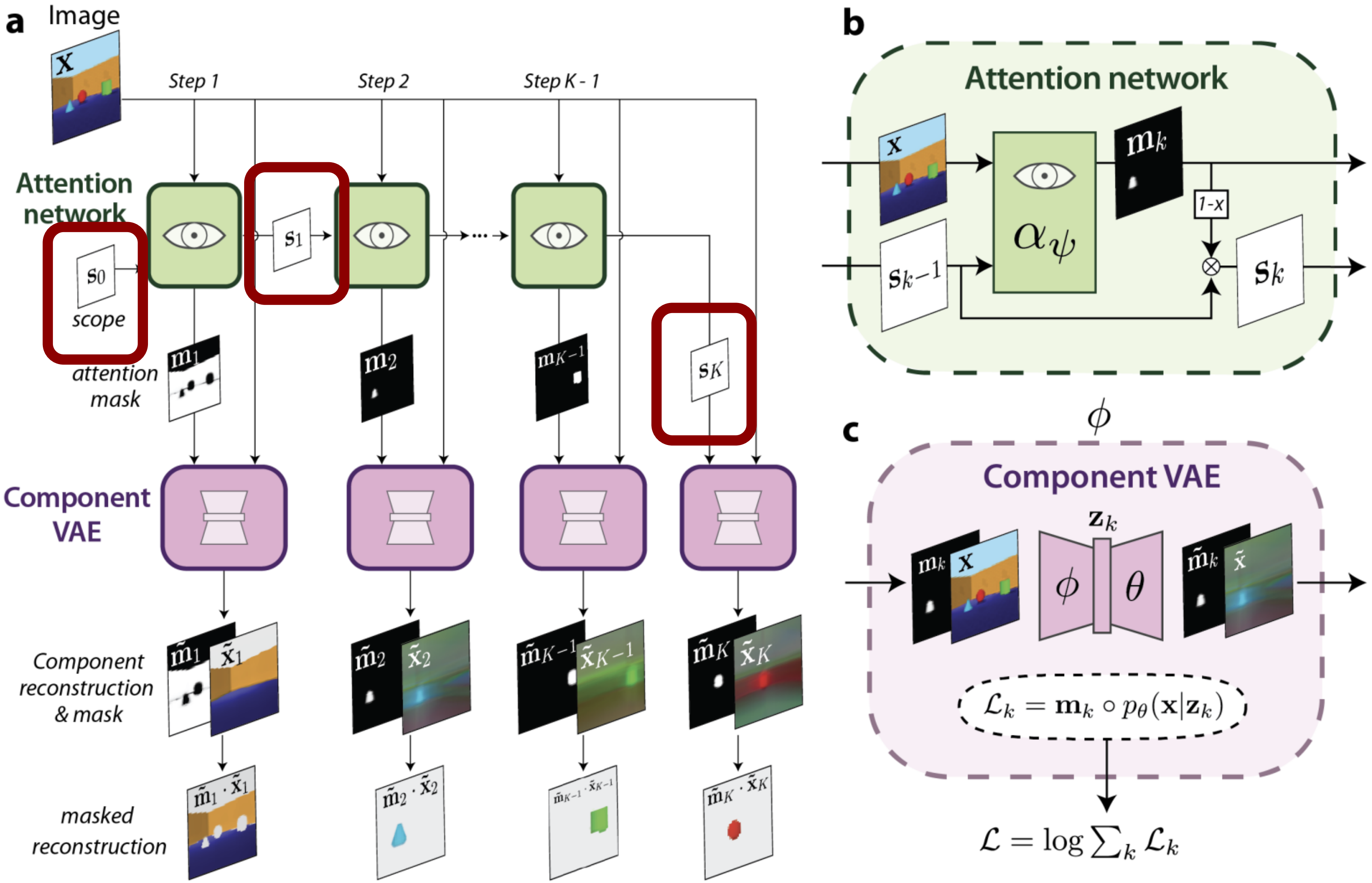
MONet


a
Image
x
Step 1
Step 2
Step K - 1
Attention network
s_0
scope
s_1
attention mask
m_1
m_2
m_{K-1}
s_K
Component VAE
Component reconstruction & mask
$\tilde{m}_1$ $\tilde{x}_1$
$\tilde{m}_2$ $\tilde{x}_2$
$\tilde{m}_{K-1}$ $\tilde{x}_{K-1}$
$\tilde{m}_K$ $\tilde{x}_K$
masked reconstruction
$\tilde{m}_1 \cdot \tilde{x}_1$
$\tilde{m}_2 \cdot \tilde{x}_2$
$\tilde{m}_{K-1} \cdot \tilde{x}_{K-1}$
$\tilde{m}_K \cdot \tilde{x}_K$
b
Attention network
x
α_ψ
m_k
1-x
s_{k-1}
s_k
ϕ
c
Component VAE
z_k
m_k x
ϕ θ
$\tilde{m}_k$ $\tilde{x}$
$\mathcal{L}_k = \mathbf{m}_k \circ p_\theta(\mathbf{x}|\mathbf{z}_k)$
$\mathcal{L} = \log \sum_k \mathcal{L}_k$

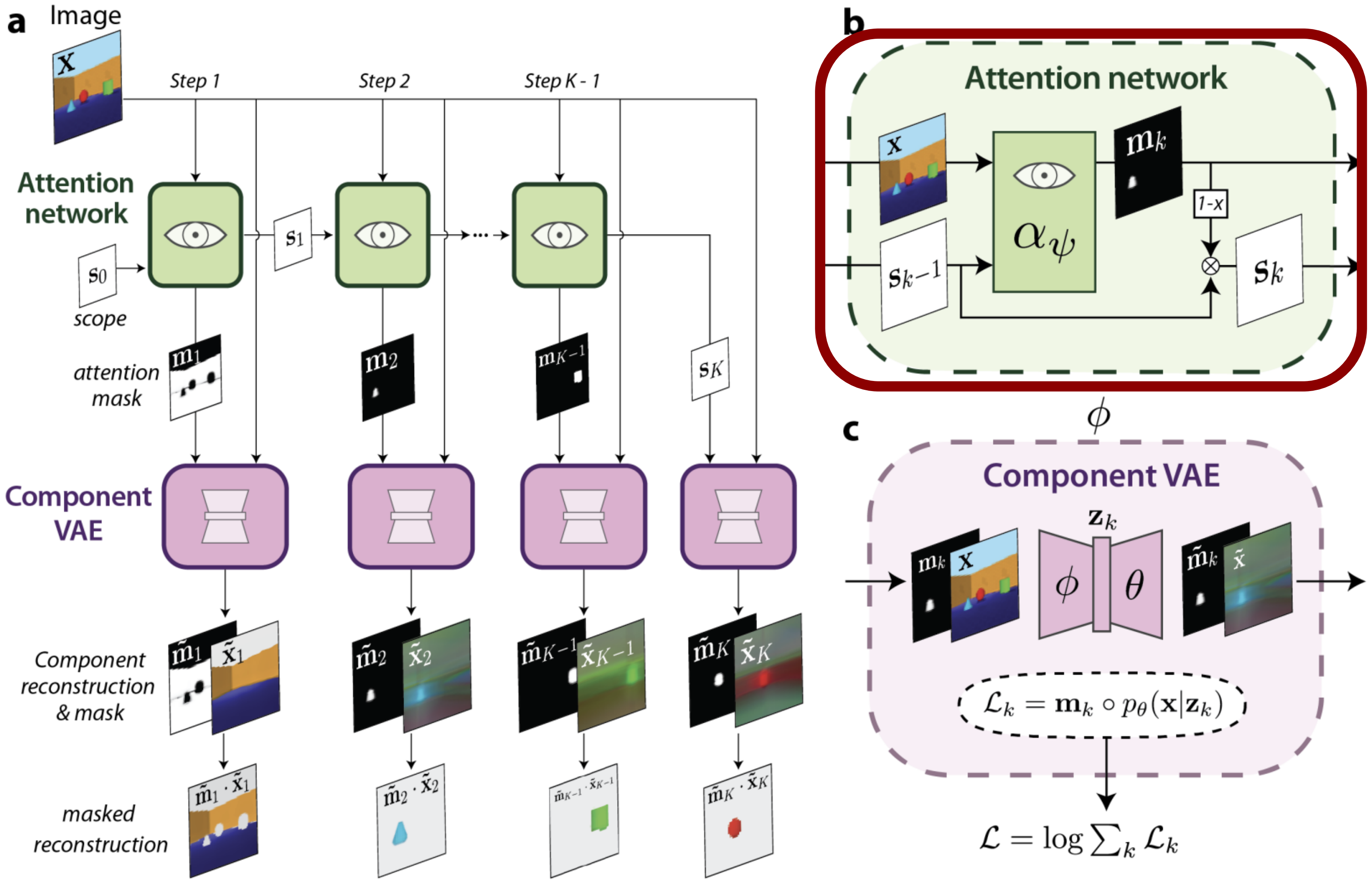
MONet


a
Image
x
Step 1
Step 2
Step K - 1
Attention network
s_0
scope
s_1
attention mask
m_1
m_2
m_{K-1}
s_K
Component VAE
Component reconstruction & mask
$\tilde{m}_1$ $\tilde{x}_1$
$\tilde{m}_2$ $\tilde{x}_2$
$\tilde{m}_{K-1}$ $\tilde{x}_{K-1}$
$\tilde{m}_K$ $\tilde{x}_K$
masked reconstruction
$\tilde{m}_1 \cdot \tilde{x}_1$
$\tilde{m}_2 \cdot \tilde{x}_2$
$\tilde{m}_{K-1} \cdot \tilde{x}_{K-1}$
$\tilde{m}_K \cdot \tilde{x}_K$
b
Attention network
x
m_k
α_ψ
1-x
s_{k-1}
s_k
c
ϕ
Component VAE
z_k
m_k x
ϕ θ
$\tilde{m}_k$ $\tilde{x}$
$\mathcal{L}_k = \mathbf{m}_k \circ p_\theta(\mathbf{x}|\mathbf{z}_k)$
$\mathcal{L} = \log \sum_k \mathcal{L}_k$

MONet

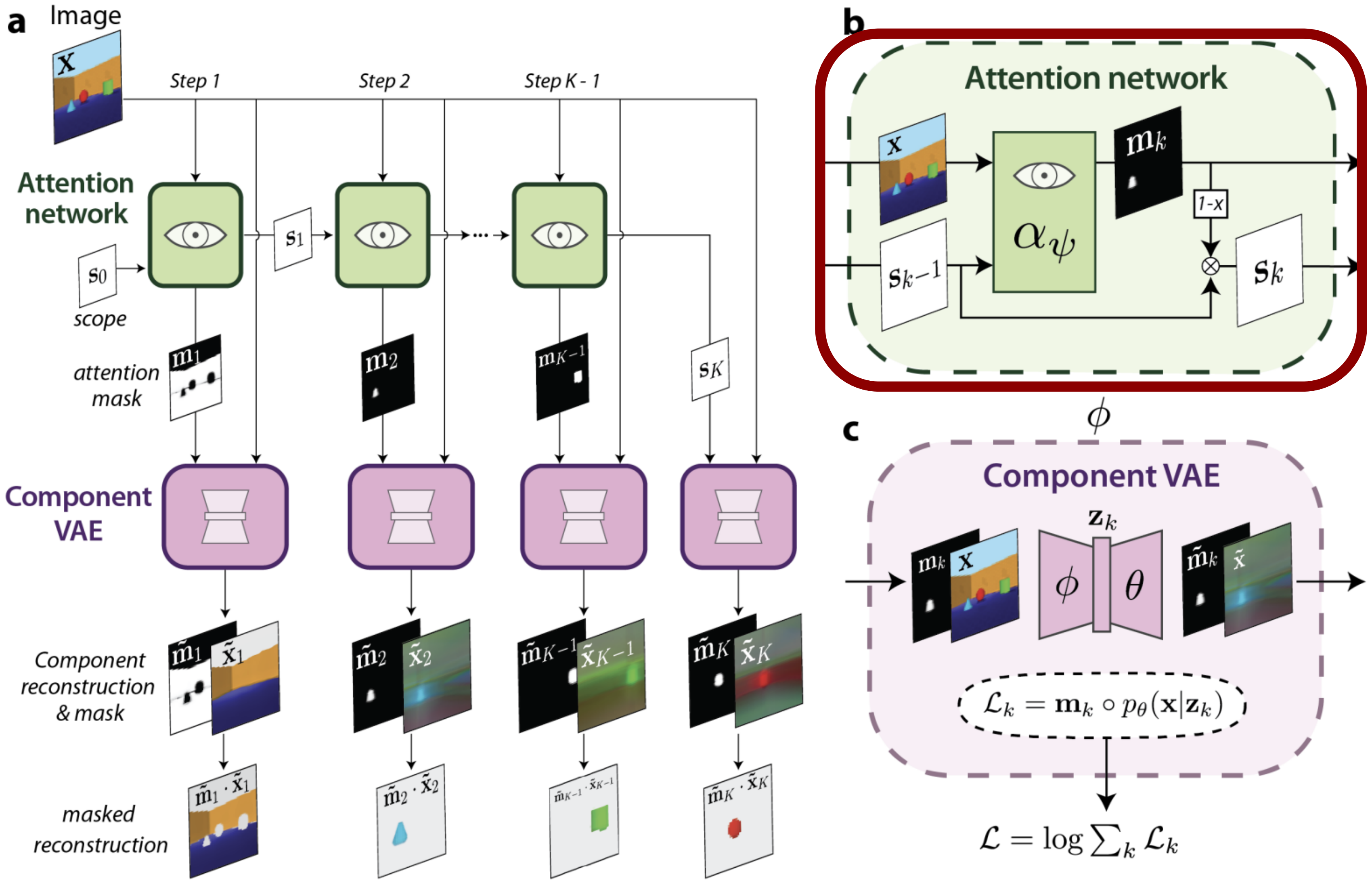

a
Image
x
Step 1
Step 2
Step K - 1
Attention network
s0
scope
s1
sK
m1
attention mask
m2
mK-1
Component VAE
Component reconstruction & mask
masked reconstruction
b
Attention network
x
mk
αψ
1-x
sk-1
sk
ϕ
c
Component VAE
zk
ϕ
θ
mk
x
m̃k
x̃
$\mathcal{L}_k = \mathbf{m}_k \circ p_\theta(\mathbf{x}|\mathbf{z}_k)$
$\mathcal{L} = \log \sum_k \mathcal{L}_k$

MONet



$$\mathbf{s}_{k+1} = \mathbf{s}_k(1 - \alpha_\psi(\mathbf{x}; \mathbf{s}_k))$$

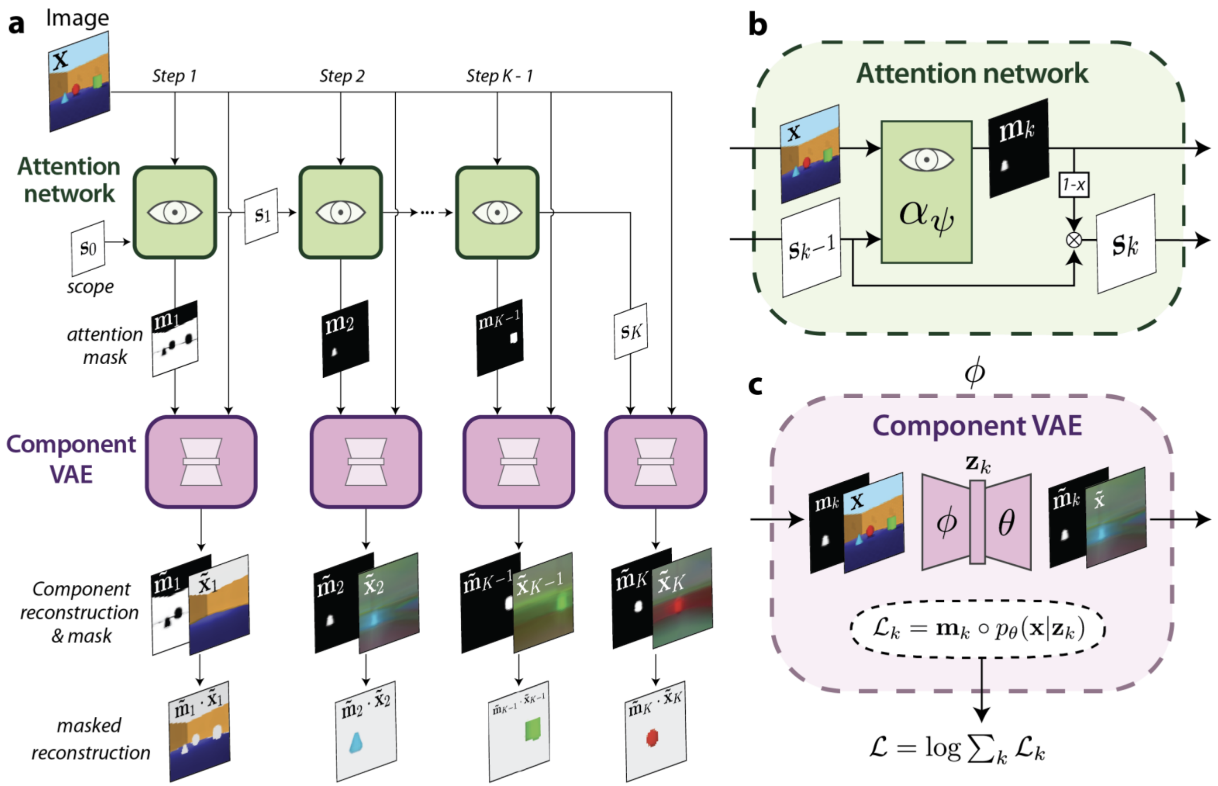
MONet



$$\mathbf{s}_{k+1} = \mathbf{s}_k(1 - \alpha_\psi(\mathbf{x}; \mathbf{s}_k))$$

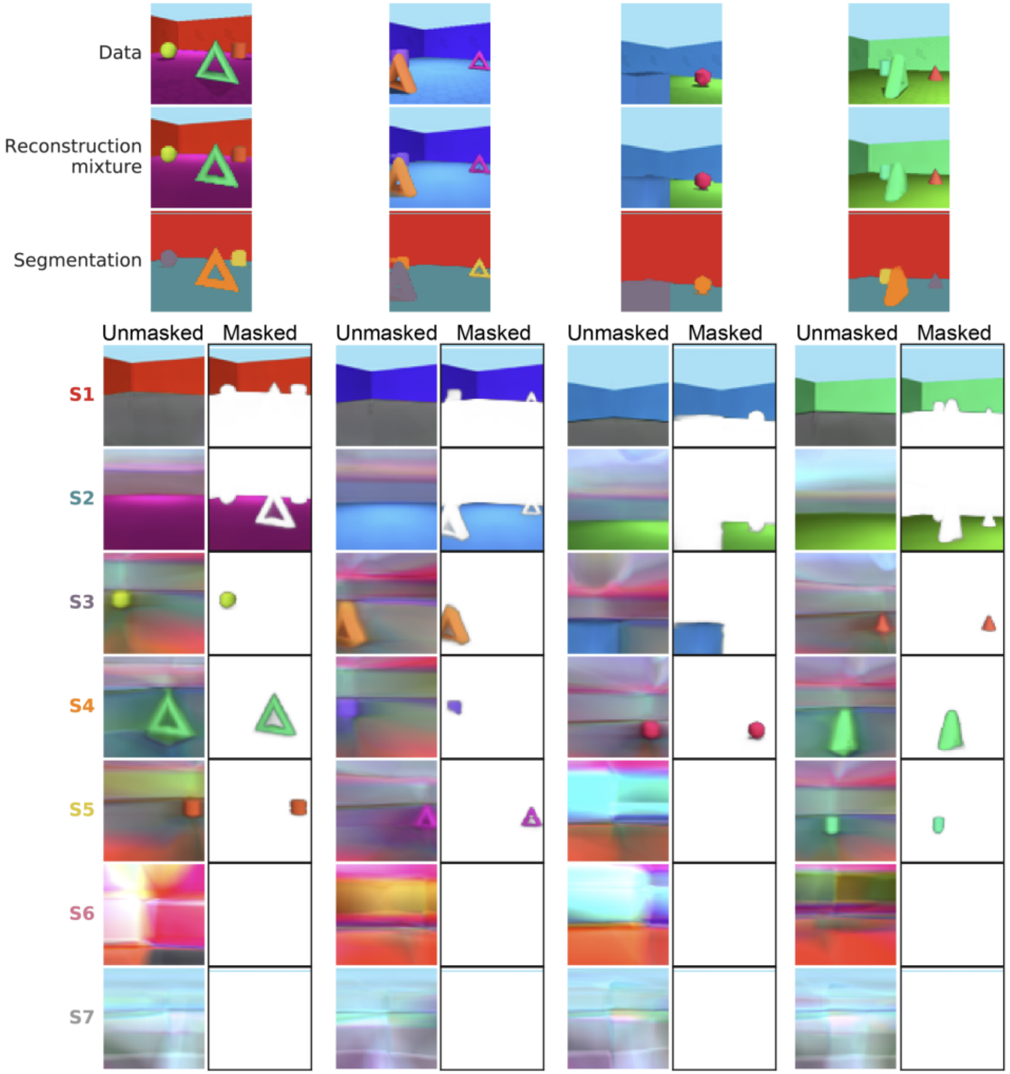
$$\mathbf{m}_k = \mathbf{s}_{k-1}\alpha_\psi(\mathbf{x}; \mathbf{s}_{k-1})$$

MONet



$$\mathcal{L}(\phi;\theta;\psi;\mathbf{x}) = -\log \sum_{k=1}^{K} \mathbf{m}_k p_\theta(\mathbf{x}|\mathbf{z}_k) + \beta D_{KL}\Big(\prod_{k=1}^{K} q_\phi(\mathbf{z}_k|\mathbf{x},\mathbf{m}_k) \parallel p(\mathbf{z})\Big) + \gamma D_{KL}\big(q_\psi(\mathbf{c}|\mathbf{x}) \parallel p_\theta(\mathbf{c}|\{\mathbf{z}_k\})\big)$$

MONet


Data
Reconstruction mixture
Segmentation
Unmasked
Masked
Unmasked
Masked
Unmasked
Masked
Unmasked
Masked
S1
S2
S3
S4
S5
S6
S7

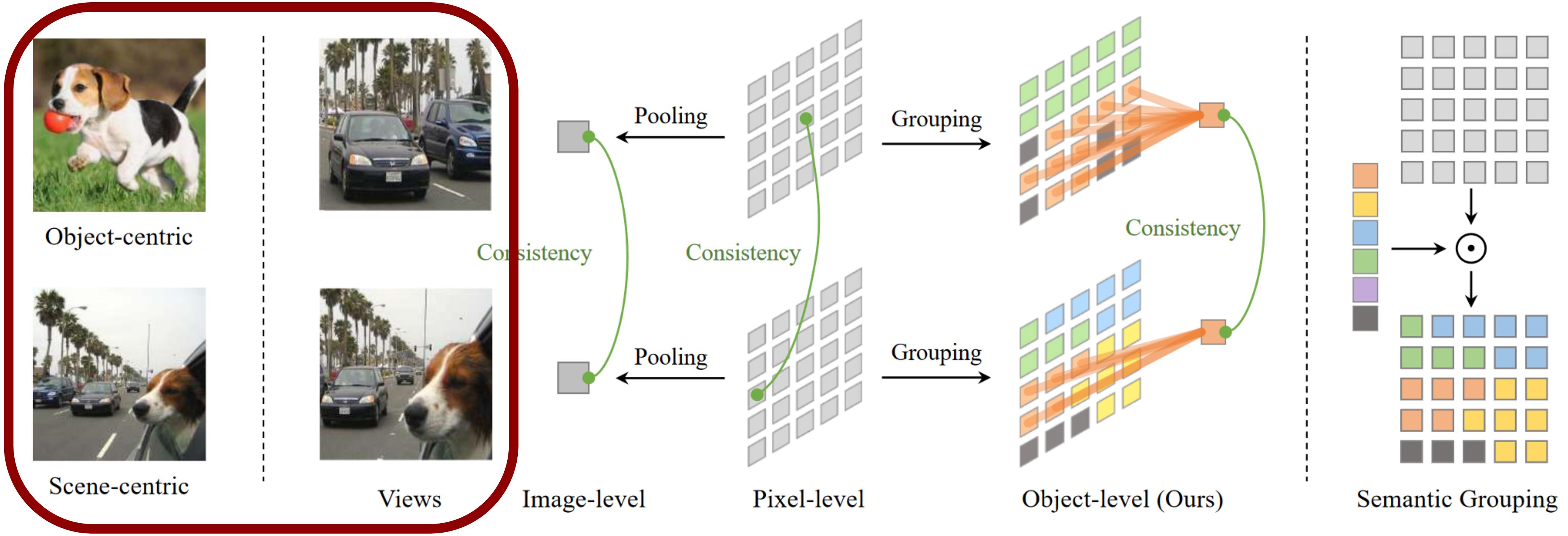
Background

- SlotCon (NeurIPS 2022)

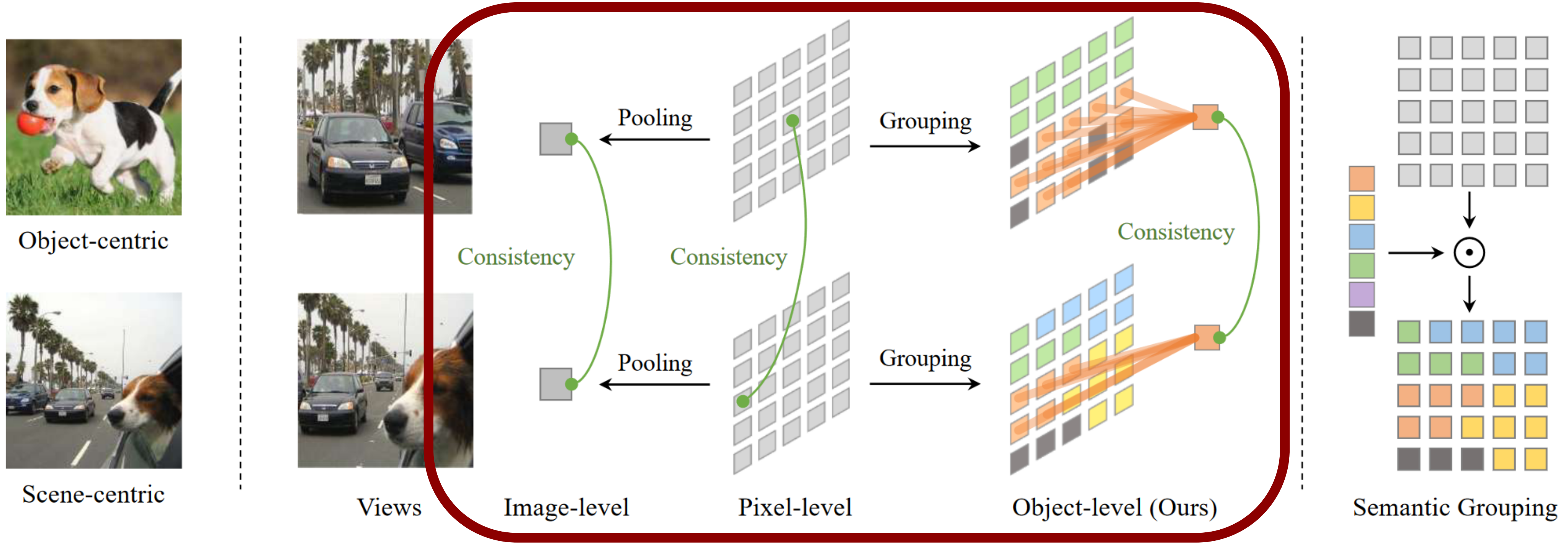
Self-Supervised Visual Representation Learning with Semantic Grouping

Xin Wen[1] Bingchen Zhao[2,3] Anlin Zheng[1,4] Xiangyu Zhang[4] Xiaojuan Qi[1]
[1]University of Hong Kong [2]University of Edinburgh [3]LunarAI [4]MEGVII Technology
{wenxin, xjqi}@eee.hku.hk zhaobc.gm@gmail.com
{zhenganlin, zhangxiangyu}@megvii.com

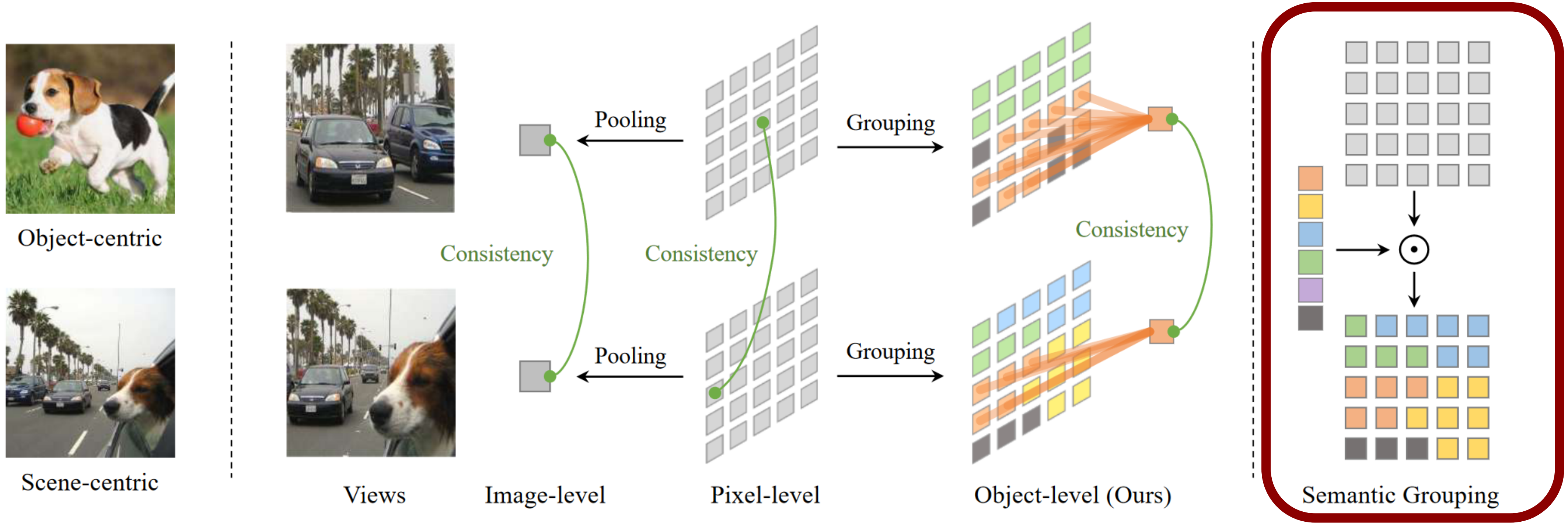
SlotCon


Object-centric
Scene-centric
Views
Pooling
Pooling
Consistency
Consistency
Grouping
Grouping
Consistency
Image-level
Pixel-level
Object-level (Ours)
Semantic Grouping

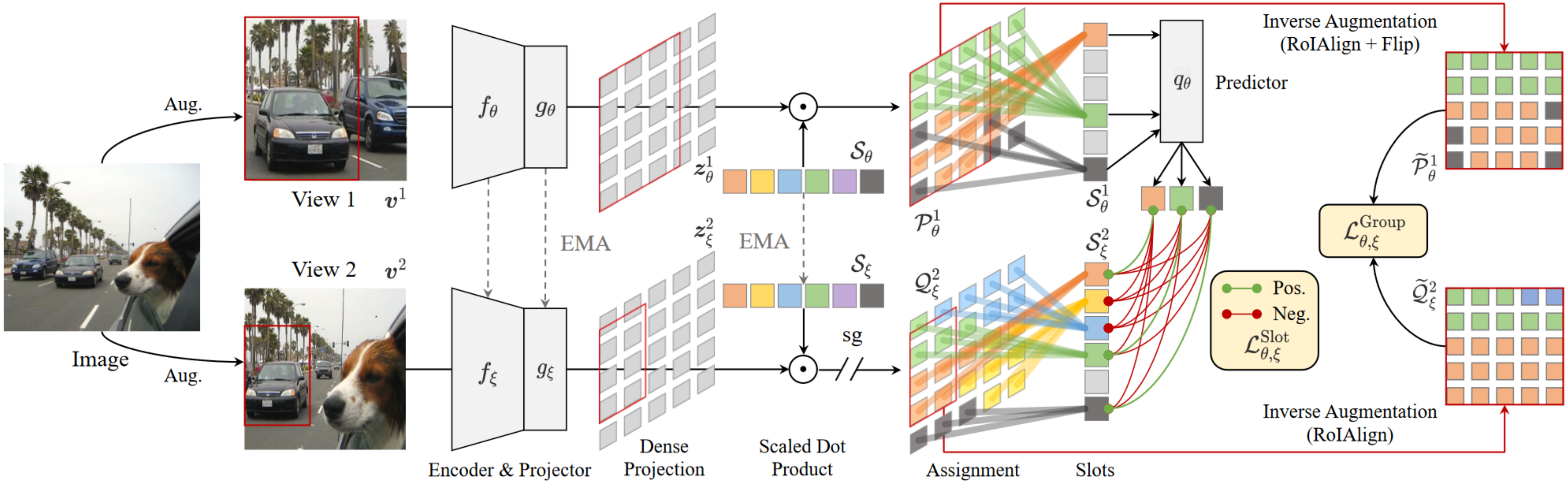
SlotCon


Object-centric
Scene-centric
Views
Pooling
Pooling
Grouping
Grouping
Consistency
Consistency
Consistency
Image-level
Pixel-level
Object-level (Ours)
Semantic Grouping

SlotCon

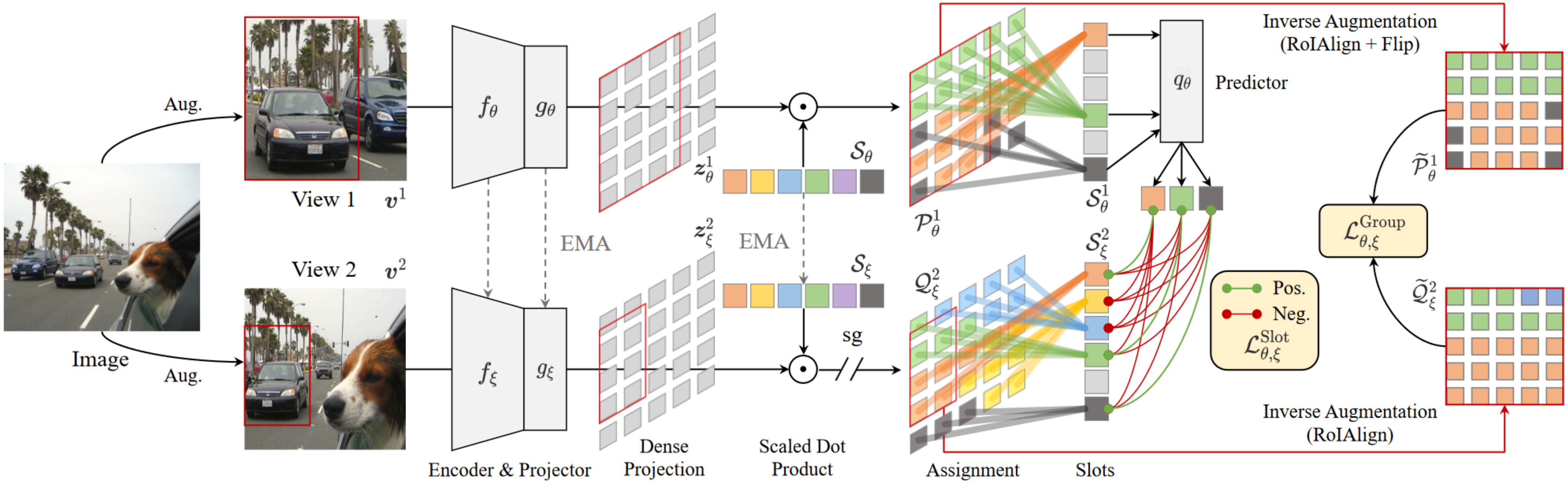

Object-centric
Scene-centric
Pooling
Pooling
Consistency
Consistency
Grouping
Grouping
Consistency
Views
Image-level
Pixel-level
Object-level (Ours)
Semantic Grouping

SlotCon



Aug.
View 1 v^1
Image
Aug.
View 2 v^2
f_θ
g_θ
f_ξ
g_ξ
EMA
z_θ^1
z_ξ^2
EMA
$\mathcal{S}_\theta$
$\mathcal{S}_\xi$
sg
$\mathcal{P}_\theta^1$
$\mathcal{Q}_\xi^2$
$\mathcal{S}_\theta^1$
$\mathcal{S}_\xi^2$
q_θ
Predictor
Inverse Augmentation
(RoIAlign + Flip)
$\tilde{\mathcal{P}}_\theta^1$
$\mathcal{L}_{\theta,\xi}^{\text{Group}}$
$\tilde{\mathcal{Q}}_\xi^2$
Pos.
Neg.
$\mathcal{L}_{\theta,\xi}^{\text{Slot}}$
Inverse Augmentation
(RoIAlign)
Encoder & Projector
Dense
Projection
Scaled Dot
Product
Assignment
Slots

SlotCon


Aug.
Image
Aug.
View 1
View 2
EMA
EMA
Encoder & Projector
Dense Projection
Scaled Dot Product
sg
Assignment
Slots
Predictor
Inverse Augmentation (RoIAlign + Flip)
Inverse Augmentation (RoIAlign)
Pos.
Neg.

Background

- CAE
- Complex-Valued Autoencoders
- Simple Design

Complex-Valued Autoencoders for Object Discovery

Sindy Löwe *loewe.sindy@gmail.com*
UvA-Bosch Delta Lab, University of Amsterdam

Phillip Lippe *p.lippe@uva.nl*
QUVA Lab, University of Amsterdam

Maja Rudolph *maja.rudolph@us.bosch.com*
Bosch Center for AI

Max Welling *m.welling@uva.nl*
UvA-Bosch Delta Lab, University of Amsterdam

Background

- CAE
- Complex-Valued Autoencoders
- Simple Design

Complex-Valued Autoencoders for Object Discovery

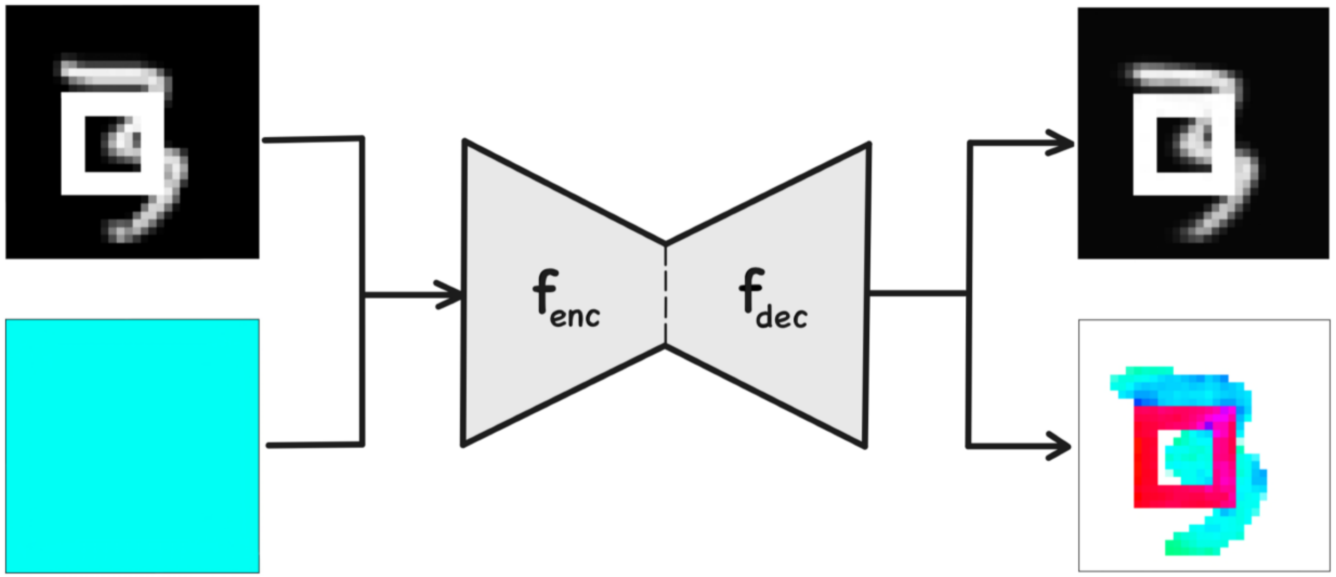
Sindy Löwe *loewe.sindy@gmail.com*
UvA-Bosch Delta Lab, University of Amsterdam

Phillip Lippe *p.lippe@uva.nl*
QUVA Lab, University of Amsterdam

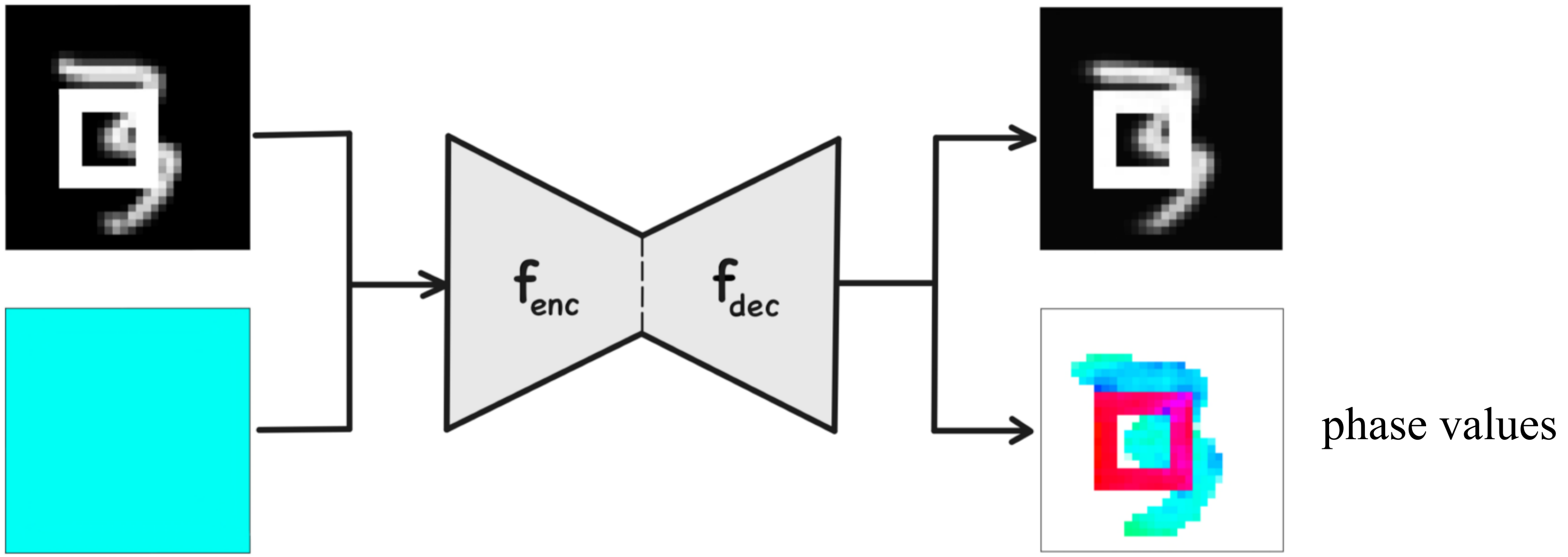
Maja Rudolph *maja.rudolph@us.bosch.com*
Bosch Center for AI

Max Welling *m.welling@uva.nl*
UvA-Bosch Delta Lab, University of Amsterdam

CAE



CAE



CAE

- **Synchronization**
  - **Additive operations**

$$\psi = f_{\mathbf{w}}(\mathbf{z}) = f_{\mathbf{w}}(\mathrm{Re}(\mathbf{z})) + f_{\mathbf{w}}(\mathrm{Im}(\mathbf{z})) \cdot i \quad \in \mathbb{C}^{d_{\mathrm{out}}}$$

- **Desynchronization**

$$\boldsymbol{m}_{\psi} = |\psi| + \boldsymbol{b}_{m} \quad \in \mathbb{R}^{d_{\mathrm{out}}} \qquad \boldsymbol{\varphi}_{\psi} = \arg(\psi) + \boldsymbol{b}_{\varphi} \quad \in \mathbb{R}^{d_{\mathrm{out}}}$$

  - **More control over the precise phase shifts:**

- **Gating**

$$\chi = f_{\mathbf{w}}(|\mathbf{z}|) + \boldsymbol{b}_{m} \quad \in \mathbb{R}^{d_{\mathrm{out}}}$$

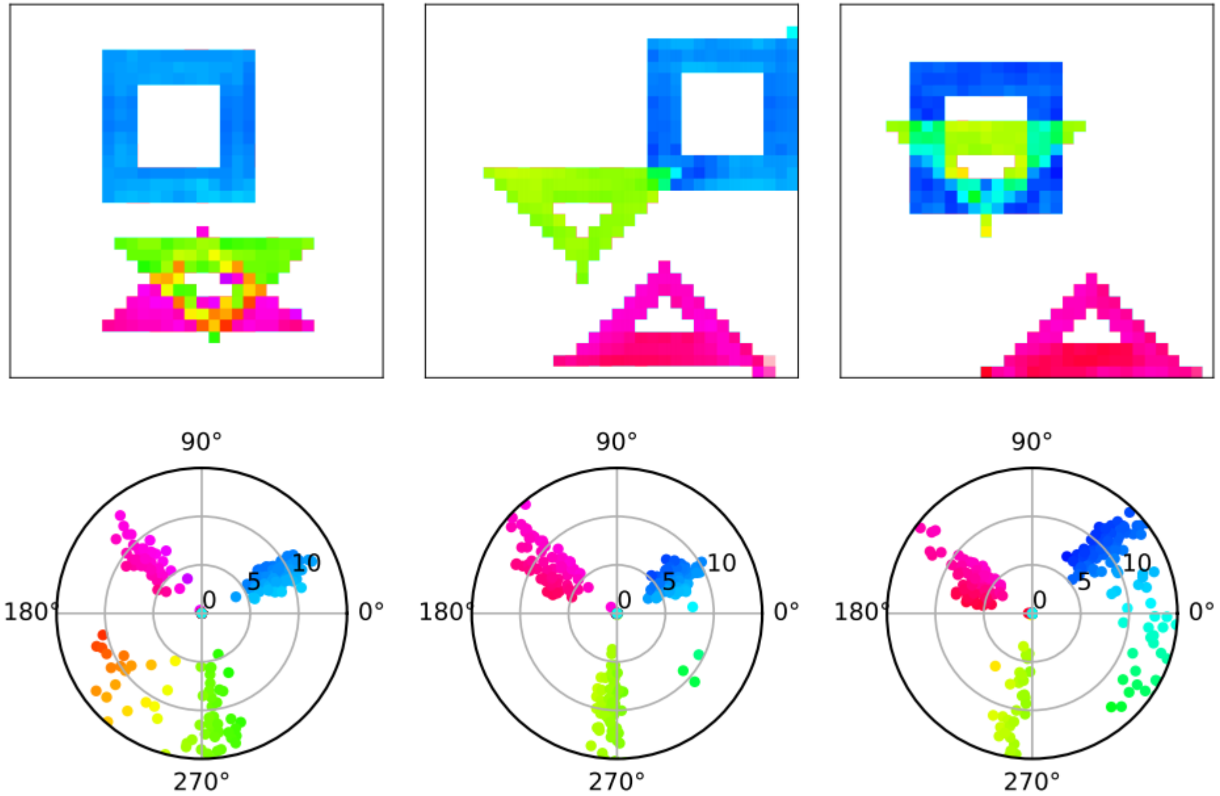
$$\boldsymbol{m}_{\mathbf{z}} = 0.5 \cdot \boldsymbol{m}_{\psi} + 0.5 \cdot \chi \quad \in \mathbb{R}^{d_{\mathrm{out}}}$$

CAE

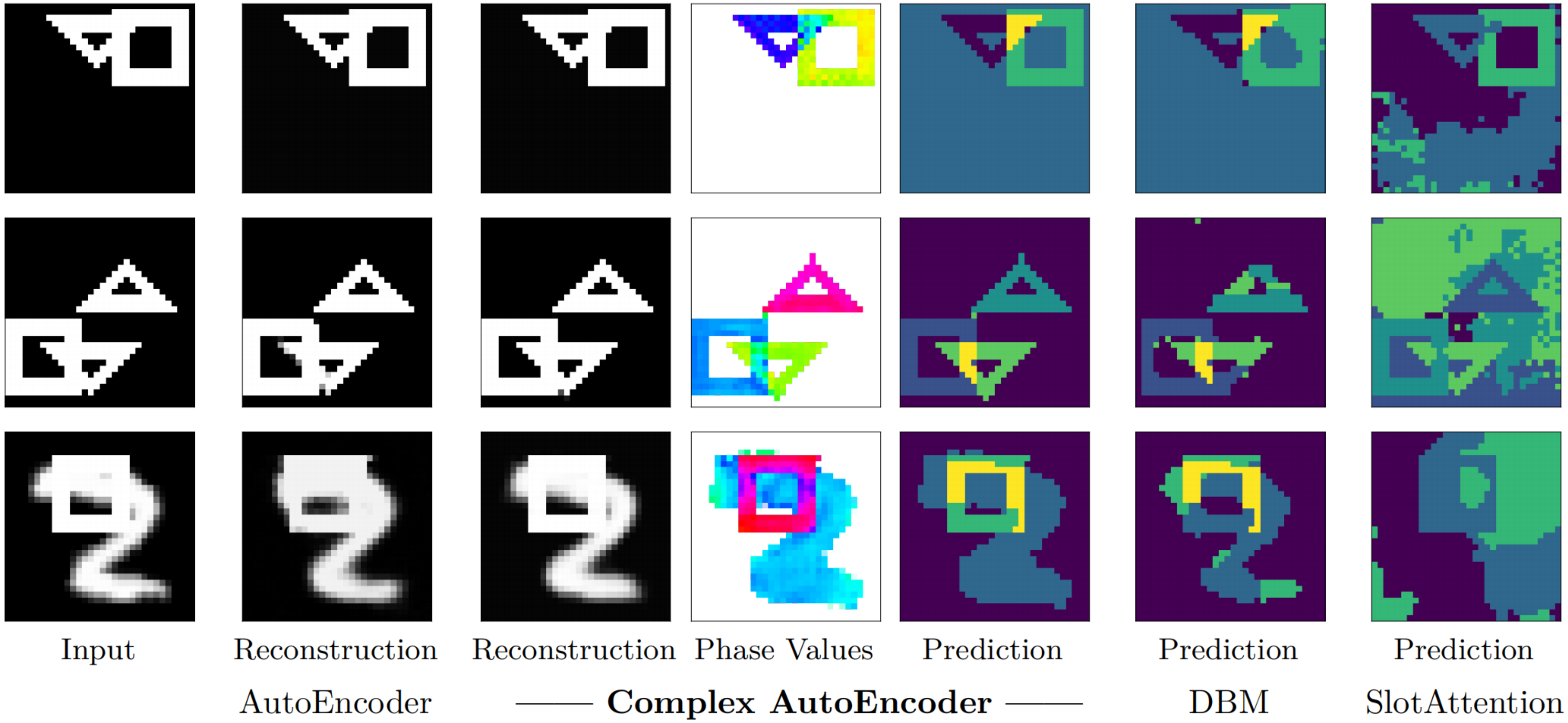
- **Complex-Valued Activation Function**

$$\mathbf{z}' = \mathrm{ReLU}(\mathrm{BatchNorm}(\boldsymbol{m}_{\mathbf{z}})) \circ e^{i\boldsymbol{\varphi}_{\boldsymbol{\psi}}} \quad \in \mathbb{C}^{d_{\text{out}}}$$

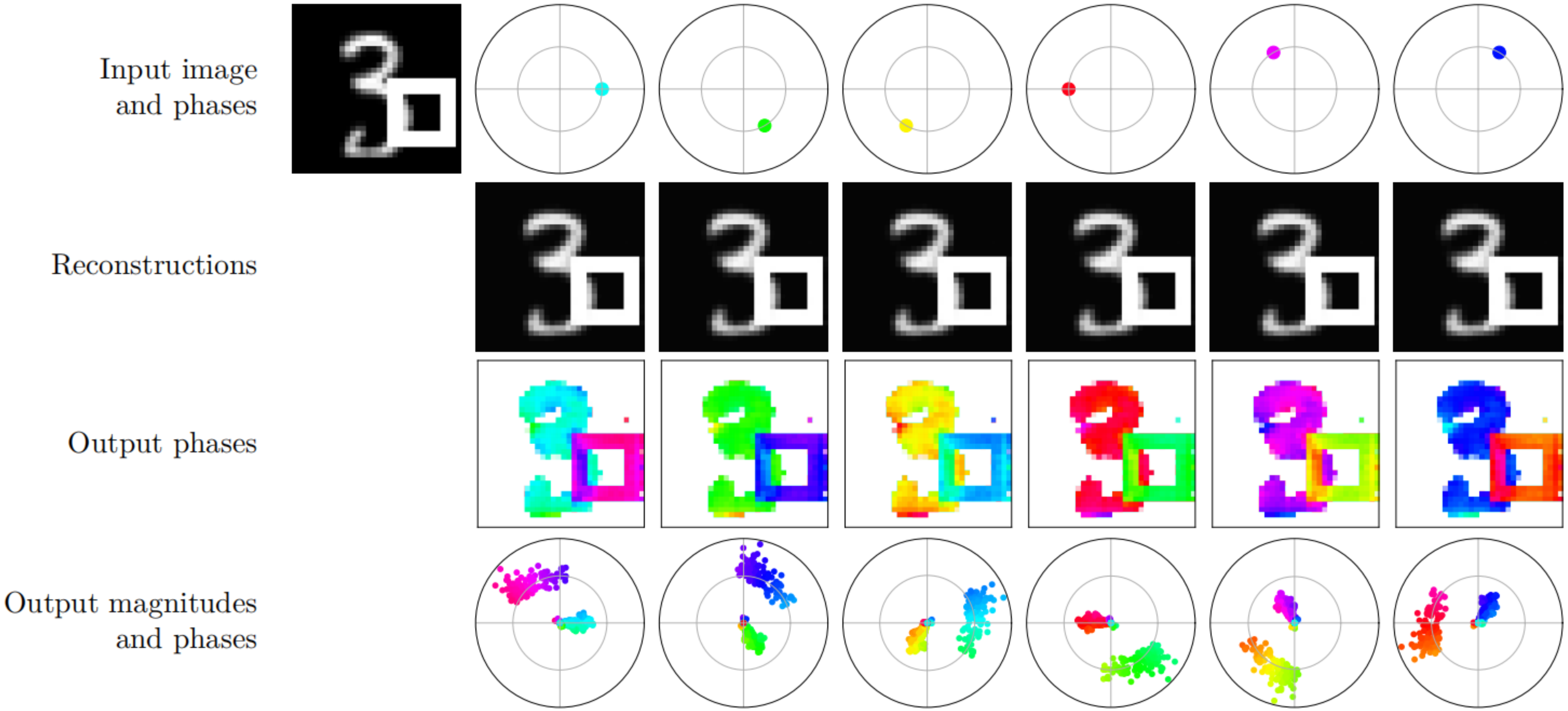
CAE



CAE


Input
Reconstruction
AutoEncoder
Reconstruction
Phase Values
Prediction
Complex AutoEncoder
Prediction
DBM
Prediction
SlotAttention

CAE

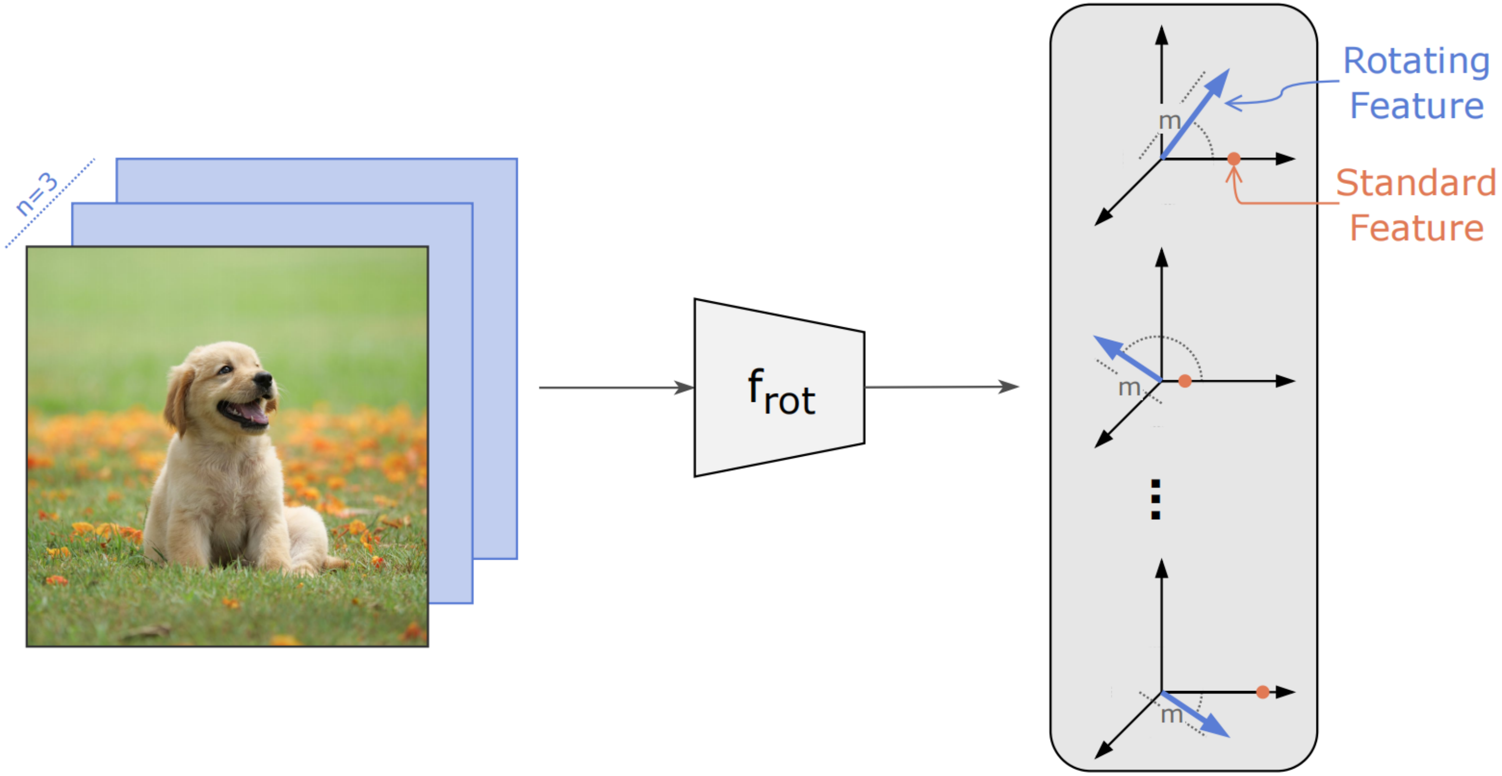

Input image
and phases
Reconstructions
Output phases
Output magnitudes
and phases

Outline

- Background
- Method
- Experiments
- Conclusion

Method

■ Rotating Features



Method

- **Rotating Features**

$\mathbf{z}_{\texttt{rotating}} \in \mathbb{R}^{n \times d}$ $\quad \left\|\mathbf{z}_{\texttt{rotating}}\right\|_2 \in \mathbb{R}^d$

Method

Rotating Features

$\mathbf{z}_{\text{in}} \in \mathbb{R}^{n \times d_{\text{in}}} \quad \mathbf{w} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}} \quad \mathbf{b} \in \mathbb{R}^{n \times d_{\text{out}}}$

$\psi = f_{\mathbf{w}}(\mathbf{z}_{\text{in}}) + \mathbf{b} \quad \in \mathbb{R}^{n \times d_{\text{out}}}$

Method

- **Rotating Features**

$$\mathbf{z}_{\text{in}} \in \mathbb{R}^{n \times d_{\text{in}}} \quad \mathbf{w} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}} \quad \mathbf{b} \in \mathbb{R}^{n \times d_{\text{out}}}$$

$$\psi = f_{\mathbf{w}}(\mathbf{z}_{\text{in}}) + \mathbf{b} \quad \in \mathbb{R}^{n \times d_{\text{out}}}$$

$$\chi = f_{\mathbf{w}}(\|\mathbf{z}_{\text{in}}\|_2) \quad \in \mathbb{R}^{d_{\text{out}}}$$

$$\mathbf{m}_{\text{bind}} = 0.5 \cdot \|\psi\|_2 + 0.5 \cdot \chi \quad \in \mathbb{R}^{d_{\text{out}}}$$

Method

Rotating Features

$$\mathbf{z}_{\text{in}} \in \mathbb{R}^{n \times d_{\text{in}}} \qquad \mathbf{w} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}} \qquad \mathbf{b} \in \mathbb{R}^{n \times d_{\text{out}}}$$

$$\boldsymbol{\psi} = f_{\mathbf{w}}(\mathbf{z}_{\text{in}}) + \mathbf{b} \quad \in \mathbb{R}^{n \times d_{\text{out}}}$$

$$\boldsymbol{\chi} = f_{\mathbf{w}}(\|\mathbf{z}_{\text{in}}\|_2) \quad \in \mathbb{R}^{d_{\text{out}}}$$

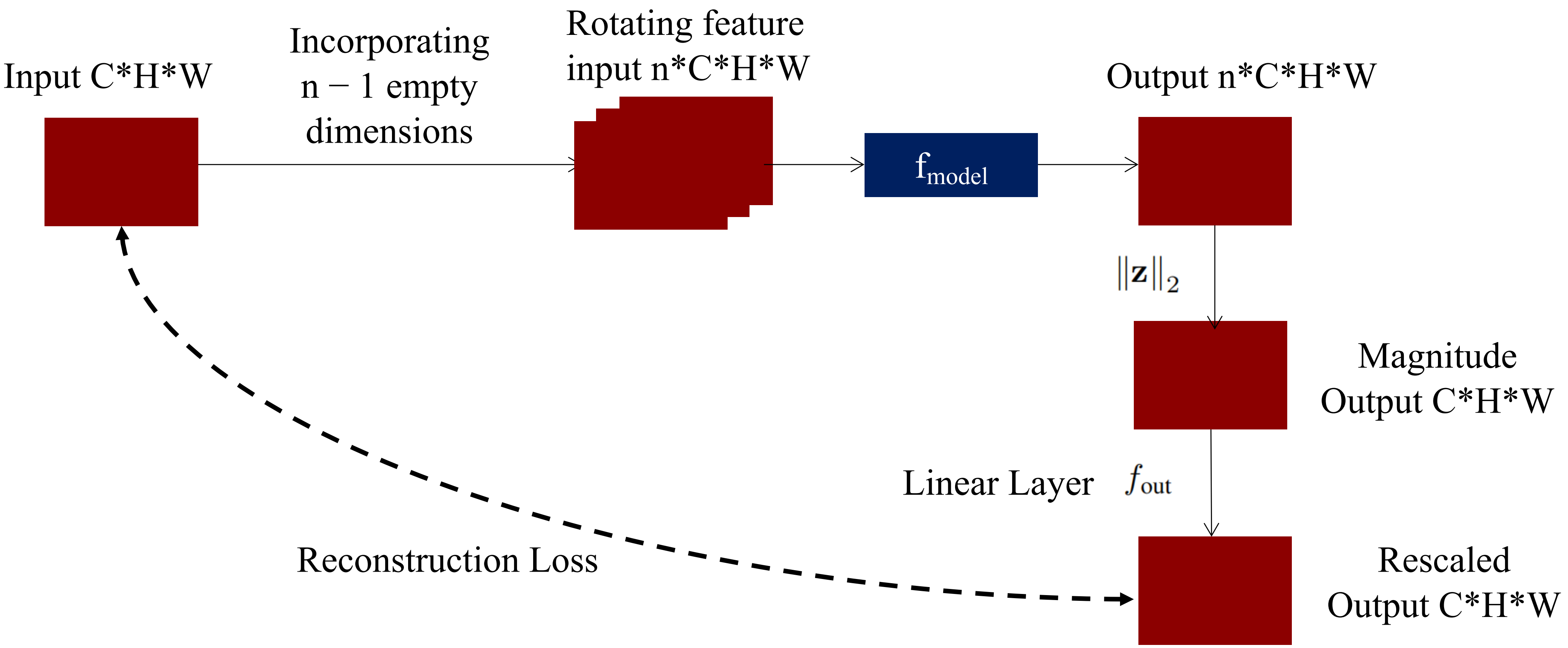
$$\mathbf{m}_{\text{bind}} = 0.5 \cdot \|\boldsymbol{\psi}\|_2 + 0.5 \cdot \boldsymbol{\chi} \quad \in \mathbb{R}^{d_{\text{out}}}$$

$$\mathbf{m}_{\text{out}} = \text{ReLU}(\text{BatchNorm}(\mathbf{m}_{\text{bind}})) \quad \in \mathbb{R}^{d_{\text{out}}}$$

$$\mathbf{z}_{\text{out}} = \frac{\boldsymbol{\psi}}{\|\boldsymbol{\psi}\|_2} \cdot \mathbf{m}_{\text{out}} \quad \in \mathbb{R}^{n \times d_{\text{out}}}$$

Method

Training Process



Method

- **Evaluating Object Separation in Rotating Features**

Method

- **Evaluating Object Separation in Rotating Features**

$$\mathbf{z}_{\text{norm}} = \frac{\mathbf{z}}{\left\|\mathbf{z}\right\|_2} \quad \in \mathbb{R}^{n \times c \times h \times w}$$

Method

- **Evaluating Object Separation in Rotating Features**

$$\mathbf{w}_{\text{eval}}^{i,j,l} = \begin{cases} 1 & \text{if } \|\mathbf{z}\|_2^{i,j,l} > t \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{z}_{\text{eval}} = \frac{\sum_{i=1}^{c} \mathbf{w}_{\text{eval}}^{i} \circ \mathbf{z}_{\text{norm}}^{i}}{\sum_{i=1}^{c} \mathbf{w}_{\text{eval}}^{i} + \varepsilon} \quad \in \mathbb{R}^{n \times h \times w}$$

Outline

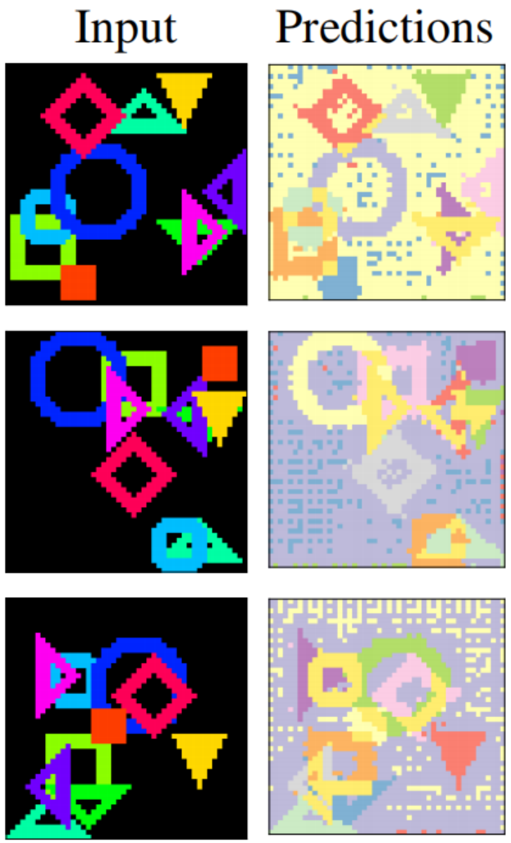
- Background
- Method
- Experiments
- Conclusion

Experiments

- Rotating Features Applied to Real-World Images

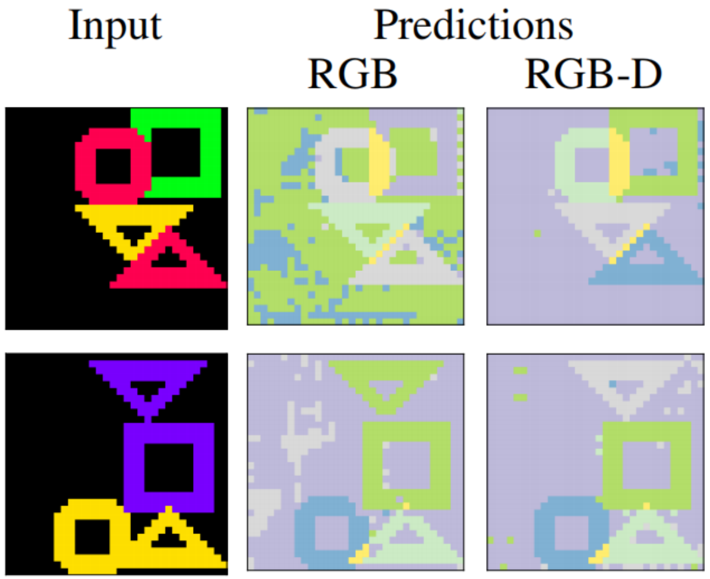
Experiments

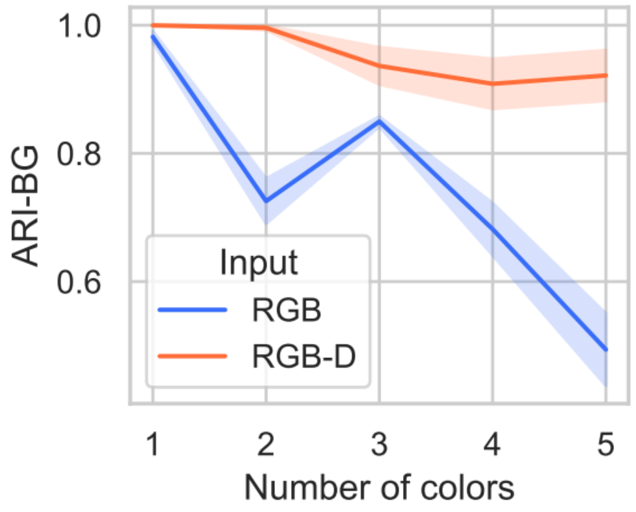
- **Rotating Features Can Represent More Objects**



Experiments

- **Rotating Features Are Applicable to Multi-Channel Images**

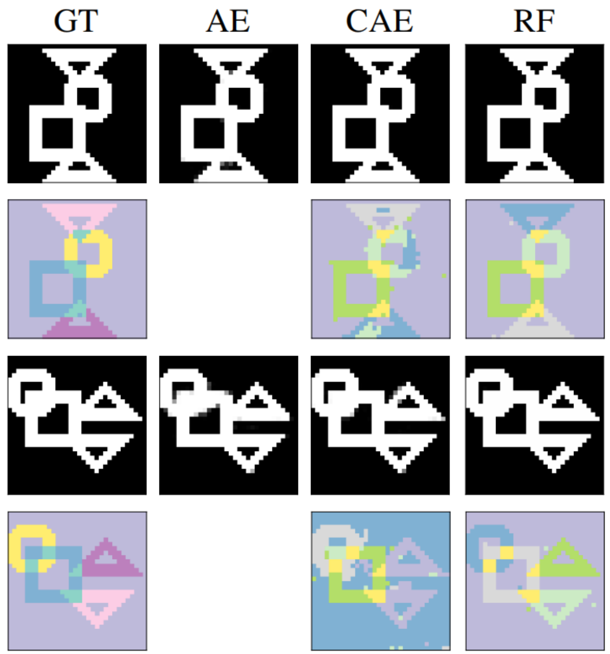




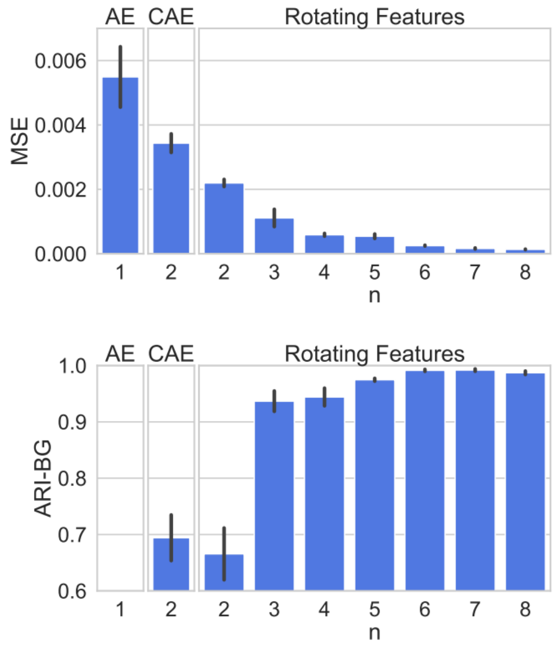
Experiments

Object Discovery

Model	$\text{MBO}_i \uparrow$	$\text{MBO}_c \uparrow$
Block Masks	0.247	0.259
Slot Attention	0.222 $\pm$ 0.008	0.237 $\pm$ 0.008
SLATE	0.310 $\pm$ 0.004	0.324 $\pm$ 0.004
Rotating Features –DINO	0.282 $\pm$ 0.006	0.320 $\pm$ 0.006
DINO k-means	0.363	0.405
DINO CAE	0.329 $\pm$ 0.009	0.374 $\pm$ 0.010
DINOSAUR Transformer	0.440 $\pm$ 0.008	0.512 $\pm$ 0.008
DINOSAUR MLP	0.395 $\pm$ 0.000	0.409 $\pm$ 0.000
Rotating Features	0.407 $\pm$ 0.001	0.460 $\pm$ 0.001



(a)



(b)

Experiments

- **Rotating Features Are Applicable to Real-World Images**

Thanks!