Unsupervised multi-object segmentation using attention and soft-argmax

Supplementary Material

1. Hyperparameter values

The hyperparameter values used for the proposed model are listed in Table 1.

2. Pseudo-code for objects encoder and decoder

The full encoding and rendering process is described in Algorithms 1 and 2.

3. Additional implementation details

The glimpse convolutional generator is described in Table 2.

Synthetic datasets and preprocessing codes were downloaded from the following public repositories:

- `https://www.robots.ox.ac.uk/~vgg/data/clevrtex/`
- `https://ogroth.github.io/shapestacks/`
- `https://github.com/deepmind/multi_object_datasets`
- `https://github.com/applied-ai-lab/genesis`.

The Segformer pretrained weights were downloaded from the following link:

`https://huggingface.co/nvidia/mit-b3`

The architecture of the U-net implemented for the ablation study is described in Table 3. It contains a sequence of downsample blocks which output feature maps of decreasing sizes, a center block which takes as input the feature map produced by the last downsample block, and upsample blocks, which take as input both the output of the previous upsample or center block and the feature map of the same size produced by corresponding downsample block.

- A downsample block is composed of a convolutional layer with stride 2 and kernel size 4, with batch normalization and CELU, followed by a residual convolutional layer with stride 1 and kernel size 3 with batch normalization and CELU.
- The center block is composed of a convolutional layer with stride 1 and kernel size 3 with batch normalization and CELU.
- An upsample block is composed of a residual convolutional layer with stride 1 and kernel size 3 with batch normalization and CELU, followed by a transpose convolutional layer with stride 2 and kernel size 4, with batch normalization and CELU.

4. Additional image samples

Additional image samples are provided in Figures 1-6.

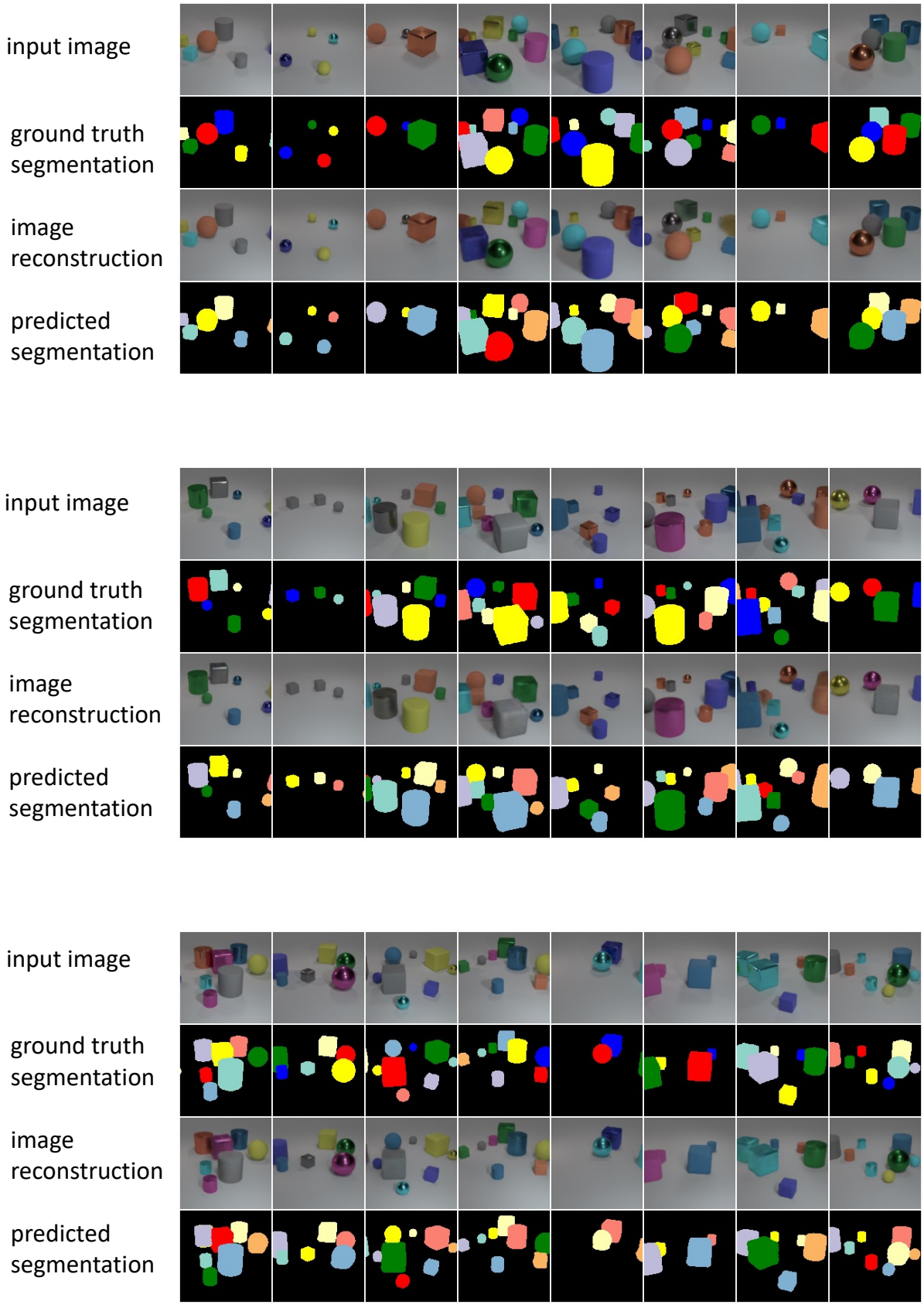

Figure 1. Examples of segmentation predictions on CLEVR test dataset

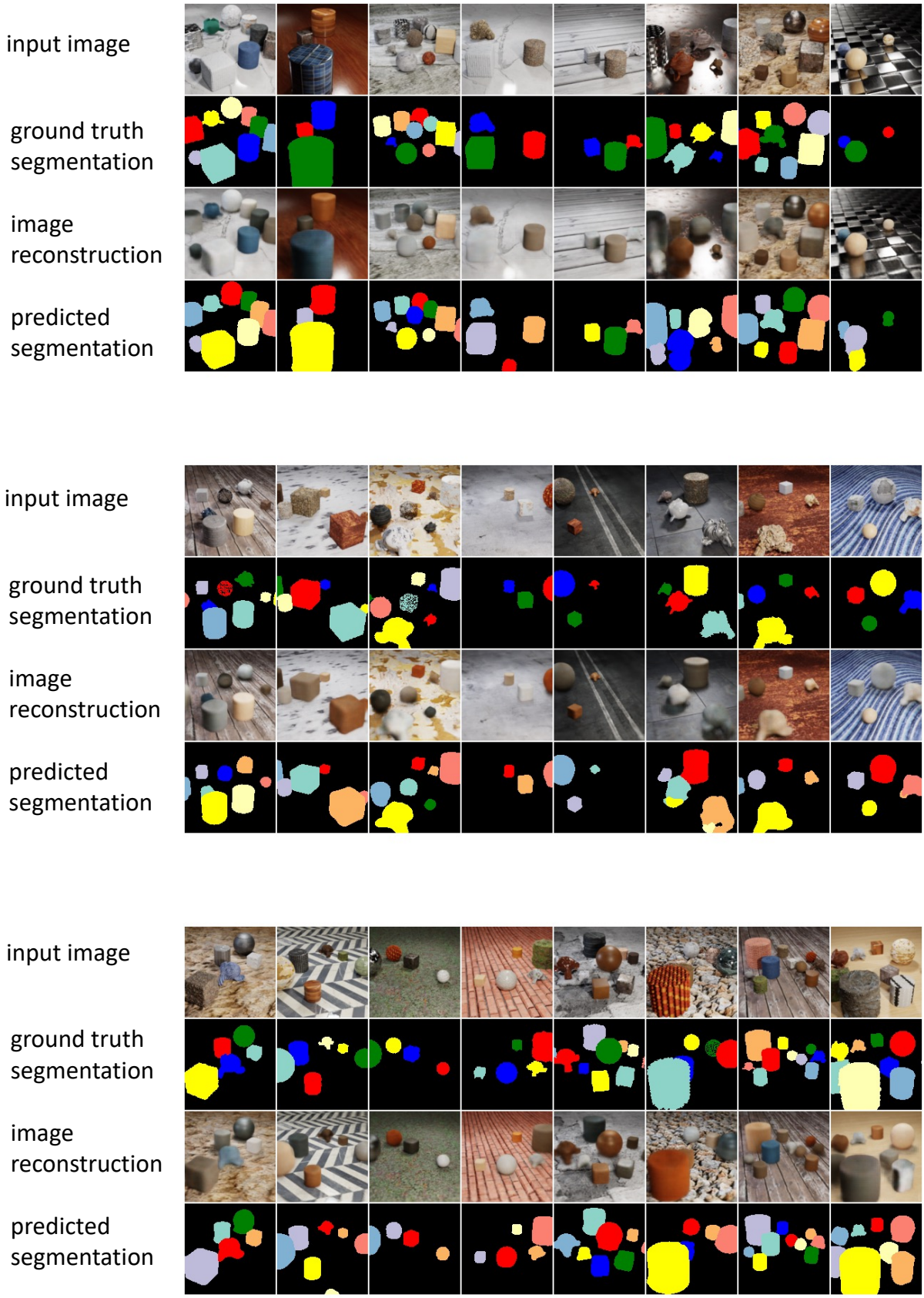

Figure 2. Examples of segmentation predictions on CLEVRTEX test dataset

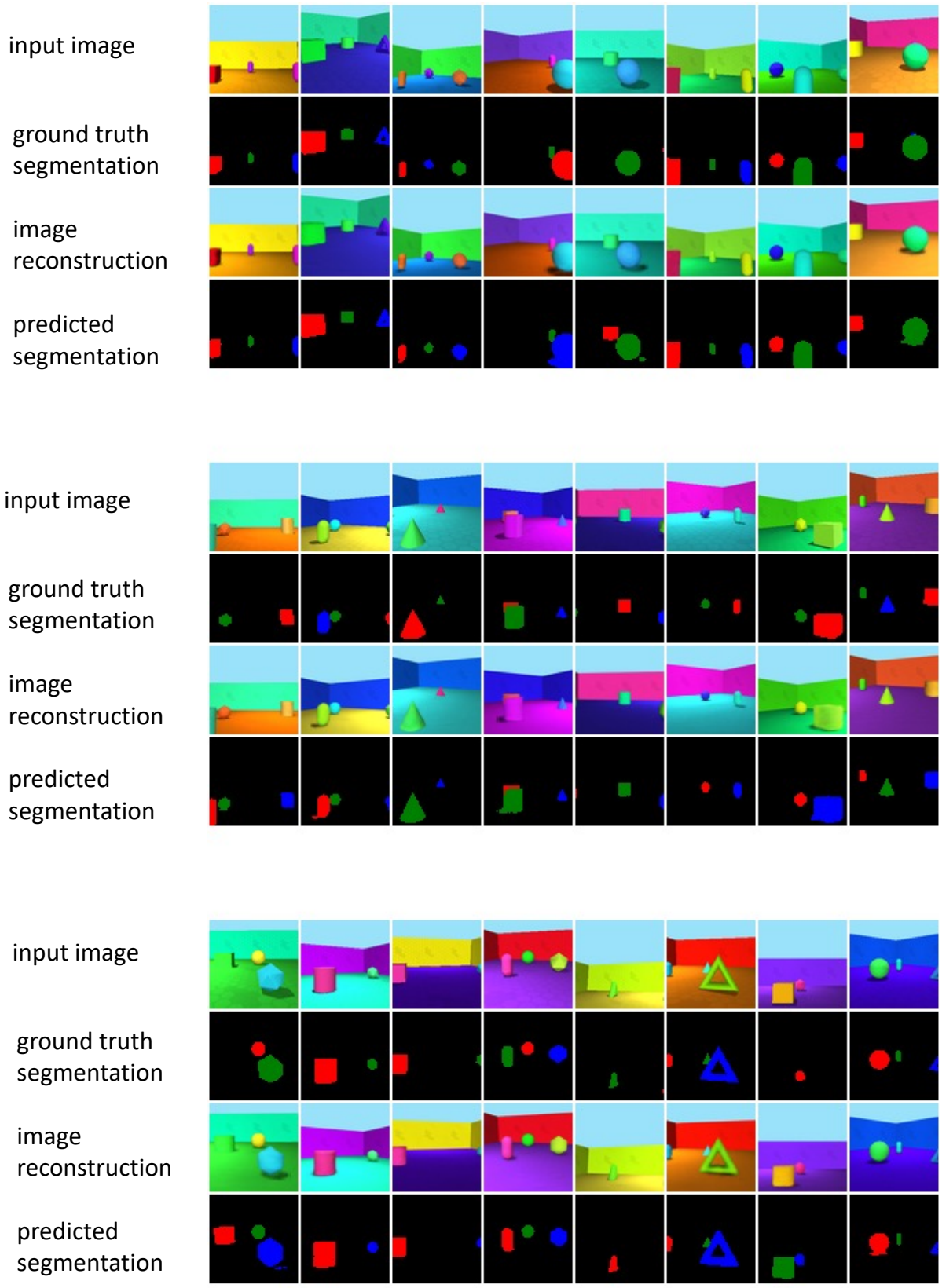

Figure 3. Examples of segmentation predictions on ObjectsRoom test dataset

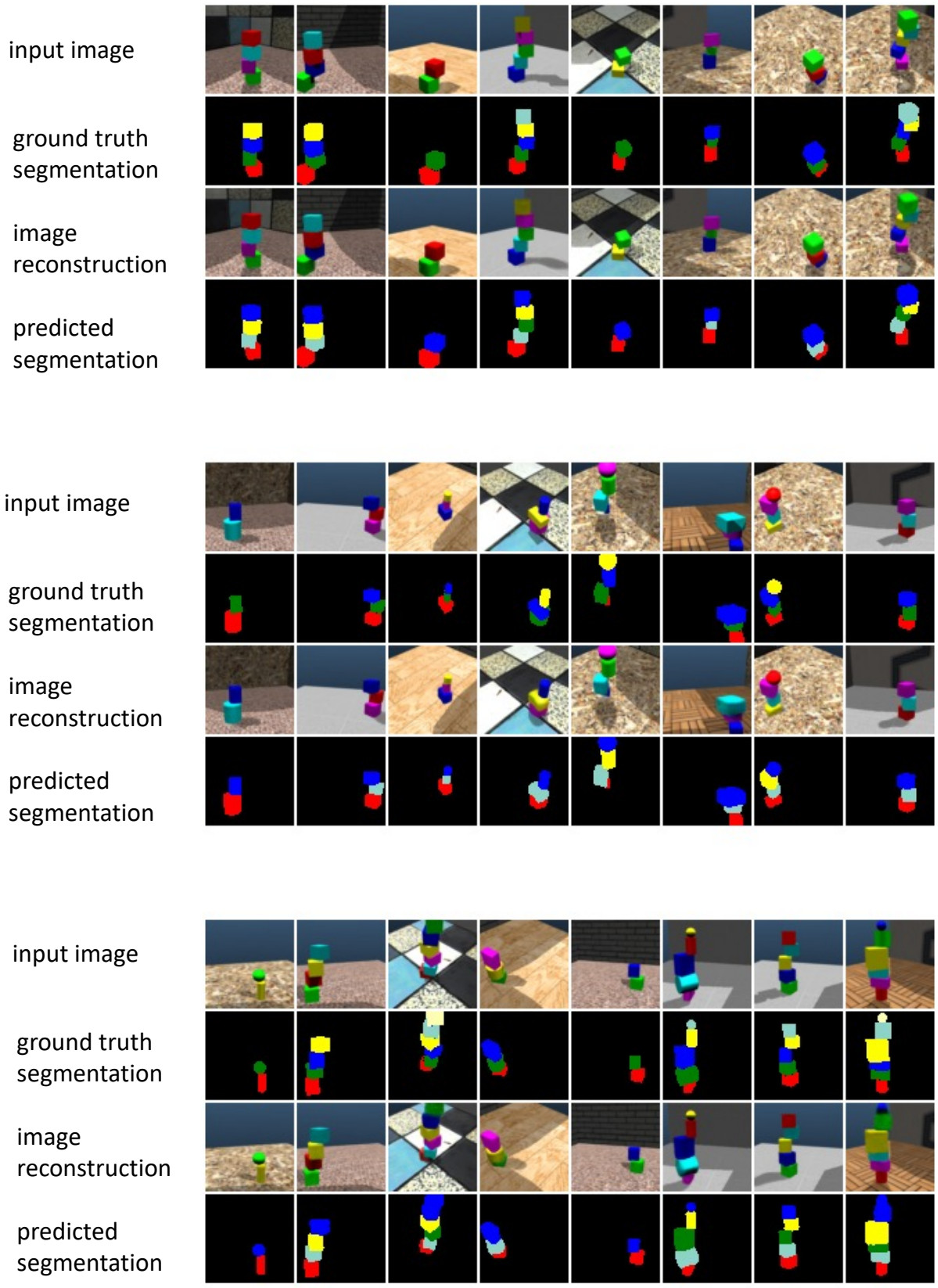

Figure 4. Examples of segmentation predictions on ShapeStacks test dataset (using a model without transformer)

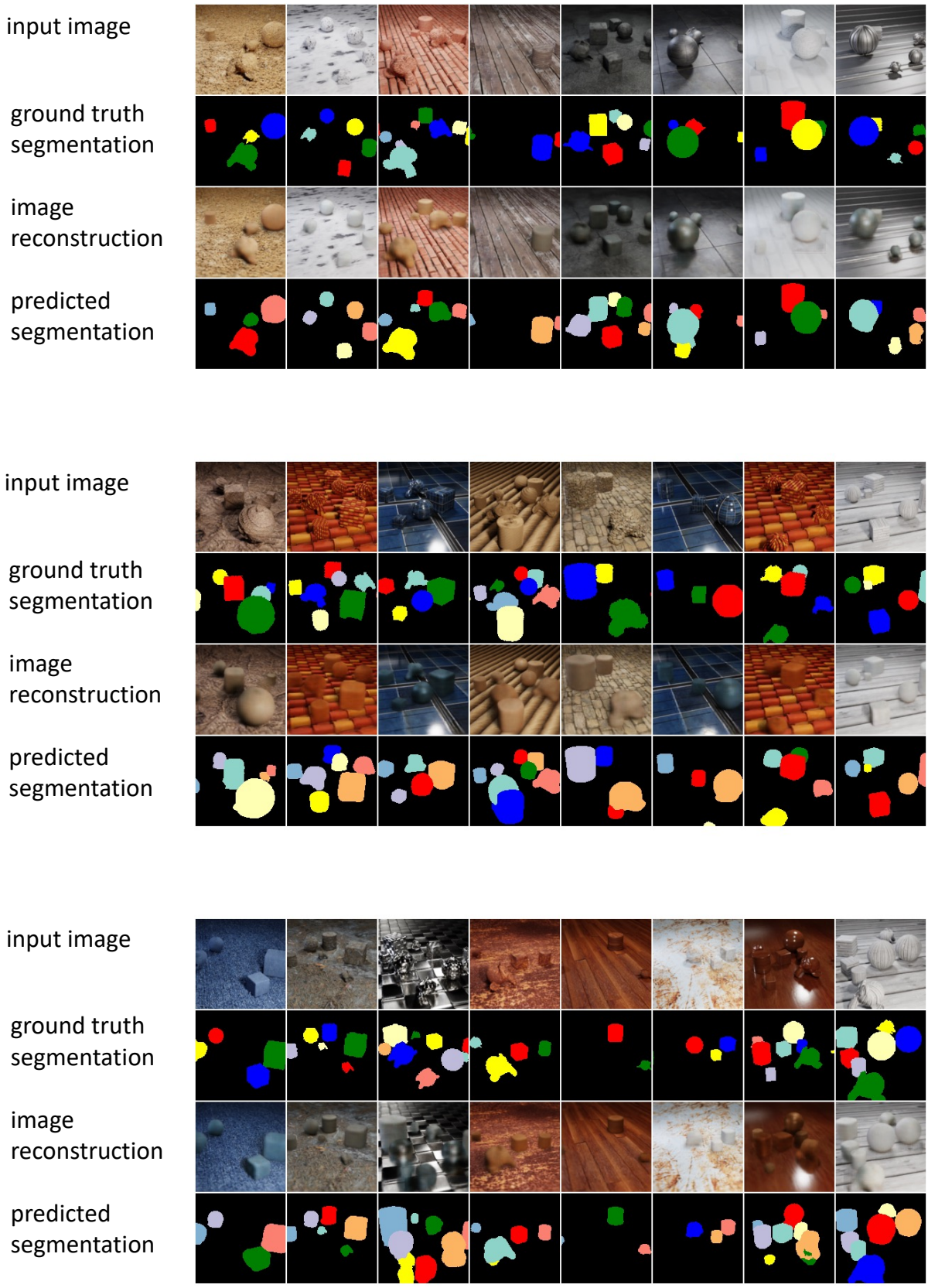

Figure 5. Examples of segmentation predictions on CAMO test dataset using a model trained on CLEVRTEX only

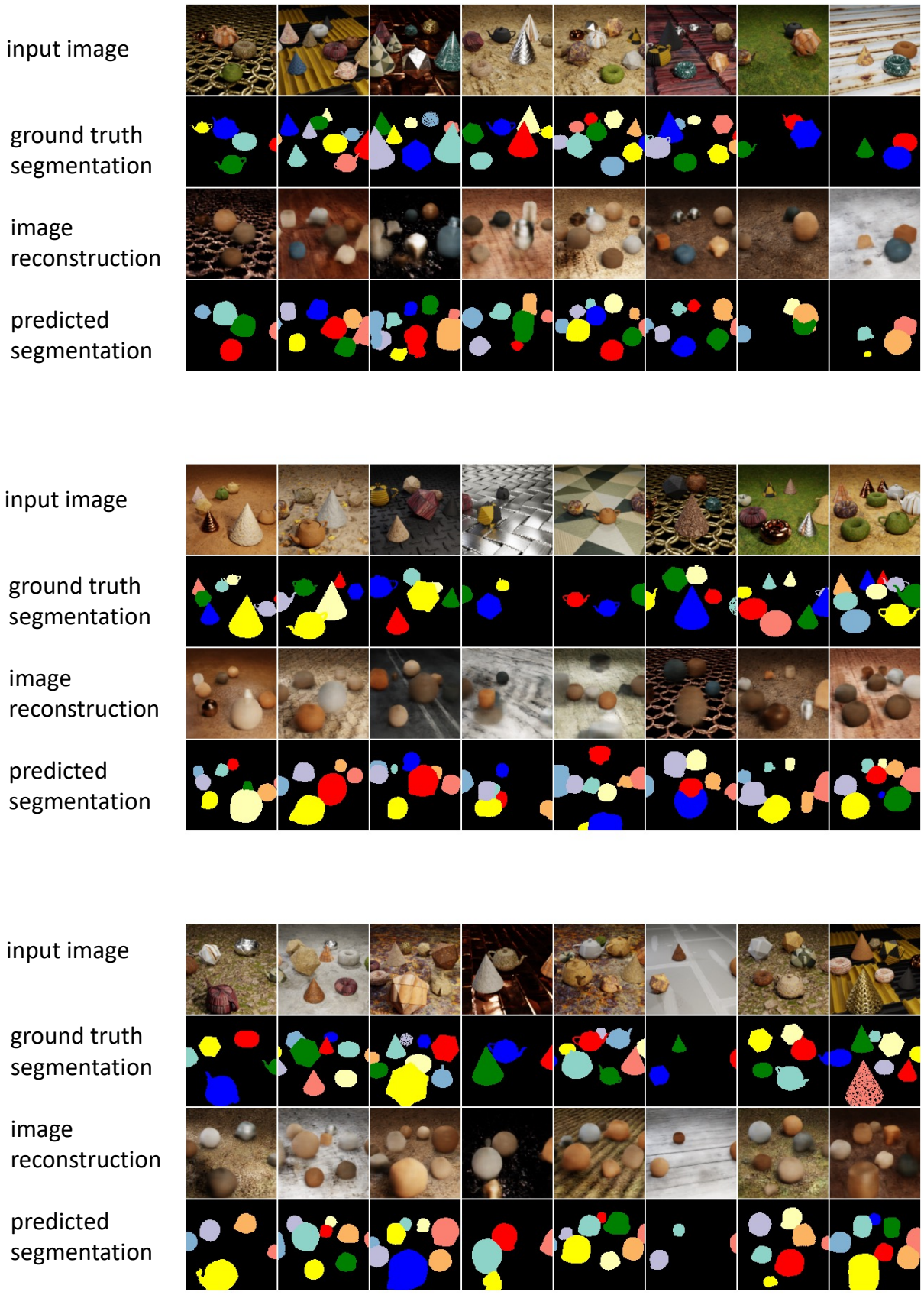

Figure 6. Examples of segmentation predictions on OOD test dataset using a model trained on CLEVRTEX only

Table 1. Hyperparameter values

hyperparameter description	notation	value
Background model pretraining:		
batch size		128
learning rate		2.10^{-3}
number of background model training iterations:		
- datasets with fixed backgrounds (CLEVR)		2500
- datasets with complex backgrounds (CLEVRTEX, ShapeStacks,ObjectsRoom)		500000
Foreground model training:		
batch size		64
learning rate		4.10^{-5}
Adam β_1		0.90
Adam β_2		0.98
Adam ϵ		10^{-9}
number of foreground model training iterations		125000
number of steps of phase 2 (CT scenario)		30000
number of steps of learning rate warmup phase		5000
number of steps of pixel entropy loss weight warmup phase	N_{pixel}	10000
initial value of background activation before training	α_0	e^{11}
dimension of z_{what}	$d_{z_{what}}$	32
pixel entropy loss weight	λ_{pixel}	1.10^{-2}
minimum value of inverse scaling factor	s_{min}	1.3
maximum value of inverse scaling factor	s_{max}	24
dimension of inputs and outputs of transformer encoder	d_T	256
number of heads of transformer encoder layer		8
dimension of feedforward transformer layer		512
number of layers of transformer encoder		6

Algorithm 1: Encoding

Input: input image $\mathbf{X}$

Output: object latents $\{z_k^{what}, x_k, y_k, s_k, \alpha_k\}_{1 \geq k \geq K}$

// feature and attention maps generation

$(\Phi, A_1, .., A_K)$ = SegformerForSemanticSegmentation($\mathbf{X}$)

for $k \leftarrow 1$ **to** K **do**

  $\mathcal{A}_k(i,j)$ = Softmax$(A_k)(i,j) = \frac{e^{A_k(i,j)}}{\sum_{i,j} e^{A_k(i,j)}}$

end

// computation of positions and feature vectors before transformer refinement

for $i \leftarrow 1$ **to** w^*, $j \leftarrow 1$ **to** h^* **do**

  $x(i) = 2\frac{i-1}{w^*-1} - 1$; $y(j) = 2\frac{j-1}{h^*-1} - 1$

end

for $k \leftarrow 1$ **to** K **do**

  $x_k^0 = \sum_{i,j} x(i)\mathcal{A}_k(i,j)$; $y_k^0 = \sum_{i,j} y(j)\mathcal{A}_k(i,j)$

  $\phi_k^0 = \sum_{i,j} \Phi(i,j)\mathcal{A}_k(i,j)$

end

// transformer refinement of positions and feature vectors

$(x_k, y_k, \phi_k)_{1 \geq k \geq K}$ = LinearProjection(TransformerEncoder(LinearEmbedding($(x_k^0, y_k^0, \phi_k^0)_{1 \geq k \geq K}$)))

// latent computations

for $k \leftarrow 1$ **to** K **do**

  x_k = clamp$(x_k, min = -1, max = 1)$; y_k = clamp$(y_k, min = -1, max = 1)$

  $(s_k, \alpha_k, z_k^{what}) = \phi_k$

  $s_k = s_{min} + (s_{max} - s_{min})\sigma(s_k)$

  $\alpha_k = e^{\alpha_k}$

end

Output: $\{z_k^{what}, x_k, y_k, s_k, \alpha_k\}_{1 \geq k \geq K}$

Algorithm 2: Rendering

Input: object latents $\{z_k^{what}, x_k, y_k, s_k, \alpha_k\}_{1 \geq k \geq K}$, background image L_0, background mask $M_0 = 1$, learned background activation α_0

Output: Image reconstruction $\hat{\mathbf{X}}$

// Obtain the object appearance $\mathbf{o}_k$ and segmentation mask $\mathbf{m}_k$

for $k \leftarrow 1$ **to** K **do**

  $\mathbf{o}_k, \mathbf{m}_k$ = GlimpseGenerator($\mathbf{z}_k^{\text{what}}$)

end

// translation and scaling using a spatial transformer network (STN)

for $k \leftarrow 1$ **to** K **do**

  L_k = STN$(\mathbf{o}_k, x_k, y_k, s_k)$

  M_k = STN$(\mathbf{m}_k, x_k, y_k, s_k)$

end

// occlusion computations

for $k \leftarrow 0$ **to** K **do**

  $w_k = \frac{\alpha_k M_k}{\sum_{i=0}^{K} \alpha_i M_i}$

end

// combination of image layers

$\hat{\mathbf{X}} = \sum_{k=0}^{K} w_k L_k$;

Output: $\hat{\mathbf{X}}$

Table 2. glimpse generator architecture

64x64 images

Layer	Size	Ch	Stride	Padding	Norm./Act.
Input	1	$d_{z_{what}}$			
Transp Conv 2×2	2	64	2	0	GroupNorm(4,64) /CELU
Transp Conv 4×4	4	32	2	1	GroupNorm(2,32)/CELU
Transp Conv4×4	8	16	2	1	GroupNorm(1,16)/CELU
Transp Conv 4×4	16	8	2	1	GroupNorm(1,8)/CELU
Transp Conv 4×4	32	4	2	1	
Sigmoid	32	4			

128x128 images

Layer	Size	Ch	Stride	Padding	Norm./Act.
Input	1	$d_{z_{what}}$			
Transp Conv 2×2	2	128	2	0	GroupNorm(8,128) /CELU
Transp Conv 4×4	4	64	2	1	GroupNorm(4,64)/CELU
Transp Conv4×4	8	32	2	1	GroupNorm(2,32)/CELU
Transp Conv 4×4	16	16	2	1	GroupNorm(1,16)/CELU
Transp Conv 4×4	32	8	2	1	GroupNorm(1,8)/CELU
Transp Conv 4×4	64	4	2	1	
Sigmoid	64	4			

Table 3. U-net architecture (ablation study)

Layer	Ch	Stride	Padding	Norm./Act.
Input	3			
Conv 3×3	80	1	1	BatchNorm /CELU
Downsample block	128			
Downsample block	192			
Downsample block	256			
Downsample block	256			
Downsample block	256			
Center block	256			
Upsample block	256			
Upsample block	256			
Upsample block	192			
Upsample block	128			
Upsample block	80			
Conv 3×3 with skip connection	d_Φ	1	1	BatchNorm /CELU
Residual Conv 3×3	d_Φ	1	1	
Conv 1×1	d_Φ	1	1	