Supplemental Material :
Handling Image and Label Resolution Mismatch in Remote Sensing

Scott Workman
DZYNE Technologies

Armin Hadzic
DZYNE Technologies

M. Usman Rafique
Kitware Inc.

1. Dataset Details

We introduced the Low2High dataset, an extension of the Chesapeake [3] dataset that includes a merged label taxonomy, additional auxiliary imagery across the United States, and a new held-out test set from Milwaukee, WI. The spatial coverage of the auxiliary imagery is shown in Figure 1. Example data from the held-out test set is shown in Figure 2. The merged label taxonomy we use for our experiments is outlined in Table 1. A summary of the number of samples per subset of the primary and auxiliary dataset components are included in Table 2 and Table 3, respectively.

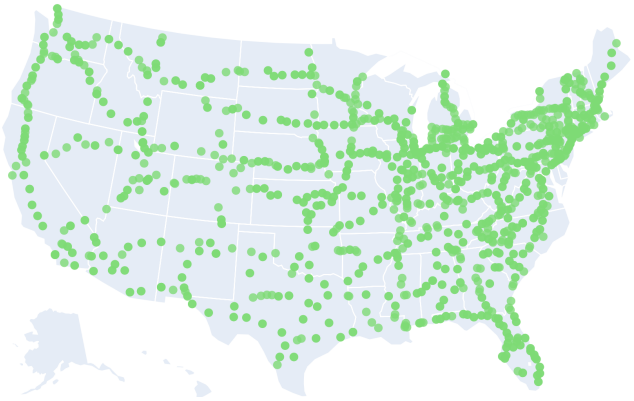

Figure 1: Spatial coverage of the auxiliary imagery in the Low2High dataset.

Figure 2: Examples of the auxiliary imagery in the Low2High dataset.

2. Self-Supervised Learning

We show that self-supervised pretraining on the auxiliary image dataset in the Low2High dataset leads to improved performance when evaluating on regions with drastically different appearance characteristics. We use image reconstruction as a pretraining strategy using the masked autoencoders (MAE) [1] framework, depicted in Figure 3. We used mean squared error between the original image (I) and reconstructed image (R) for a given batch B as the objective function, as follows:

$$\frac{1}{B}\sum_{i}^{B}(I[i] - R[i])^2 \tag{1}$$

In Figure 4 we show a qualitative example of reconstruction for a given overhead image from our test set. The resulting composite (masked region combined with reconstruction) image appears to be a near duplicate of the input image after training for 300 epochs.

3. Additional Results

Figure 5 shows example qualitative results from the region aggregation component of our method. Region aggregation allows the network to output fine-grained predictions, without requiring the target label to be upsampled to the native image resolution. In other words, the network outputs high-resolution predictions which are aggregated (by summing the logits of pixels in the region) and passed to the objective function for comparison with the low-resolution target label. As observed, this process is often not sufficient by itself due to the low quality of the target label (NLCD is 30m per pixel), as well as discrepancies with the high-resolution label (i.e., missing objects). However, our full method (Figure 5, right) is able to leverage region aggregation and adversarial learning to yield high quality predictions. Figure 6 shows qualitative results from the Milwaukee, WI test set, a diverse geographic region not observed during training.

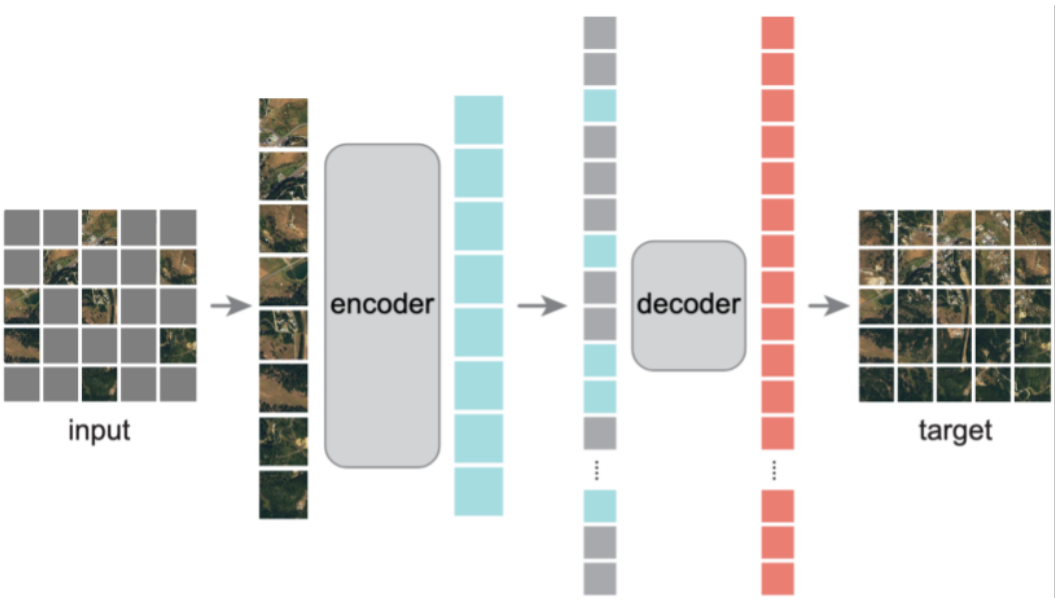

Figure 3: An overview of the masked autoencoders architecture.

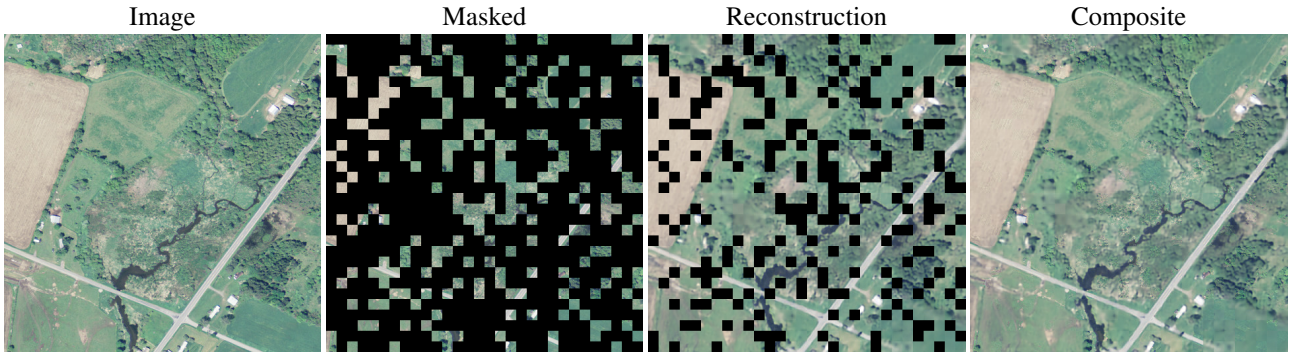

Figure 4: An example reconstruction from masked autoencoders on overhead imagery.

4. Detailed Architecture

We provide detailed architecture descriptions for the components of our network. Table 4 shows the feature encoder used for the overhead imagery (ResNet-18). Table 5 shows our U-Net style decoder used for generating the segmentation output. Finally, Table 6 shows the discriminator architecture.

References

[1] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[2] Andrew Pilant, Keith Endres, Daniel Rosenbaum, and Gillian Gundersen. Us epa enviroatlas meter-scale urban land cover (mulc): 1-m pixel land cover class definitions and guidance. *Remote sensing*, 12(12):1909, 2020.

[3] Caleb Robinson, Le Hou, Kolya Malkin, Rachel Soobitsky, Jacob Czawlytko, Bistra Dilkina, and Nebojsa Jojic. Large scale high-resolution land cover mapping with multi-resolution data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[4] Limin Yang, Suming Jin, Patrick Danielson, Collin Homer, Leila Gass, Stacie M Bender, Adam Case, Catherine Costello, Jon Dewitz, Joyce Fry, et al. A new generation of the united states national land cover database: Requirements, research priorities, design, and implementation strategies. *ISPRS Journal of Photogrammetry and Remote Sensing*, 146:108–123, 2018.

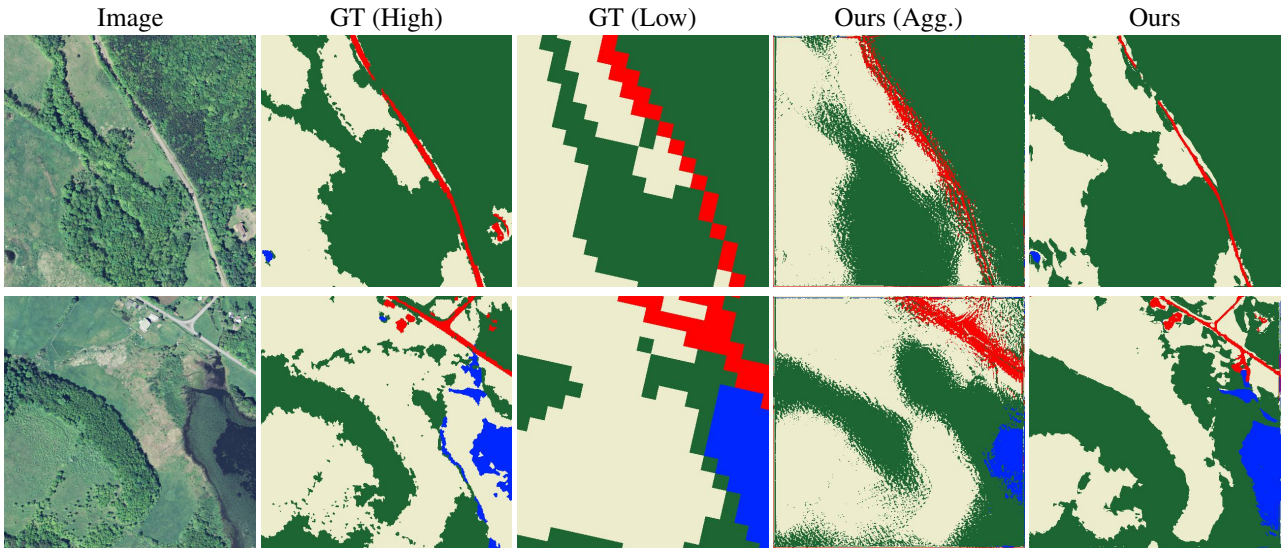

Figure 5: Example qualitative results from our approach.

Figure 6: Example qualitative results for the Milwaukee, WI test set. (top) NAIP image, (middle) remapped ground-truth label from EnviroAtlas, and (bottom) our prediction.

Table 1: Our merged label taxonomy.

Chesapeake [3]	Remapped
Water	Water
Forest	Forest
Field	Field
Barren	Impervious
Impervious (other)	Impervious
Impervious (road)	Impervious
NLCD [4]	
Open Water	Water
Deciduous Forest	Forest
Evergreen Forest	Forest
Mixed Forest	Forest
Dwarf Scrub	Forest
Shrub/Scrub	Forest
Woody Wetlands	Forest
Grassland/Herbaceous	Field
Sedge/Herbaceous	Field
Lichens	Field
Moss	Field
Pasture/Hay	Field
Cultivated Crops	Field
Emergent Herbaceous Wetlands	Field
Developed, Open Space	Impervious
Developed, Low Intensity	Impervious
Developed, Medium Intensity	Impervious
Developed, High Intensity	Impervious
Barren Land (Rock/Sand/Clay)	Impervious
Perennial Ice/Snow	*Ignore*
EPA EnviroAtlas [2]	
Water	Water
Tree	Forest
Shrub	Forest
Orchards	Forest
Wooded Wetlands	Forest
Soil	Field
Grass	Field
Agriculture	Field
Wetlands	Field
Impervious	Impervious

Table 2: Low2High dataset summary (images of size 512×512).

	Train	Validation	Held-out	Test
Delaware	4560	571	571	540
New York	3915	490	490	492
Maryland	4886	611	611	540
Pennsylvania	4433	555	555	540
Virginia	4440	556	556	552
West Virginia	3029	379	379	540
Total	25263	3162	3162	3204

Table 3: Variant of the Low2High dataset, including auxiliary imagery, that is used for our model generalization experiments. Rest of USA excludes Alaska, Rhode Island, Hawaii and the other states listed below.

	Auxiliary	Train	Validation	Held-out	Test
Delaware	18	4560	571	571	540
New York	558	3915	490	490	492
Maryland	117	4886	611	611	540
Pennsylvania	630	4433	555	555	540
Virginia			1159		552
West Virginia	243	3029	379	379	540
Wisconsin			270		3262
Rest of USA	10458				
Total	12024	20823	4035	2606	6466

Table 4: Feature encoder architecture.

Layer (type:depth-idx)	Input Shape	Kernel Shape	Output Shape	Param #
Encoder	[1, 3, 512, 512]	–	[1, 64, 128, 128]	–
— Conv2d: 1-1	[1, 3, 512, 512]	[7, 7]	[1, 64, 256, 256]	9,408
— BatchNorm2d: 1-2	[1, 64, 256, 256]	–	[1, 64, 256, 256]	128
— ReLU: 1-3	[1, 64, 256, 256]	–	[1, 64, 256, 256]	–
— MaxPool2d: 1-4	[1, 64, 256, 256]	3	[1, 64, 128, 128]	–
— Sequential: 1-5	[1, 64, 128, 128]	–	[1, 64, 128, 128]	–
— — BasicBlock: 2-1	[1, 64, 128, 128]	–	[1, 64, 128, 128]	–
— — — Conv2d: 3-1	[1, 64, 128, 128]	[3, 3]	[1, 64, 128, 128]	36,864
— — — BatchNorm2d: 3-2	[1, 64, 128, 128]	–	[1, 64, 128, 128]	128
— — — ReLU: 3-3	[1, 64, 128, 128]	–	[1, 64, 128, 128]	–
— — — Conv2d: 3-4	[1, 64, 128, 128]	[3, 3]	[1, 64, 128, 128]	36,864
— — — BatchNorm2d: 3-5	[1, 64, 128, 128]	–	[1, 64, 128, 128]	128
— — — ReLU: 3-6	[1, 64, 128, 128]	–	[1, 64, 128, 128]	–
— — BasicBlock: 2-2	[1, 64, 128, 128]	–	[1, 64, 128, 128]	–
— — — Conv2d: 3-7	[1, 64, 128, 128]	[3, 3]	[1, 64, 128, 128]	36,864
— — — BatchNorm2d: 3-8	[1, 64, 128, 128]	–	[1, 64, 128, 128]	128
— — — ReLU: 3-9	[1, 64, 128, 128]	–	[1, 64, 128, 128]	–
— — — Conv2d: 3-10	[1, 64, 128, 128]	[3, 3]	[1, 64, 128, 128]	36,864
— — — BatchNorm2d: 3-11	[1, 64, 128, 128]	–	[1, 64, 128, 128]	128
— — — ReLU: 3-12	[1, 64, 128, 128]	–	[1, 64, 128, 128]	–
— Sequential: 1-6	[1, 64, 128, 128]	–	[1, 128, 64, 64]	–
— — BasicBlock: 2-3	[1, 64, 128, 128]	–	[1, 128, 64, 64]	–
— — — Conv2d: 3-13	[1, 64, 128, 128]	[3, 3]	[1, 128, 64, 64]	73,728
— — — BatchNorm2d: 3-14	[1, 128, 64, 64]	–	[1, 128, 64, 64]	256
— — — ReLU: 3-15	[1, 128, 64, 64]	–	[1, 128, 64, 64]	–
— — — Conv2d: 3-16	[1, 128, 64, 64]	[3, 3]	[1, 128, 64, 64]	147,456
— — — BatchNorm2d: 3-17	[1, 128, 64, 64]	–	[1, 128, 64, 64]	256
— — — Sequential: 3-18	[1, 64, 128, 128]	–	[1, 128, 64, 64]	8,448
— — — ReLU: 3-19	[1, 128, 64, 64]	–	[1, 128, 64, 64]	–
— — BasicBlock: 2-4	[1, 128, 64, 64]	–	[1, 128, 64, 64]	–
— — — Conv2d: 3-20	[1, 128, 64, 64]	[3, 3]	[1, 128, 64, 64]	147,456
— — — BatchNorm2d: 3-21	[1, 128, 64, 64]	–	[1, 128, 64, 64]	256
— — — ReLU: 3-22	[1, 128, 64, 64]	–	[1, 128, 64, 64]	–
— — — Conv2d: 3-23	[1, 128, 64, 64]	[3, 3]	[1, 128, 64, 64]	147,456
— — — BatchNorm2d: 3-24	[1, 128, 64, 64]	–	[1, 128, 64, 64]	256
— — — ReLU: 3-25	[1, 128, 64, 64]	–	[1, 128, 64, 64]	–
— Sequential: 1-7	[1, 128, 64, 64]	–	[1, 256, 32, 32]	–
— — BasicBlock: 2-5	[1, 128, 64, 64]	–	[1, 256, 32, 32]	–
— — — Conv2d: 3-26	[1, 128, 64, 64]	[3, 3]	[1, 256, 32, 32]	294,912
— — — BatchNorm2d: 3-27	[1, 256, 32, 32]	–	[1, 256, 32, 32]	512
— — — ReLU: 3-28	[1, 256, 32, 32]	–	[1, 256, 32, 32]	–
— — — Conv2d: 3-29	[1, 256, 32, 32]	[3, 3]	[1, 256, 32, 32]	589,824
— — — BatchNorm2d: 3-30	[1, 256, 32, 32]	–	[1, 256, 32, 32]	512
— — — Sequential: 3-31	[1, 128, 64, 64]	–	[1, 256, 32, 32]	33,280
— — — ReLU: 3-32	[1, 256, 32, 32]	–	[1, 256, 32, 32]	–
— — BasicBlock: 2-6	[1, 256, 32, 32]	–	[1, 256, 32, 32]	–
— — — Conv2d: 3-33	[1, 256, 32, 32]	[3, 3]	[1, 256, 32, 32]	589,824
— — — BatchNorm2d: 3-34	[1, 256, 32, 32]	–	[1, 256, 32, 32]	512
— — — ReLU: 3-35	[1, 256, 32, 32]	–	[1, 256, 32, 32]	–
— — — Conv2d: 3-36	[1, 256, 32, 32]	[3, 3]	[1, 256, 32, 32]	589,824
— — — BatchNorm2d: 3-37	[1, 256, 32, 32]	–	[1, 256, 32, 32]	512
— — — ReLU: 3-38	[1, 256, 32, 32]	–	[1, 256, 32, 32]	–
— Sequential: 1-8	[1, 256, 32, 32]	–	[1, 512, 16, 16]	–
— — BasicBlock: 2-7	[1, 256, 32, 32]	–	[1, 512, 16, 16]	–
— — — Conv2d: 3-39	[1, 256, 32, 32]	[3, 3]	[1, 512, 16, 16]	1,179,648
— — — BatchNorm2d: 3-40	[1, 512, 16, 16]	–	[1, 512, 16, 16]	1,024
— — — ReLU: 3-41	[1, 512, 16, 16]	–	[1, 512, 16, 16]	–
— — — Conv2d: 3-42	[1, 512, 16, 16]	[3, 3]	[1, 512, 16, 16]	2,359,296
— — — BatchNorm2d: 3-43	[1, 512, 16, 16]	–	[1, 512, 16, 16]	1,024
— — — Sequential: 3-44	[1, 256, 32, 32]	–	[1, 512, 16, 16]	132,096
— — — ReLU: 3-45	[1, 512, 16, 16]	–	[1, 512, 16, 16]	–
— — BasicBlock: 2-8	[1, 512, 16, 16]	–	[1, 512, 16, 16]	–
— — — Conv2d: 3-46	[1, 512, 16, 16]	[3, 3]	[1, 512, 16, 16]	2,359,296
— — — BatchNorm2d: 3-47	[1, 512, 16, 16]	–	[1, 512, 16, 16]	1,024
— — — ReLU: 3-48	[1, 512, 16, 16]	–	[1, 512, 16, 16]	–
— — — Conv2d: 3-49	[1, 512, 16, 16]	[3, 3]	[1, 512, 16, 16]	2,359,296
— — — BatchNorm2d: 3-50	[1, 512, 16, 16]	–	[1, 512, 16, 16]	1,024
— — — ReLU: 3-51	[1, 512, 16, 16]	–	[1, 512, 16, 16]	–

Table 5: Decoder architecture.

Layer (type:depth-idx)	Input Shape	Kernel Shape	Output Shape	Param #
Decoder	[1, 64, 256, 256]	–	[1, 12, 512, 512]	–
— Upsample: 1-1	[1, 512, 16, 16]	–	[1, 512, 32, 32]	–
— Sequential: 1-2	[1, 768, 32, 32]	–	[1, 256, 32, 32]	–
— — Conv2d: 2-1	[1, 768, 32, 32]	[3, 3]	[1, 256, 32, 32]	1,769,728
— — ReLU: 2-2	[1, 256, 32, 32]	–	[1, 256, 32, 32]	–
— — Conv2d: 2-3	[1, 256, 32, 32]	[3, 3]	[1, 256, 32, 32]	590,080
— — ReLU: 2-4	[1, 256, 32, 32]	–	[1, 256, 32, 32]	–
— Upsample: 1-3	[1, 256, 32, 32]	–	[1, 256, 64, 64]	–
— Sequential: 1-4	[1, 384, 64, 64]	–	[1, 128, 64, 64]	–
— — Conv2d: 2-5	[1, 384, 64, 64]	[3, 3]	[1, 128, 64, 64]	442,496
— — ReLU: 2-6	[1, 128, 64, 64]	–	[1, 128, 64, 64]	–
— — Conv2d: 2-7	[1, 128, 64, 64]	[3, 3]	[1, 128, 64, 64]	147,584
— — ReLU: 2-8	[1, 128, 64, 64]	–	[1, 128, 64, 64]	–
— Upsample: 1-5	[1, 128, 64, 64]	–	[1, 128, 128, 128]	–
— Sequential: 1-6	[1, 192, 128, 128]	–	[1, 64, 128, 128]	–
— — Conv2d: 2-9	[1, 192, 128, 128]	[3, 3]	[1, 64, 128, 128]	110,656
— — ReLU: 2-10	[1, 64, 128, 128]	–	[1, 64, 128, 128]	–
— — Conv2d: 2-11	[1, 64, 128, 128]	[3, 3]	[1, 64, 128, 128]	36,928
— — ReLU: 2-12	[1, 64, 128, 128]	–	[1, 64, 128, 128]	–
— Upsample: 1-7	[1, 64, 128, 128]	–	[1, 64, 256, 256]	–
— Sequential: 1-8	[1, 128, 256, 256]	–	[1, 64, 256, 256]	–
— — Conv2d: 2-13	[1, 128, 256, 256]	[3, 3]	[1, 64, 256, 256]	73,792
— — ReLU: 2-14	[1, 64, 256, 256]	–	[1, 64, 256, 256]	–
— — Conv2d: 2-15	[1, 64, 256, 256]	[3, 3]	[1, 64, 256, 256]	36,928
— — ReLU: 2-16	[1, 64, 256, 256]	–	[1, 64, 256, 256]	–
— Upsample: 1-9	[1, 64, 256, 256]	–	[1, 64, 512, 512]	–
— Conv2d: 1-10	[1, 64, 512, 512]	[3, 3]	[1, 12, 512, 512]	6,924

Table 6: Discriminator architecture.

Layer (type:depth-idx)	Input Shape	Kernel Shape	Output Shape	Param #
Discriminator	[1, 1, 256, 256]	–	–	–
— ModuleList: 1-1	–	–	–	–
— — DiscriminatorBlock: 2-1	[1, 1, 256, 256]	–	[1, 32, 128, 128]	–
— — — Conv2d: 3-1	[1, 1, 256, 256]	[1, 1]	[1, 32, 128, 128]	64
— — — Sequential: 3-2	[1, 1, 256, 256]	–	[1, 32, 256, 256]	9,568
— — — Sequential: 3-3	[1, 32, 256, 256]	–	[1, 32, 128, 128]	9,248
— — DiscriminatorBlock: 2-2	[1, 32, 128, 128]	–	[1, 64, 64, 64]	–
— — — Conv2d: 3-4	[1, 32, 128, 128]	[1, 1]	[1, 64, 64, 64]	2,112
— — — Sequential: 3-5	[1, 32, 128, 128]	–	[1, 64, 128, 128]	55,424
— — — Sequential: 3-6	[1, 64, 128, 128]	–	[1, 64, 64, 64]	36,928
— — DiscriminatorBlock: 2-3	[1, 64, 64, 64]	–	[1, 128, 32, 32]	–
— — — Conv2d: 3-7	[1, 64, 64, 64]	[1, 1]	[1, 128, 32, 32]	8,320
— — — Sequential: 3-8	[1, 64, 64, 64]	–	[1, 128, 64, 64]	221,440
— — — Sequential: 3-9	[1, 128, 64, 64]	–	[1, 128, 32, 32]	147,584
— — DiscriminatorBlock: 2-4	[1, 128, 32, 32]	–	[1, 128, 16, 16]	–
— — — Conv2d: 3-10	[1, 128, 32, 32]	[1, 1]	[1, 128, 16, 16]	16,512
— — — Sequential: 3-11	[1, 128, 32, 32]	–	[1, 128, 32, 32]	295,168
— — — Sequential: 3-12	[1, 128, 32, 32]	–	[1, 128, 16, 16]	147,584
— — DiscriminatorBlock: 2-5	[1, 128, 16, 16]	–	[1, 128, 8, 8]	–
— — — Conv2d: 3-13	[1, 128, 16, 16]	[1, 1]	[1, 128, 8, 8]	16,512
— — — Sequential: 3-14	[1, 128, 16, 16]	–	[1, 128, 16, 16]	295,168
— — — Sequential: 3-15	[1, 128, 16, 16]	–	[1, 128, 8, 8]	147,584
— — DiscriminatorBlock: 2-6	[1, 128, 8, 8]	–	[1, 128, 4, 4]	–
— — — Conv2d: 3-16	[1, 128, 8, 8]	[1, 1]	[1, 128, 4, 4]	16,512
— — — Sequential: 3-17	[1, 128, 8, 8]	–	[1, 128, 8, 8]	295,168
— — — Sequential: 3-18	[1, 128, 8, 8]	–	[1, 128, 4, 4]	147,584
— — DiscriminatorBlock: 2-7	[1, 128, 4, 4]	–	[1, 128, 2, 2]	–
— — — Conv2d: 3-19	[1, 128, 4, 4]	[1, 1]	[1, 128, 2, 2]	16,512
— — — Sequential: 3-20	[1, 128, 4, 4]	–	[1, 128, 4, 4]	295,168
— — — Sequential: 3-21	[1, 128, 4, 4]	–	[1, 128, 2, 2]	147,584
— — DiscriminatorBlock: 2-8	[1, 128, 2, 2]	–	[1, 128, 2, 2]	–
— — — Conv2d: 3-22	[1, 128, 2, 2]	[1, 1]	[1, 128, 2, 2]	16,512
— — — Sequential: 3-23	[1, 128, 2, 2]	–	[1, 128, 2, 2]	295,168
— ModuleList: 1-2	–	–	–	–
— Conv2d: 1-3	[1, 128, 2, 2]	[3, 3]	[1, 128, 2, 2]	147,584
— Flatten: 1-4	[1, 128, 2, 2]	–	[1, 512]	–
— Linear: 1-5	[1, 512]	–	[1, 1]	513