DLCV Spring 2018 HW3

d05921018 林家慶

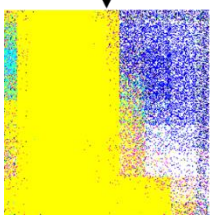
1. (5%) Print the network architecture of your VGG16-FCN32s model.

input image
256x256x3

[NOTE] satellite images are resized into 256x256 in order to benefit from larger batch size (up to 32)

conv1_1	3x3x64, BN, RELU
conv1_2	3x3x64, BN, RELU
pool1	2x2, stride=2
conv2_1	3x3x128, BN, RELU
conv2_2	3x3x128, BN, RELU
pool2	2x2, stride=2
conv3_1	3x3x256, BN, RELU
conv3_2	3x3x256, BN, RELU
conv3_2	3x3x256, BN, RELU
pool3	2x2, stride=2
conv4_1	3x3x512, BN, RELU
conv4_2	3x3x512, BN, RELU
conv4_3	3x3x512, BN, RELU
pool4	2x2, stride=2
conv5_1	3x3x512, BN, RELU
conv5_2	3x3x512, BN, RELU
conv5_3	3x3x512, BN, RELU
pool5	2x2, stride=2
conv6	**3x3x4096, BN, RELU**
conv7	**1x1x4096, BN, RELU**
conv8	**1x1x7 (no BN, no activation function)**
transpose convolution	**64x64x7, stride=32, (no bias term)**

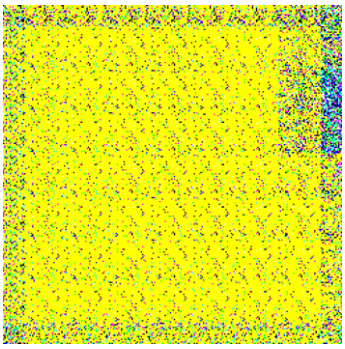
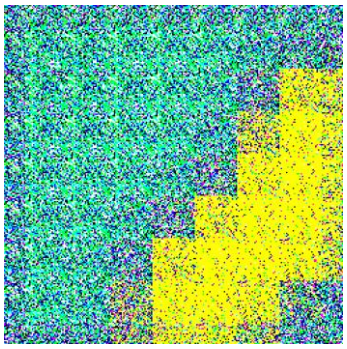
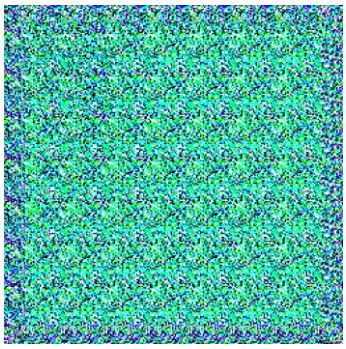
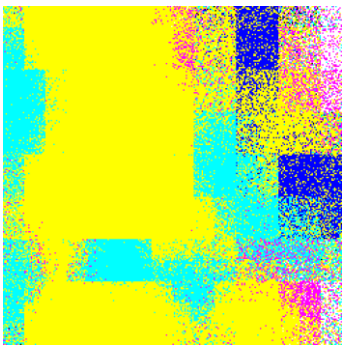
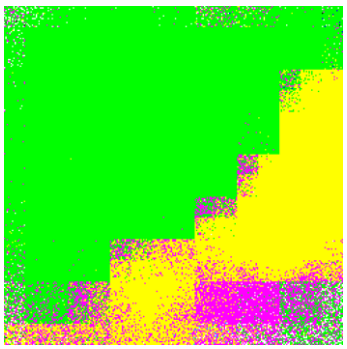
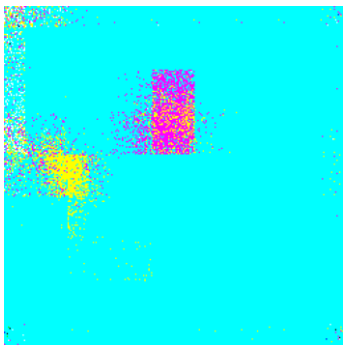
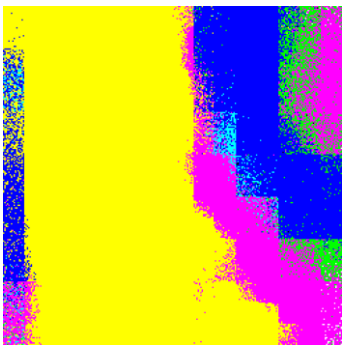
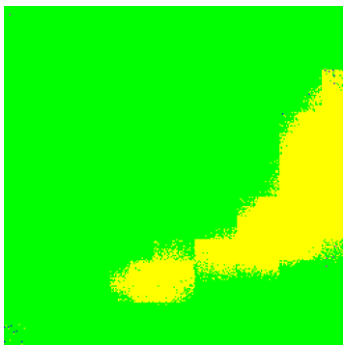
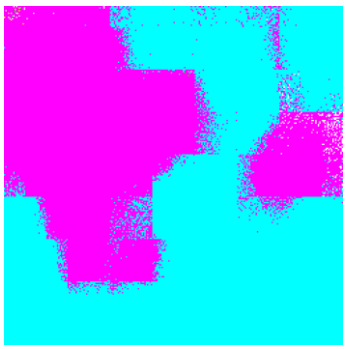
VGG16



output mask
256x256x1

[NOTE] the final output masks will be resized into 512x512 by applying numpy.repeat() function in both H and W dimensions

2. (10%) Show the predicted segmentation mask of “validation/0008_sat.jpg”, “validation/0097_sat.jpg”, and “validation/0107_sat.jpg” during the early, middle, and the final stage during the training stage.

	0008	0097	0107
Satellite			
Epoch 1			
Epoch 10			
Epoch 20			

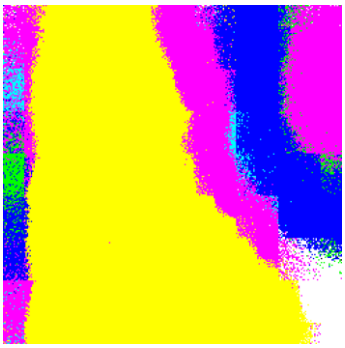
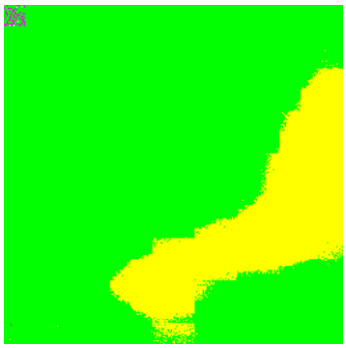
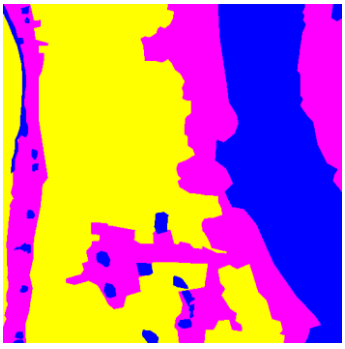
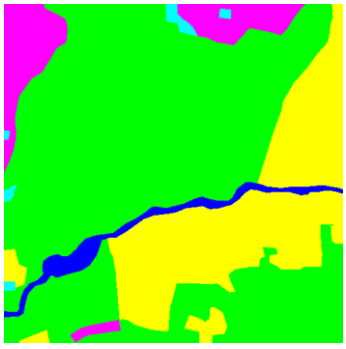
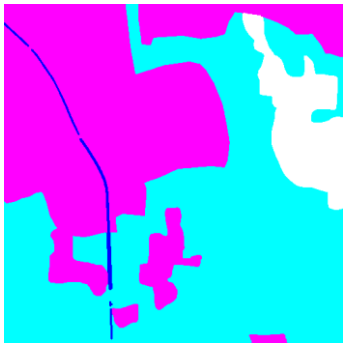
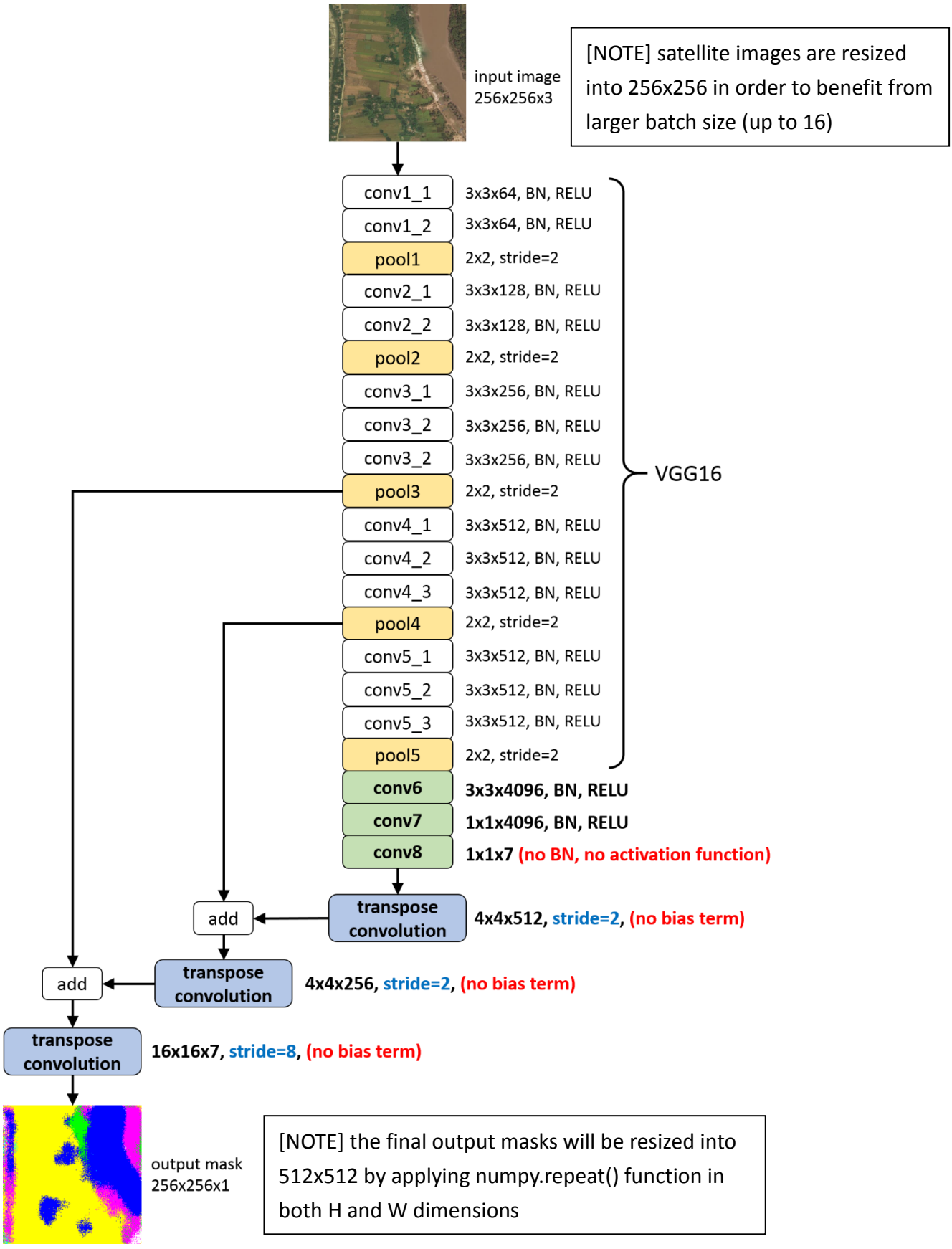
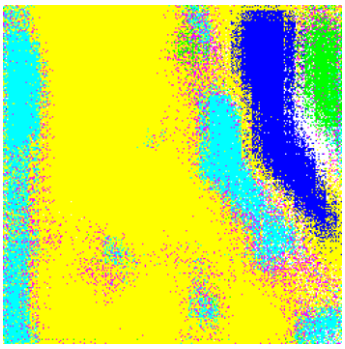
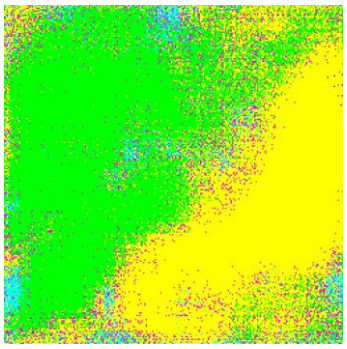
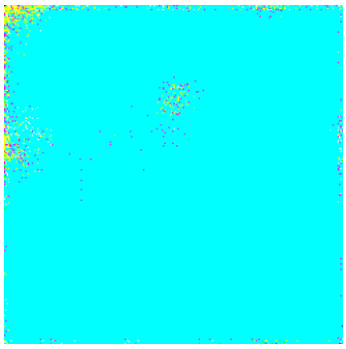
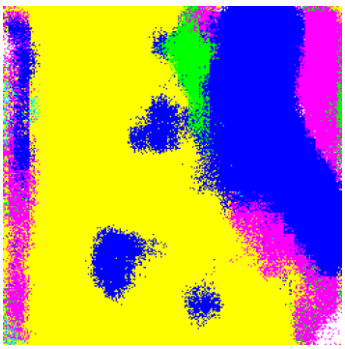
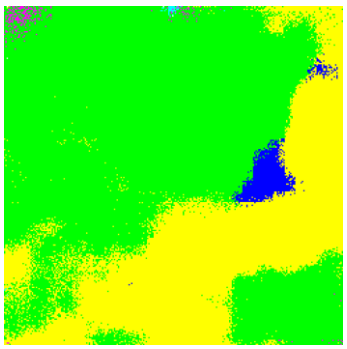
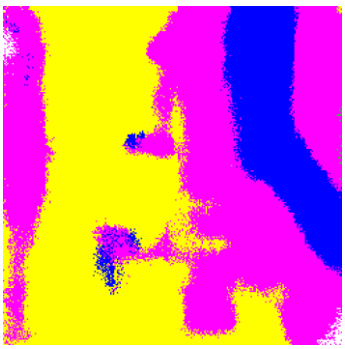
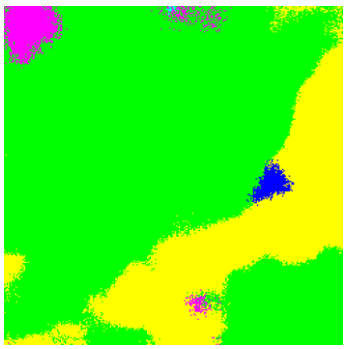
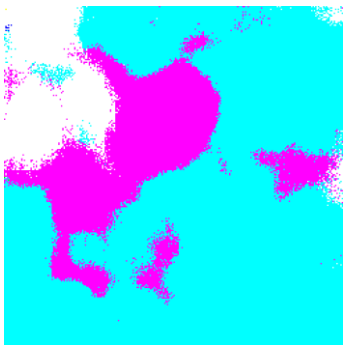
Epoch 30 (Final)			
Ground Truth			

Table 1, predicted segmentation masks of **VGG16_FCN32s** during training process

3. (15%) Implement an improved model which performs better than your baseline model. Print the network architecture of this model.



4. (10%) Show the predicted segmentation mask of “validation/0008_sat.jpg”, “validation/0097_sat.jpg”, “validation/0107_sat.jpg” during the early, middle, and the final stage during the training process of this improved model.

	0008	0097	0107
Satellite			
Epoch 1			
Epoch 10			
Epoch 20			

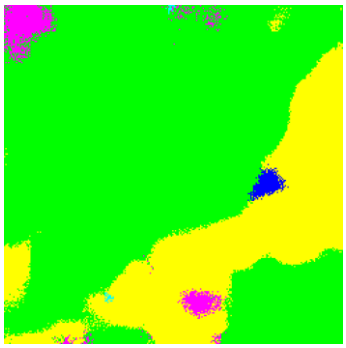
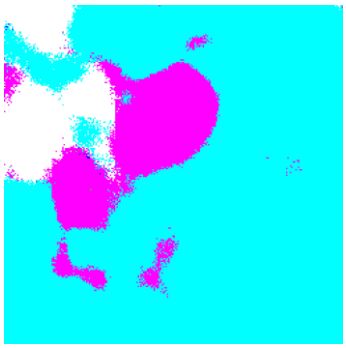
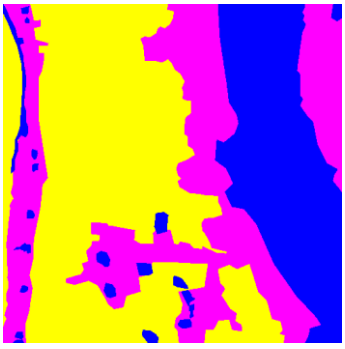
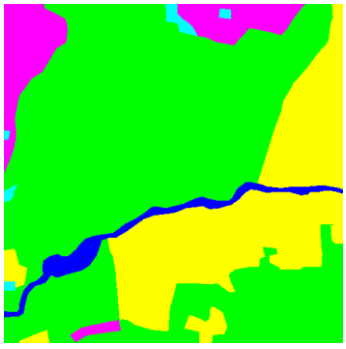
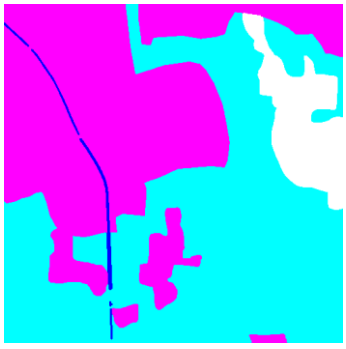
Epoch 30 (Final)			
Ground Truth			

Table 2, predicted segmentation masks of **VGG16_FCN8s** during training process

5. (15%) Report mIoU score of both models on the validation set. Discuss the reason why the improved model performs better than the baseline one. You may conduct some experiments and show some evidences to support your reasoning.

As can be seen from the predicted masks during training processes, **VGG16_FCN32s** produces more coarse masks due to its large upsampling rate (transpose convolution with stride 32), especially in the early training stage (e.g., epoch 1 and 10). As for **VGG16_FCN8s**, it produces masks with finer granularities due to its smaller upsampling rate at the final stage (transpose convolution with stride 8), even in early training stages (e.g., epoch 1).

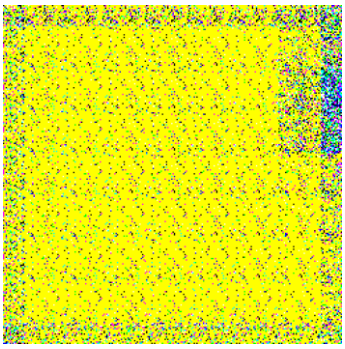
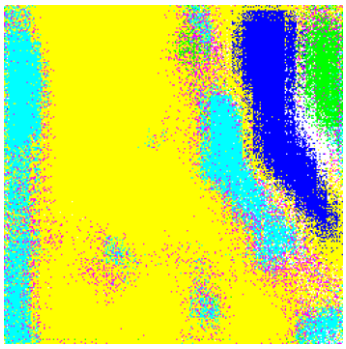
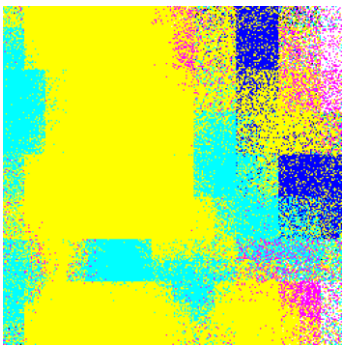
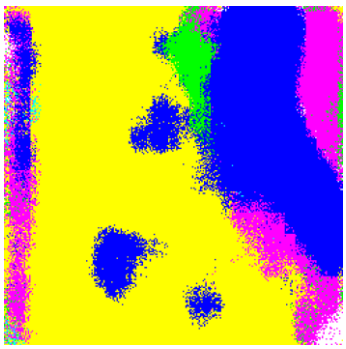
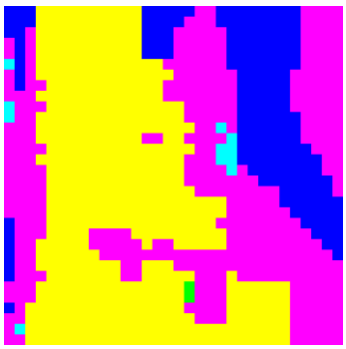
I also implemented **VGG16_dilated**: the frontend module of dilated convolutions proposed in [2]. The network architecture can be found in Appendix A. By default, **VGG16_dilated** produces masks with size 1/8 of the input image (i.e., 32x32). Surprisingly, even if I didn't implement interpolation nor any further enhancement network (for example, the context module), its IOU performance are quite good, as can be seen from the following tables.

	VGG16_FCN32s	**VGG16_FCN8s**	**VGG16_dilated**
Epoch 1	0.17940	0.41786	0.50013
Epoch 10	0.56510	0.58825	0.62551
Epoch 20	0.64557	0.69053	0.67623
Epoch 30	**0.67754**	**0.70554**	**0.69120**

Table 3, mean_IOU during training process

	VGG16_FCN32s	VGG16_FCN8s	VGG16_dilated
class #0	0.69703	0.73903	0.73291
class #1	0.87118	0.88106	0.87298
class #2	0.30488	0.37889	0.35306
class #3	0.80328	0.80991	0.79046
class #4	0.70292	0.73472	0.71138
class #5	0.68593	0.68962	0.68637
mean_iou	**0.67754**	**0.70554**	**0.69120**

Table 4, detail of final (Epoch 30) performances

	VGG16_FCN32s	VGG16_FCN8s	VGG16_dilated
Satellite			
Epoch 1			
Epoch 10			

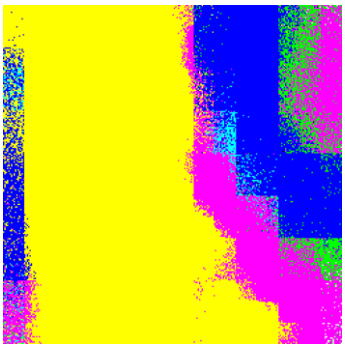
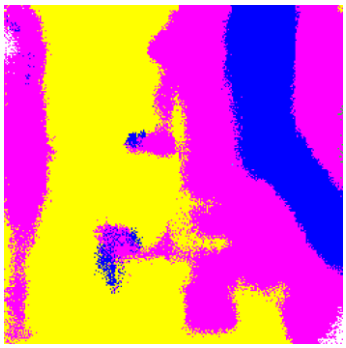
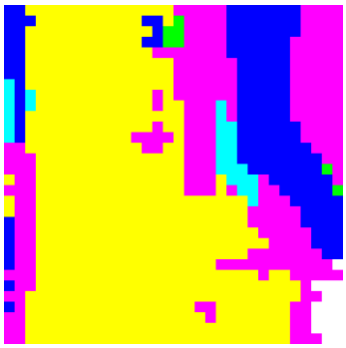
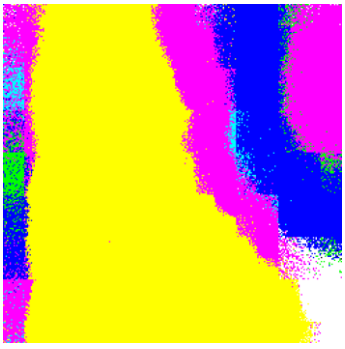
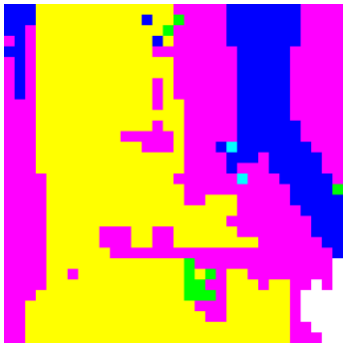
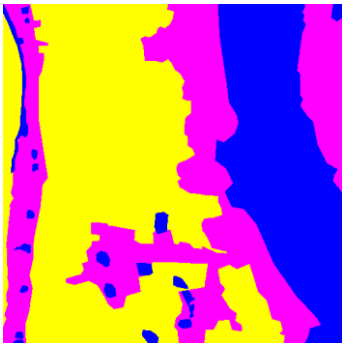
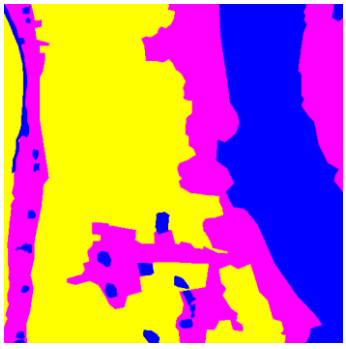
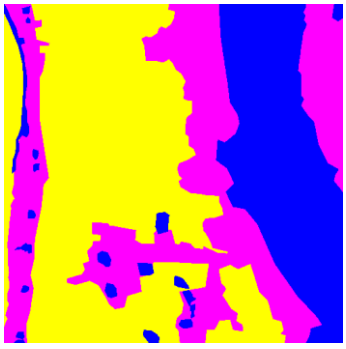
Epoch 20			
Epoch 30 (Final)			
Ground Truth			

Table 5, predicted “0008_mask.png” of 3 implemented models during training process

The main reason of the good performance of **VGG16_dilated** is its dilated convolution, with which the final output masks can benefit from larger receptive fields (as done by pooling layers in FCN) without severe downsampling (as caused by pooling layers in FCN).

Appendix A: network architecture of VGG16_dilated

input image
256x256x3

[NOTE] satellite images are resized into 256x256 in order to benefit from larger batch size (up to 16)

conv1_1	3x3x64, BN, RELU
conv1_2	3x3x64, BN, RELU
pool1	2x2, stride=2
conv2_1	3x3x128, BN, RELU
conv2_2	3x3x128, BN, RELU
pool2	2x2, stride=2
conv3_1	3x3x256, BN, RELU
conv3_2	3x3x256, BN, RELU
conv3_2	3x3x256, BN, RELU
pool3	2x2, stride=2
conv4_1	3x3x512, BN, RELU
conv4_2	3x3x512, BN, RELU
conv4_3	3x3x512, BN, RELU
	(no pooling)
conv5_1	3x3x512, **dilated rate = 2**, BN, RELU
conv5_2	3x3x512, **dilated rate = 2**, BN, RELU
conv5_3	3x3x512, **dilated rate = 2**, BN, RELU
	(no pooling)
conv6	**3x3x4096, dilated rate = 4, BN, RELU**
conv7	**1x1x4096, BN, RELU**
conv8	**1x1x7 (no BN, no activation function)**

output mask
32x32x1

[NOTE] the final output masks will be resized into 512x512 by applying numpy.repeat() function in both H and W dimensions

Appendix B: some implementation details

As mentioned in all network architectures, I used input images with smaller sizes (256x256) and a large batch size (32 for **VGG16_FCN32s**; 16 for **VGG16_FCN8s** and **VGG16_dilated**). Also, I used the Adam optimizer in tensorflow with a relatively large learning rate (5e-5) and hand-crafted early-stopping trick (monitoring validation loss with patience=10 or 5) to control the training flow. This large learning rate leads to fast training (around 20 epochs to achieve the baseline, see Table 3) but may sometimes cause loss='nan'. It is suggested to run the training script multiple times, or just use a smaller learning rate (e.g., 1e-6) with larger number of epochs (e.g., 100).

I also tried different kernel sizes and different number of kernels of the 2 convolutional layers right after VGG16 (conv6 and conv7), and found that a smaller kernel size in conv6 (3x3 instead of 7x7 as used by the authors of the FCN paper [1]) leads to better performance, as can be seen from the following table:

Reference

[1] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation". In *CVPR*, 2015.

[2] Fisher Yu and Vladlen Koltun, "Multi-scale context aggregation by dilated convolutions". In *ICLR*, 2016.

(Bonus)

$$x(\underline{z}^{(n)};\underline{w}) = \frac{1}{1+e^{-\underline{w}^T\underline{z}^{(n)}}} \Rightarrow 1-x(\underline{z}^{(n)};\underline{w}) = \frac{e^{-\underline{w}^T\underline{z}^{(n)}}}{1+e^{-\underline{w}^T\underline{z}^{(n)}}}$$

$$\text{Also, } \frac{d\,x(\underline{z}^{(n)};\underline{w})}{d\underline{w}} = \frac{e^{-\underline{w}^T\underline{z}^{(n)}}\,\underline{z}^{(n)}}{\left[1+e^{-\underline{w}^T\underline{z}^{(n)}}\right]^2}$$

For each n, let $G^{(n)}(\underline{w}) = t^{(n)} \ln x(\underline{z}^{(n)};\underline{w}) + (1-t^{(n)}) \ln(1-x(\underline{z}^{(n)};\underline{w}))$

$$\Rightarrow \frac{d}{d\underline{w}} G^{(n)}(\underline{w}) = \frac{d\,G^{(n)}(\underline{w})}{d\,x(\underline{z}^{(n)};\underline{w})} \cdot \frac{d\,x(\underline{z}^{(n)};\underline{w})}{d\underline{w}}$$

$$= \left(\frac{t^{(n)}}{x(\underline{z}^{(n)};\underline{w})} + \frac{t^{(n)}-1}{1-x(\underline{z}^{(n)};\underline{w})}\right) \cdot \frac{d\,x(\underline{z}^{(n)};\underline{w})}{d\underline{w}}$$

$$= \left[t^{(n)}\left(1+e^{-\underline{w}^T\underline{z}^{(n)}}\right) + \frac{(t^{(n)}-1)\left(1+e^{-\underline{w}^T\underline{z}^{(n)}}\right)}{e^{-\underline{w}^T\underline{z}^{(n)}}}\right] \cdot \frac{e^{-\underline{w}^T\underline{z}^{(n)}}\,\underline{z}^{(n)}}{\left[1+e^{-\underline{w}^T\underline{z}^{(n)}}\right]^2}$$

$$= \left[t^{(n)}\left(\frac{e^{-\underline{w}^T\underline{z}^{(n)}}}{1+e^{-\underline{w}^T\underline{z}^{(n)}}}\right) + (t^{(n)}-1)\left(\frac{1}{1+e^{-\underline{w}^T\underline{z}^{(n)}}}\right)\right] \underline{z}^{(n)}$$

$$= \left[t^{(n)}\left(1-x(\underline{z}^{(n)};\underline{w})\right) + (t^{(n)}-1)\,x(\underline{z}^{(n)};\underline{w})\right] \underline{z}^{(n)}$$

$$= \left[t^{(n)} - x(\underline{z}^{(n)};\underline{w})\right] \underline{z}^{(n)}$$

$$\Rightarrow \frac{d}{d\underline{w}} G(\underline{w}) = -\sum_n \left(\frac{d}{d\underline{w}} G^{(n)}(\underline{w})\right) = -\sum_n \left(t^{(n)} - x(\underline{z}^{(n)};\underline{w})\right) \underline{z}^{(n)} \quad \#$$