Supplementary material: Mix and match networks: encoder-decoder alignment for zero-pair image translation

Yaxing Wang, Joost van de Weijer, Luis Herranz
Computer Vision Center, Universitat Autònoma de Barcelona
Barcelona, Spain
{wang,joost,lherranz}@cvc.uab.es

1. Network architecture

Table 3 shows the architecture (convolutional and pooling layers) of the encoders used in the cross-modal experiment. Tables 4 and 1 show the corresponding decoders. Table 2 shows the discriminator used for RGB. . Every convolutional layer of the encoders, decoders and the discriminator is followed by a batch normalization layer and a ReLU layer (LeakyReLU for the discriminator). The only exception is the RGB encoder, which is is initialized with weights from the VGG16 model[1] and does not use batch normalization.

layer	Input →Output	Kernel, stride
conv1	[6,8,8,512] → $[6, 16, 16, 512]$	[3, 3], 1
conv2	[6,16,16,512] → $[6, 32, 32, 256]$	[3, 3], 1
conv3	[6,32,32,256] → $[6, 64, 64, 128]$	[3, 3], 1
conv4	[6,64,64,128] → $[6, 128, 128, 64]$	[3, 3], 1
conv5	[6,128,128,64]→ $[6, 256, 256, 3]$	[3, 3], 1

Table 1: Convolutional and pooling layers of the RGB decoder.

layer	Input →Output	Kernel, stride
deconv1	$[6, 256, 256, 3] \rightarrow [6, 128, 128, 64]$	[5, 5], 2
deconv2	$[6, 128, 128, 64] \rightarrow [6, 64, 64, 128]$	[5, 5], 2
deconv3	$[6, 64, 64, 128] \rightarrow [6, 32, 32, 256]$	[5,5], 2
deconv4	$[6, 32, 32, 256] \rightarrow [6, 16, 16, 512]$	[5,5], 2

Table 2: RGB discriminator.

2. Multimodal fusion

Figure 1 shows the performance for different values of α for multimodal semantic segmentation. It also compares the performance when the semantic segmentation decoder uses the pooling indices from the depth encoder instead of the ones from the RGB encoder.

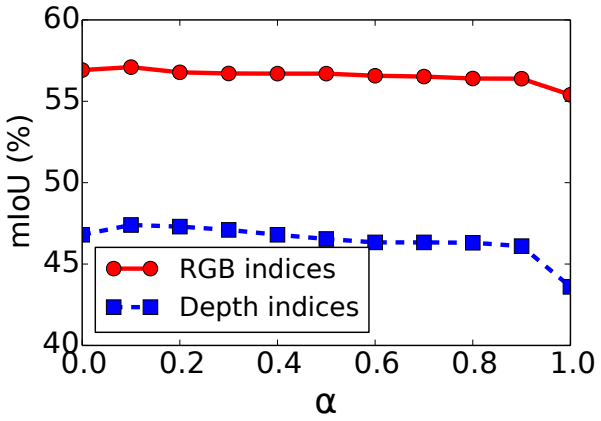

Figure 1: Multimodal semantic segmentation: pooling indices modality and modality weight α ($\alpha = 0$ for RGB only, $\alpha = 1$ for depth only).

References

[1] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Layer	Input →Output	Kernel, stride
conv1 (RGB)	[6,256,256,3] → [6,256,256,64]	[3,3], 1
conv1 (Depth)	$[6, 256, 256, 1] \rightarrow [6, 256, 256, 64]$	[3,3], 1
conv1 (Segm.)	[6,256,256,14] → [6,256,256,64]	[3,3], 1
conv2	[6,256,256,64] → [6,256,256,64]	[3,3], 1
pool2 (max)	[6,256,256,64] → [6,128,128,64]+indices2	[2,2], 2
conv3	[6,128,128,64] → [6,128,128,128]	[3,3], 1
conv4	[6,128,128,128] → [6,128,128,128]	[3,3], 1
pool4 (max)	[6,128,128,128] → [6,64,64,128]+indices4	[2,2], 2
conv5	[6,64,64,128] → [6,64,64,256]	[3,3], 1
conv6	[6,64,64,256] → [6,64,64,256]	[3,3], 1
conv7	[6,64,64,256] → [6,64,64,256]	[3,3], 1
pool7 (max)	[6,64,64,256] → [6,32,32,256]+indices7	[2,2], 2
conv8	[6,32,32,256] → [6,32,32,512]	[3,3], 1
conv9	[6,32,32,512] → [6,32,32,512]	[3,3], 1
con10	[6,32,32,512] → [6,32,32,512]	[3,3], 1
pool10 (max)	[6,32,32,512] → [6,16,16,512]+indices10	[2,2], 2
conv11	[6,16,16,512] → [6,16,16,512]	[3,3], 1
conv12	[6,16,16,512] → [6,16,16,512]	[3,3], 1
conv13	[6,16,16,512] → [6,16,16,512]	[3,3], 1
pool13 (max)	[6,16,16,512] → [6,8,8,512]+indices13	[2,2], 2

Table 3: Convolutional and pooling layers of the encoders.

layer	Input →Output	Kernel, stride
unpool1	indices13 + [6,8,8,512] → $[6, 16, 16, 512]$	[2, 2], 2
conv1	[6,16,16,512] → $[6, 16, 16, 512]$	[3,3], 1
conv2	[6,16,16,512] → $[6, 16, 16, 512]$	[3,3], 1
conv3	[6,16,16,512] → $[6, 16, 16, 512]$	[3,3], 1
unpool4	indices10 + [6,16,16,512] → $[6, 32, 32, 512]$	[2, 2], 2
conv4	[6,32,32,512] → $[6, 32, 32, 512]$	[3,3], 1
conv5	[6,32,32,512] → $[6, 32, 32, 512]$	[3,3], 1
conv6	[6,32,32,512] → $[6, 32, 32, 256]$	[3,3], 1
unpool7	indices7 + [6,32,32,256] → $[6, 64, 64, 256]$	[2, 2], 2
conv7	[6,64,64,256] → $[6, 64, 64, 256]$	[3,3], 1
conv8	[6,64,64,256]→ $[6, 64, 64, 256]$	[3,3], 1
conv9	[6,64,64,256]→ $[6, 64, 64, 128]$	[3,3], 1
unpool10	indices4 + [6,64,64,128] → $[6, 128, 128, 128]$	[2, 2], 2
conv10	[6,128,128,128] → $[6, 128, 128, 128]$	[3,3], 1
conv11	[6,128,128,128] → $[6, 128, 128, 64]$	[3,3], 1
unpool12	indices2 + [6,128,128,64] → $[6, 256, 256, 64]$	[2, 2], 2
conv12	[6,256,256,64] → $[6, 256, 256, 64]$	[3,3], 1
conv13 (Depth)	[6,256,256,64] → $[6, 256, 256, 1]$	[3,3], 1
conv13 (Segm.)	[6,256,256,64] → $[6, 256, 256, 14]$	[3,3], 1

Table 4: Convolutional and pooling layers of the segmentation and depth decoders.