NETWORK STRUCTURE

This document provides more details about network structures and training configurations.

A. Auto-encoding 3D shapes

(1) CNN-AE

Encoder:

* BN stands for batch normalization.

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1,d2,d3, channel)*
input voxels	-	-	-	(64,64,64,1)
conv3d	(4,4,4)	(2,2,2)	BN LReLU	(32,32,32,32)
conv3d	(4,4,4)	(2,2,2)	BN LReLU	(16,16,16,64)
conv3d	(4,4,4)	(2,2,2)	BN LReLU	(8,8,8,128)
conv3d	(4,4,4)	(2,2,2)	BN LReLU	(4,4,4,256)
conv3d	(4,4,4)	-	Sigmoid	(1,1,1,128)

Decoder:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1,d2,d3, channel)*
feature code	-	-	-	(1,1,1,128)
deconv3d	(4,4,4)	-	BN LReLU	(4,4,4,256)
deconv3d	(4,4,4)	(2,2,2)	BN LReLU	(8,8,8,128)
deconv3d	(4,4,4)	(2,2,2)	BN LReLU	(16,16,16,64)
deconv3d	(4,4,4)	(2,2,2)	BN LReLU	(32,32,32,32)
deconv3d	(4,4,4)	(2,2,2)	Sigmoid	(64,64,64,1)

(2) IM-AE

Encoder: same to the encoder of CNN-AE.

Decoder:

Layer	*Skip connection from*	*Input shape*	*Activation*	*Output shape*	*label*
f. code + coordinates	-	(128+3)	-	(131)	α
fully-connected	-	(131)	LReLU	(2048)	-
fully-connected	α	(2048+131)	LReLU	(1024)	-
fully-connected	α	(1024+131)	LReLU	(512)	-
fully-connected	α	(512+131)	LReLU	(256)	-
fully-connected	α	(256+131)	LReLU	(128)	-
fully-connected	-	(128)	Sigmoid	(1)	-

B. Generative models for 3D shapes

CNN-GAN and IM-GAN are using the same latent-GAN structure.

Generator:

Layer	*Activation function*	*Output shape*
latent vector	-	(128)
fully-connected	LReLU	(2048)
fully-connected	LReLU	(2048)
fully-connected	Sigmoid	(128)

Discriminator:

Layer	*Activation function*	*Output shape*
feature code	-	(128)
fully-connected	LReLU	(2048)
fully-connected	LReLU	(2048)
fully-connected	-	(1)

C. Generative models for 2D shapes

(1) CNN-AE-2D for 28^2 inputs

Encoder:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1, d2, channel)*
input pixels	-	-	-	(28,28,1)
conv2d	(4,4)	(1,1)	BN LReLU	(28,28,16)
conv2d	(4,4)	(2,2)	BN LReLU	(14,14,32)
conv2d	(4,4)	(2,2)	BN LReLU	(7,7,64)
conv2d	(4,4)	(2,2)	BN LReLU	(4,4,128)
conv2d	(4,4)	-	Sigmoid	(1,1,100)

Decoder:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1, d2, channel)*
feature code	-	-	-	(1,1,100)
deconv2d	(4,4)	-	BN LReLU	(4,4,128)
deconv2d	(4,4)	(2,2)	BN LReLU	(7,7,64)
deconv2d	(4,4)	(2,2)	BN LReLU	(14,14,32)
deconv2d	(4,4)	(2,2)	BN LReLU	(28,28,16)
deconv2d	(4,4)	(1,1)	Sigmoid	(28,28,1)

(2) CNN-AE-2D for 64^2 inputs

Encoder:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1, d2, channel)*
input pixels	-	-	-	(64,64,1)
conv2d	(4,4)	(2,2)	BN LReLU	(32,32,16)
conv2d	(4,4)	(2,2)	BN LReLU	(16,16,32)
conv2d	(4,4)	(2,2)	BN LReLU	(8,8,64)
conv2d	(4,4)	(2,2)	BN LReLU	(4,4,128)
conv2d	(4,4)	-	Sigmoid	(1,1,100)

Decoder:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1, d2, channel)*
feature code	-	-	-	(1,1,100)
deconv2d	(4,4)	-	BN LReLU	(4,4,128)
deconv2d	(4,4)	(2,2)	BN LReLU	(8,8,64)
deconv2d	(4,4)	(2,2)	BN LReLU	(16,16,32)
deconv2d	(4,4)	(2,2)	BN LReLU	(32,32,16)
deconv2d	(4,4)	(2,2)	Sigmoid	(64,64,1)

(3) IM-AE-2D

Encoder: same to CNN-AE-2D.

Decoder:

Layer	*Skip connection from*	*Input shape*	*Activation*	*Output shape*	*label*
f. code + coordinates	-	(100+2)	-	(102)	α
fully-connected	-	(102)	LReLU	(1024)	-
fully-connected	α	(1024+102)	LReLU	(512)	-
fully-connected	α	(512+102)	LReLU	(256)	-
fully-connected	α	(256+102)	LReLU	(128)	-
fully-connected	α	(128+102)	LReLU	(64)	-
fully-connected	-	(64)	Sigmoid	(1)	-

(4) Latent-GAN-2D

Same to latent-GAN for 3D shapes.

(4) DCGAN/WGAN

Generator: same to the decoder of CNN-AE-2D, or IM-AE-2D for $WGAN_{IM}$.

Discriminator: replace the last layer of the encoder of CNN-AE-2D by

conv2d	(4,4)	-	Sigmoid/-	(1,1,1)

(5) VAE

Decoder: same to the decoder of CNN-AE-2D, or IM-AE-2D for VAE_{IM}.

Encoder: replace the last layer of the encoder of CNN-AE-2D by

conv2d	(4,4)	-	-	(1,1,256)

Where the output 256-d vector is 128-d mu and 128-d sigma.

D. Single-view 3D reconstruction

Decoder: same to the decoder of IM-AE.

Encoder: (Slightly modified from ResNet-18. Pooling layers are removed or replaced by conv. layers)

Layer	*Kernel size*	*Stride*	*Activation function*	*Input shape*	*Output shape*
input pixels	-	-	-	-	(128,128,1)
conv2d	(7,7)	(2,2)	BN LReLU	(128,128,1)	(64,64,64)
ResNet block	(3,3)	-	-	(64,64,64)	(64,64,64)
ResNet block	(3,3)	-	-	(64,64,64)	(64,64,64)
ResNet block	(3,3)	-	-	(64,64,64)	(32,32,128)
ResNet block	(3,3)	-	-	(32,32,128)	(32,32,128)
ResNet block	(3,3)	-	-	(32,32,128)	(16,16,256)
ResNet block	(3,3)	-	-	(16,16,256)	(16,16,256)
ResNet block	(3,3)	-	-	(16,16,256)	(8,8,512)
ResNet block	(3,3)	-	-	(8,8,512)	(8,8,512)
conv2d	(4,4)	(2,2)	BN LReLU	(8,8,512)	(4,4,512)
conv2d	(4,4)	-	Sigmoid	(4,4,512)	(1,1,128)

E. Training configurations

The networks were implemented with Tensorflow and using Adam optimizer (learning_rate=5e-5, beta1=0.5, beta2=0.999, epsilon=1e-8).

For leaky ReLU, alpha=0.02.

For batch normalization, decay=0.999, epsilon=1e-5.

The training batch size is: 32 for 3D CNN-based models, 50 for 2D CNN-based models, 1 for implicit-decoder-based models, 50 for latent-GANs. Notice that for implicit-decoder-based models, the batch size is one shape, the actual batch size for implicit decoder varies according to the resolution of the training data.

The hyper-parameters were not selected by extensive testing, thus are not guaranteed to be optimal.