LEED: Label-Free Expression Editing via Disentanglement

Rongliang Wu[0000−0002−5586−0628] and Shijian Lu[0000−0002−6766−2506]

Nanyang Technological University
ronglian001@e.ntu.edu.sg, shijian.lu@ntu.edu.sg

1 Network Architecture

Our network has five major components: an extractor $\boldsymbol{\mathcal{X}}$ for extracting expression attribute from the reference image; an interpolator $\boldsymbol{\mathcal{I}}$ for fusing the extracted expression attribute and the identity attribute of the input image; an encoder $\boldsymbol{E}$ for mapping the facial images into a compact expression and identity embedded space; a decoder $\boldsymbol{D}$ for projecting the interpolated code to image space and a pre-trained GAN for synthesizing the neutral faces. Besides, a discriminator $\boldsymbol{\mathcal{D}}$ is designed for distinguishing the real/interpolated codes, a regularizer $\boldsymbol{Q}$ and siamese network $\boldsymbol{S}$ for optimal expression disentanglement and consistent synthesis, respectively. The detailed architectures are shown in Tables 1-6 [1].

2 Training Details

We adopt Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$ for optimization. We set λ_Q, λ_{recon}, λ_S and λ_{gp} to be 0.01, 10, 1000 and 100 to balance the magnitude of different losses. The batchsize is set to 24. The total number of epochs is set to 100. The initial learning rate is set to 1e-4 for the first 50 epochs, then linearly decay to 0 over another 50 epochs. The training process takes 7 hours on RaFD [3] and 13 hours on CFEED [2] on a single Tesla V100 GPU, respectively.

3 More Results

We also present more results generated by LEED in the following pages.

References

1. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8789–8797 (2018)

[1] We pretrain StarGAN [1] on the corresponding dataset and use it for synthesizing neutral faces, with official implementation at https://github.com/yunjey/stargan.

2. Du, S., Tao, Y., Martinez, A.M.: Compound facial expressions of emotion. Proceedings of the National Academy of Sciences **111**(15), E1454–E1462 (2014)
3. Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D.H., Hawk, S.T., Van Knippenberg, A.: Presentation and validation of the radboud faces database. Cognition and emotion **24**(8), 1377–1388 (2010)

Fig. 1. Additional expression editing results on wild images. In each triplet, the first column is input facial image, the second column is the image with desired expression and the last column is the synthesized result.

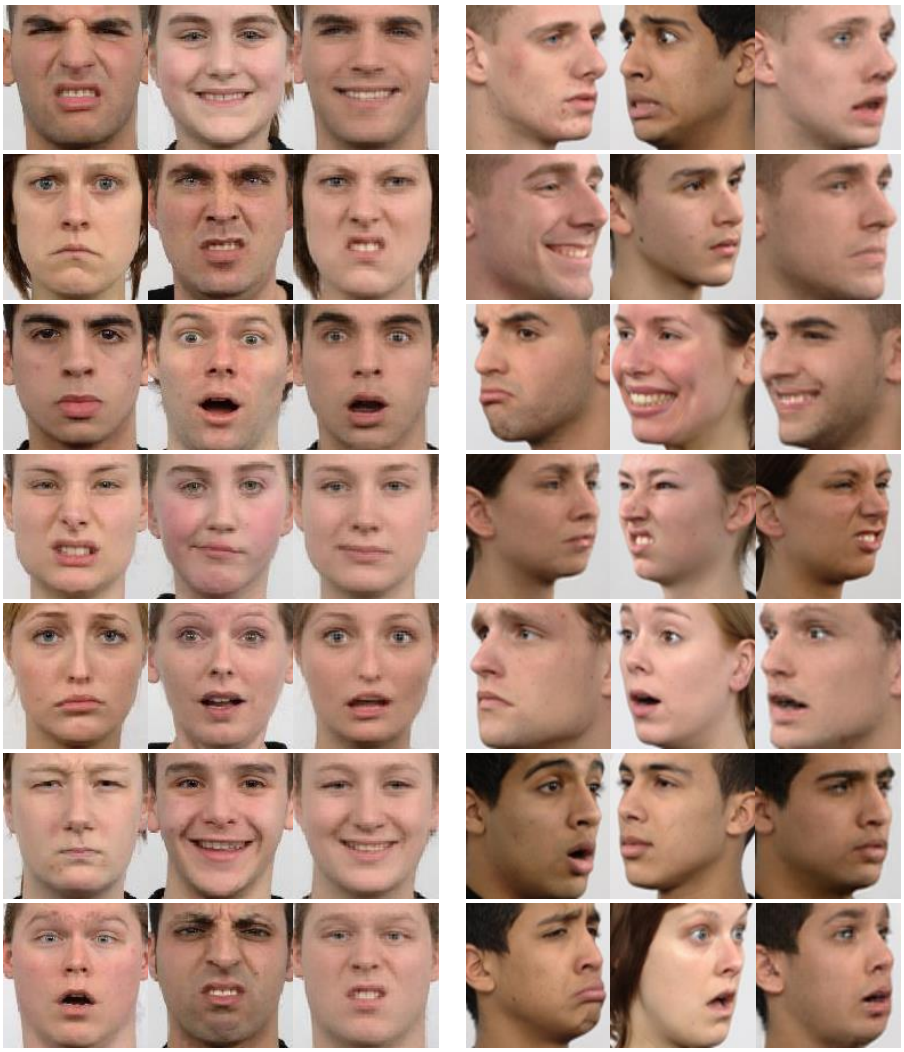

Fig. 2. Additional expression editing results on RaFD [3]. In each triplet, the first column is input facial image, the second column is the image with desired expression and the last column is the synthesized result.

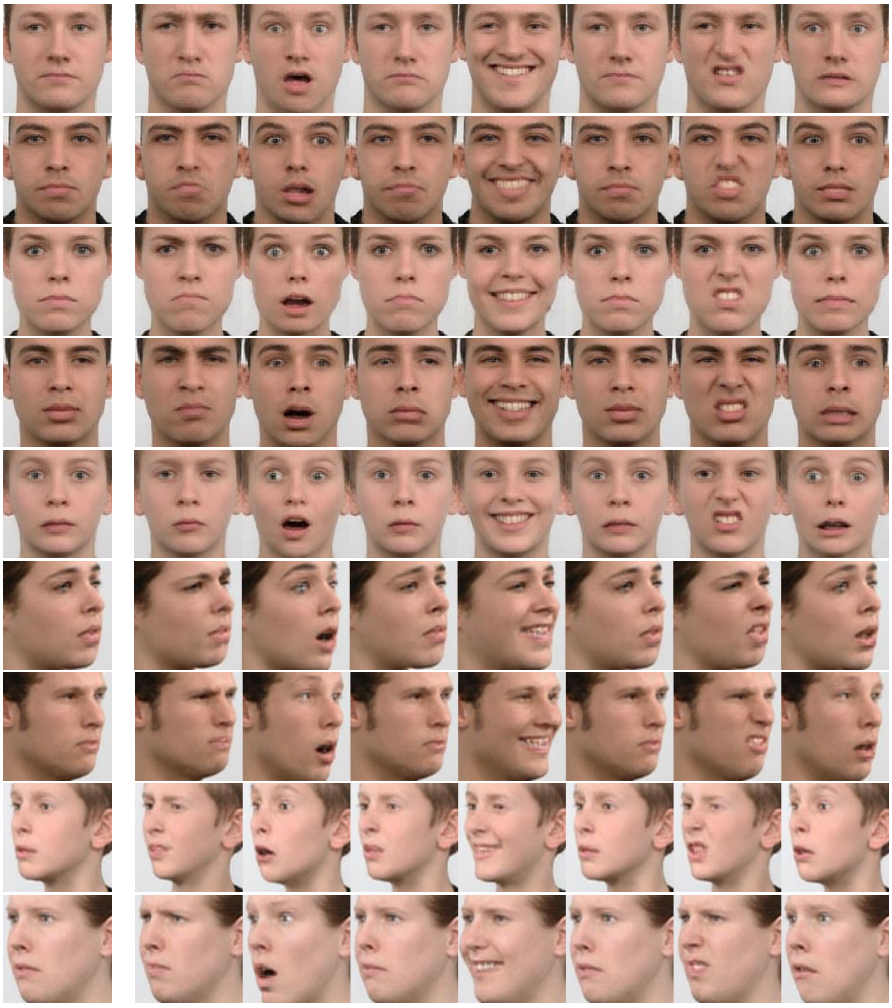

Fig. 3. Additional expression editing results on RaFD [3]. For each row, the left most one is the input facial image, and the rest gives the synthesized expressions (Angry, Surprised, Sad, Happy, Neutral, Disgusted, Fearful).

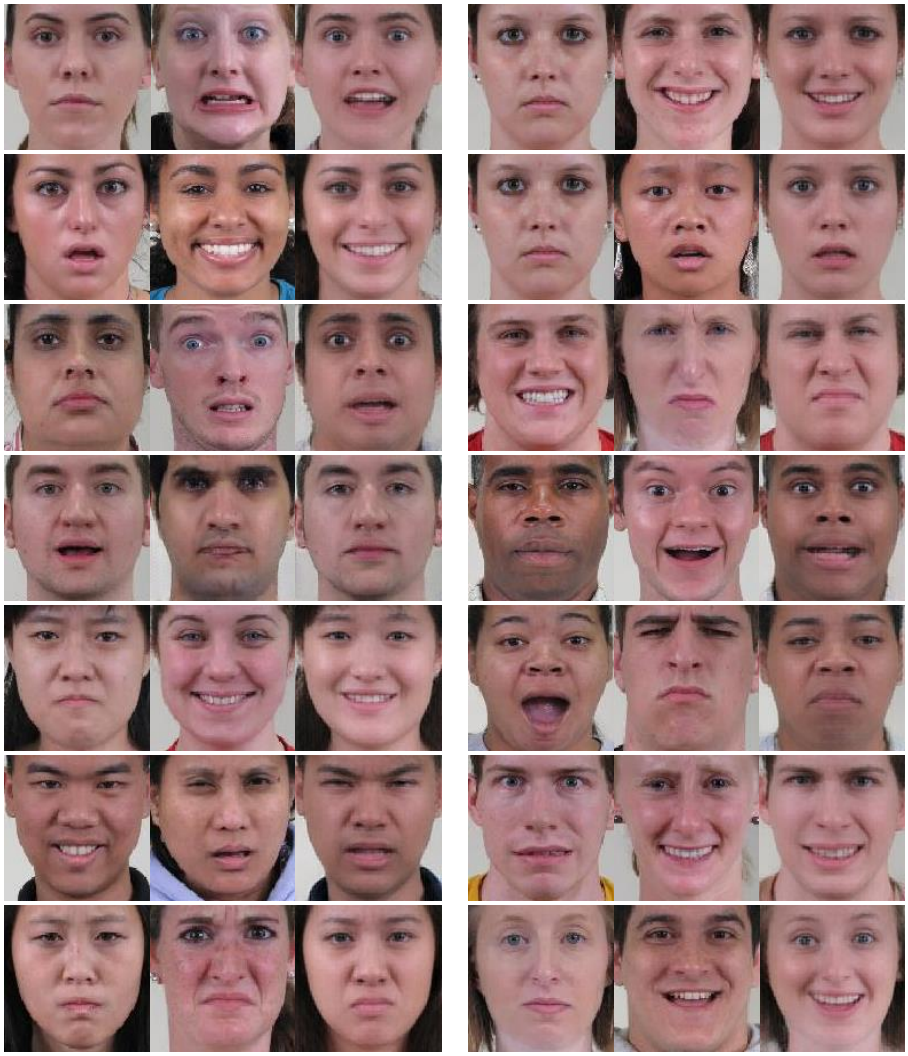

Fig. 4. Additional expression editing results on CFEED [2]. In each triplet, the first column is input facial image, the second column is the image with desired expression and the last column is the synthesized result.

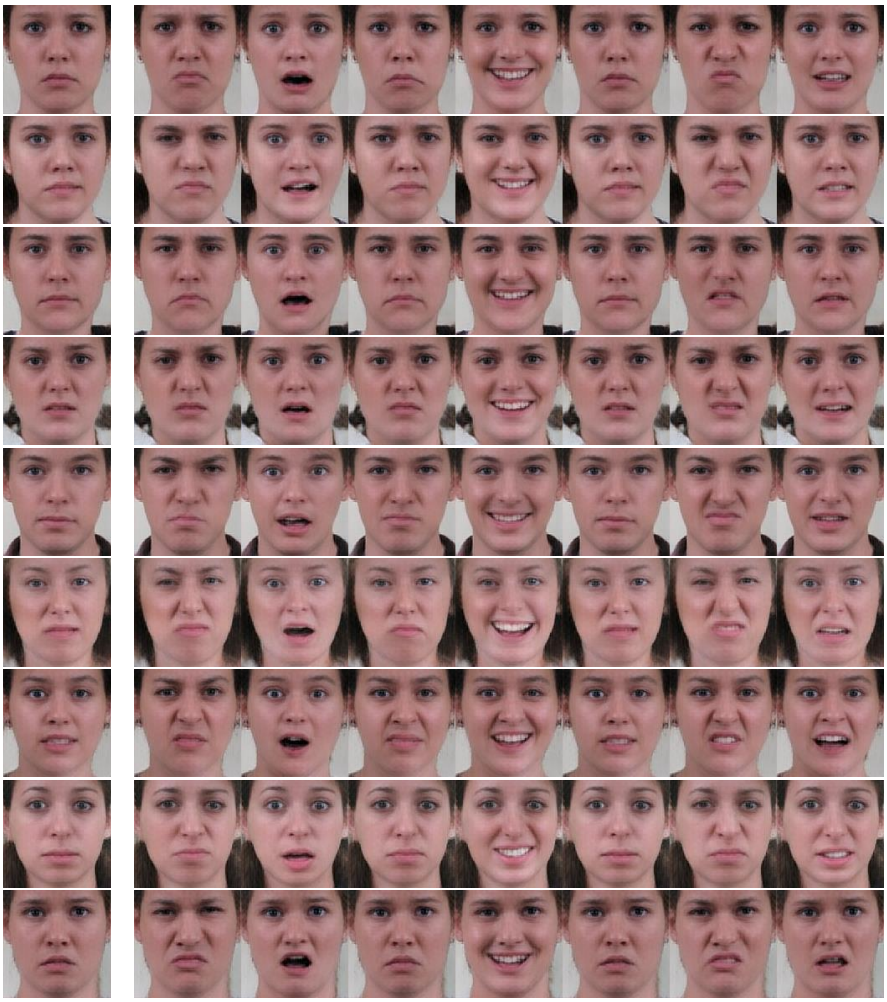

Fig. 5. Additional expression editing results on CFEED [2]. For each row, the left most one is the input facial image, and the rest gives the synthesized expressions (Angry, Surprised, Sad, Happy, Neutral, Disgusted, Fearful).

Table 1. Architecture of extractor $\boldsymbol{\mathcal{X}}$ and interpolator $\boldsymbol{\mathcal{I}}$. $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{I}}$ share the same architecture.

Layer Type	Output Size	Channel	Kernel	Stride	Padding	Normalization	Activation
Conv2d	512 $\times 8 \times 8$	512	3	1	1	-	LeakyReLU
Conv2d	512 $\times 8 \times 8$	512	3	1	1	-	LeakyReLU
Conv2d	512 $\times 8 \times 8$	512	3	1	1	-	-

Table 2. Architecture of discriminator $\boldsymbol{\mathcal{D}}$. IN stands for instance normalization.

Layer Type	Output Size	Channel	Kernel	Stride	Padding	Normalization	Activation
Conv2d	256 $\times 8 \times 8$	256	1	1	0	IN	LeakyReLU
Conv2d	512 $\times 4 \times 4$	512	4	2	1	IN	LeakyReLU
Conv2d	1024 $\times 2 \times 2$	1024	4	2	1	IN	LeakyReLU
Conv2d	1024 $\times 1 \times 1$	1024	2	2	0	-	-

Table 3. Architecture of encoder $\boldsymbol{E}$.

Layer Type	Output Size	Channel	Kernel	Stride	Padding	Normalization	Activation
Conv2d	64 $\times 128 \times 128$	64	3	1	1	-	ReLU
Conv2d	64 $\times 128 \times 128$	64	3	1	1	-	ReLU
MaxPool2d	64 $\times 64 \times 64$	-	2	2	0	-	-
Conv2d	128 $\times 64 \times 64$	128	3	1	1	-	ReLU
Conv2d	128 $\times 64 \times 64$	128	3	1	1	-	ReLU
MaxPool2d	128 $\times 32 \times 32$	-	2	2	0	-	-
Conv2d	256 $\times 32 \times 32$	256	3	1	1	-	ReLU
Conv2d	256 $\times 32 \times 32$	256	3	1	1	-	ReLU
Conv2d	256 $\times 32 \times 32$	256	3	1	1	-	ReLU
Conv2d	256 $\times 32 \times 32$	256	3	1	1	-	ReLU
MaxPool2d	256 $\times 16 \times 16$	-	2	2	0	-	-
Conv2d	512 $\times 16 \times 16$	512	3	1	1	-	ReLU
Conv2d	512 $\times 16 \times 16$	512	3	1	1	-	ReLU
Conv2d	512 $\times 16 \times 16$	512	3	1	1	-	ReLU
Conv2d	512 $\times 16 \times 16$	512	3	1	1	-	ReLU
MaxPool2d	512 $\times 8 \times 8$	-	2	2	0	-	-
Conv2d	512 $\times 8 \times 8$	512	3	1	1	-	-

Table 4. Architecture of decoder $\boldsymbol{D}$. BN stands for batch normalization.

Layer Type	Output Size	Channel	Kernel	Stride	Padding	Normalization	Activation
Conv2d	512 $\times$8 $\times$ 8	512	3	1	1	BN	ReLU
Upsample	512 $\times$16 $\times$ 16	-	-	-	-	-	-
Conv2d	256 $\times$16 $\times$ 16	256	3	1	1	BN	ReLU
Conv2d	256 $\times$16 $\times$ 16	256	3	1	1	BN	ReLU
Conv2d	256 $\times$16 $\times$ 16	256	3	1	1	BN	ReLU
Conv2d	256 $\times$16 $\times$ 16	256	3	1	1	BN	ReLU
Upsample	256 $\times$32 $\times$ 32	-	-	-	-	-	-
Conv2d	128 $\times$32 $\times$ 32	128	3	1	1	BN	ReLU
Conv2d	128 $\times$32 $\times$ 32	128	3	1	1	BN	ReLU
Conv2d	128 $\times$32 $\times$ 32	128	3	1	1	BN	ReLU
Conv2d	128 $\times$32 $\times$ 32	128	3	1	1	BN	ReLU
Upsample	128 $\times$64 $\times$ 64	-	-	-	-	-	-
Conv2d	64 $\times$64 $\times$ 64	64	3	1	1	BN	ReLU
Conv2d	64 $\times$64 $\times$ 64	64	3	1	1	BN	ReLU
Upsample	64 $\times$128 $\times$ 128	-	-	-	-	-	-
Conv2d	64 $\times$128 $\times$ 128	64	3	1	1	BN	ReLU
Conv2d	3 $\times$128 $\times$ 128	3	3	1	1	-	-

Table 5. Architecture of regularizer $\boldsymbol{Q}$. BN stands for batch normalization.

Layer Type	Output Size	Channel	Kernel	Stride	Padding	Normalization	Activation
Conv2d	512 $\times$1 $\times$ 1	512	8	1	0	BN	LeakyReLU
FC	128	128	-	-	-	BN	LeakyReLU
FC	16	16	-	-	-	-	-

Table 6. Architecture of siamese network $\boldsymbol{S}$. IN stands for instance normalization.

Layer Type	Output Size	Channel	Kernel	Stride	Padding	Normalization	Activation
Conv2d	64 $\times$64 $\times$ 64	64	4	2	1	IN	LeakyReLU
Conv2d	128 $\times$32 $\times$ 32	128	4	2	1	IN	LeakyReLU
Conv2d	256 $\times$16 $\times$ 16	256	4	2	1	IN	LeakyReLU
Conv2d	512 $\times$8 $\times$ 8	512	4	2	1	IN	LeakyReLU
Conv2d	1024 $\times$4 $\times$ 4	1024	4	2	1	IN	LeakyReLU
Conv2d	2048 $\times$2 $\times$ 2	1024	4	2	1	IN	LeakyReLU
FC	1024	1024	-	-	-	-	-