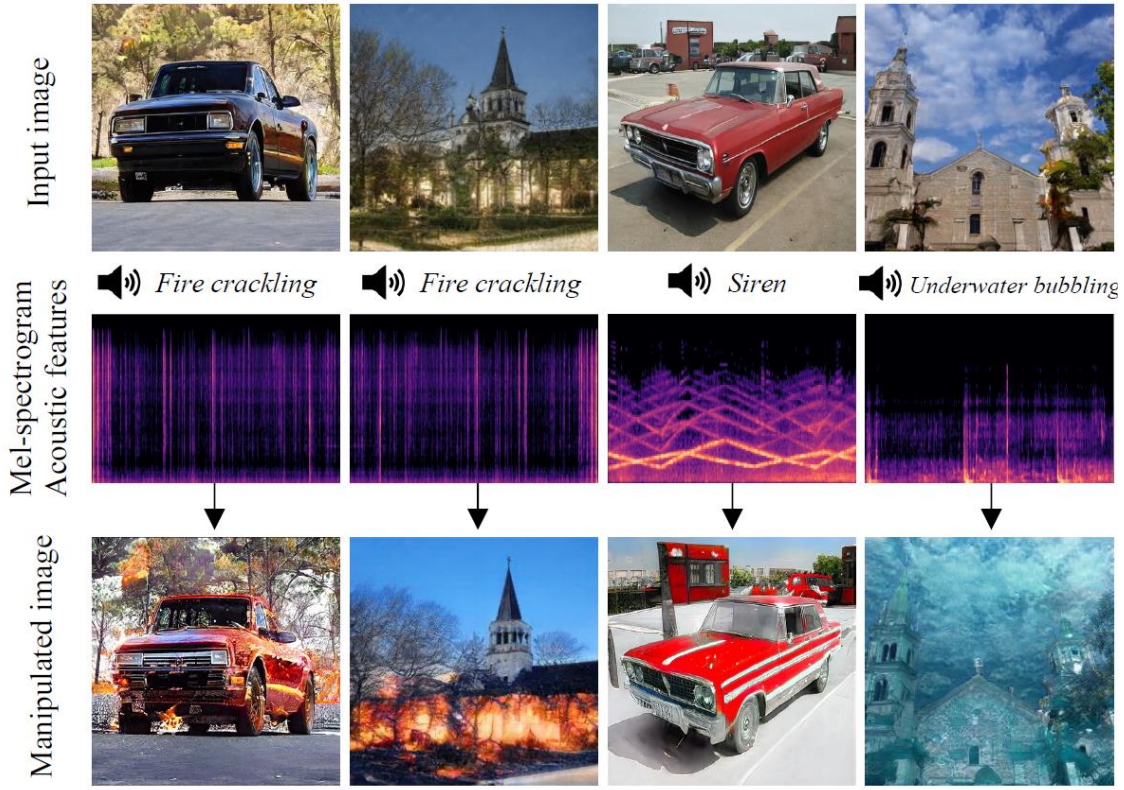
Generative Adversarial Nets: Learning and Applications

Wangmeng Zuo

Centre on Machine Learning Research
Harbin Institute of Technology

Image Generation

Image Generation

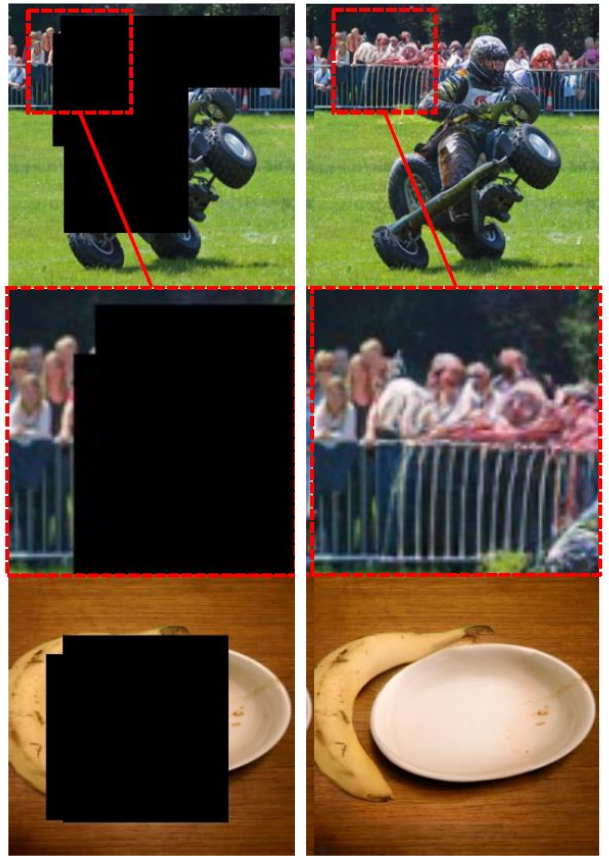


Conditional Image Generation

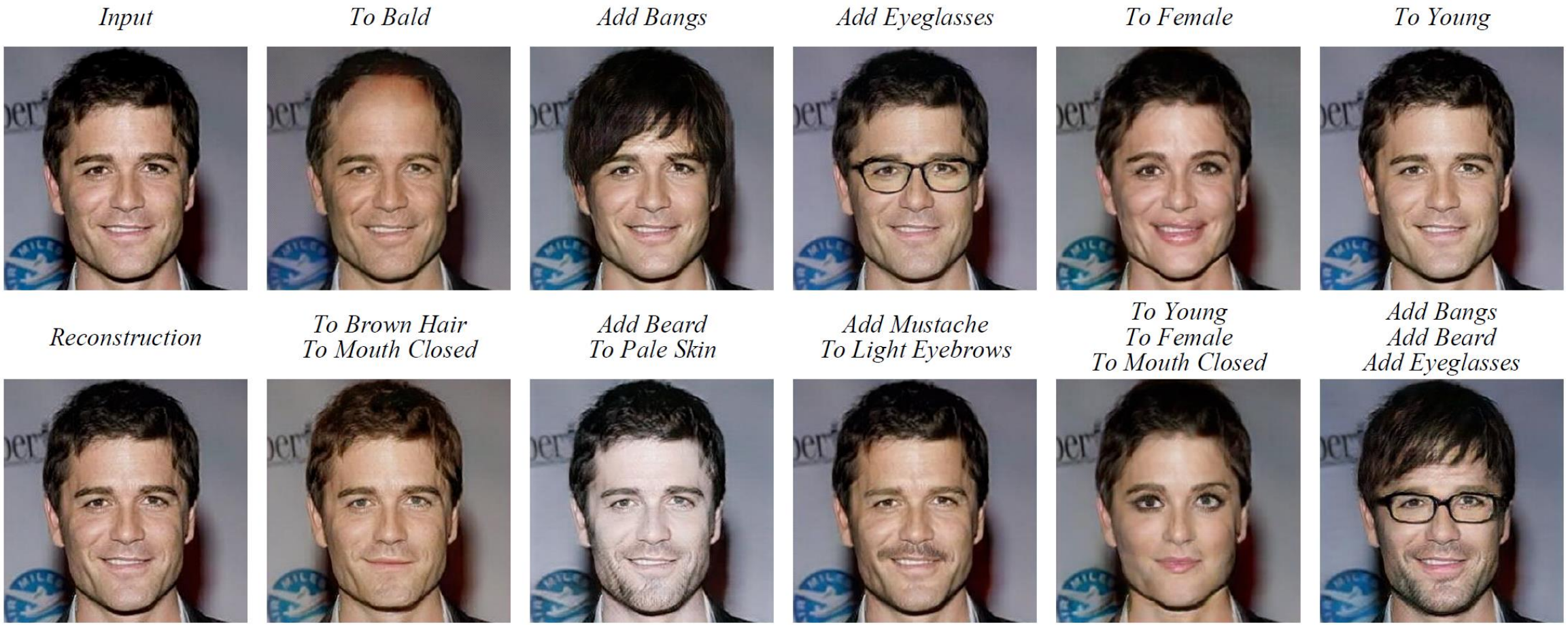
Image Inpainting

(a) Input context (b) Human artist

(c) Context Encoder ($L2$ loss) (d) Context Encoder ($L2$ + Adversarial loss)

Defective Image	NÜWA-LIP (finetune)	Guidance Text
		*Racers riding four wheelers while a crowd watches.*
		The banana is laying next to an almost empty bowl.

Face Attribute Editing



LeCun, NIPS 2016

- Reinforcement learning (cherry)
- Supervised learning (Chocolate)
- Unsupervised/Predictive learning (Cake)
  - Generative adversarial nets (GAN)
  - Self-supervised learning
  - World model

Content

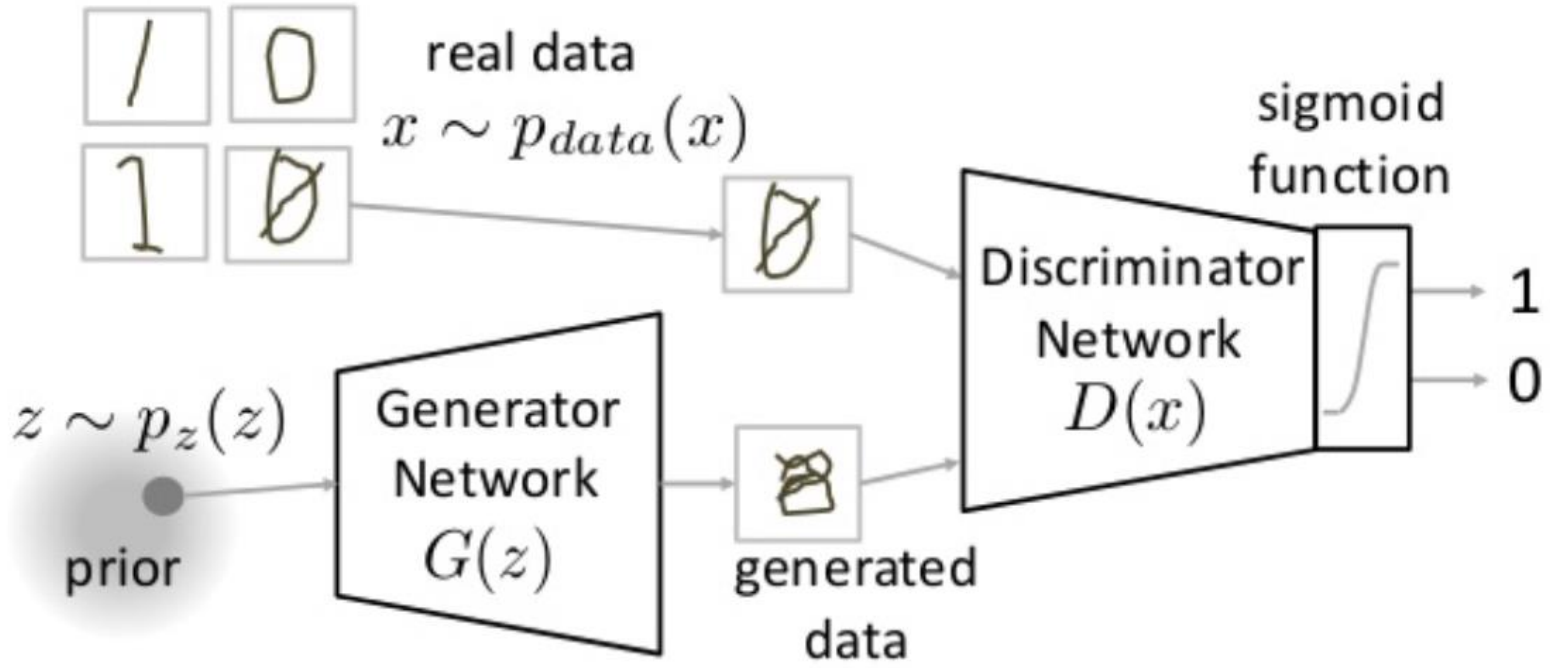
- Improve the trainability of GANs
  - Theoretical solution
  - Incorporating with other learning models
  - Designing generator based on signal/image characteristics
  - Regularization techniques

- Applications
  - Image Generation
  - Image Restoration

Improve the trainability of GANs

Generative Adversarial Networks (Goodfellow et al., NIPS 2014)

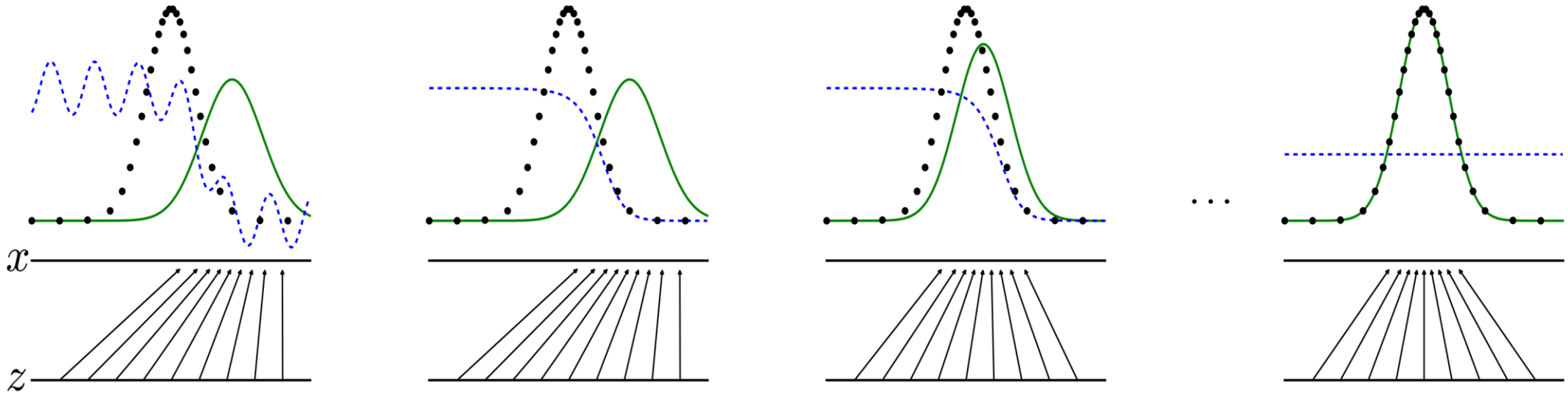
- Update the generator to generate more realistic image
- Update the discriminator to discriminate the synthetic images from real ones



Two-player minimax game

- Value function

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Theoretical Results: Global Optimality

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

$$\begin{aligned} C(G) =& \max_D V(G, D) \\ =& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\ =& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\ =& \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] \end{aligned}$$

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Theoretical Results: Convergence

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

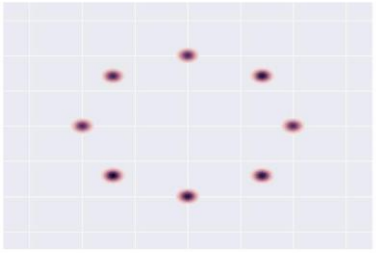
$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))]$$

then p_g converges to p_{data}

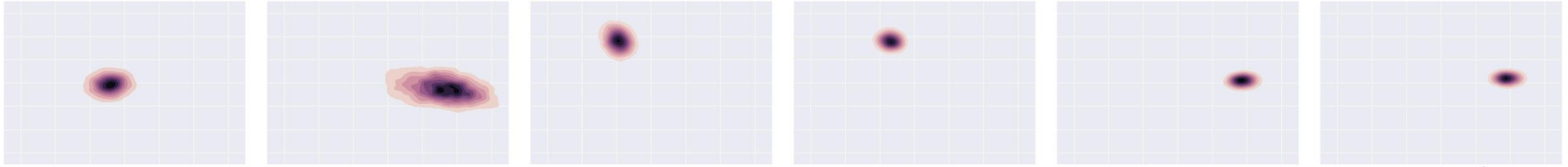
Mode Collapse

$$\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D)$$

- D in inner loop: convergence to correct distribution
- G in inner loop: place all mass on most likely point



Target



Step 0 Step 5k Step 10k Step 15k Step 20k Step 25k

Wasserstein GAN (Arjovsky et al., Arxiv 2017)

- Wasserstein-1 distance

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

- Actually, it is the original MMD exactly (Borgwardt, Bioinformatics 2006):

$$\mathrm{MMD}^2(s,t) = \sup_{\|\phi\|_{\mathcal{H}} \leq 1} \left\| E_{\mathbf{x}^s \sim s}\left[\phi(\mathbf{x}^s)\right] - E_{\mathbf{x}^t \sim t}\left[\phi(\mathbf{x}^t)\right] \right\|_{\mathcal{H}}^2$$

- Moreover, similar method has already be adopted in improved GAN (Salimans et al., NIPS 2016)

$$\sup_{\|f\|_L \leq 1} \; ||\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \mathbf{f}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} \mathbf{f}(G(\boldsymbol{z}))||_2^2$$

Wasserstein GAN

- With kernel two-sample test, MMD is optimized via quaternary based mini-batch.

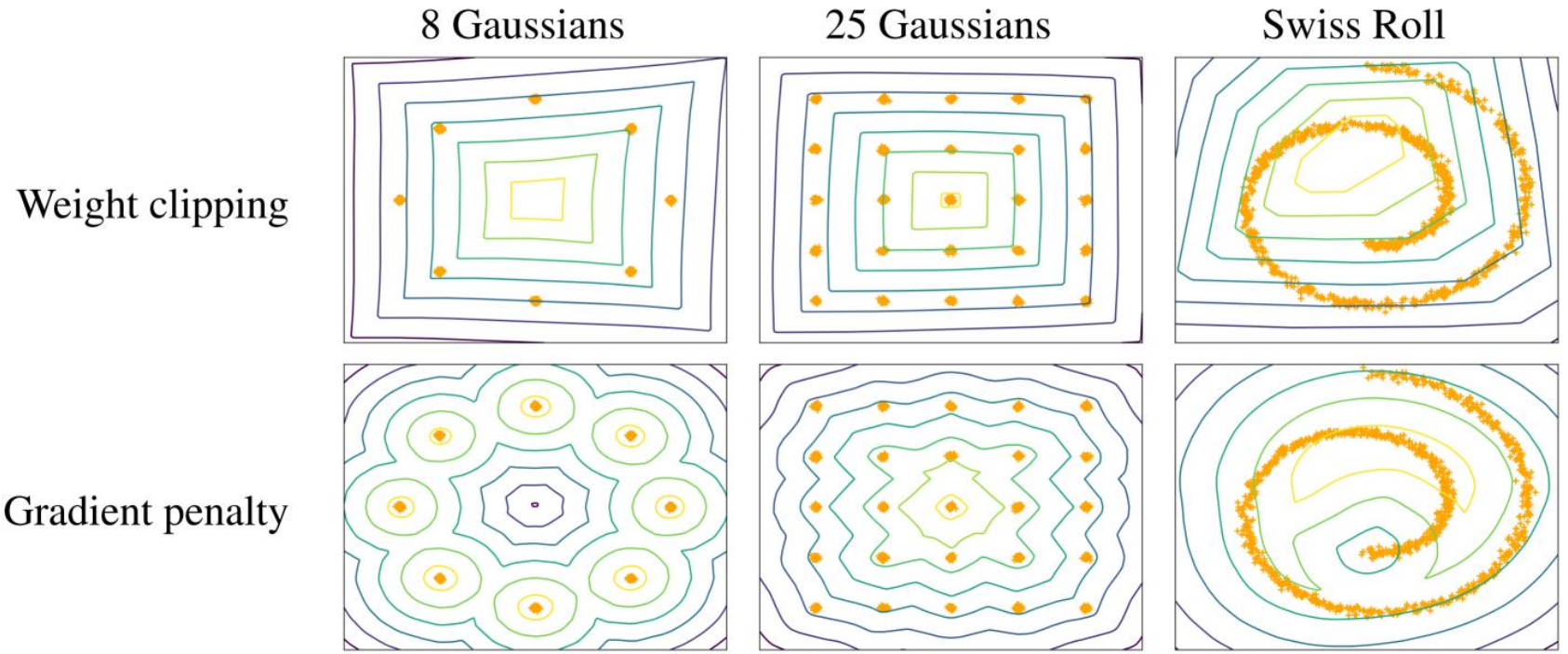
- While Wasserstein GAN is minimized optimized directly on two batches

for $t = 0, ..., n_{\text{critic}}$ **do**
  Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
  $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
  $w \leftarrow \text{clip}(w, -c, c)$
end for

Improved Wasserstein GAN

- Gradient penalty

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}$$

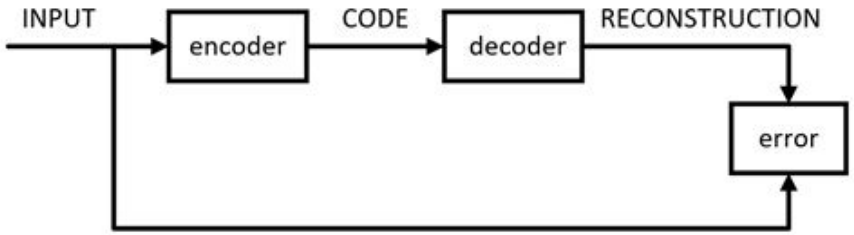


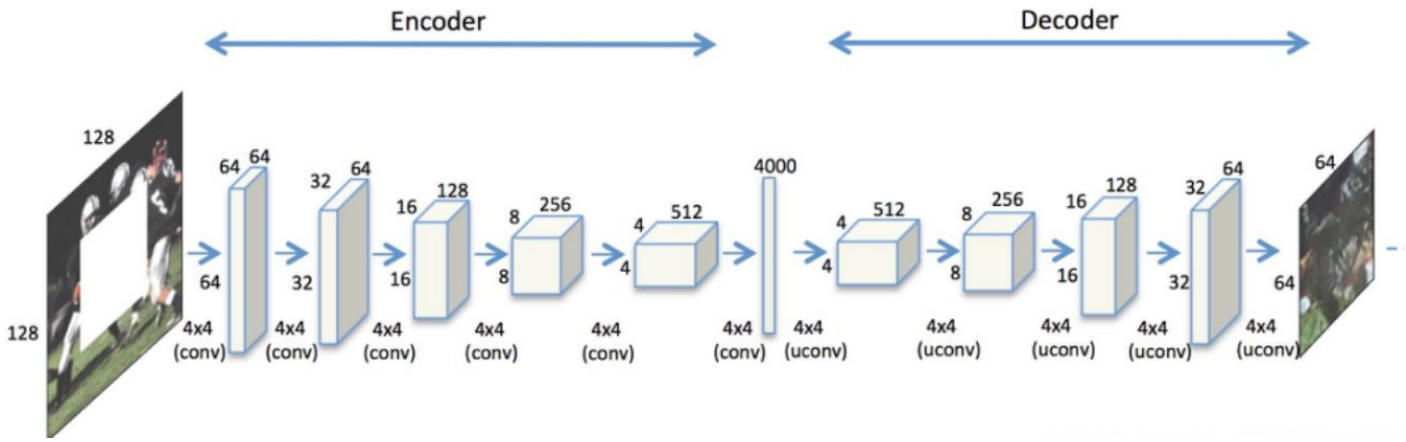
Incorporating with Other Learning Tasks

- Incorporating GAN learning with:
  - Image Reconstruction
  - Image Classification
  - Latent Codes
  - ...

Auto-encoder

- Auto-encoder





$$\phi, \psi = argmin_{\phi,\psi} L(X, (\psi \circ \phi)X)$$

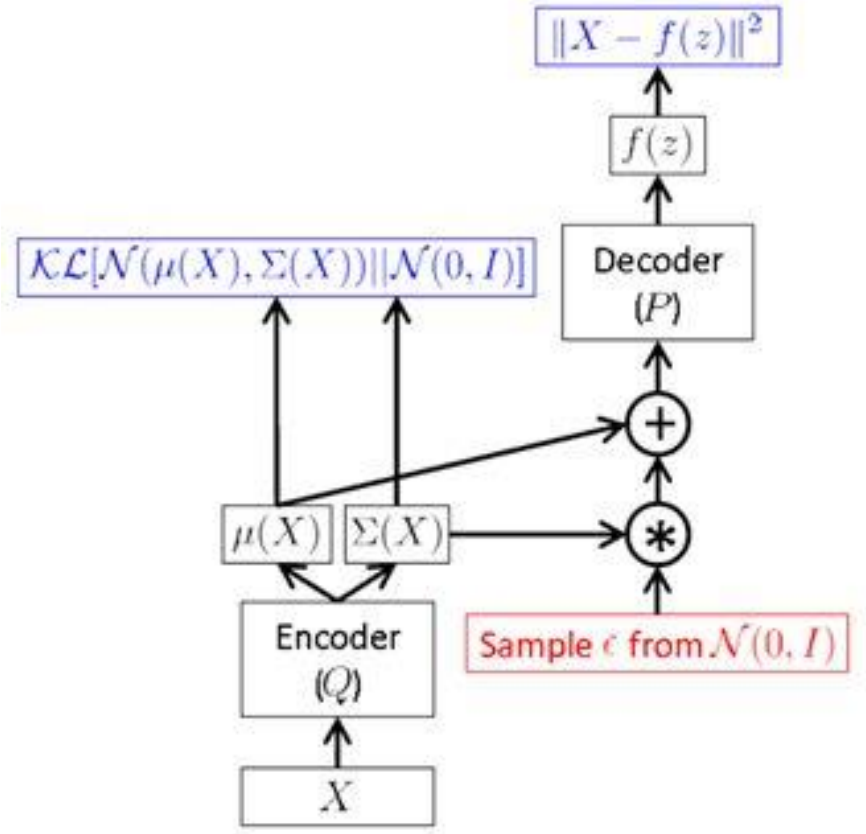
- Denoising auto-encoder

Variational AutoEncoder

- Variational AutoEncoder

$$l_i = -E_{z\sim q}[logp(x_i|z)] + KL[q(z|x_i)||p(z)]$$

  - Relaxation of discrete variables

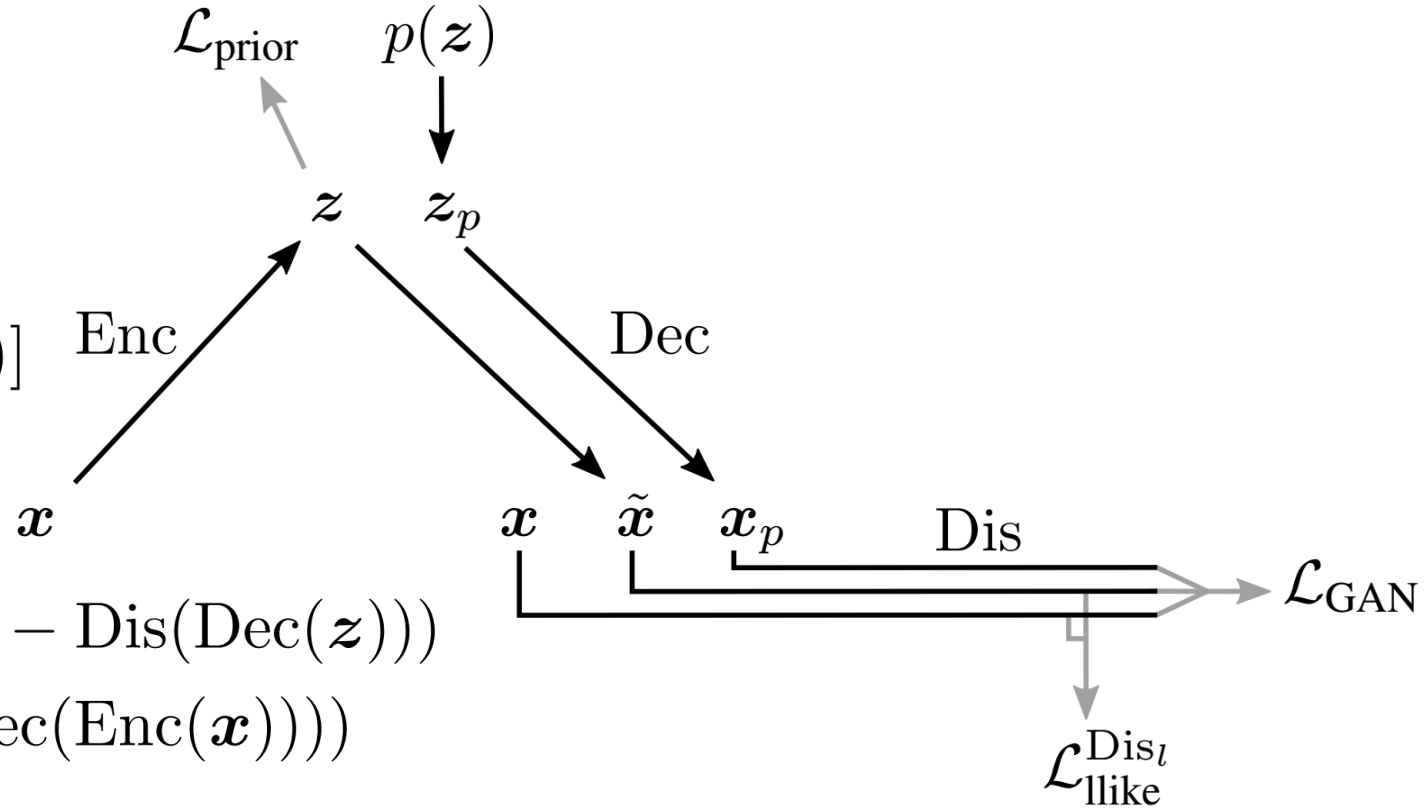


VAE/GAN (Larsen et al., ICML 2016)

- VAE

$$\mathcal{L}_{\text{prior}} = D_{\text{KL}}(q(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}))$$

$$\mathcal{L}_{\text{llike}}^{\text{Dis}_l} = -\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\text{Dis}_l(\boldsymbol{x})|\boldsymbol{z})\right]$$



- GAN

$$\begin{aligned}\mathcal{L}_{\text{GAN}} = \quad & \log(\text{Dis}(\boldsymbol{x})) + \log(1 - \text{Dis}(\text{Dec}(\boldsymbol{z}))) \\ & + \log(1 - \text{Dis}(\text{Dec}(\text{Enc}(\boldsymbol{x}))))\end{aligned}$$

- VAE/GAN

$$\mathcal{L} = \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{llike}}^{\text{Dis}_l} + \mathcal{L}_{\text{GAN}}$$

InfoGAN (Chen et al., NIPS 2016)

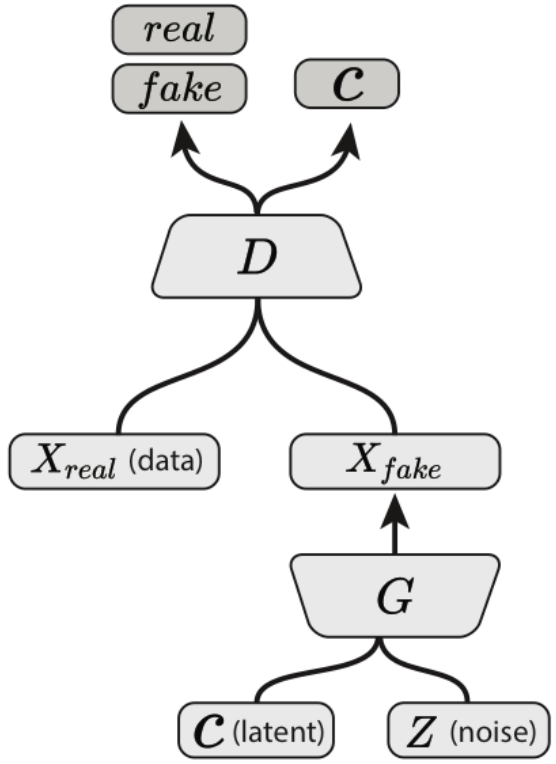
- GAN

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim \text{noise}}[\log(1 - D(G(z)))]$$

- InfoGAN (Chen et al., NIPS 2016)
  - Input: z, c

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

  - Interpretable and disentangled representations
  - Easy to train



InfoGAN
(Chen, et al., 2016)

AC-GAN (Odena et al., ICML 2017)

- Class-conditional image synthesis with Auxiliary Classifier GANs

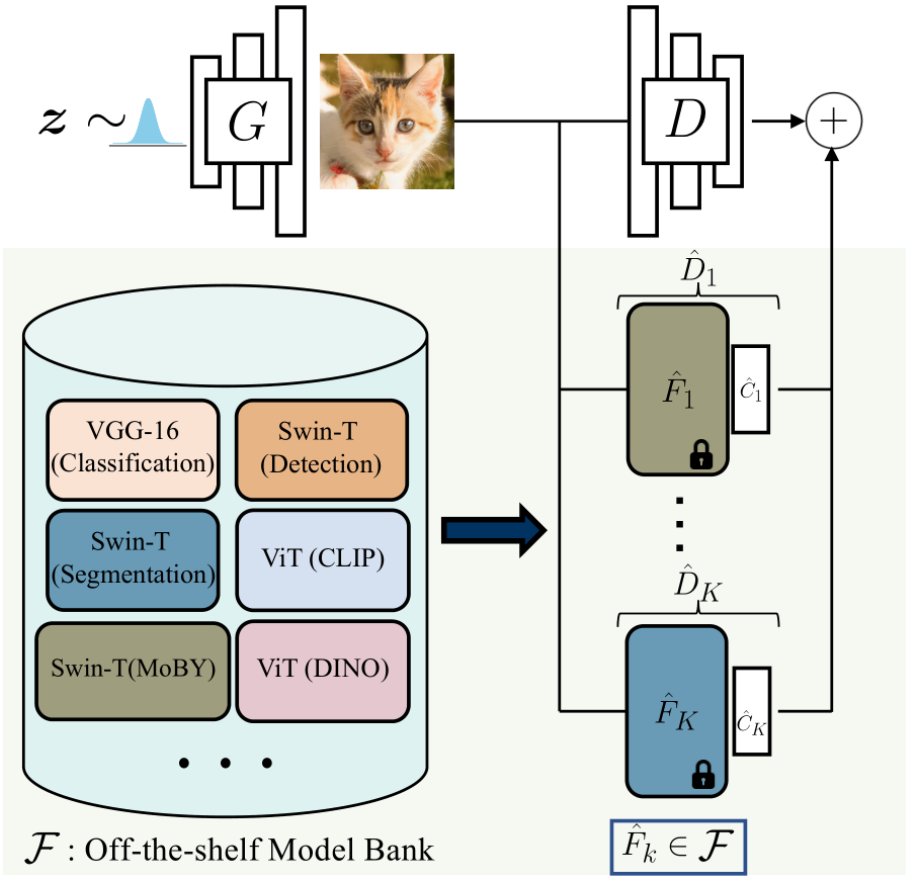
- The log-likelihood of the correct source:

$$L_S = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})]$$

- The log-likelihood of the correct class:

$$L_C = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})]$$

Ensembling Off-the-shelf Models



Dataset		StyleGAN2	DiffAugment	ADA	Ours (w/ ADA)			Ours (w/ DiffAugment)		
					+1st D	+2nd D	+3rd D	+1st D	+2nd D	+3rd D
FFHQ	1k	62.16	27.20	19.57	11.43	**10.39**	10.58	**12.33**	13.39	12.76
	2k	42.62	16.63	16.06	10.17	8.73	**8.18**	10.01	**9.24**	10.99
	10k	16.07	8.15	8.38	6.90	6.39	**5.90**	6.94	**6.26**	6.43
LSUN Cat	1k	185.75	43.32	41.14	15.49	12.90	**12.19**	13.52	12.52	**11.01**
	2k	68.03	25.70	23.32	13.44	13.35	**11.51**	12.20	11.79	**11.33**
	10k	18.59	12.56	13.25	8.37	7.13	**6.86**	8.19	7.90	**7.79**
LSUN Church	1k	-	19.38	19.66	11.39	9.78	**9.56**	10.15	**9.87**	9.94
	2k	-	13.46	11.17	5.25	**5.06**	5.26	6.09	6.37	**5.56**
	10k	-	6.69	6.12	4.80	4.82	**4.47**	3.42	3.41	**3.25**

Ensembling Off-the-shelf Models for GAN Training, Arxiv 2021.

Take home message

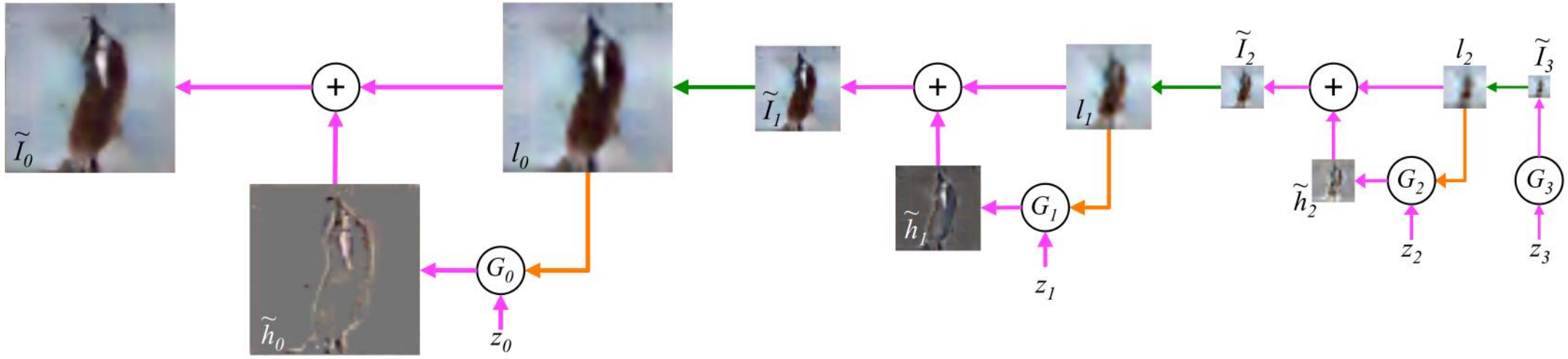
- Incorporating auto-encoder to improve the trainability of generator;
- Incorporating deep classification model to improve the trainability of discriminator

Let's then turn to the objective of GANs

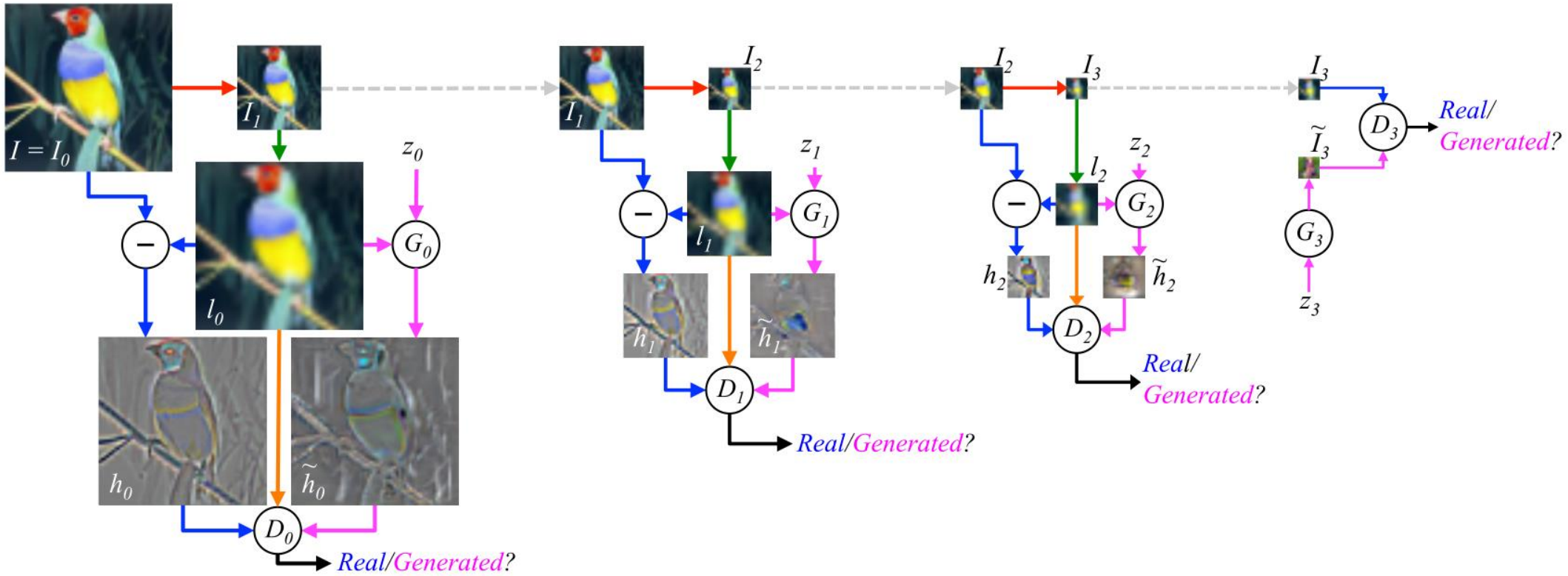
- Image generation
- What's the characteristics of an image
  - Multi-scale property

LAPGANs (Denton et al., NIPS 2015)

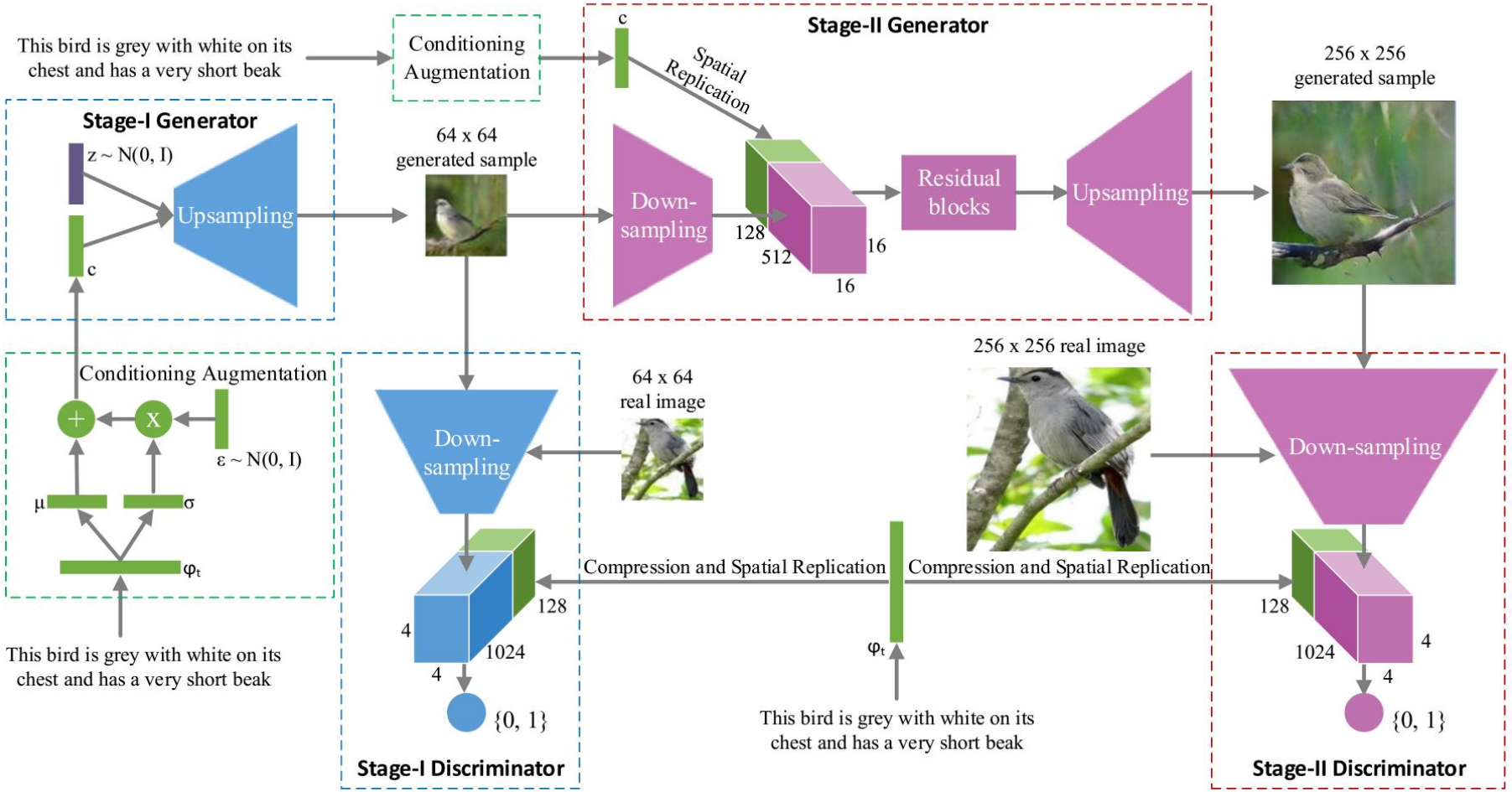
$$\tilde{I}_k = u(\tilde{I}_{k+1}) + \tilde{h}_k = u(\tilde{I}_{k+1}) + G_k(z_k, u(\tilde{I}_{k+1}))$$



LAPGANs (Denton et al., 2015)

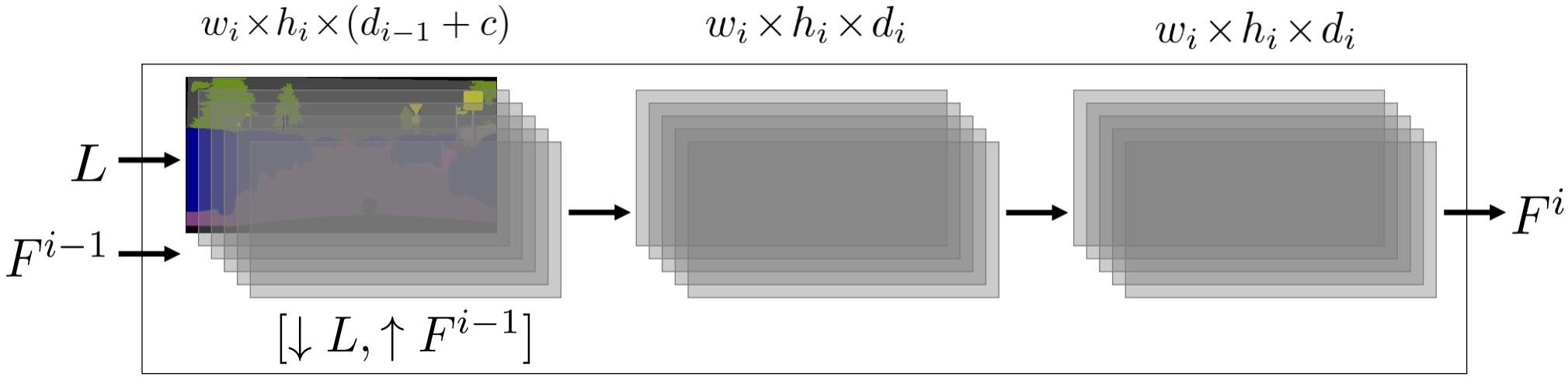


Stack-GAN (Zhang et al., ICCV 2017)

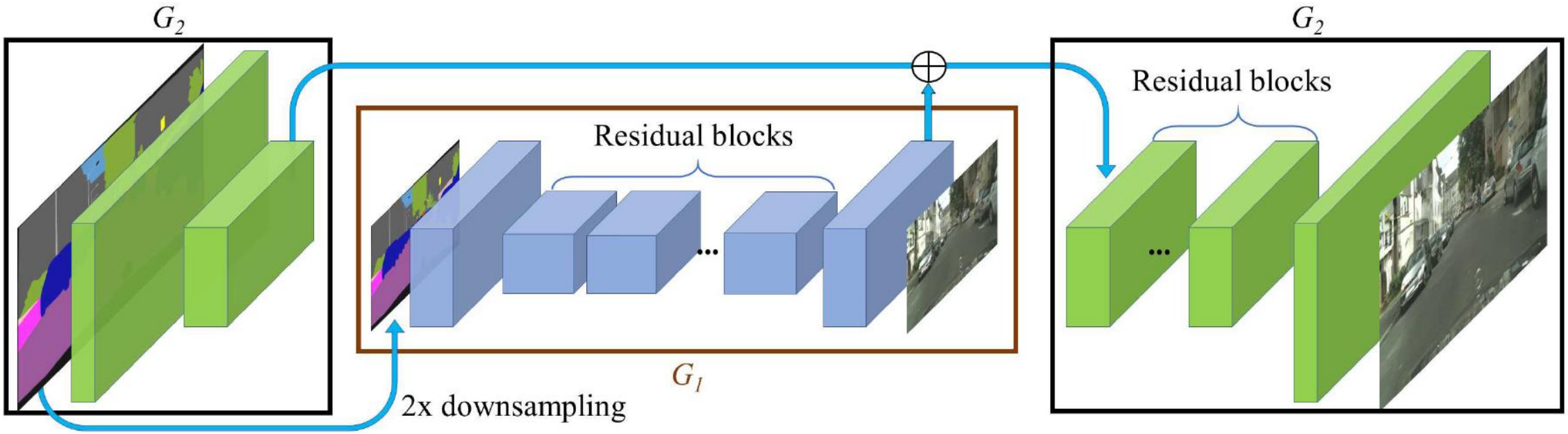


Cascaded Refinement Networks (Chen & Koltun, ICCV 2017)

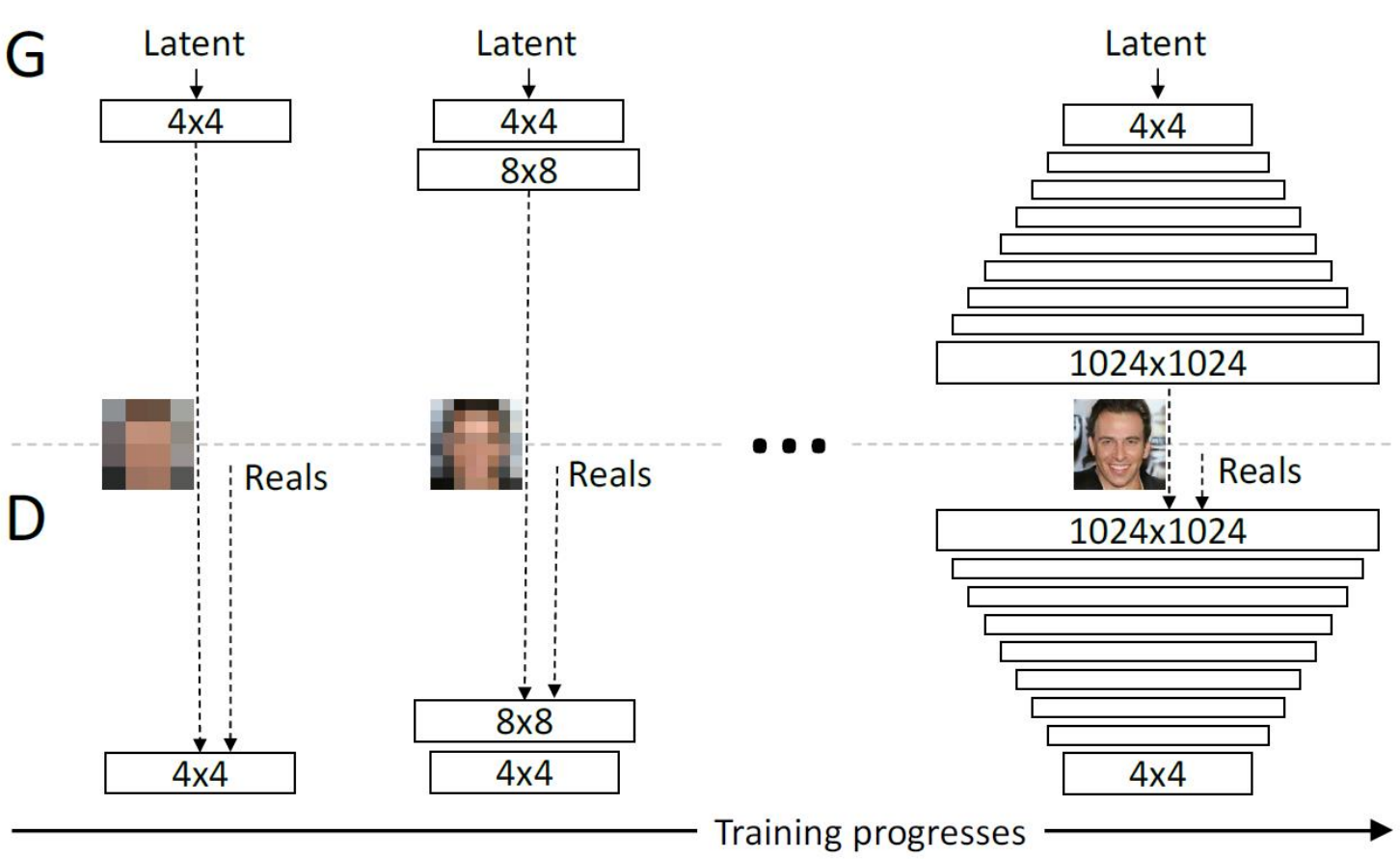
- CRN: not rely on adversarial training

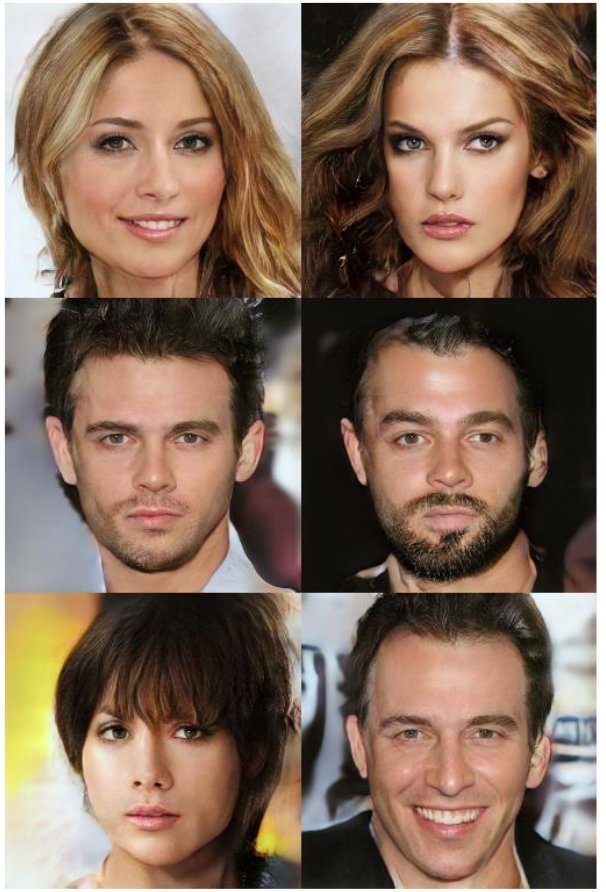


Pix2PixHD (Wang et al., 2018)



Progressive GAN (ICLR 2018)



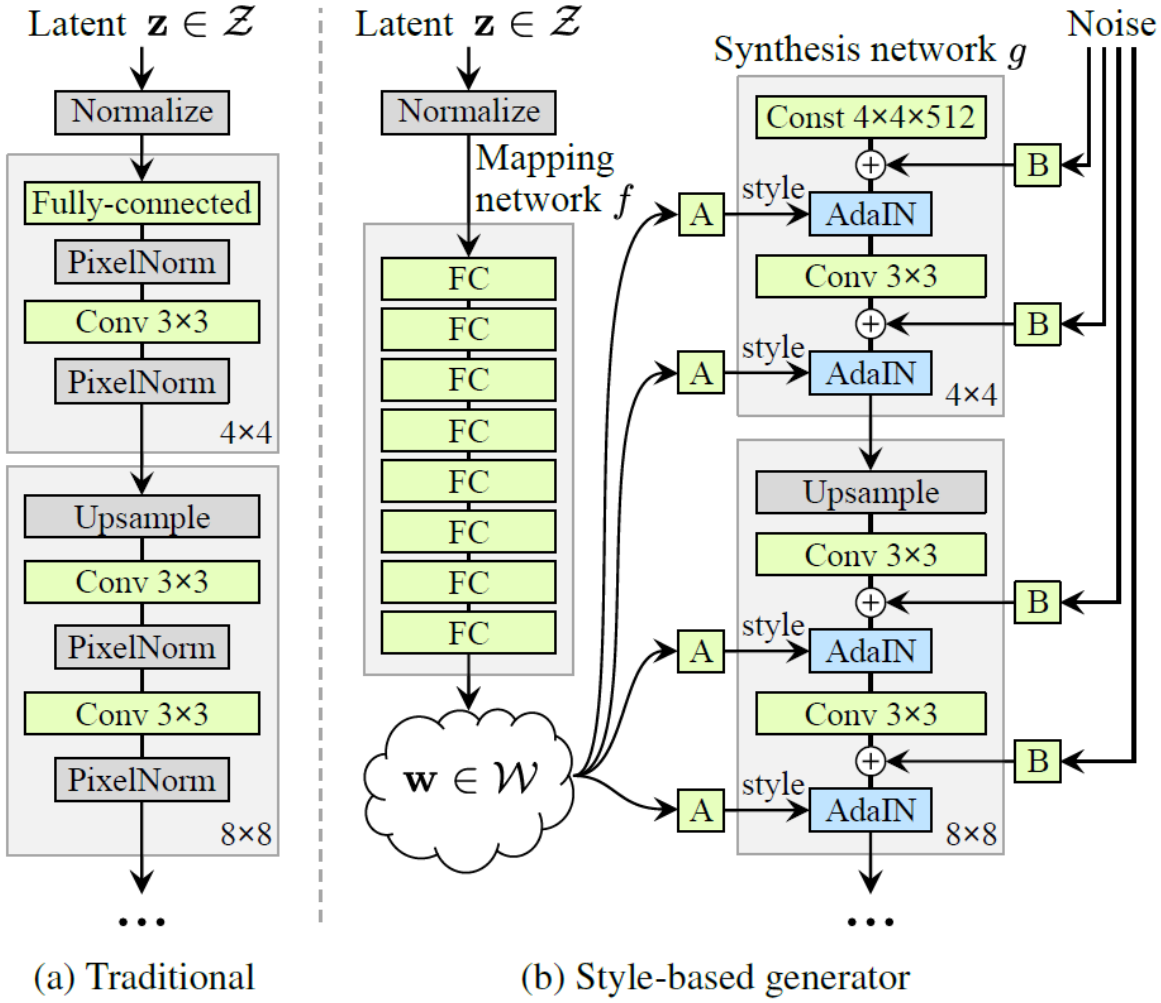


BigGAN

- Orthogonal regularization $R_\beta(W) = \beta \|W^\top W \odot (\mathbf{1} - I)\|_{\mathrm{F}}^2$
- Truncation trick: trade-off between sample fidelity and variety
- Multi-layer manipulation on style and noise

StyleGAN (Karras, CVPR 2019)

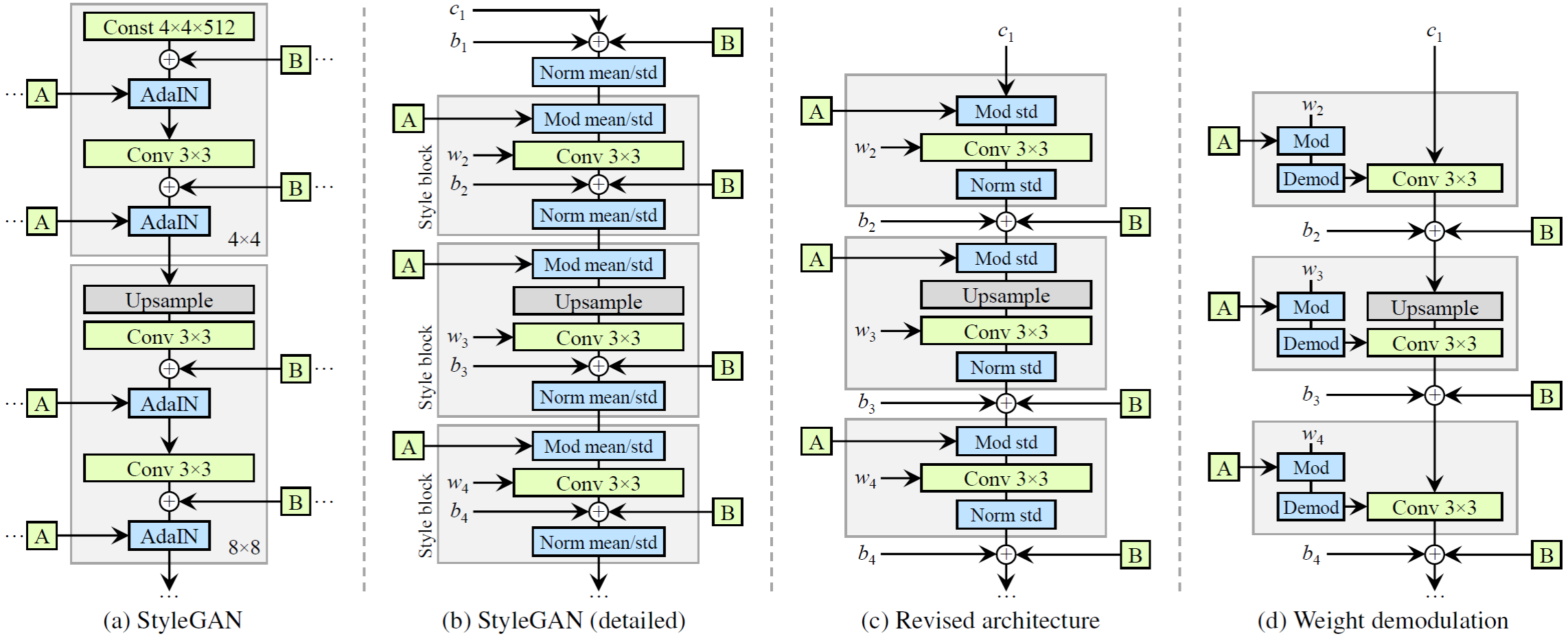
- Progressive GAN
- Multi-layer manipulation on style and noise
- Truncation trick



StyleGAN2 CVPR 2020

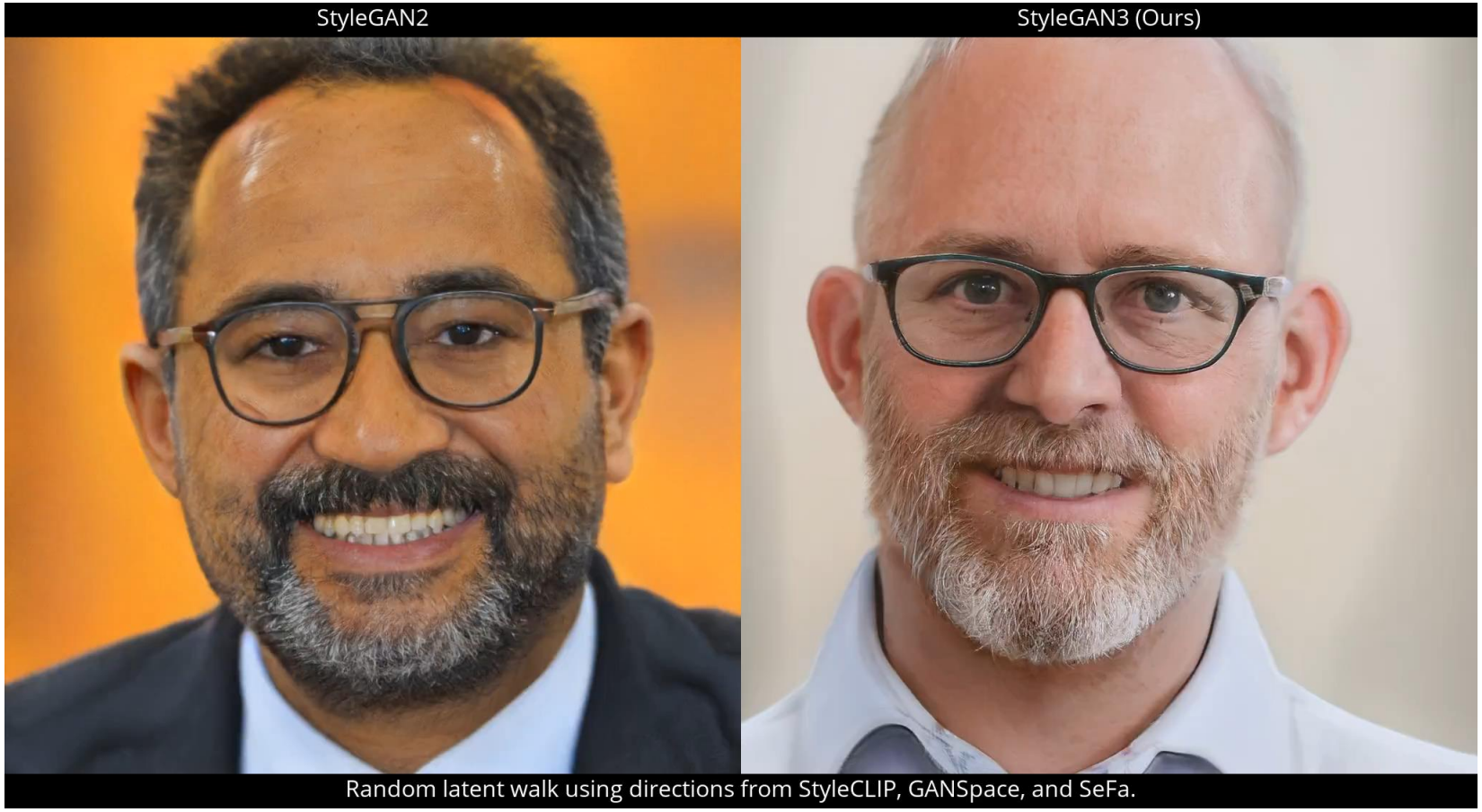
Configuration	FFHQ, 1024×1024				LSUN Car, 512×384			
	FID ↓	Path length ↓	Precision ↑	Recall ↑	FID ↓	Path length ↓	Precision ↑	Recall ↑
A Baseline StyleGAN [21]	4.40	212.1	**0.721**	0.399	3.27	1484.5	**0.701**	0.435
B + Weight demodulation	4.39	175.4	0.702	0.425	3.04	862.4	0.685	0.488
C + Lazy regularization	4.38	158.0	0.719	0.427	2.83	981.6	0.688	0.493
D + Path length regularization	4.34	**122.5**	0.715	0.418	3.43	651.2	0.697	0.452
E + No growing, new G & D arch.	3.31	124.5	0.705	0.449	3.19	471.2	0.690	0.454
F + Large networks (StyleGAN2)	**2.84**	145.0	0.689	**0.492**	**2.32**	**415.5**	0.678	**0.514**
Config A with large networks	3.98	199.2	0.716	0.422	–	–	–	–

StyleGAN2 CVPR 2020

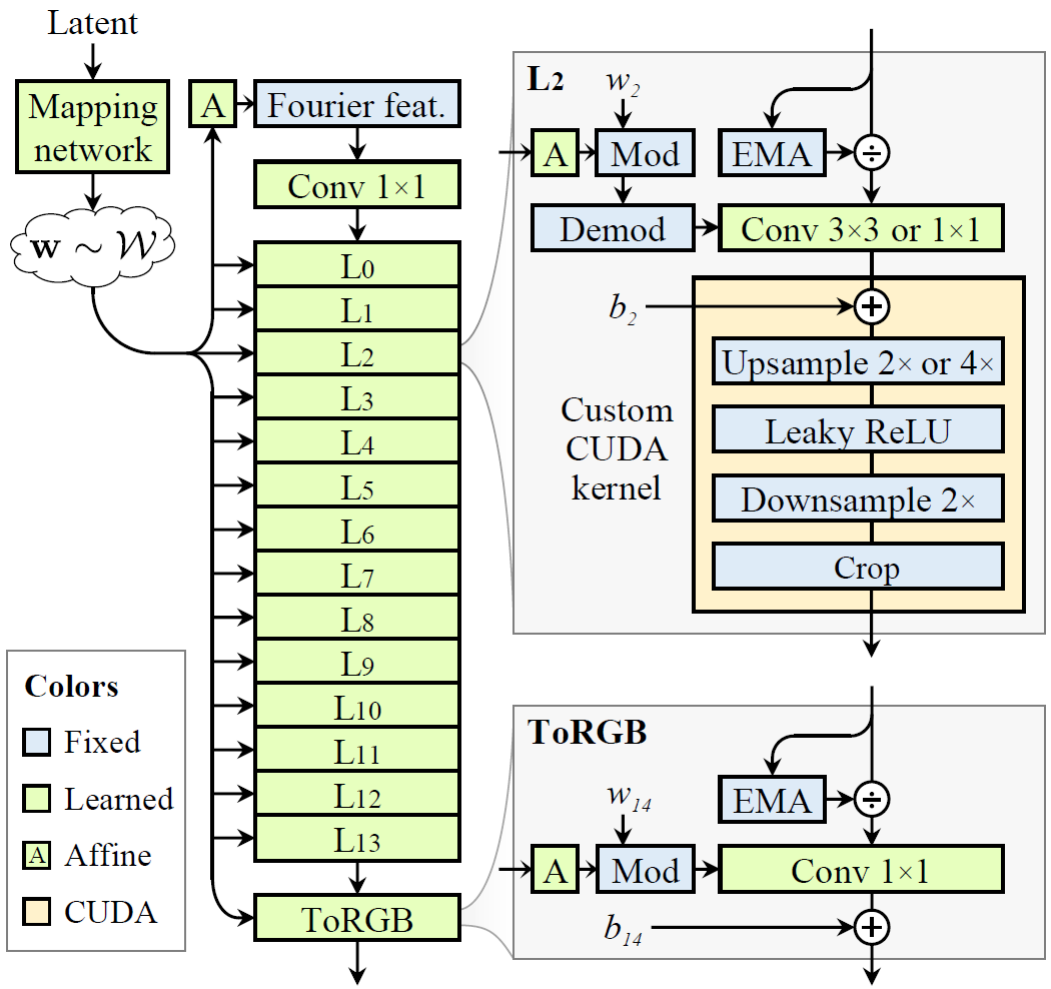


(a) StyleGAN (b) StyleGAN (detailed) (c) Revised architecture (d) Weight demodulation

StyleGAN3 NIPS2021



StyleGAN3 NIPS2021



	Configuration	FID ↓	EQ-T ↑	EQ-R ↑
A	StyleGAN2	5.14	–	–
B	+ Fourier features	4.79	16.23	10.81
C	+ No noise inputs	4.54	15.81	10.84
D	+ Simplified generator	5.21	19.47	10.41
E	+ Boundaries & upsampling	6.02	24.62	10.97
F	+ Filtered nonlinearities	6.35	30.60	10.81
G	+ Non-critical sampling	4.78	43.90	10.84
H	+ Transformed Fourier features	4.64	45.20	10.61
T	+ Flexible layers (StyleGAN3-T)	4.62	63.01	13.12
R	+ Rotation equiv. (StyleGAN3-R)	**4.50**	**66.65**	**40.48**

Regularization Techniques for GANs

- Orthogonal Regularization
- Weight normalization
- Spectral normalization
- Path length regularization

From feature to weight normalization

- BN

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

- 正交正则化

$$\|W^{\mathrm{T}}W - I\|_F^2$$

- 正交正则化的改进

$$R_\beta(W) = \beta\|W^\top W \odot (\mathbf{1} - I)\|_{\mathrm{F}}^2$$

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML 2015*, 2015.
K. Jia, D. Tao, S. Gao, and X. Xu, Improving training of deep neural networks via Singular Value Bounding, CVPR 2017

Weight Normalization

- Weight normalization

$$\sigma_1(\bar{W}_{\mathrm{WN}})^2 + \sigma_2(\bar{W}_{\mathrm{WN}})^2 + \cdots + \sigma_T(\bar{W}_{\mathrm{WN}})^2 = d_o$$

- Spectral normalization

$$\bar{W}_{\mathrm{SN}}(W) := W/\sigma(W)$$

- Path length regularization

$$\mathcal{L}_{\mathrm{pl}} = \mathbb{E}_{\mathbf{w}}\mathbb{E}_{\mathbf{y}}\left(\left\|\mathbf{J}_{\mathbf{w}}^T\mathbf{y}\right\|_2 - a\right)^2$$

T. Salimans and D.P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In NIPS, pp. 901–909, 2016.
T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, ICLR 2018

Take home message

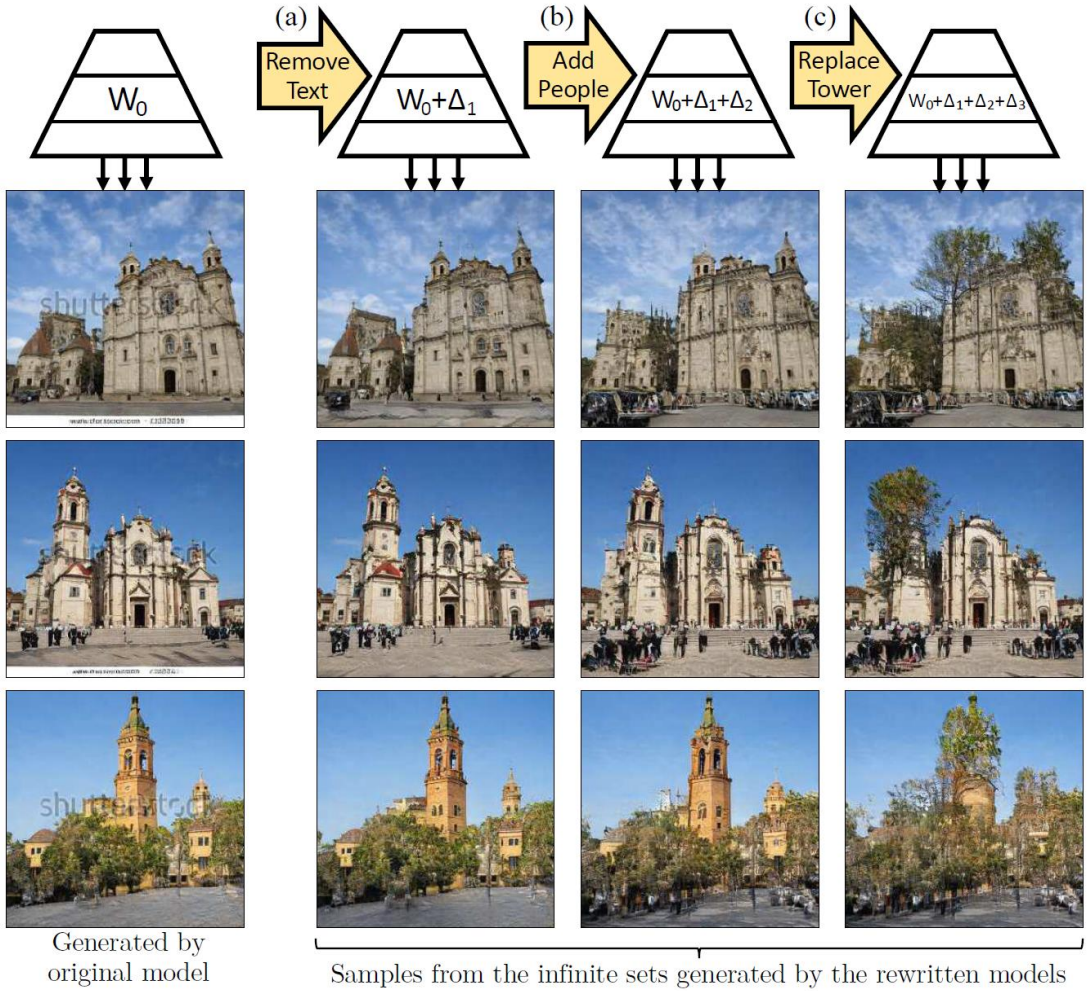
- Exploiting image property to improve GANs
- Progressive training
- Regularization Techniques

Understanding GAN

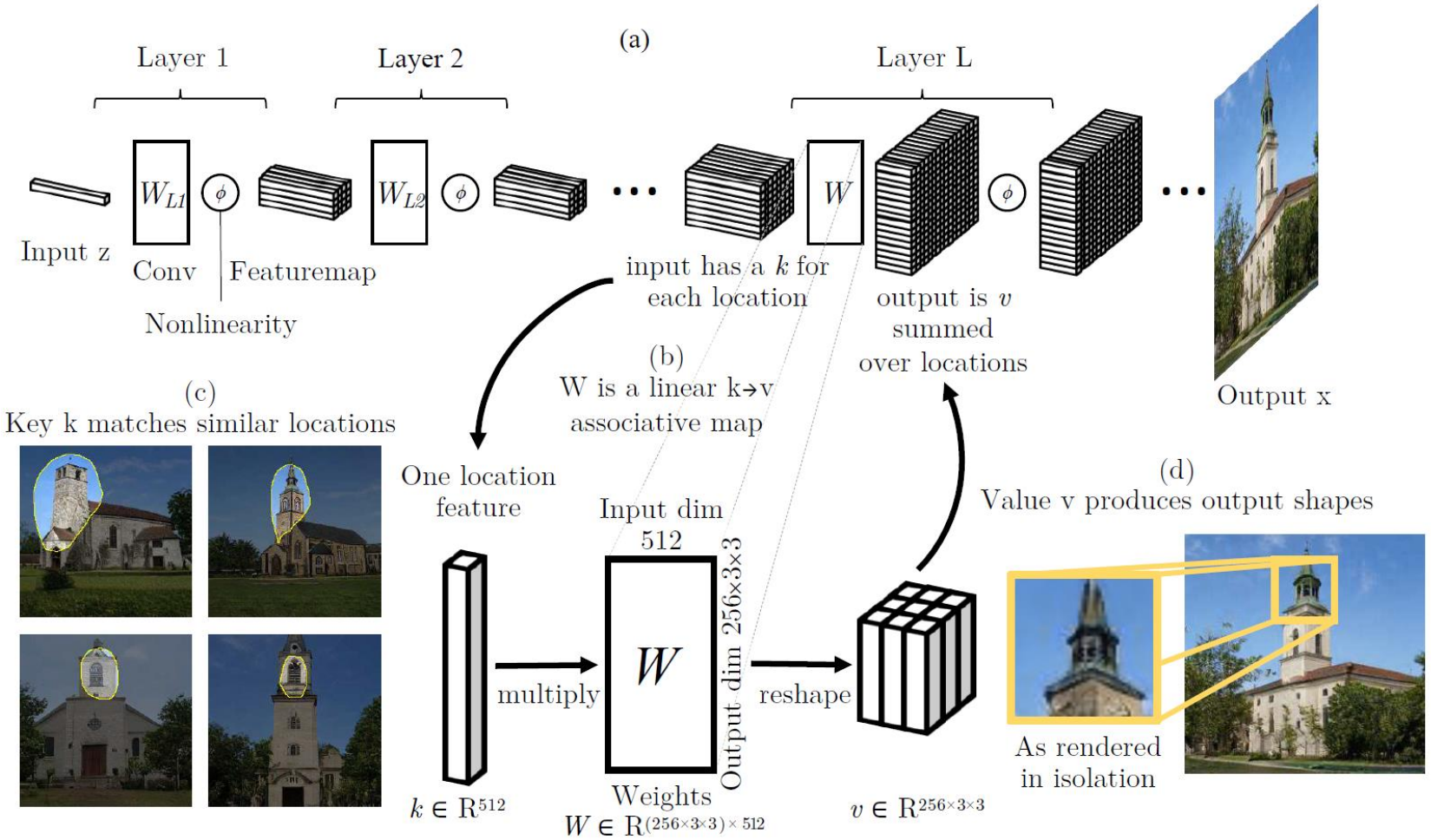
- Neurons/Weights

- Input/Regularization

Rewriting a Deep Generative Model (Bau et al., ECCV 2020)



Rewriting a Deep Generative Model (Bau et al., ECCV 2020)

- 

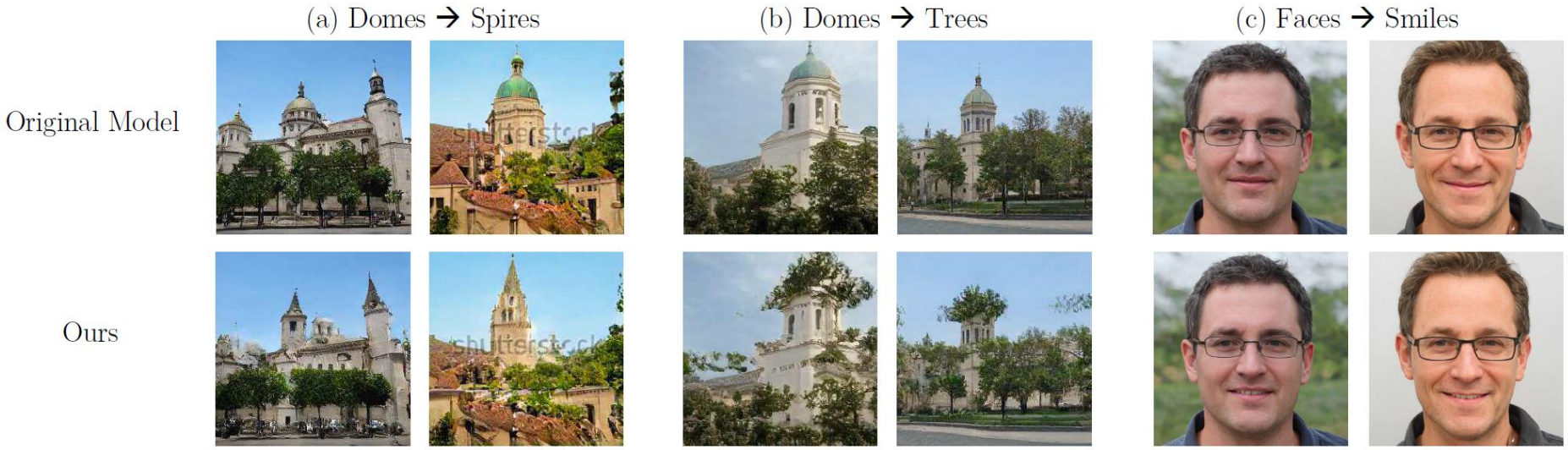
Rewriting a Deep Generative Model (Bau et al., ECCV 2020)

$$W_1 = \arg\min_W ||V - WK||^2$$

$$\text{subject to } v_* = W_1 k_*$$

$$W_1 K K^T = W_0 K K^T + \Lambda k_*{}^T$$

$$W_1 = W_0 + \Lambda (C^{-1} k_*)^T$$



Understanding GAN

- Neurons/Weights

- Input/Regularization

Closed-Form Factorization of Latent Semantics in GANs (CVPR 2021)

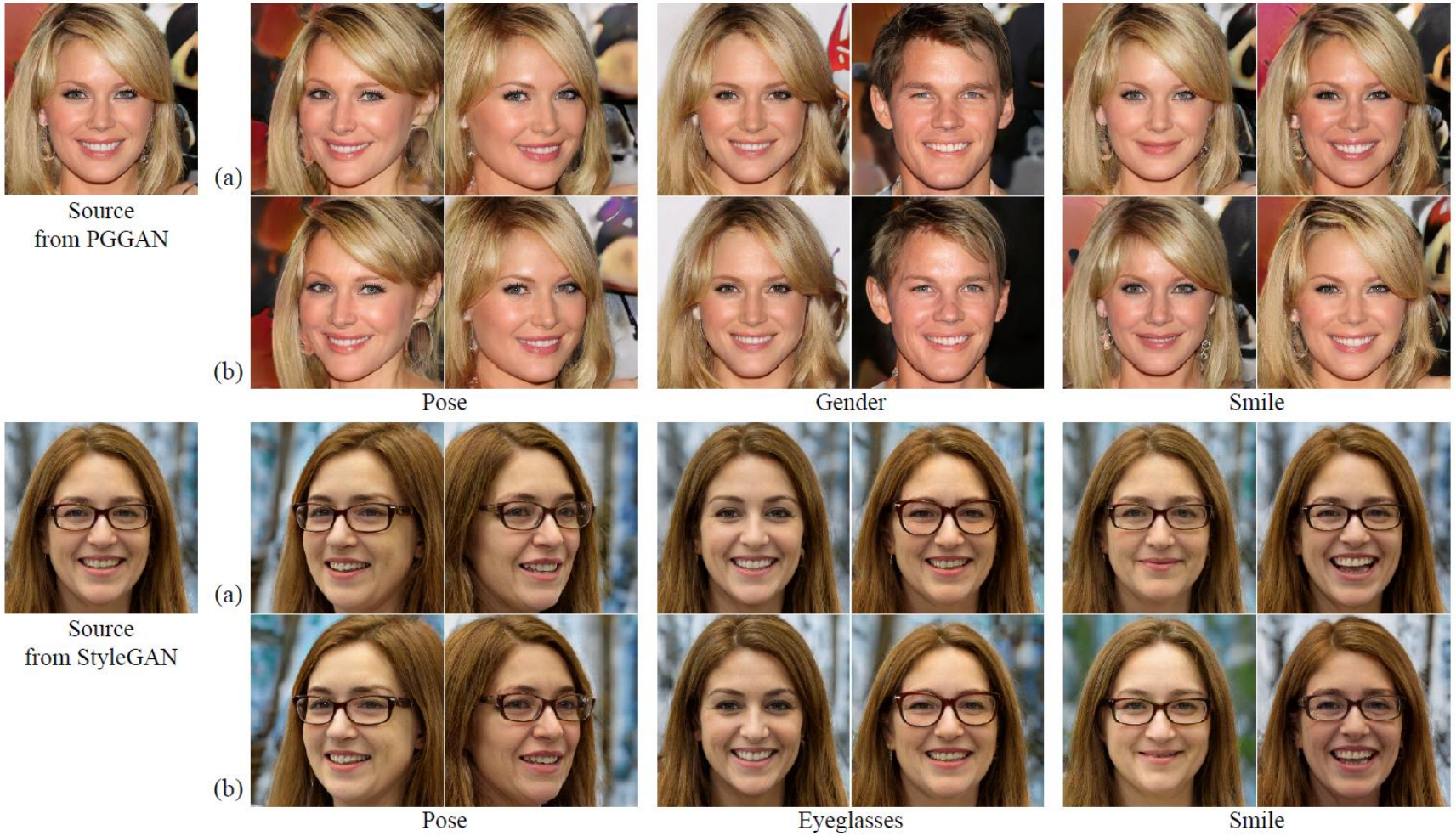
- Only the first fully-connected layer

$$\texttt{edit}(G(\mathbf{z})) = G(\mathbf{z}') = G(\mathbf{z} + \alpha\mathbf{n})$$

- Find n

$$\begin{aligned}\mathbf{N}^* &= \arg\max_{\mathbf{N}\in\mathbb{R}^{d\times k}} \sum_{i=1}^{k} ||\mathbf{A}\mathbf{n}_i||_2^2 - \sum_{i=1}^{k} \lambda_i(\mathbf{n}_i^T\mathbf{n}_i - 1) \\ &= \arg\max_{\mathbf{N}\in\mathbb{R}^{d\times k}} \sum_{i=1}^{k} (\mathbf{n}_i^T\mathbf{A}^T\mathbf{A}\mathbf{n}_i - \lambda_i\mathbf{n}_i^T\mathbf{n}_i + \lambda_i).\end{aligned}$$

Closed-Form Factorization of Latent Semantics in GANs (CVPR 2021)

- 

Closed-Form Factorization of Latent Semantics in GANs (CVPR 2021)

- 
Pose on CelebA-HQ Faces (PGGAN)


Orientation on LSUN Cars (StyleGAN)


Expression on Anime Faces (StyleGAN)


Body Pose on LSUN Cats (StyleGAN)


Pose on ImageNet Magpies (BigGAN)


Layout on LSUN Bedrooms (StyleGAN2)

Hessian Penalty (Peebles et al., ECCV 2020)

- Hessian Penalty

$$H_{ij} = \frac{\partial^2 G}{\partial z_i \partial z_j} = \frac{\partial}{\partial z_j}\left(\frac{\partial G}{\partial z_i}\right) = 0$$

$$\mathcal{L}_H(G) = \sum_{i=1}^{|z|} \sum_{j \neq i}^{|z|} H_{ij}^2$$

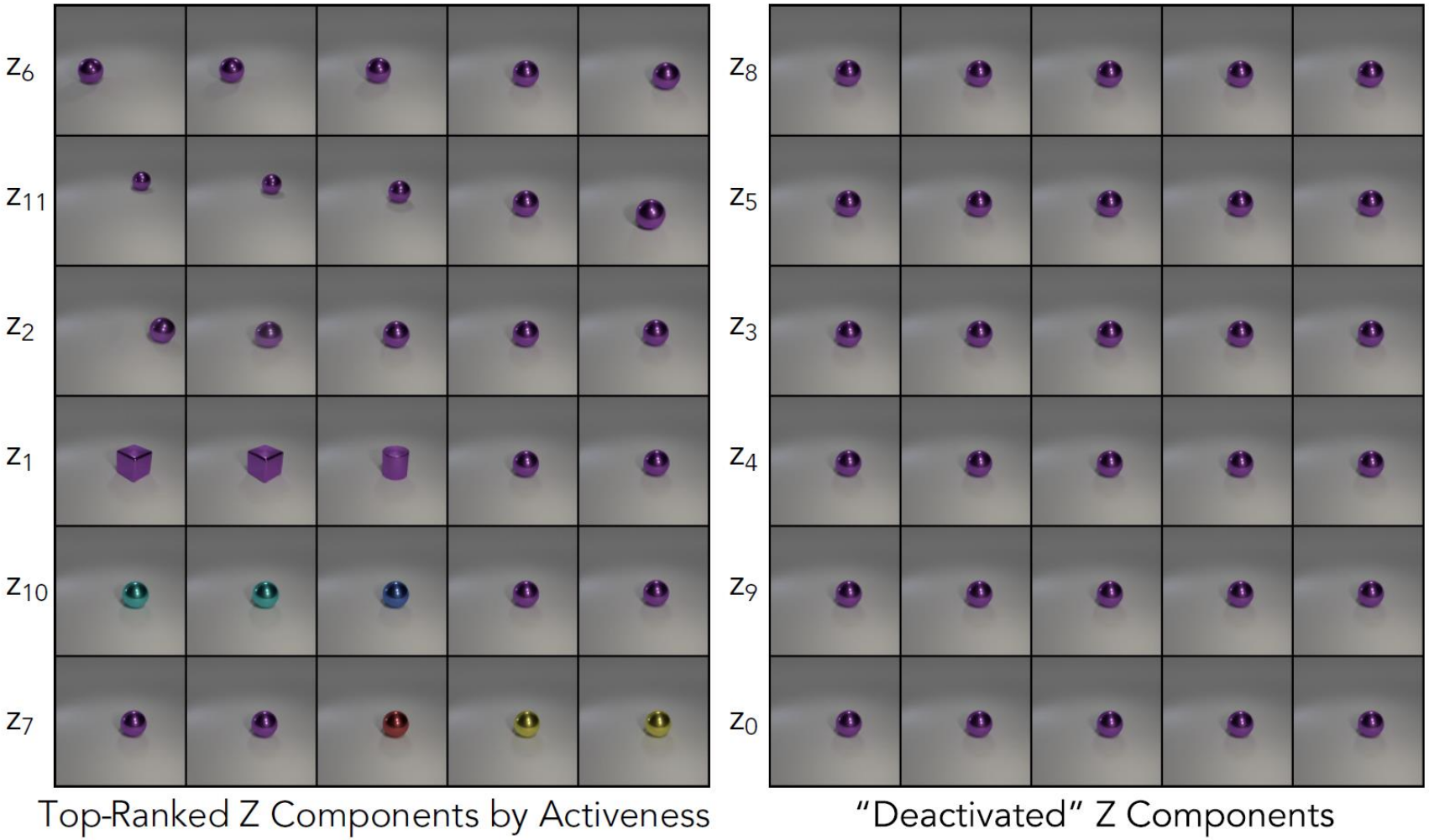
- Generalization to Vector-Valued Functions

$$\mathcal{L}_{\mathbf{H}}(G) = \max_i \mathcal{L}_{\mathbf{H}_i}(G)$$

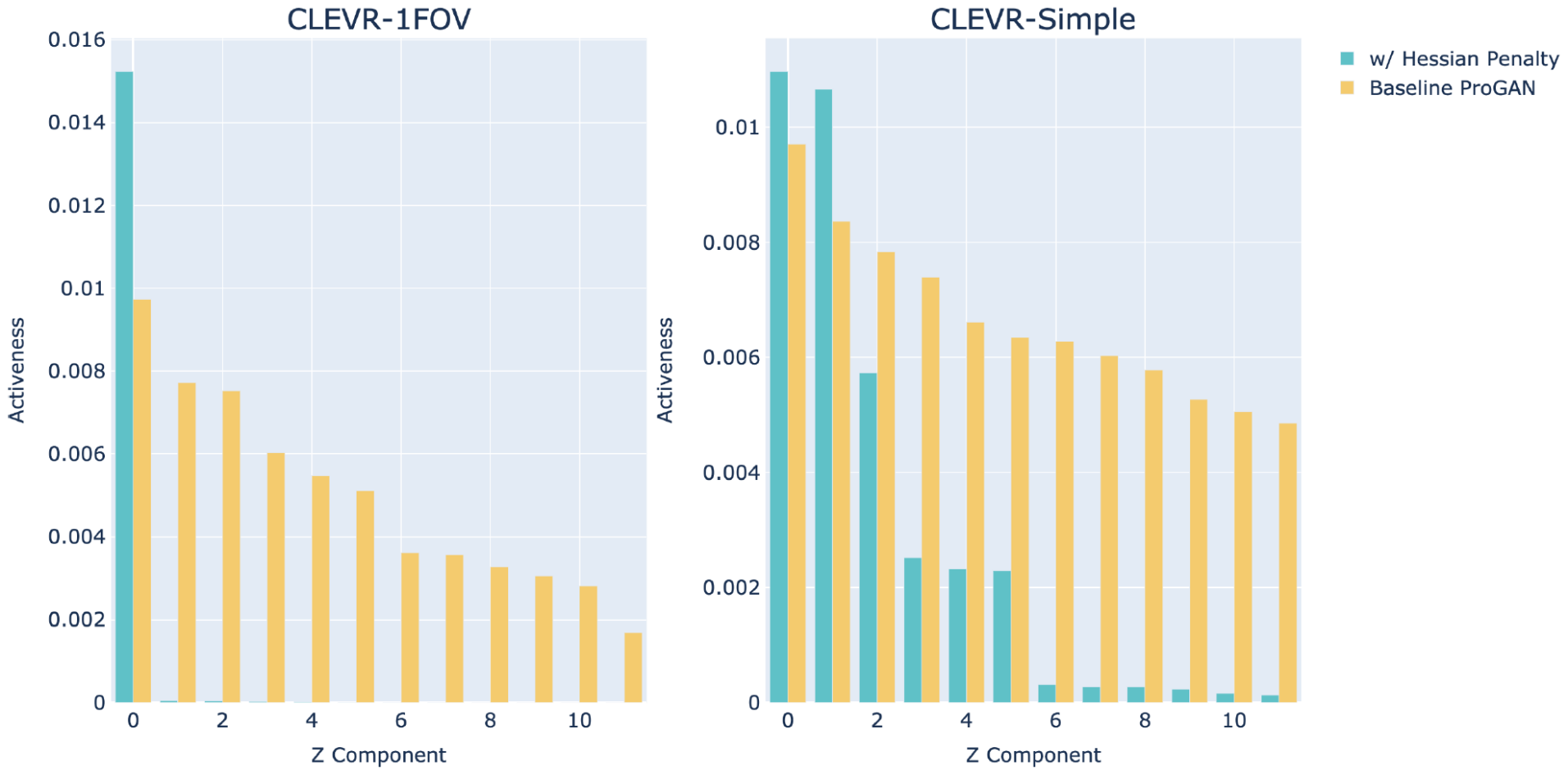
- Efficient Approximation

$$\mathcal{L}_H(G) = \mathrm{Var}_v\left(v^T H v\right)$$

Hessian Penalty (Peebles et al., ECCV 2020)

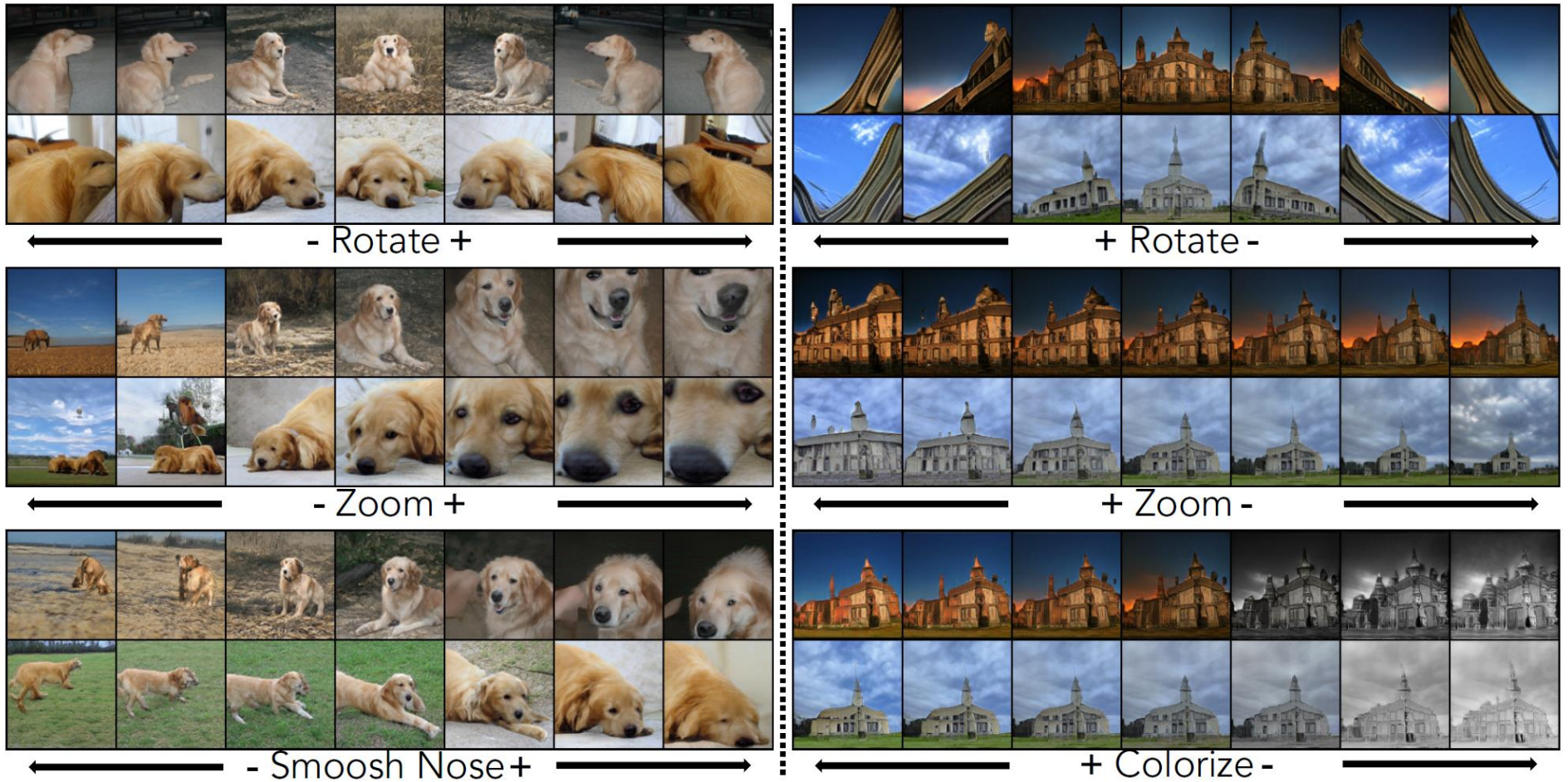
- 

Hessian Penalty (Peebles et al., ECCV 2020)

- 

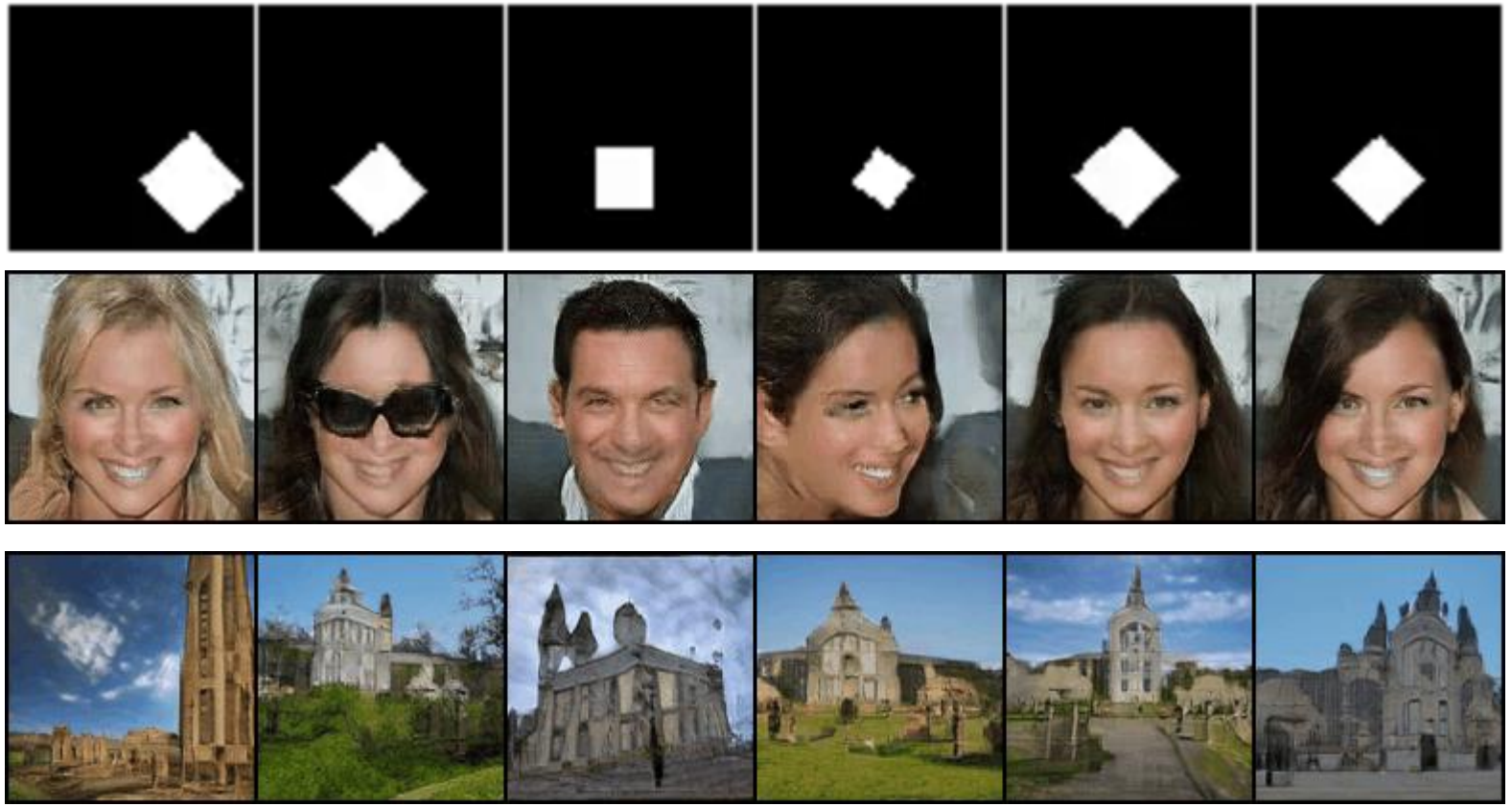
Hessian Penalty (Peebles et al., ECCV 2020)

$$A^* = \arg\min_A \mathbb{E}_{z,w_i,\eta} \mathcal{L}_{\mathbf{H}}(G(z + \eta A w_i))$$

- 

Unsupervised Disentanglement Learning

- Generative Model
  - $G(z_1, z_2, \ldots, z_m)$
- The effect of z_is is disentangled
- Finding redundant z_is



Yuxiang Wei*, Yupeng Shi, Xiao Liu, Zhilong Ji, Yuan Gao, Zhongqin Wu, Wangmeng Zuo, Orthogonal Jacobian Regularization for Unsupervised Disentanglement in Image Generation, ICCV 2021.

OroJaR: Orthogonal Jacobian Regularization

- Orthogonal Jacobian Regularization

$$\mathcal{L}_J(G) = \sum_{d=1}^{D} \|\mathbf{J}_d^T \mathbf{J}_d \circ (\mathbf{1} - \mathbf{I})\| = \sum_{d=1}^{D} \sum_{i=1}^{m} \sum_{j \neq i} \left| \left[\frac{\partial G_d}{\partial z_i} \right]^T \frac{\partial G_d}{\partial z_j} \right|^2$$

- Training Deep Generative Models

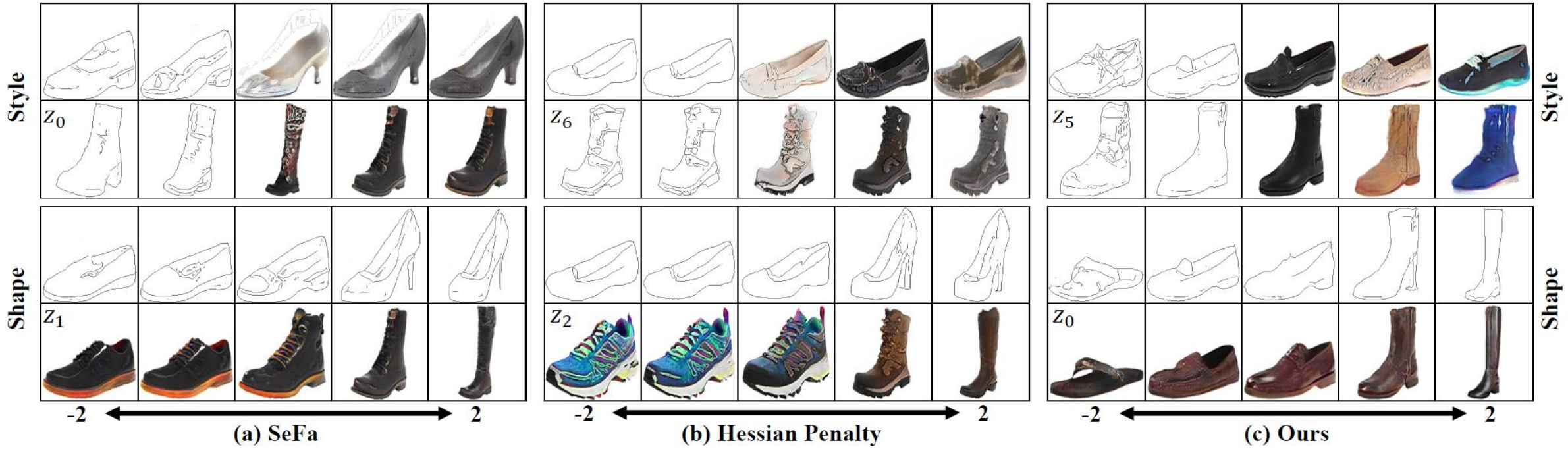
$$\mathcal{L}_G^{oro} = \mathbb{E}_{\mathbf{z}}[f(1 - D(G(\mathbf{z})))] + \lambda \mathbb{E}_{\mathbf{z}}[\mathcal{L}_J(G(\mathbf{z}))]$$

- Interpreting Pre-trained Generator

$$\mathbf{A}^* = \arg\min_{\mathbf{A}} \mathbb{E}_{\mathbf{z},\omega_i} \mathcal{L}_J(G(\mathbf{z} + \eta \mathbf{A} \omega_i))$$

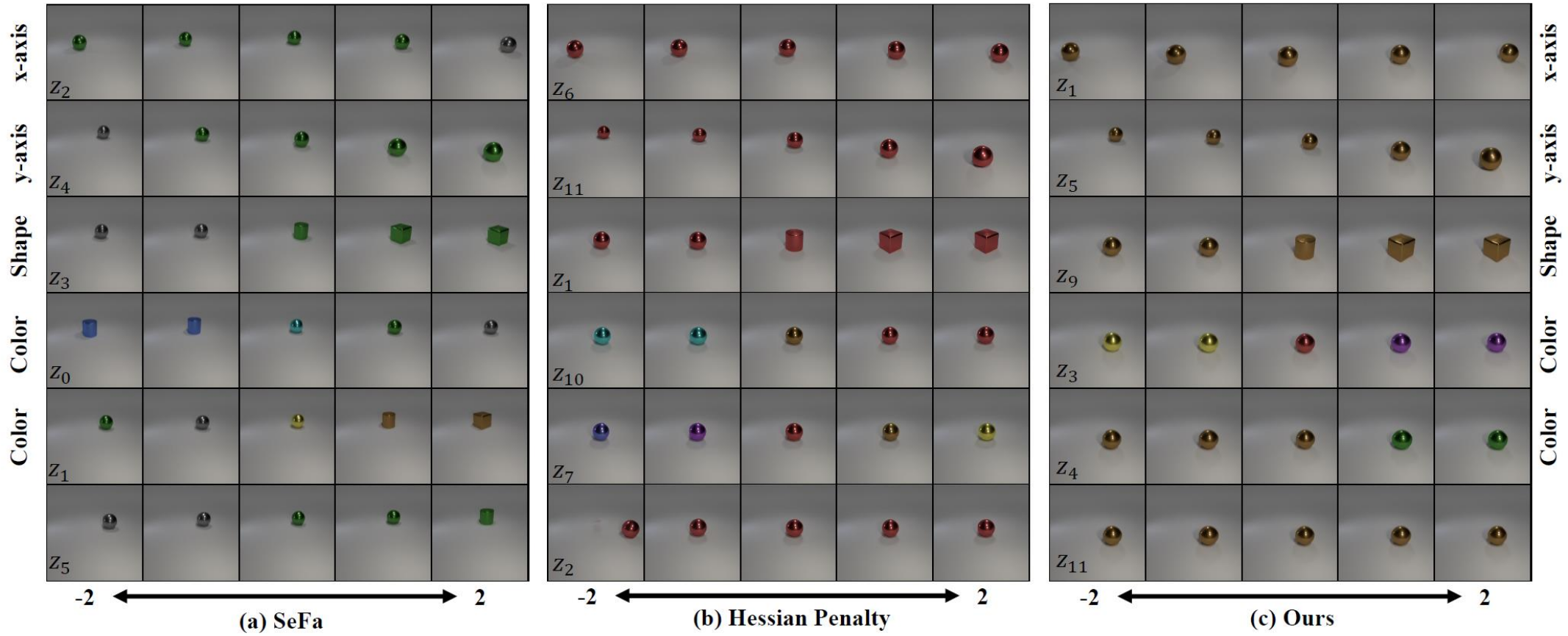
OroJaR: Results

- Edges+Shoes



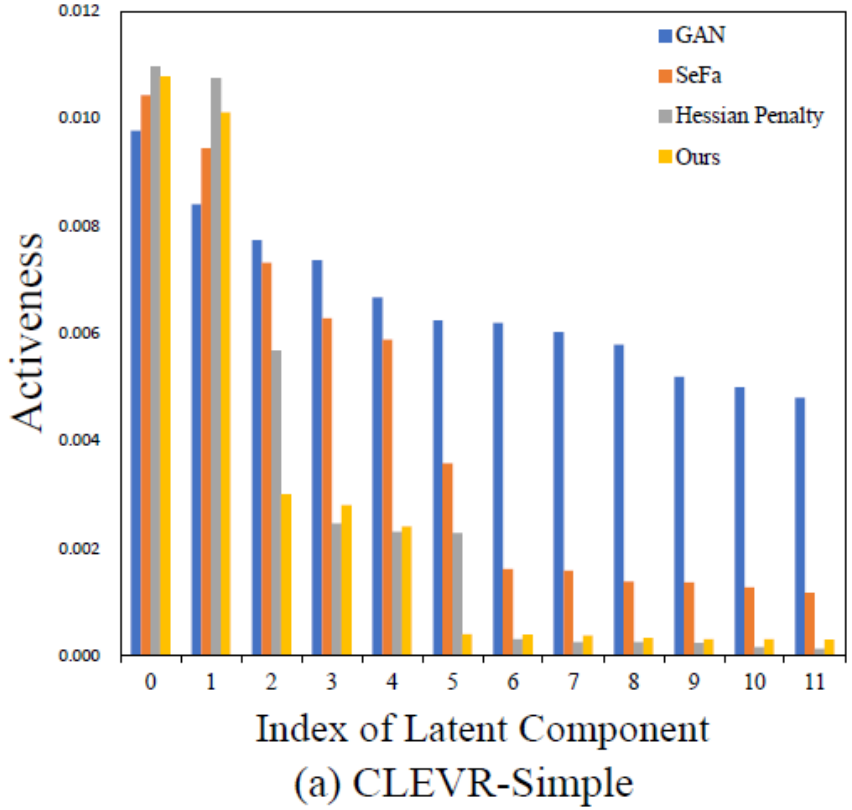
OroJaR: Results

- CLEVR-Simple

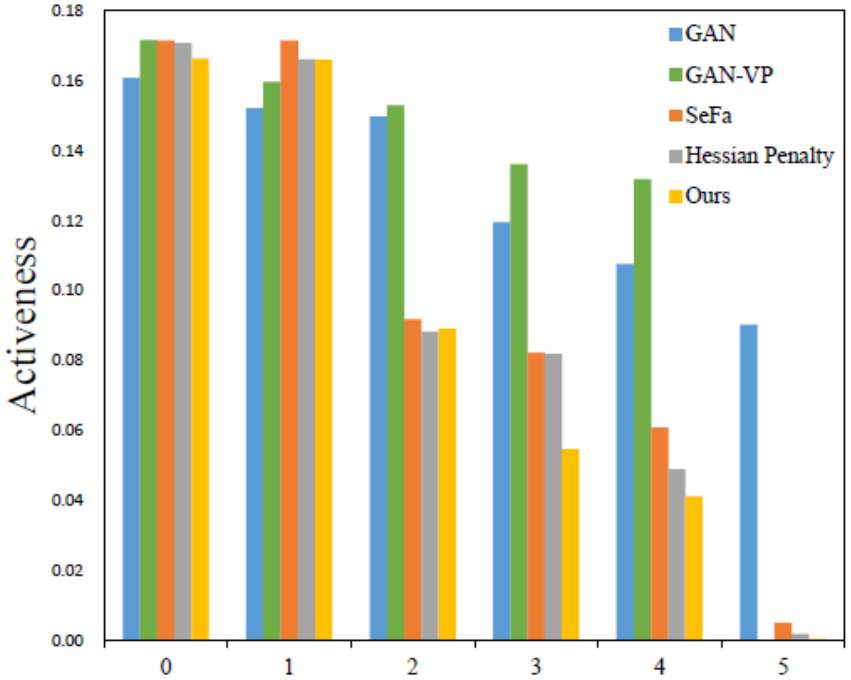


OroJaR: Results

- Finding redundant z_is



(a) CLEVR-Simple

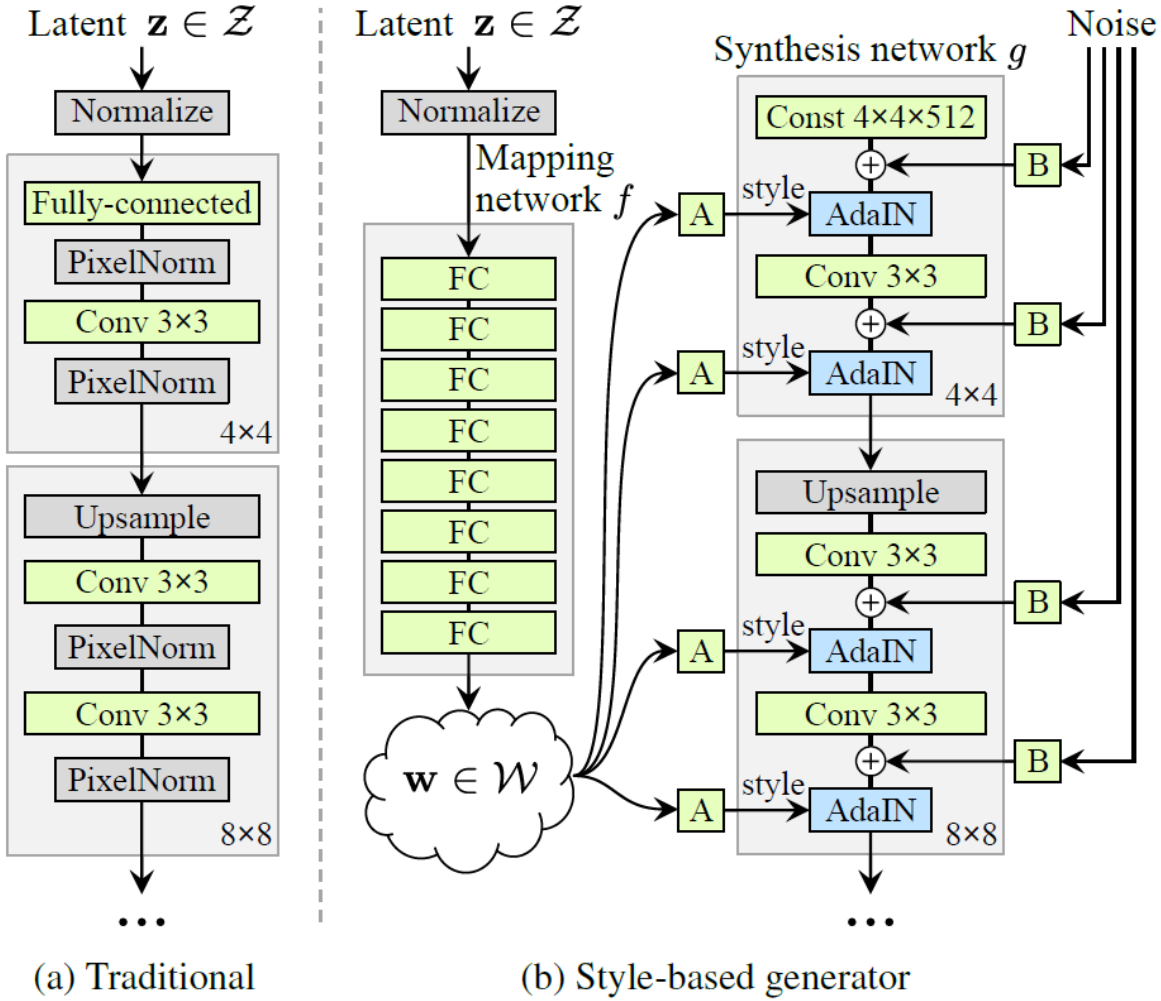


(b) Dsprites

GAN Applications

StyleGAN (Karras, CVPR 2019)

- Progressive GAN
- Multi-layer manipulation on style and noise
- Truncation trick



Tero Karras, Samuli Laine, Timo Aila, A style-based generator architecture for generative adversarial networks, CVPR 2019.

Image2StyleGAN

- Robustness of latent space (Affine / Mask / Class)
- Analyze StyleGAN space and propose $\mathcal{W}^+$ space
- $$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}_{percept}(G(\mathbf{w}), \mathbf{x}) + \lambda_{mse} \left\| G(\mathbf{w}) - \mathbf{x} \right\|_2^2$$

Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?

Rameen Abdal
KAUST
rameen.abdal@kaust.edu.sa

Yipeng Qin
KAUST
yipeng.qin@kaust.edu.sa

Peter Wonka
KAUST
pwonka@gmail.com

Image2StyleGAN

- Applications *Style Mixing (last 9 layers)*

Image2StyleGAN++

- Introducing $\mathcal{N}$ space
- Optimization objective for editing

Image2StyleGAN++: How to Edit the Embedded Images?

Rameen Abdal
KAUST
rameen.abdal@kaust.edu.sa

Yipeng Qin
Cardiff University
qiny16@cardiff.ac.uk

Peter Wonka
KAUST
pwonka@gmail.com

Image2StyleGAN++

- Optimization objective for editing

$$\mathcal{L} = \lambda_{mse_1} \left\| M_m \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{x}) \right\|_2^2 + \lambda_p \mathcal{L}_{percept}(M_p, G(\mathbf{w},\mathbf{n}), \mathbf{x})$$

$$+\lambda_{mse_2} \left\| (1 - M_m) \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{y}) \right\|_2^2 + \lambda_s \mathcal{L}_{style}(M_s, G(\mathbf{w},\mathbf{n}), \mathbf{y})$$

where M_s, M_m, M_p are spatial masks, $\mathcal{L}_{style}$ means $conv3\_3$ of VGG-16, $\mathcal{L}_{percept}$ uses $conv1\_1,1\_2,2\_2,3\_3$ of VGG-16

- $\mathcal{W}^+$ optimization: $\lambda_s = \lambda_{mse_2} = 0, \lambda_{mse_1} = \lambda_p = 10^{-5}$
- $\mathcal{N}$ optimization: $\lambda_s = \lambda_p = 0, \lambda_{mse_1} = \lambda_{mse_2} = 10^{-5}$
- Style Transfer: $\lambda_s = 5 \times 10^{-7}, \lambda_{mse_1} = \lambda_{mse_2} = \lambda_p = 0$

Image2StyleGAN++

$$\mathcal{L} = \lambda_{mse_1} \left\| M_m \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{x}) \right\|_2^2 + \lambda_p \mathcal{L}_{percept}(M_p, G(\mathbf{w},\mathbf{n}), \mathbf{x})$$
$$+ \lambda_{mse_2} \left\| (1 - M_m) \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{y}) \right\|_2^2 + \lambda_s \mathcal{L}_{style}(M_s, G(\mathbf{w},\mathbf{n}), \mathbf{y})$$

- Reconstruction

$\mathcal{W}^+$ for 5k iters, and $\mathcal{N}$ for 3k iters. PSNR 44~45 dB

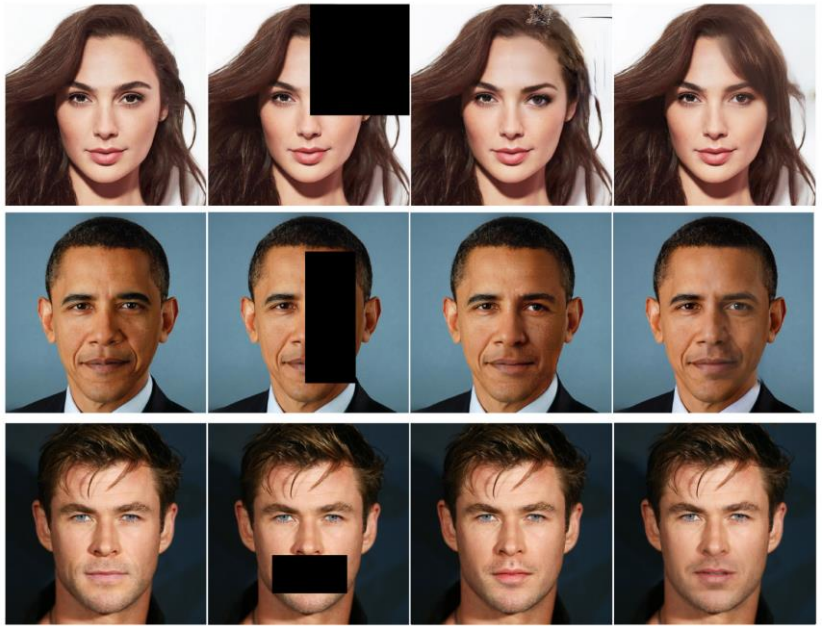
Input I2S I2S++

Image2StyleGAN++

$$\mathcal{L} = \lambda_{mse_1} \left\| M_m \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{x}) \right\|_2^2 + \lambda_p \mathcal{L}_{percept}(M_p, G(\mathbf{w},\mathbf{n}), \mathbf{x})$$
$$+ \lambda_{mse_2} \left\| (1 - M_m) \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{y}) \right\|_2^2 + \lambda_s \mathcal{L}_{style}(M_s, G(\mathbf{w},\mathbf{n}), \mathbf{y})$$

- Image Inpainting

Only optimize 1-9 and 17/18 th *w*



GT Input PConv I2S++

Algorithm 4: Image Inpainting

Input: image $I_{def} \in \mathbb{R}^{n \times m \times 3}$; masks M, M_{blur+}
Output: the embedded code (w_{out}, n_{out})

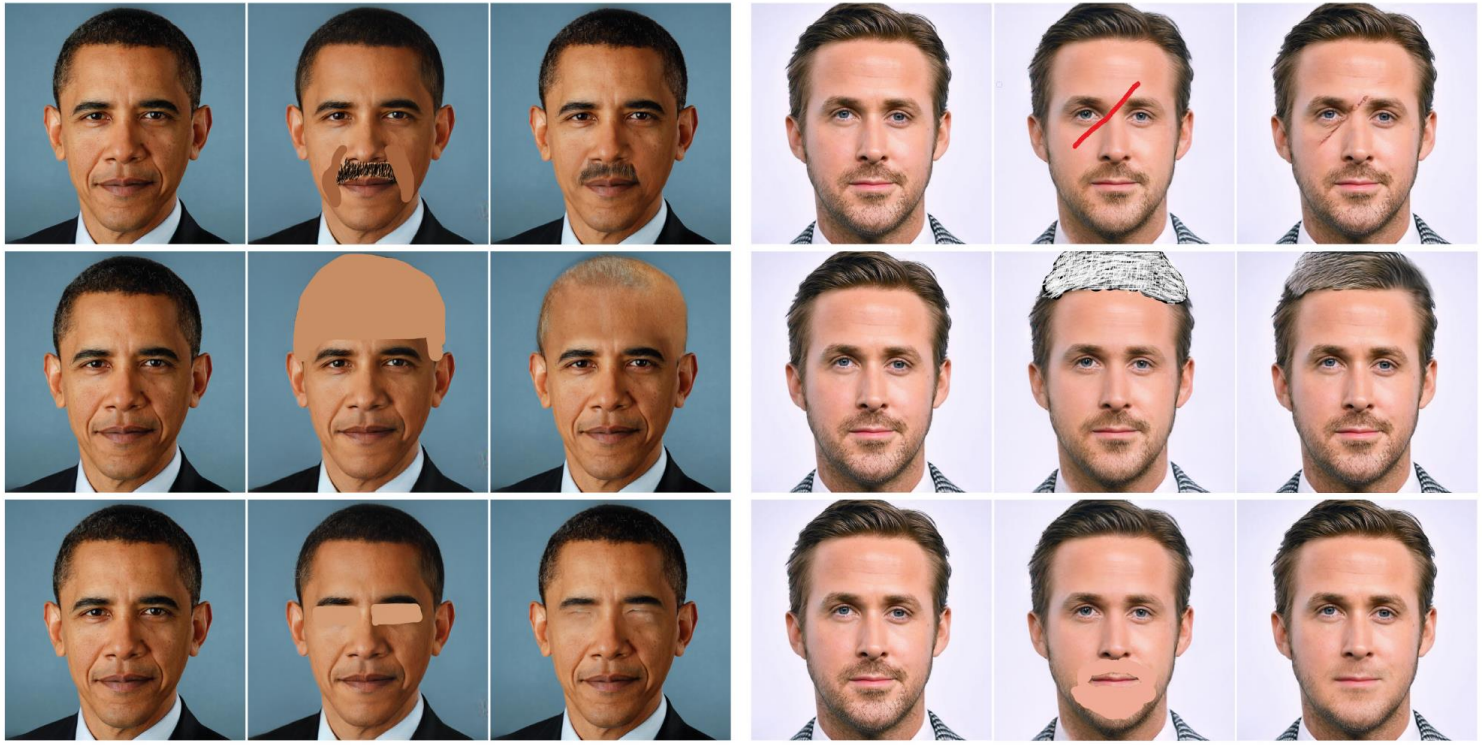
1 $(w_{ini}, n_{ini}) \leftarrow$ initialize();
2 $w_{out} = W_l(1 - M, 1 - M, w_m, w_{ini}, n_{ini}, I_{def})$;
3 $n_{out} =$
$Mk_n(1 - M_{blur+}, w_{out}, n_{ini}, I_{def}, G(w_{out}))$;

Image2StyleGAN++

$$\mathcal{L} = \lambda_{mse_1} \left\| M_m \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{x}) \right\|_2^2 + \lambda_p \mathcal{L}_{percept}(M_p, G(\mathbf{w},\mathbf{n}), \mathbf{x})$$
$$+\lambda_{mse_2} \left\| (1 - M_m) \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{y}) \right\|_2^2 + \lambda_s \mathcal{L}_{style}(M_s, G(\mathbf{w},\mathbf{n}), \mathbf{y})$$

- Local editing

Only optimize 4-6 th *w*



Algorithm 5: Local Edits using Scribble

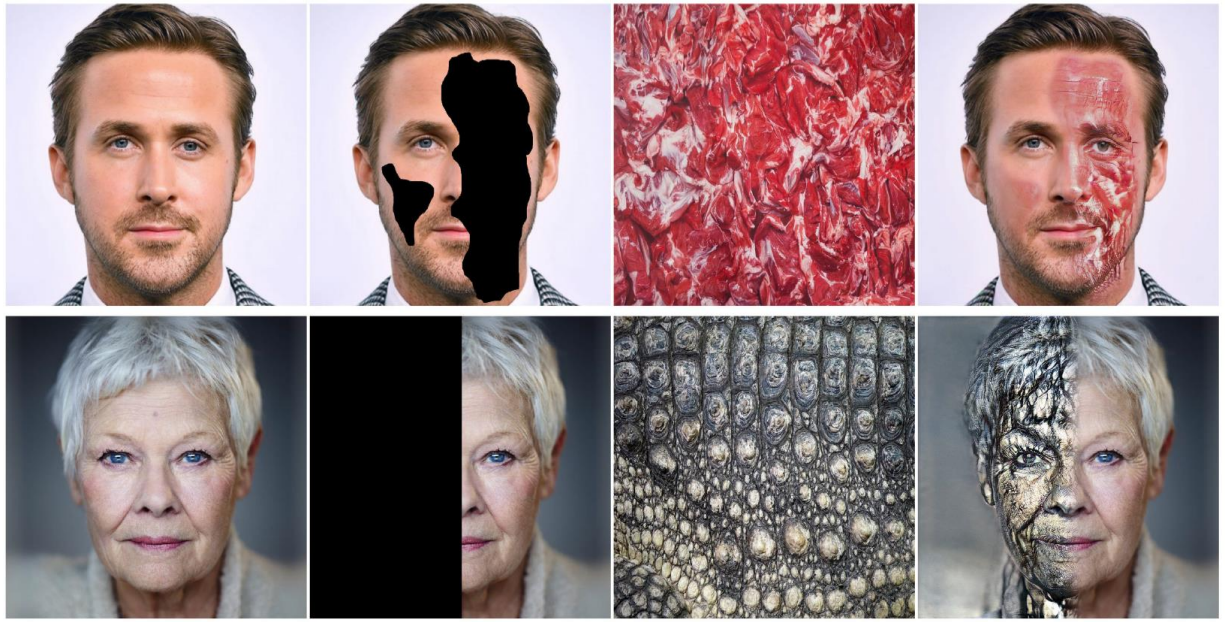
Input: image $I_{scr} \in \mathbb{R}^{n \times m \times 3}$; masks M_{blur}
Output: the embedded code (w_{out}, n_{out})

1 $(w^*, n_{ini}) \leftarrow$ initialize();
2 $w_{out} = W_l(1, 1, w_m, w^*, n_{ini}, I_{scr})$
$+\lambda \|w^* - w_{out}\|_2$;
3 $n_{out} = Mk_n(M_{blur}, w_{out}, n_{ini}, I_{scr}, G(w_{out}))$;

Image2StyleGAN++

$$\mathcal{L} = \lambda_{mse_1} \left\| M_m \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{x}) \right\|_2^2 + \lambda_p \mathcal{L}_{percept}(M_p, G(\mathbf{w},\mathbf{n}), \mathbf{x})$$
$$+ \lambda_{mse_2} \left\| (1 - M_m) \odot (G(\mathbf{w},\mathbf{n}) - \mathbf{y}) \right\|_2^2 + \lambda_s \mathcal{L}_{style}(M_s, G(\mathbf{w},\mathbf{n}), \mathbf{y})$$

- Local style transfer



Algorithm 6: Local Style Transfer

Input: images $I_1, I_2 \in \mathbb{R}^{n \times m \times 3}$; masks M_{blur}
Output: the embedded code (w_{out}, n_{out})

$(w^*, n_{ini}) \leftarrow$ initialize();
$w_{out} = W_l(M_{blur}, M_{blur}, 1, w^*, n_{ini}, I_1)$
$\quad + M_{st}(1 - M_{blur}, w^*, n_{ini}, I_2)$;
$n_{out} = Mk_n(M_{blur}, w_{out}, n_{ini}, I_1, G(w_{out}))$;

pSp

- Learning-based Encoder $pSp(\mathbf{x}) \coloneqq G(E(\mathbf{x}) + \bar{\mathrm{w}})$
- A U-Net shaped Encoder
- LPIPS Loss + $\bar{w}$ regularization + ArcFace-based ID Loss
- *"Invert first, edit later" → One stop operation*

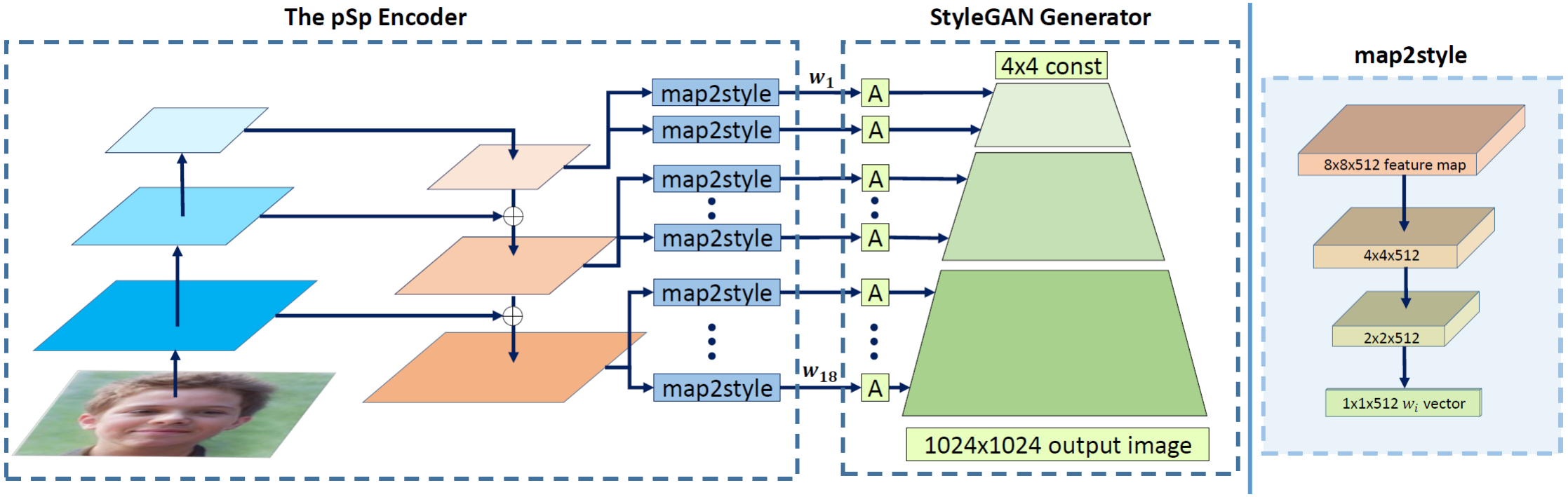
Encoding in Style: a StyleGAN Encoder for Image-to-Image Translation

Elad Richardson[1] Yuval Alaluf[1,2] Or Patashnik[1,2] Yotam Nitzan[2]
Yaniv Azar[1] Stav Shapiro[1] Daniel Cohen-Or[2]

[1]Penta-AI [2]Tel-Aviv University

pSp

- Left: ResNet-50 coarse/medium/fine: 0-2/3-6/7-18



pSp

- Loss function
  - L2

$$\mathcal{L}_2(\mathbf{x}) = \left\|\mathbf{x} - pSp(\mathbf{x})\right\|_2, \text{ where } pSp(\mathbf{x}) = G(E(\mathbf{x}) + \bar{\mathbf{w}})$$

  - LPIPS

$$\mathcal{L}_{LPIPS}(\mathbf{x}) = \left\|F(\mathbf{x}) - F(pSp(\mathbf{x}))\right\|_2$$

  - $\bar{\mathbf{w}}$ Regularization

$$\mathcal{L}_{reg}(\mathbf{x}) = \left\|E(\mathbf{x}) - \bar{\mathbf{w}}\right\|_2$$

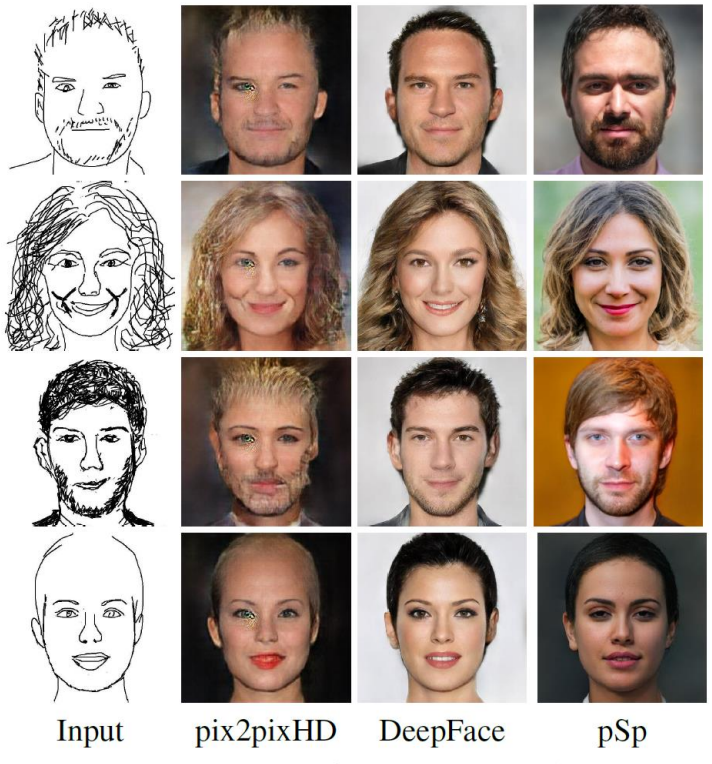
  - ID (R is pre-trained ArcFace Model)

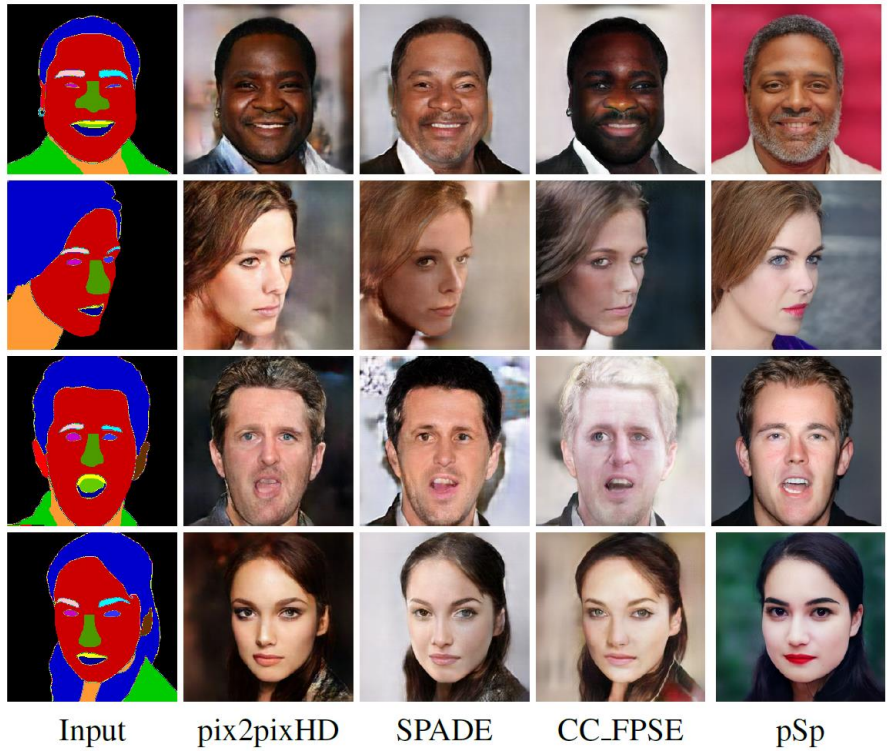
$$\mathcal{L}_{id} = 1 - \left\langle R(\mathbf{x}), R(pSp(\mathbf{x}))\right\rangle$$

pSp

- sketch & label to image

$$w = [w_{input}^{1-7}, w_{rand}^{8-18}]$$





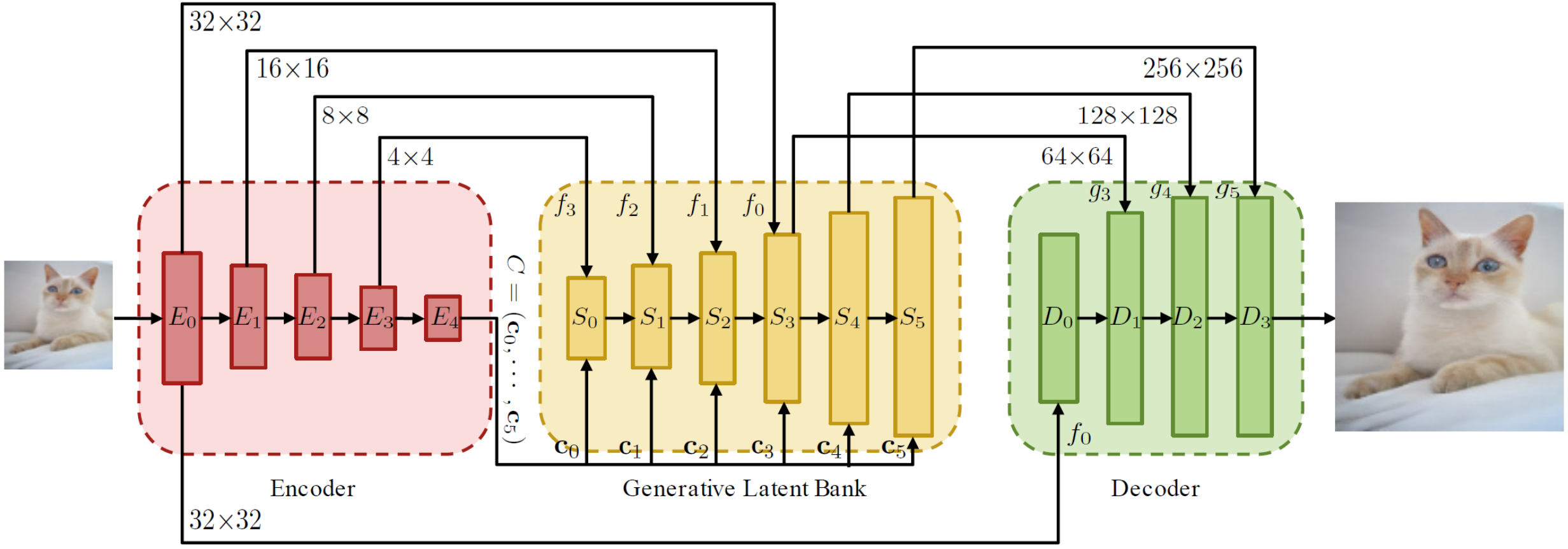
GLEAN

- Learning-based encoder for IR (so many in CVPR 2021)
- Extra refinement network behind StyleGAN

GLEAN: Generative Latent Bank for Large-Factor Image Super-Resolution

Kelvin C.K. Chan[1] Xintao Wang[2] Xiangyu Xu[1] Jinwei Gu[3,4] Chen Change Loy[1*]
[1]S-Lab, Nanyang Technological University
[2]Applied Research Center, Tencent PCG [3]Tetras.AI. [4]Shanghai AI Laboratory
{chan0899, xiangyu.xu, ccloy}@ntu.edu.sg xintao.wang@outlook.com gujinwei@tetras.ai

GLEAN (l2+perceptual+adversarial loss)



GLEAN

- SR

(a) Low-Resolution (b) ESRGAN (c) PULSE (d) GLEAN (ours) (e) Ground-truth

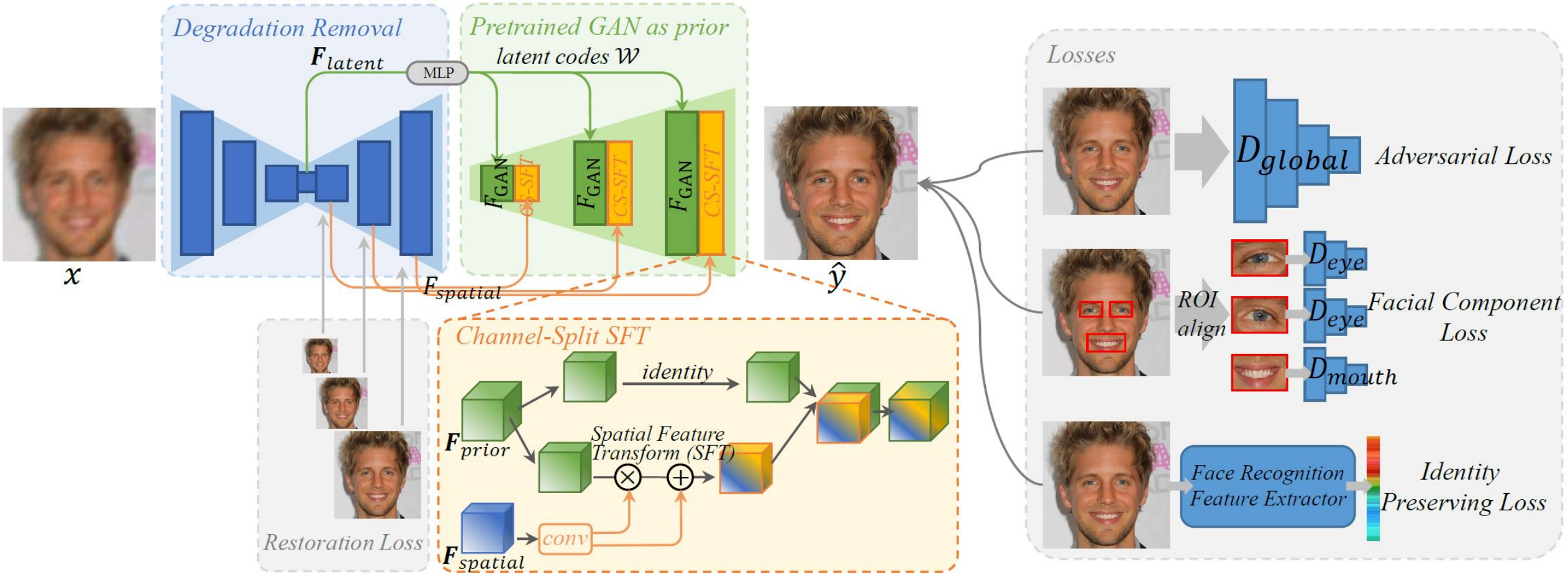
GFPGAN

- A U-Net based Encoder for *degradation removal*
- Applying Spatial Feature Transform (STF) in Pre-trained GAN

Towards Real-World Blind Face Restoration with Generative Facial Prior

Xintao Wang Yu Li Honglun Zhang Ying Shan
Applied Research Center (ARC), Tencent PCG
{xintaowang, ianyli, honlanzhang, yingsshan}@tencent.com

GFPGAN


Degradation Removal
F_{latent}
MLP
Pretrained GAN as prior
latent codes $\mathcal{W}$
F_{GAN}
CS-SFT
$F_{spatial}$
x
$\hat{y}$
Losses
D_{global}
Adversarial Loss
ROI align
D_{eye}
D_{eye}
D_{mouth}
Facial Component Loss
Face Recognition Feature Extractor
Identity Preserving Loss
Restoration Loss
Channel-Split SFT
identity
F_{prior}
Spatial Feature Transform (SFT)
conv
$F_{spatial}$

GFPGAN

- Learning Objective

$$\mathcal{L}_{rec} = \lambda_{l1} \left\| \hat{y} - y \right\|_1 + \lambda_{per} \left\| \phi(\hat{y}) - \phi(y) \right\|_1$$

$$\mathcal{L}_{adv} = -\lambda_{adv} \mathbb{E}_{\hat{y}} \text{softplus}(D(\hat{y}))$$

$$\mathcal{L}_{comp} = \sum_{\text{ROI}} \lambda_{local} \mathbb{E}_{\hat{y}_{\text{ROI}}} [\log(1 - D_{\text{ROI}}(\hat{\mathbf{y}}_{\text{ROI}})] + \lambda_{fs} \left\| Gram(\psi(\hat{y}_{ROI})) - Gram(\psi(y_{ROI})) \right\|_1$$

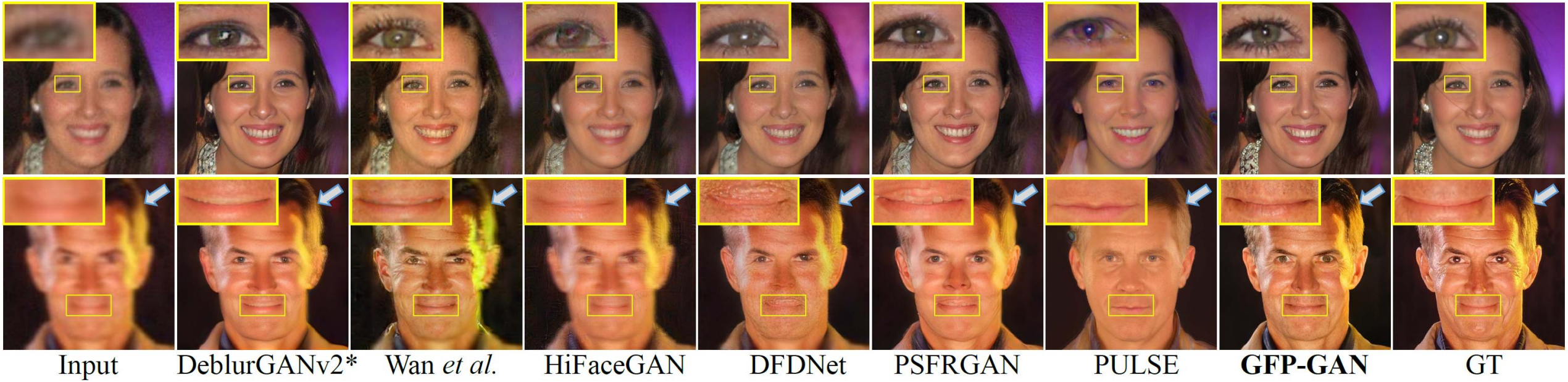
- Degradation Model

$$\mathbf{x} = [(\mathbf{y} * \mathbf{k}_\sigma) \downarrow_r + \mathbf{n}_\delta]_{JPEG_q}$$

$$\sigma = \{0.2 : 10\}, r = \{1 : 8\}, \delta = \{0 : 15\}, q = \{60 : 100\}$$

GFPGAN

- SR results



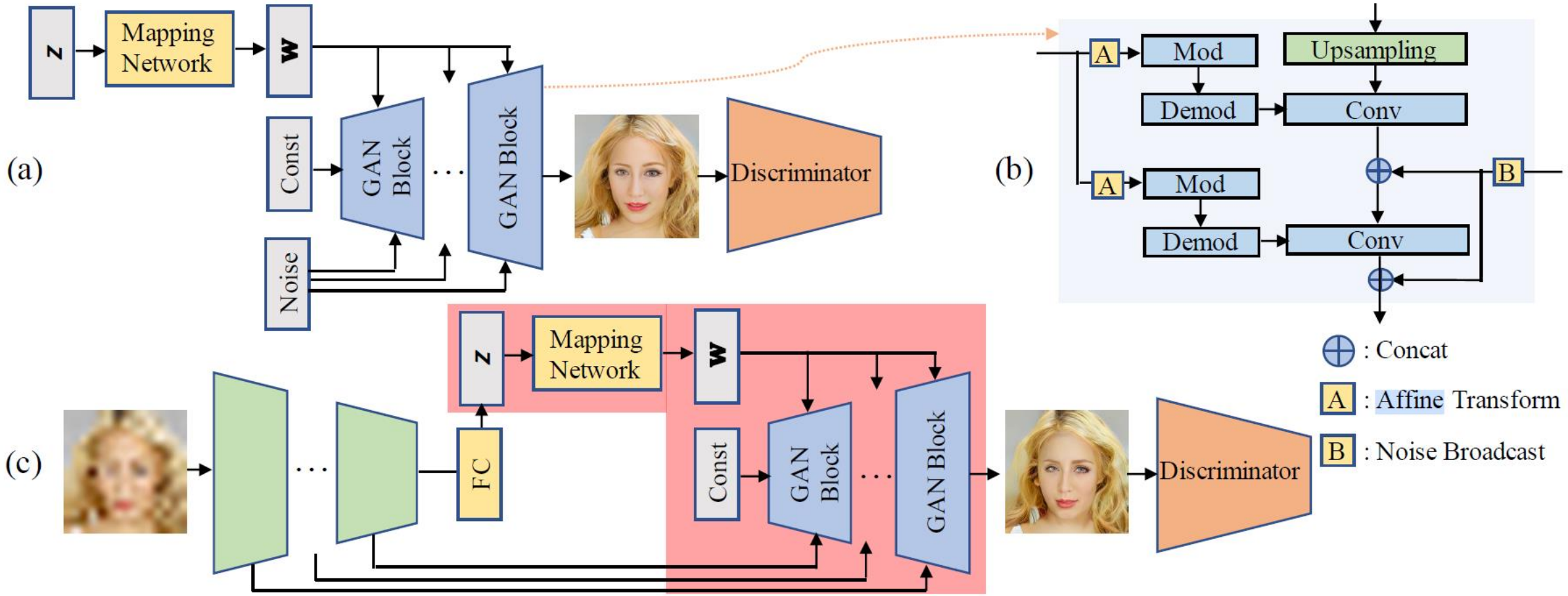
GPEN

- Both $\mathcal{W}^+$ and $\mathcal{N}$ space
- Jointly fine-tune the Generator

GAN Prior Embedded Network for Blind Face Restoration in the Wild

Tao Yang[1], Peiran Ren[1], Xuansong Xie[1], and Lei Zhang[1,2*]
[1]DAMO Academy, Alibaba Group
[2]Department of Computing, The Hong Kong Polytechnic University
yangtao9009@gmail.com, peiran_r@sohu.com, xingtong.xxs@taobao.com, cslzhang@comp.polyu.edu.hk

GPEN


z
Mapping Network
w
Const
GAN Block
GAN Block
Noise
Discriminator
(a)
A
Mod
Upsampling
Demod
Conv
B
A
Mod
Demod
Conv
(b)
: Concat
A : Affine Transform
B : Noise Broadcast
z
Mapping Network
w
FC
Const
GAN Block
GAN Block
Discriminator
(c)

GPEN

- Learning Objective ($LR_{enc}: LR_{dec}: LR_{dis} = 100: 10: 1$)

$$L_C = \min_G \left\| X - \tilde{X} \right\|_1$$

$$L_A = \min_G \max_D E_X \log(1 + \exp(-D(G(\tilde{X}))))$$

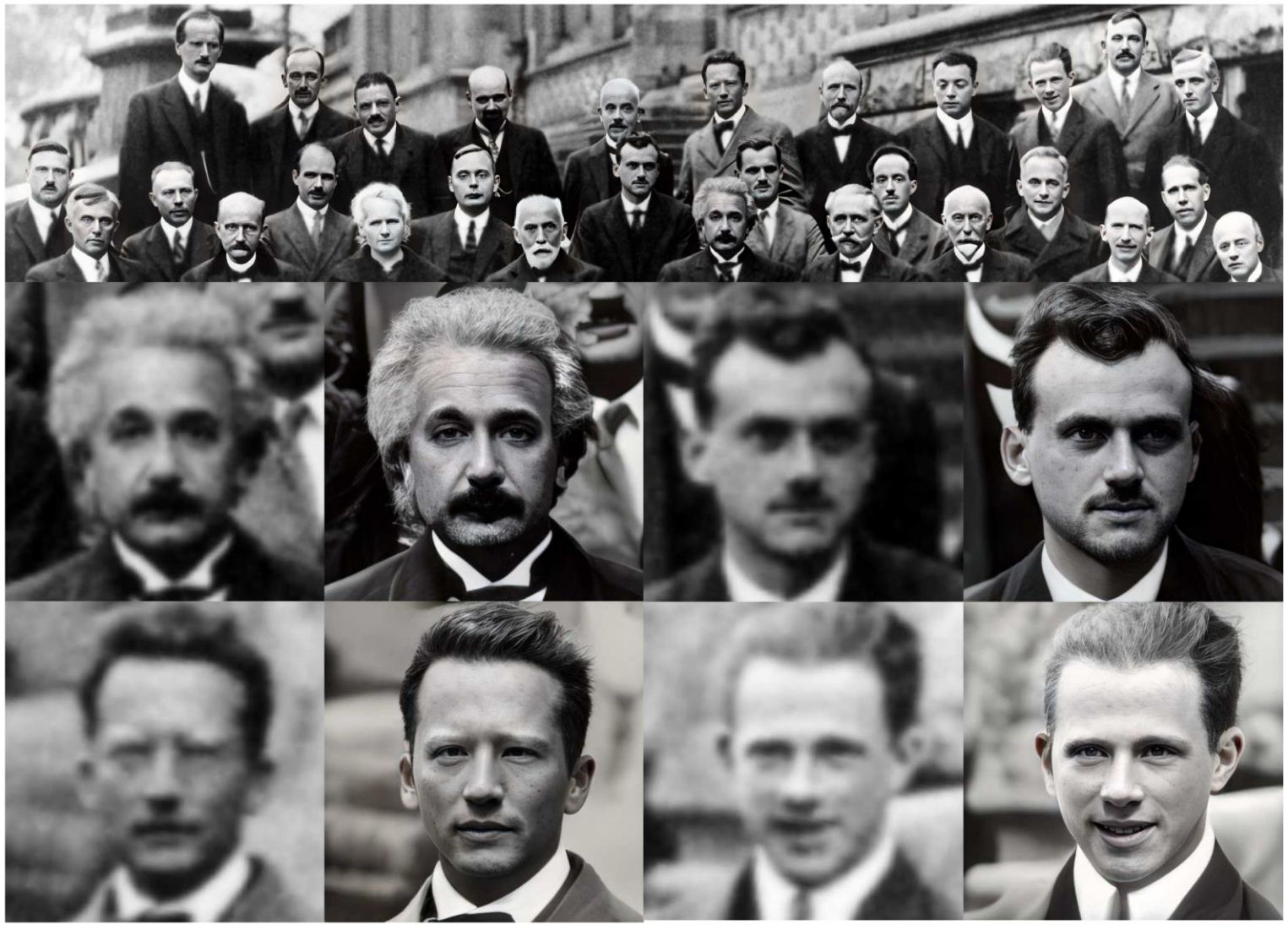
$$L_F = \min_G E_X \left(\sum_{i=0}^{T} \left\| D^i(X) - D^i(G(\tilde{X})) \right\|_2 \right)$$

- Degradation Model

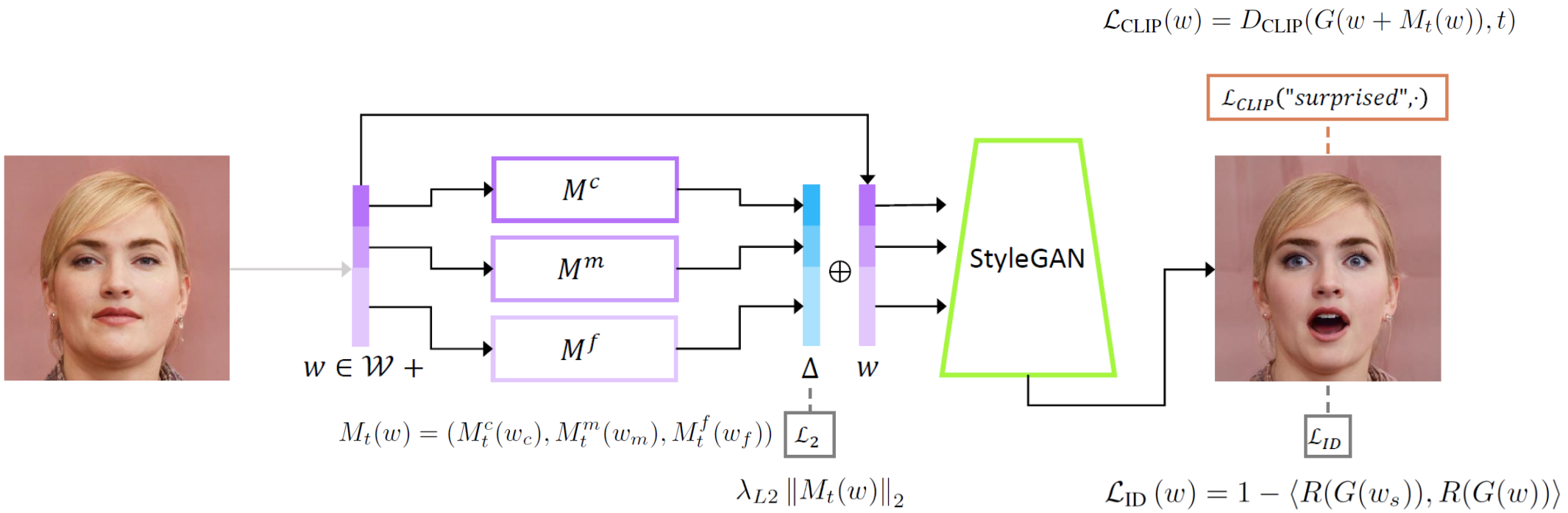
$$I^d = ((I \otimes \mathbf{k}) \downarrow_s + \mathbf{n}_\sigma)_{JPEG_q}$$

$$s = \{10:200\}, \sigma = \{0:25\}, q = \{5:50\}$$

GPEN

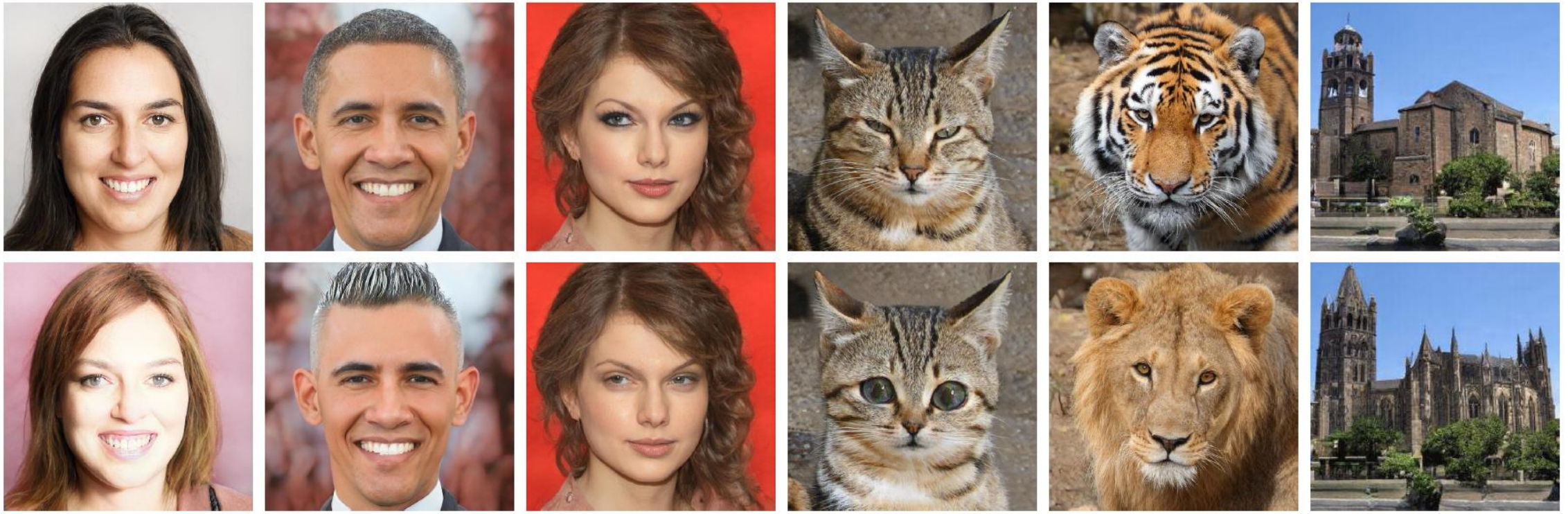


StyleCLIP: Text-Driven Manipulation



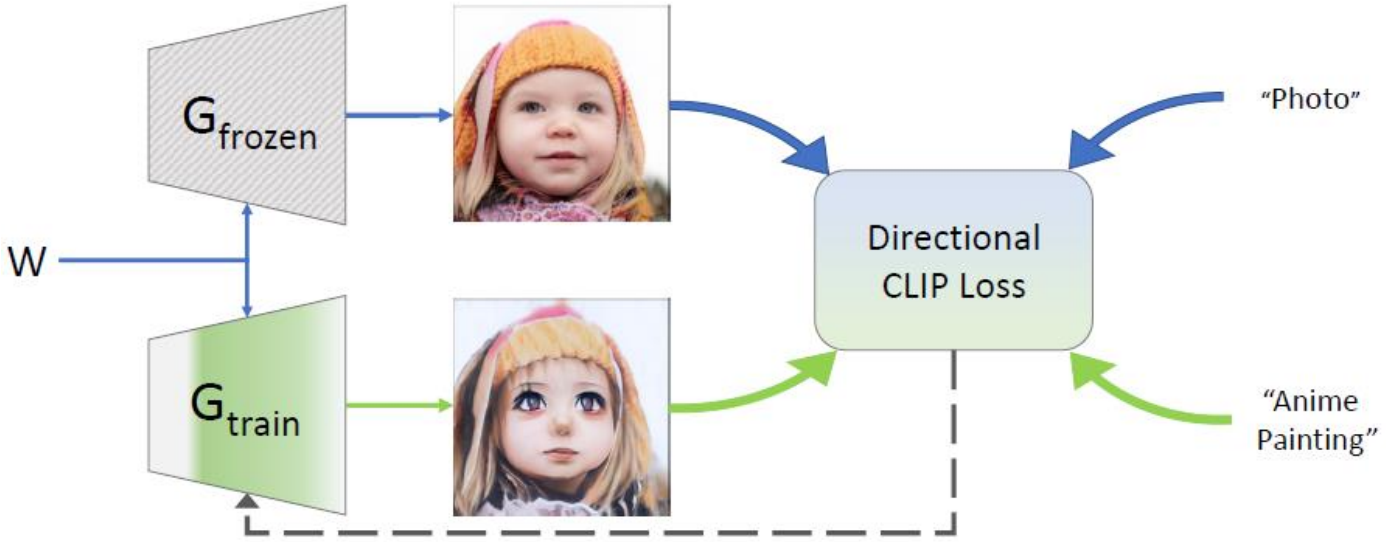
Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, Dani Lischinski, StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery, ICCV 2021.

StyleCLIP: Text-Driven Manipulation

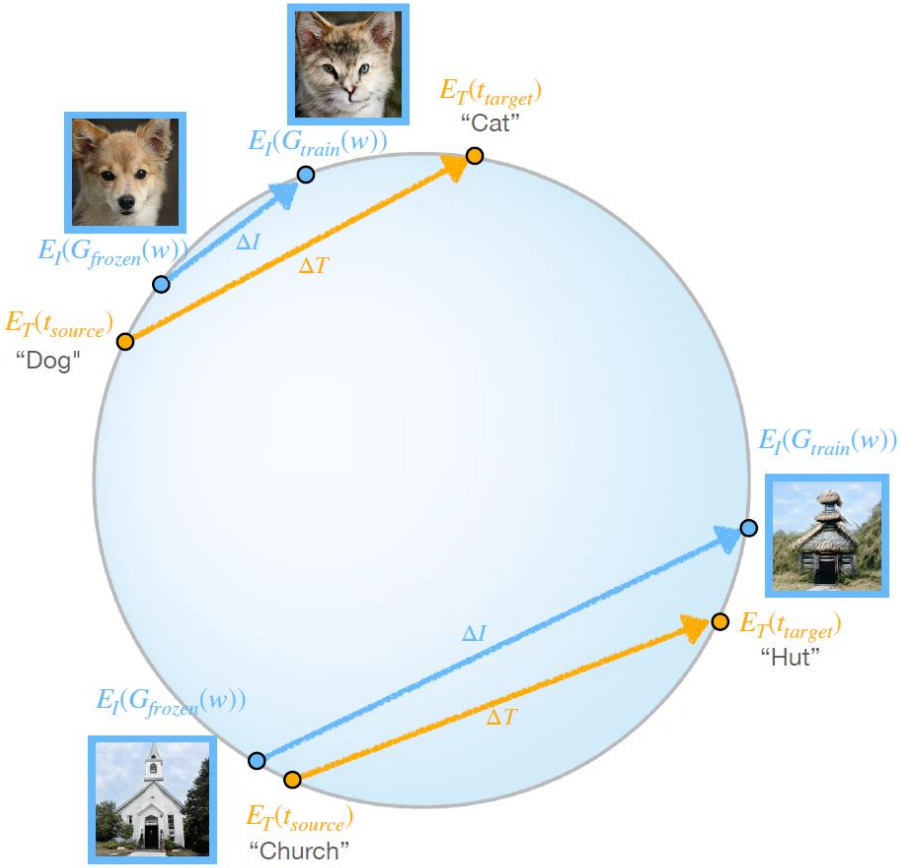


"Emma Stone" "Mohawk hairstyle" "Without makeup" "Cute cat" "Lion" "Gothic church"

StyleGAN-NADA: CLIP-Guided Adaptation



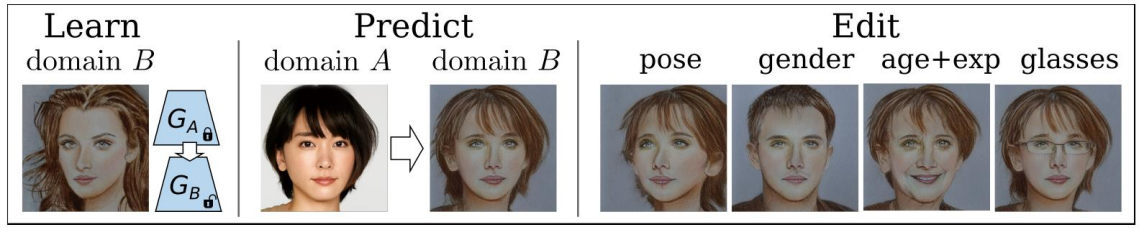
$$\Delta T = E_T\left(t_{target}\right) - E_T\left(t_{source}\right) ,$$
$$\Delta I = E_I\left(G_{train}\left(w\right)\right) - E_I\left(G_{frozen}\left(w\right)\right)$$
$$\mathcal{L}_{direction} = 1 - \frac{\Delta I \cdot \Delta T}{|\Delta I|\,|\Delta T|} ,$$
$$\mathcal{L}_{norm} = \left|E_I\left(G\left(w\right)\right) - E_I\left(G\left(M\left(w\right)\right)\right)\right|^2$$



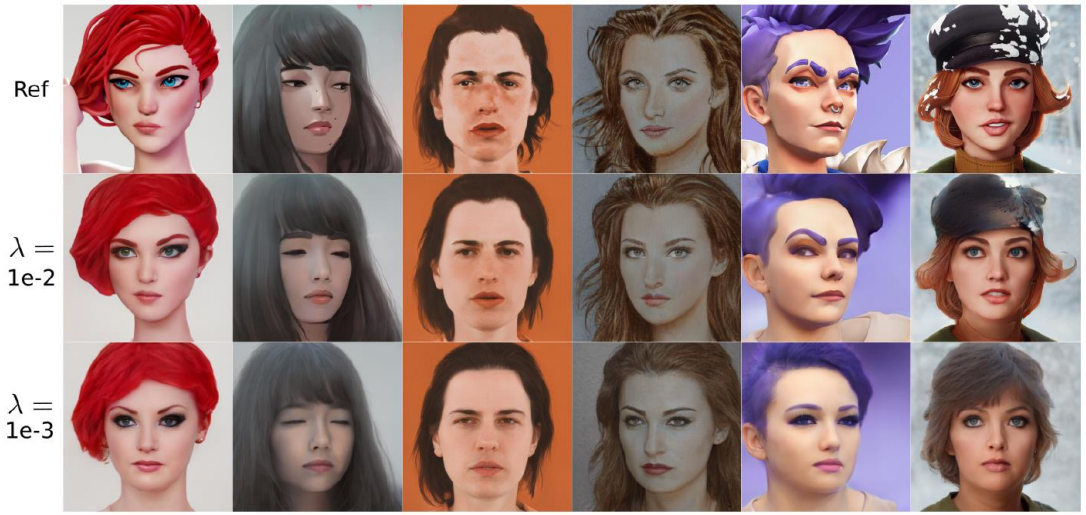
Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, Daniel Cohen-Or, StyleGAN-NADA: CLIP-Guided Domain Adaptation of Image Generators, Arxiv 2021.

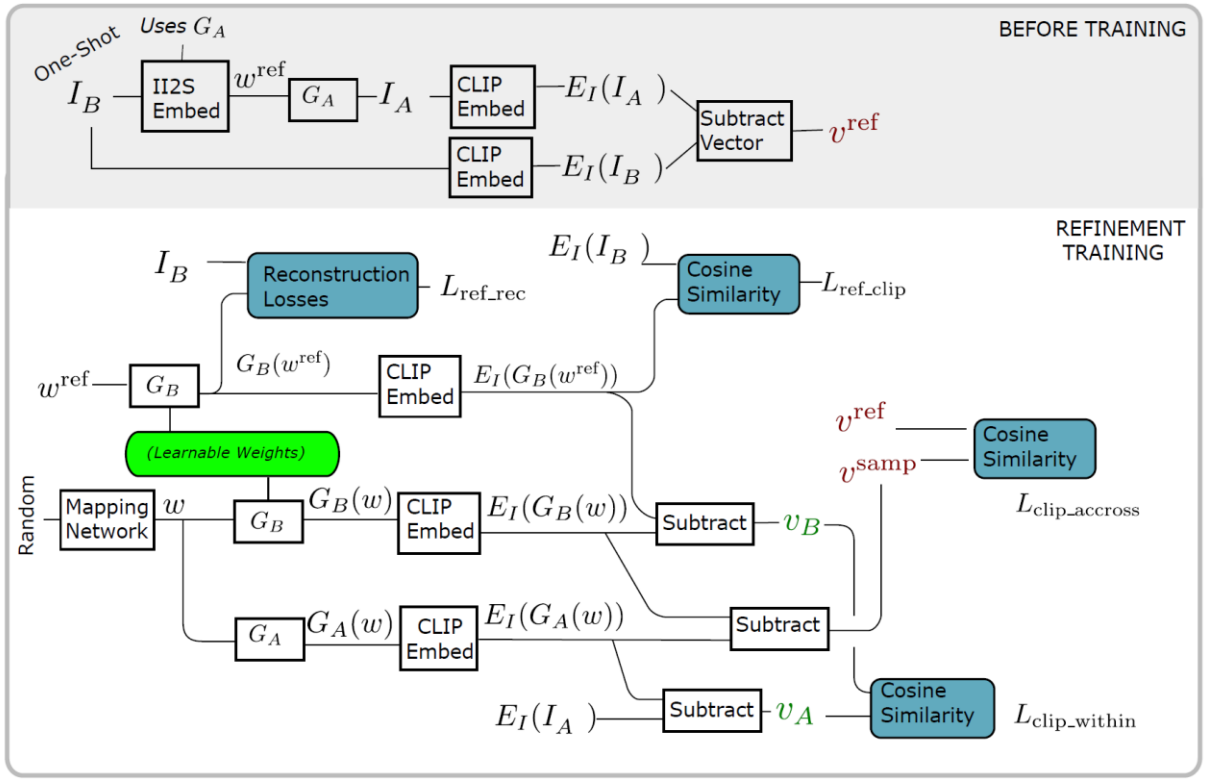
StyleGAN-NADA

CLIP for Single Shot Domain Adaptation



$$v^{\text{ref}} = E_I(I_B) - E_I(I_A)$$

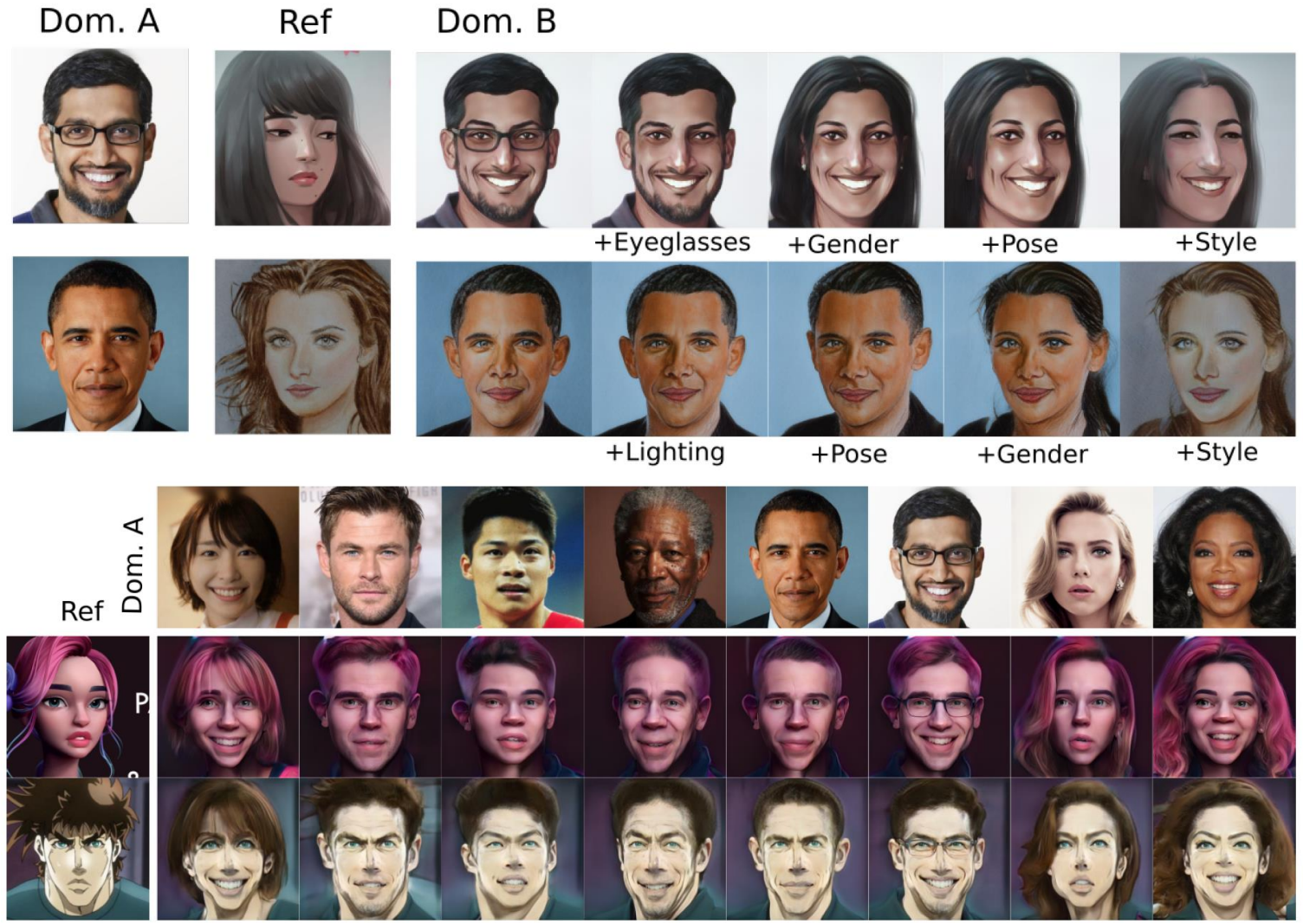




- Style Mixing

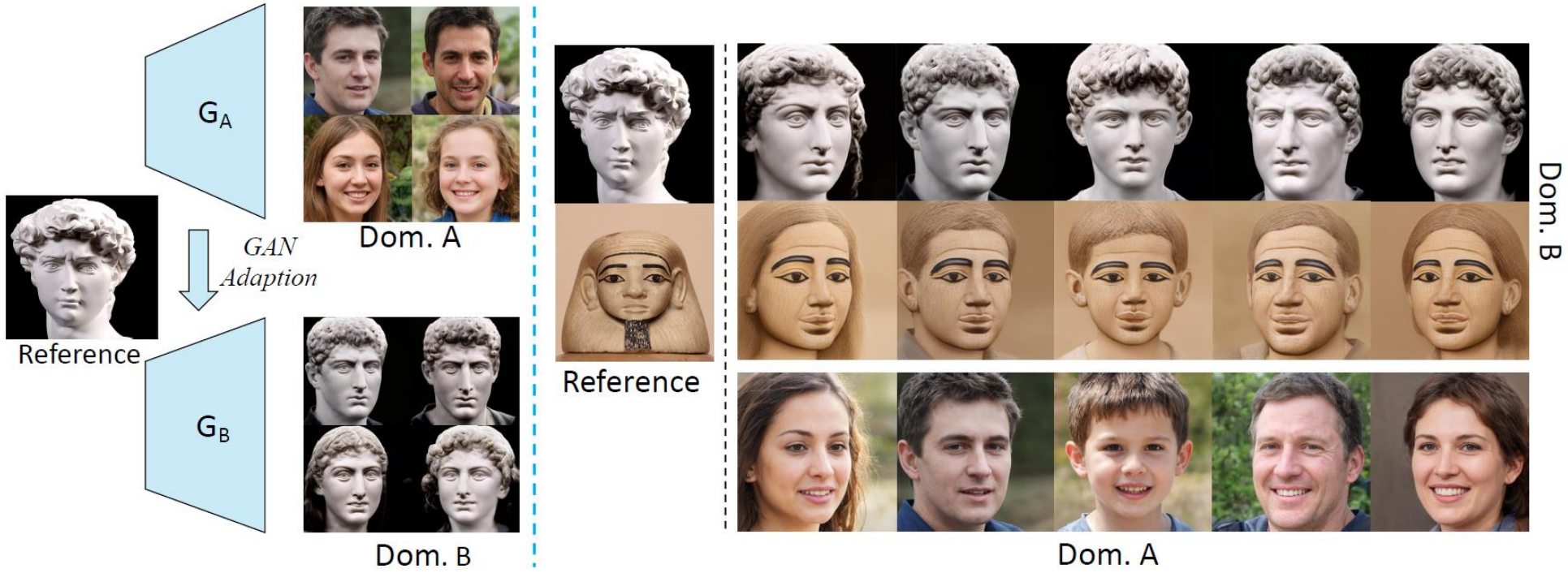
Peihao Zhu, Rameen Abdal, John Femiani, Peter Wonka, Mind the Gap: Domain Gap Control for Single Shot Domain Adaptation for Generative Adversarial Networks, Arxiv 2021.

CLIP for Single Shot Domain Adaptation

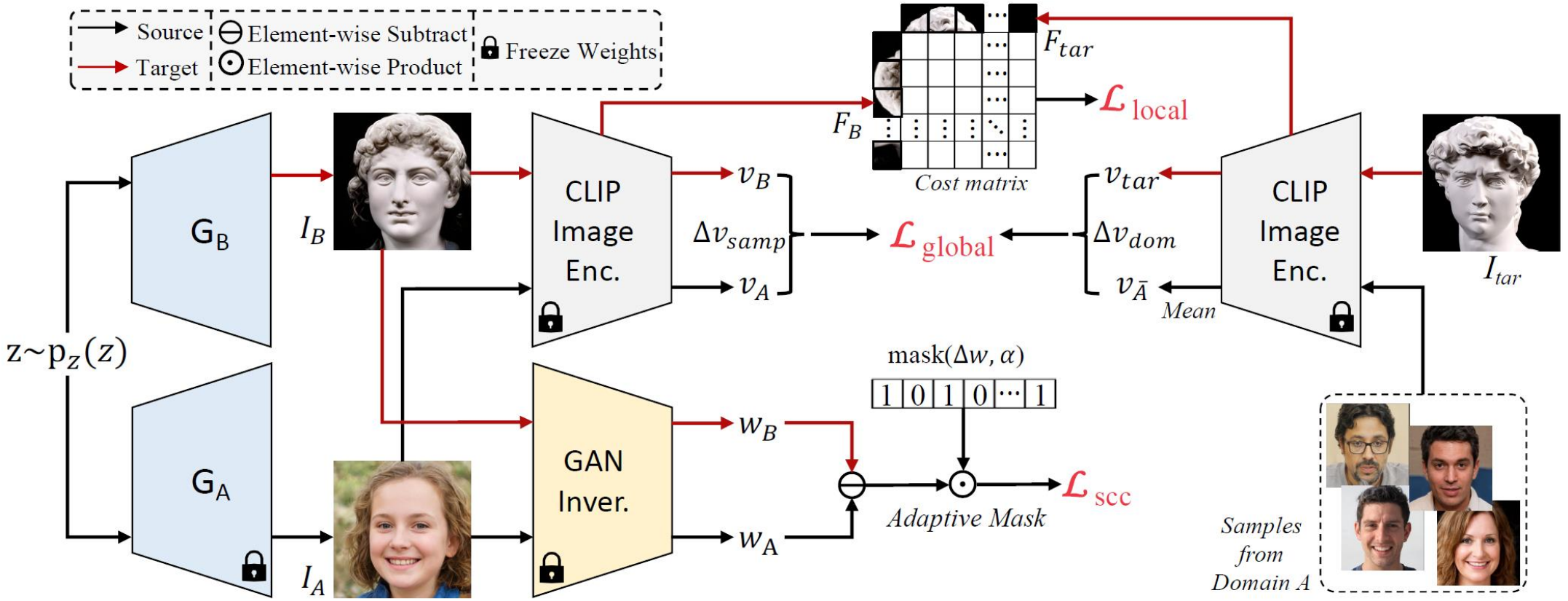


One-shot Generative Domain Adaption

- Input: Pre-trained Generator, Conditional Image
- Output: Adapted Generator

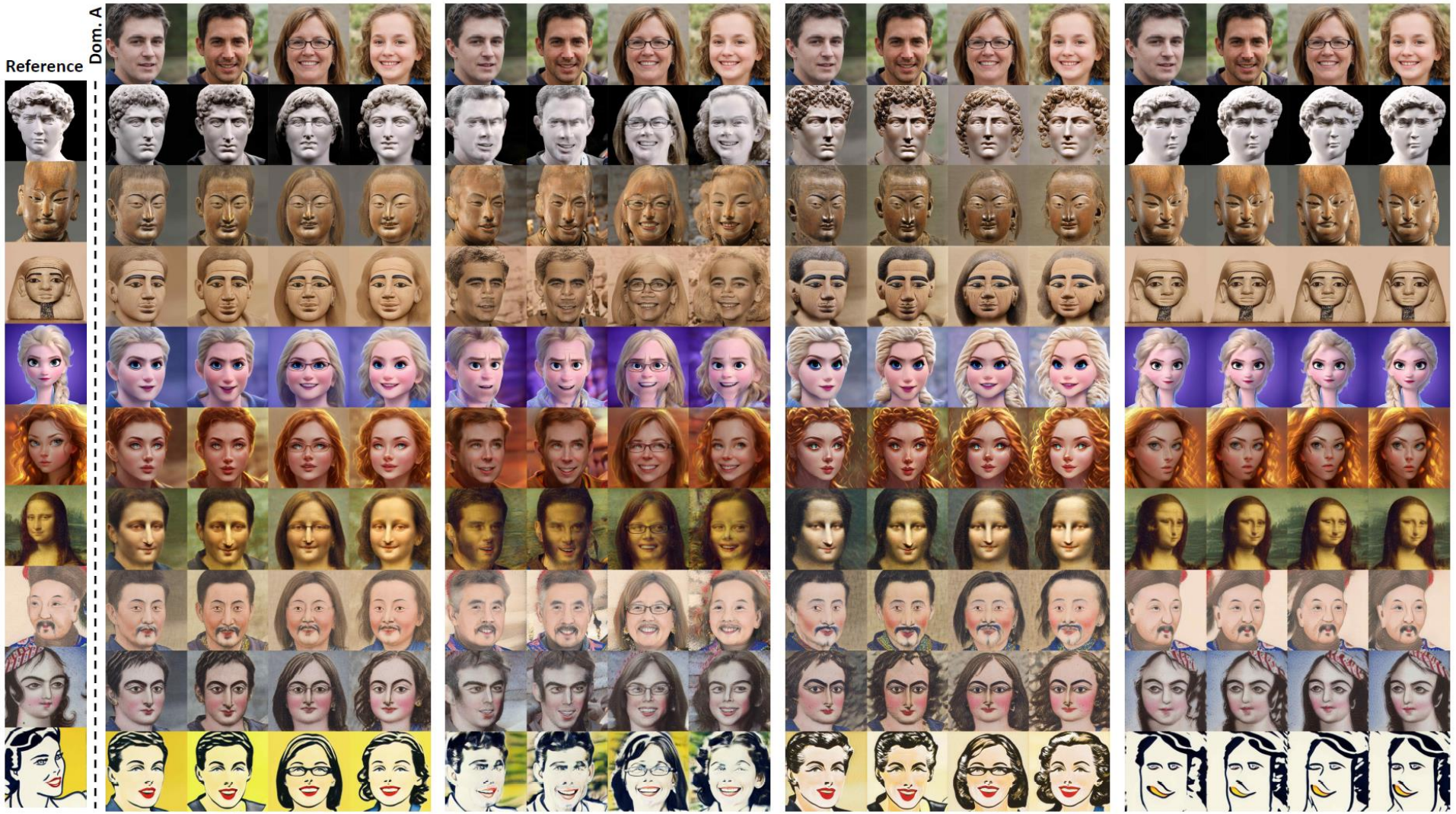


DiFa



Towards Diverse and Faithful One-shot Adaption of Generative Adversarial Networks, Submitted

FFHQ


Reference
Dom. A
(a) Ours
(b) Mind The Gap
(c) StyleGAN-NADA
(d) Few-Shot Adaption

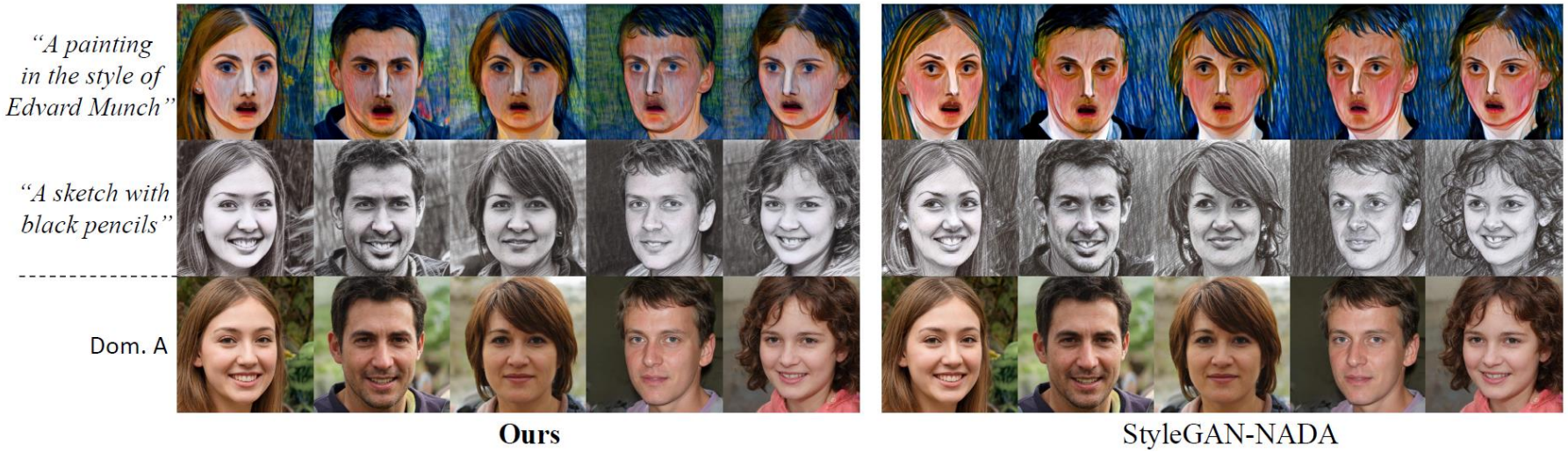
AFHQ-Cat

(a) Ours (b) Mind The Gap (c) StyleGAN-NADA

Extensions

- Latent Space Editing

- Zero-shot generative domain adaption



总结

- 理论研究
  - 如果更好更稳定地训练GAN
  - 网络结构、损失函数与正则化

- 应用研究
  - 预训练网络
  - 理解和应用：编辑/生成、算法->网络学习、更好地理解设计GAN Inversion网络