

# Divide-and-Conquer Inspired Adversarial Learning for Image & Video Generation and Enhancement

Zhiwu Huang

Computer Vision Lab @ ETH Zurich

## Presented Papers

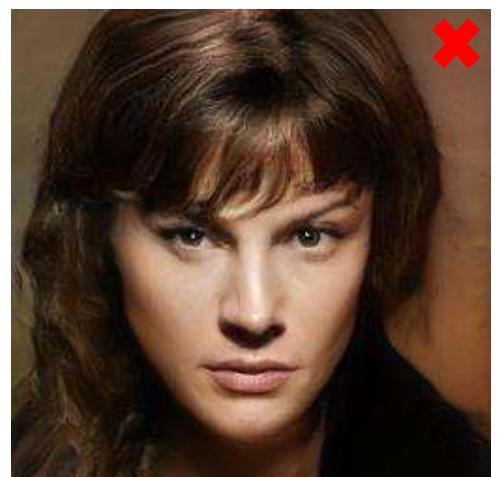
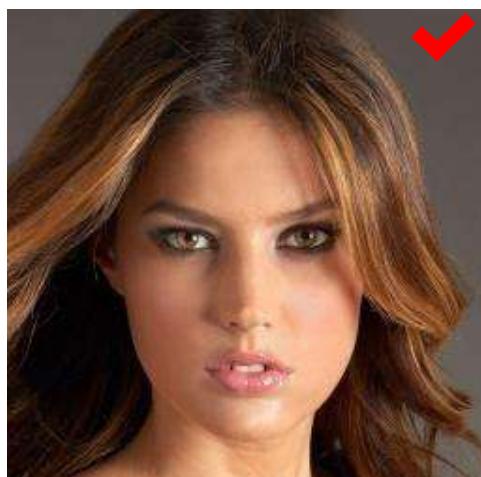
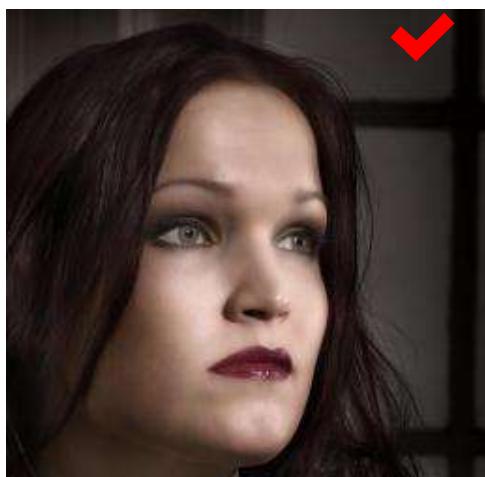
- **Sliced Wasserstein Generative Models**

Jiqing Wu\*, **Zhiwu Huang\***, Dinesh Acharya, Wen Li, Janine Thoma, Danda Pani Paudel, Luc Van Gool. (\**indicates equal contributions*)  
*Computer Vision and Pattern Recognition (CVPR), 2019*

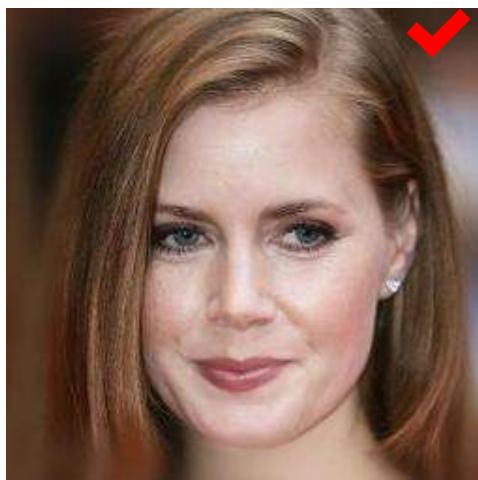
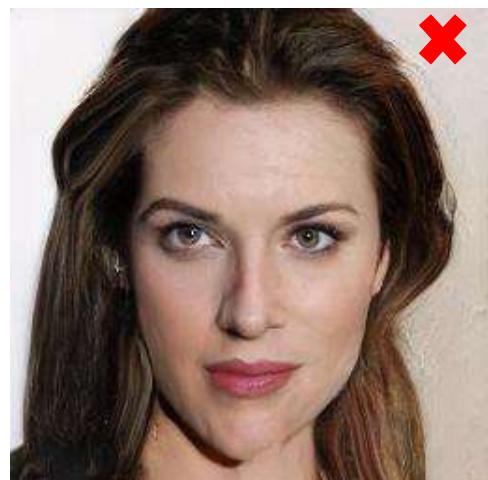
- **Divide-and-Conquer Adversarial Learning for High-resolution Photo Enhancement**

**Zhiwu Huang**, Danda Pani Paudel, Guanju Li, Jiqing Wu, Radu Timofte, Luc Van Gool.  
*Submitted to The International Conference on Learning Representations (ICLR), 2020*

## Real or Fake?



## Real or Fake?



## Hints for Real or Fake?

**Text is uninterpretable**



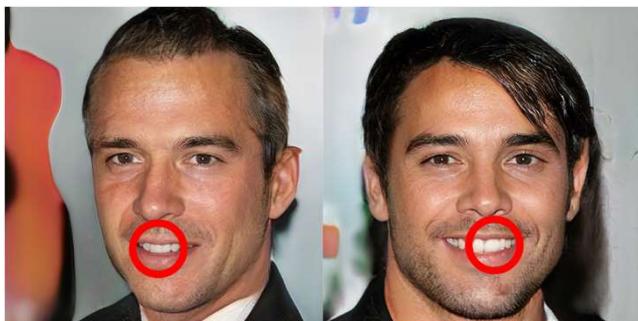
**Background is surreal**



**Asymmetry**



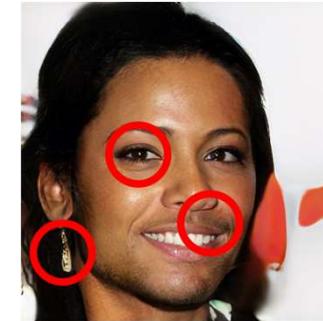
**Weird teeth**



**Messy hair**

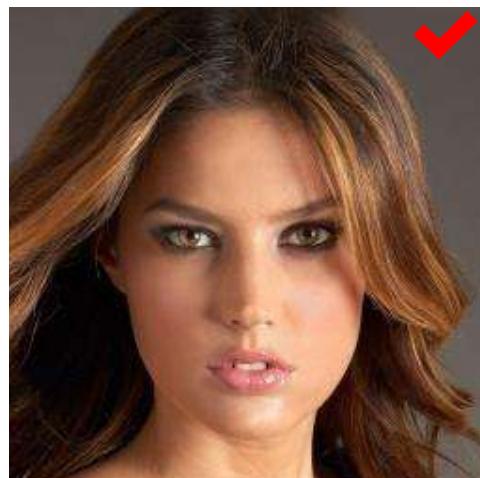
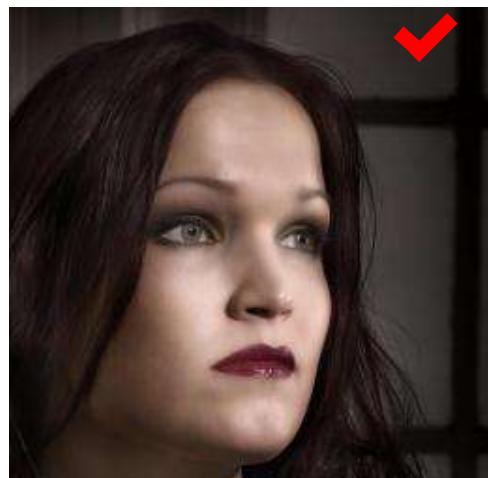


**Non-stereotypical gender**



Images from <https://medium.com/@kcimc/how-to-recognize-fake-ai-generated-images-4d1f6f9a2842>

## Real or Fake?

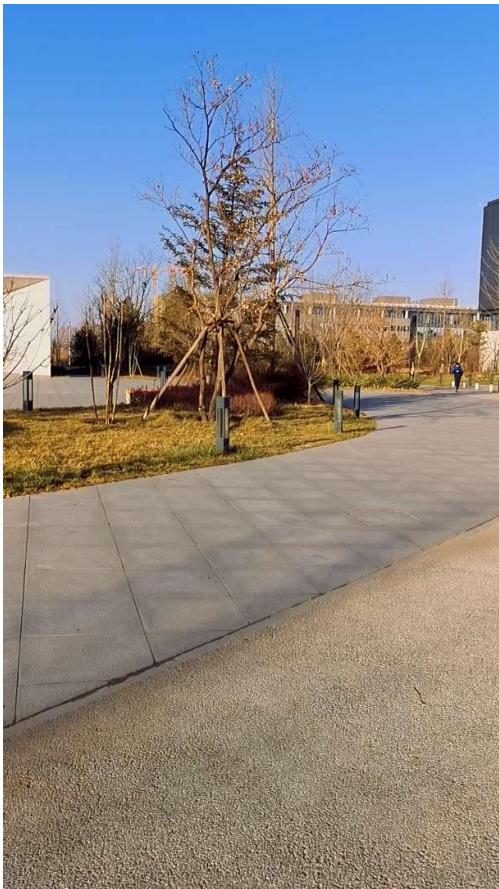




## Before or After?



# Before or After?

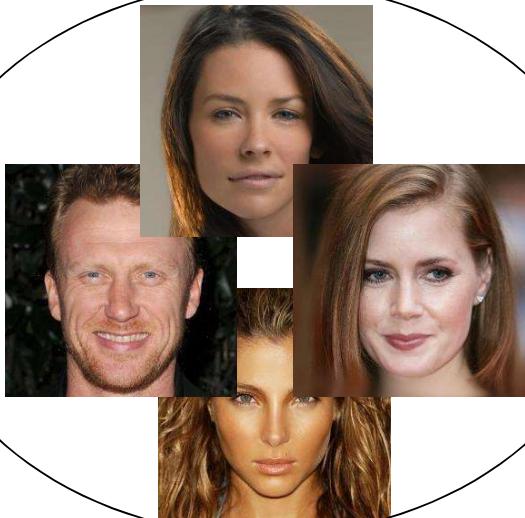


## Img Generation

Generated



Real



Distribution Distance Measurement ???

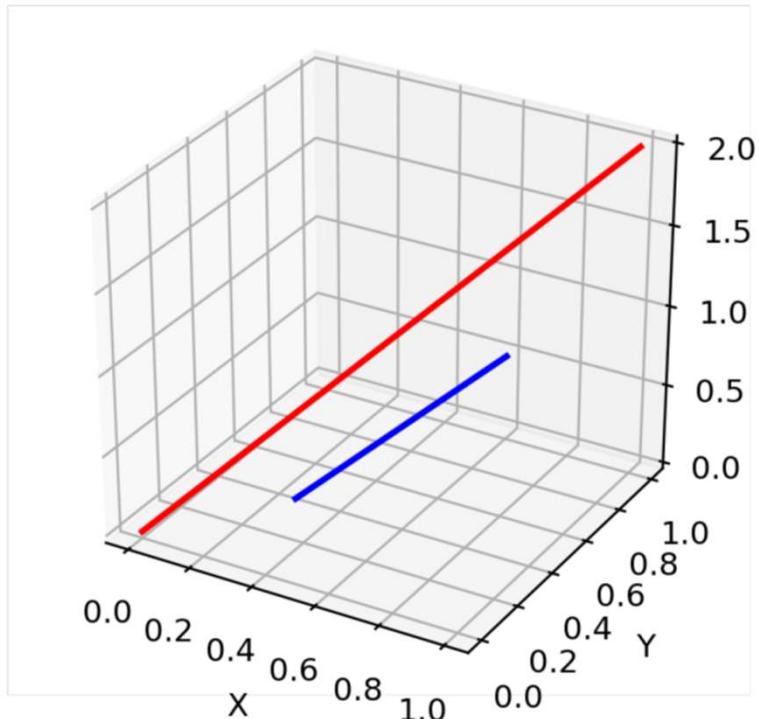
Enhanced



Target



# From KL and JS Divergence to Wasserstein Distance



low dimensional manifolds in high dimension space hardly have overlaps

$$D_{KL}(P\|Q) = \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{KL}(Q\|P) = \sum_{x=\theta, y \sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

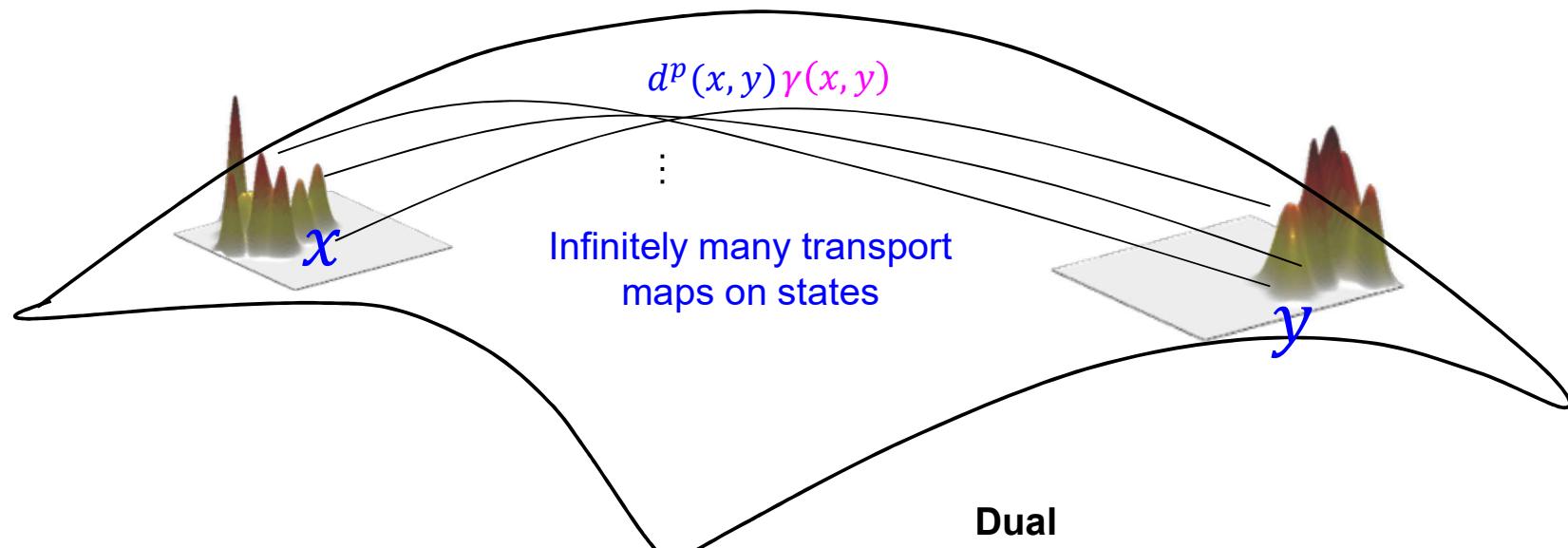
$$D_{JS}(P, Q) = \frac{1}{2} \left( \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{1/2} \right) = \log 2$$

$$W(P, Q) = |\theta| \quad \checkmark$$

# Wasserstein Distance – High Sample Complexity

**Primal**

$$W_p(P, Q) = \inf_{\gamma \in \Pi} \langle D, \Gamma \rangle_F = \left( \inf_{\gamma \in \Pi} d^p(x, y) \gamma(x, y) \right)^{\frac{1}{p}} = \left( \inf_{\gamma \in \Pi} \mathbb{E}_{x, y \sim \gamma} d^p(x, y) \right)^{\frac{1}{p}}$$



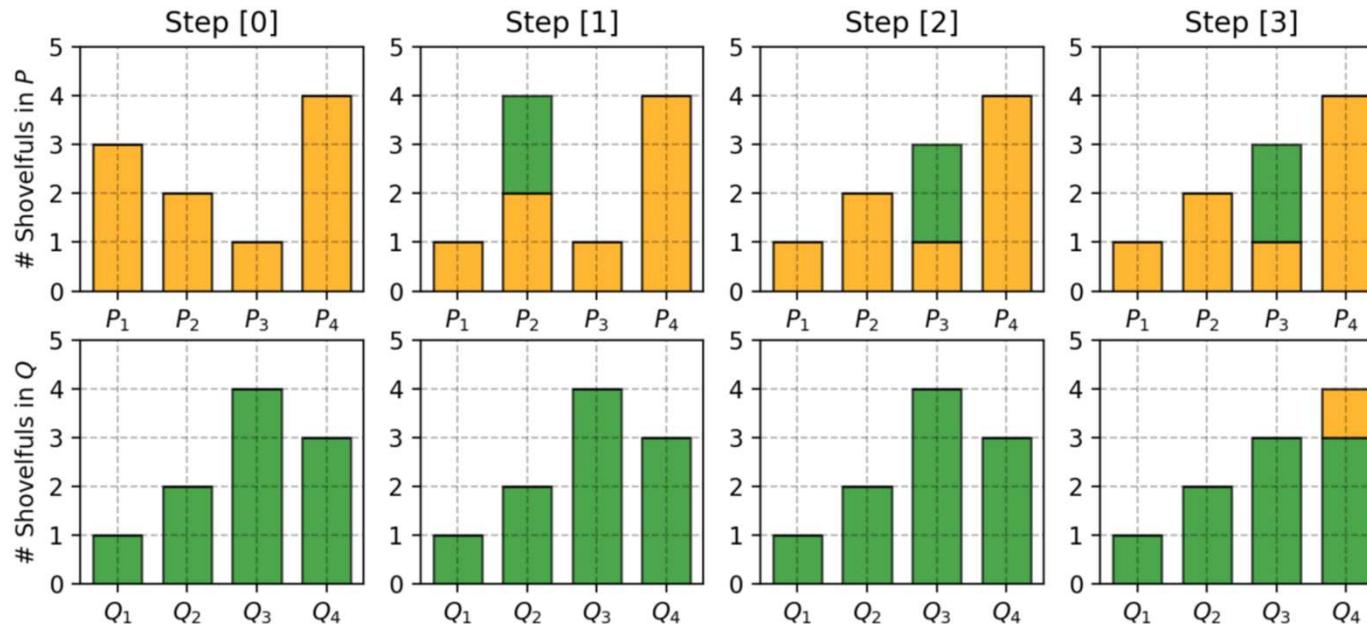
**Dual**

$$W_p(P, Q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{y \sim p_\theta}[f(y)]$$

# Divide and Conquer



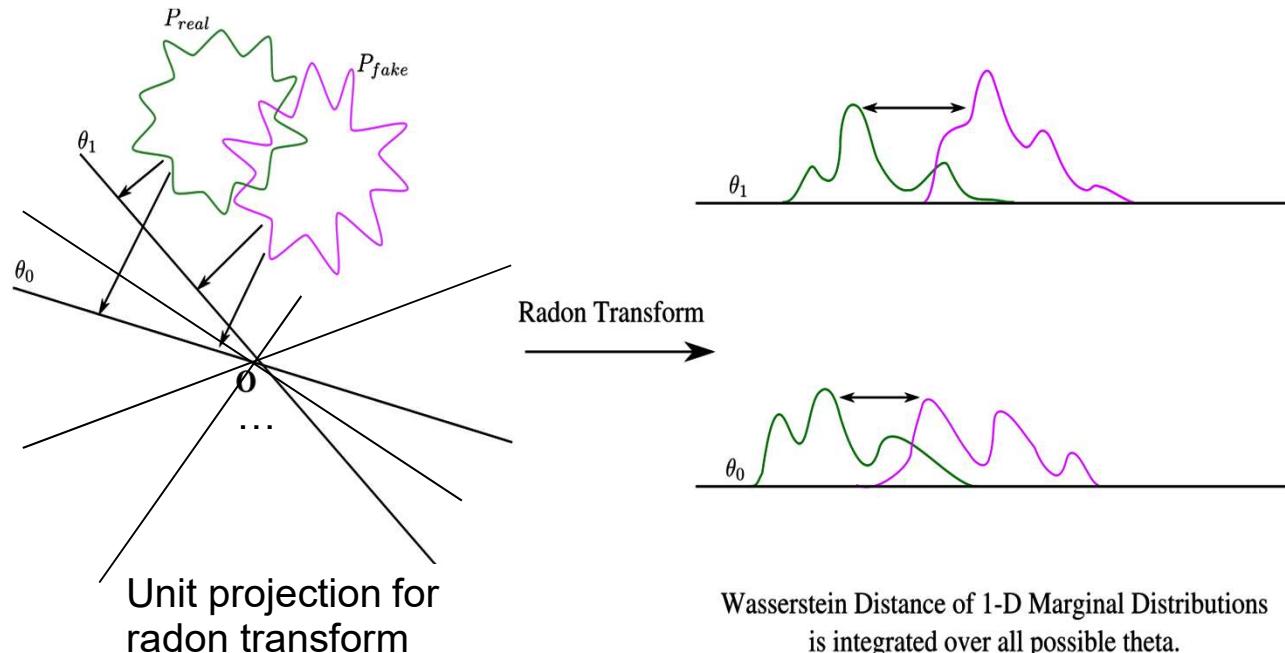
## Sliced Wasserstein Distance – 1D case



- ❖ Optimal map (closed form, increasing arrangement):  $\tau(x) = F_Q^{-1} \circ F_P(x)$ , where  $F_P, F_Q$  is the corresponding cumulative distribution functions (CDFs)

## Sliced Wasserstein Distance – High Projection Complexity

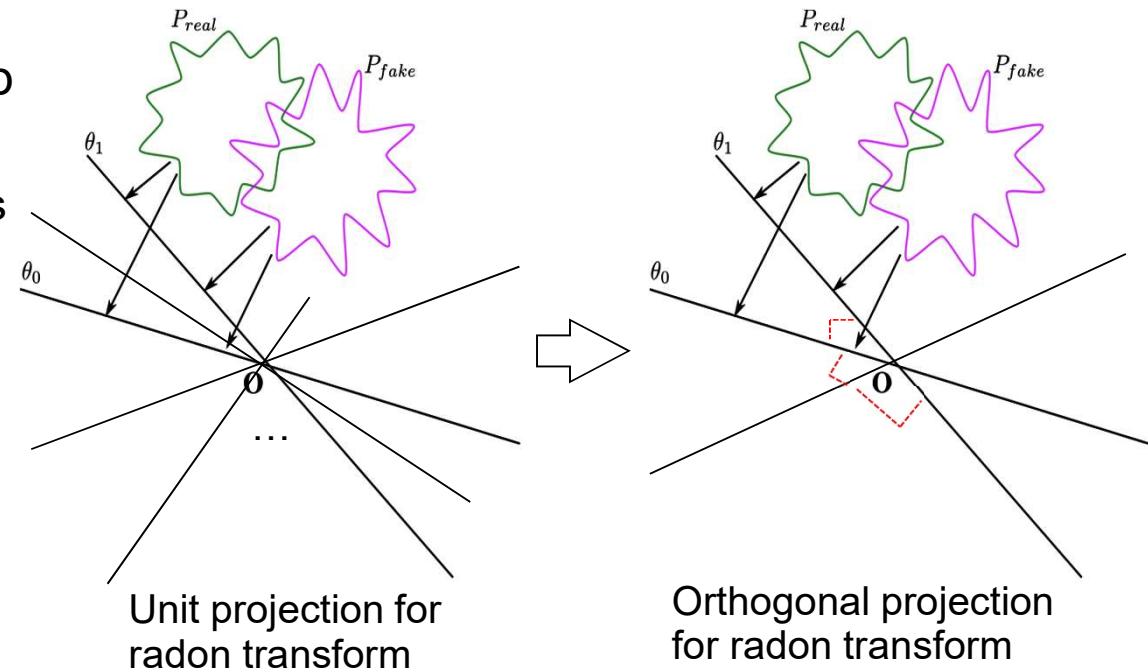
$$SW_p(P, Q) = \left( \int_{\Omega} W_p(\pi_{\theta}^* P, \pi_{\theta}^* Q) d\theta \right)^{\frac{1}{p}}, \text{ where } \Omega \text{ is a unit sphere, } \pi_{\theta}^* P = P \circ \pi_{\theta}, \\ \pi_{\theta}(x) = \theta^T x$$



Infinitely many random unit projections

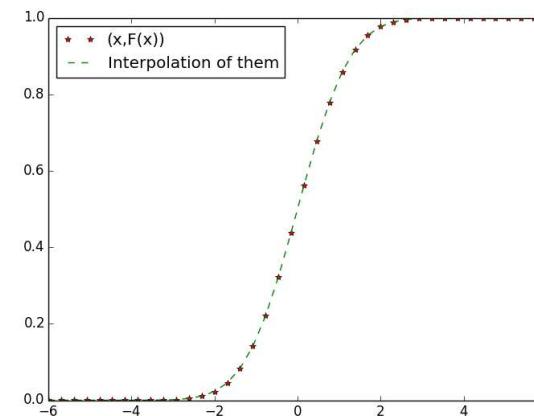
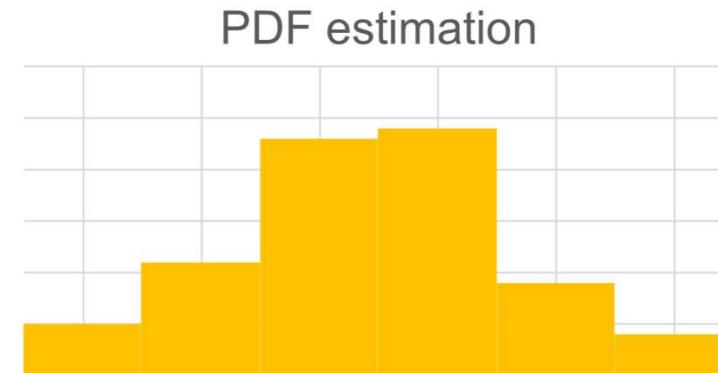
## Proposed Sliced Wasserstein Distance

- Learnable orthogonal projections for Radon Transform  $\pi_\theta(x) = \theta^T x$  in a deep learning manner
  - Orthogonal projections vs. unit projections
    - More efficient to cover entire space
  - Trainable net weights vs. random weights
    - More compact



## Proposed Sliced Wasserstein Distance

- Learnable orthogonal projections for Radon Transform  $\pi_\theta(x) = \theta^T x$  in a deep learning manner
- Differentiable 1D transport map  $\tau_\theta = (\pi_\theta^* F_Q)^{-1} \circ \pi_\theta^* F_P$ ,
  - 1D PDF estimation using soft histogram assignment
    - $\frac{e^{-\alpha \|y - c_i\|^2}}{\sum_{j=1}^l e^{-\alpha \|y - c_j\|^2}}$  to the  $i$ -th bin, where  $c_i$  are the  $i$ -th bin center
  - (Inverse) 1D CDF estimation using linear interpolation



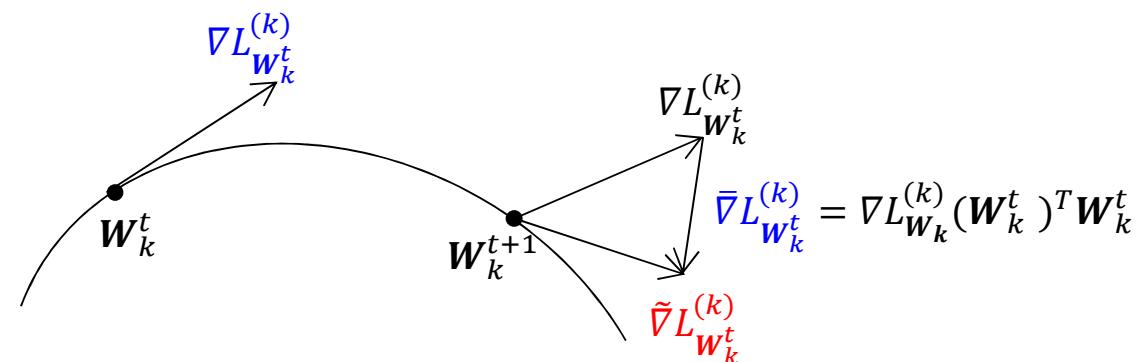
## Proposed Sliced Wasserstein Distance

- Learnable orthogonal projections for Radon Transform  $\pi_\theta(x) = \theta^T x$  in a deep learning manner
- Differentiable 1D transport map  $\tau_\theta = (\pi_\theta^* F_Q)^{-1} \circ \pi_\theta^* F_P$ ,
- Dual form (non-linearity  $f$ ):

$$\int_{\mathbb{S}^{n-1}} \left( \sup_{f \in \text{Lip}^k} \mathbb{E}_{X_\theta \sim \pi_\theta^* P_X} [f(X_\theta)] - \mathbb{E}_{Y_\theta \sim \pi_\theta^* P_Y} [f(Y_\theta)] \right) d\theta$$

## Proposed Sliced Wasserstein Distance

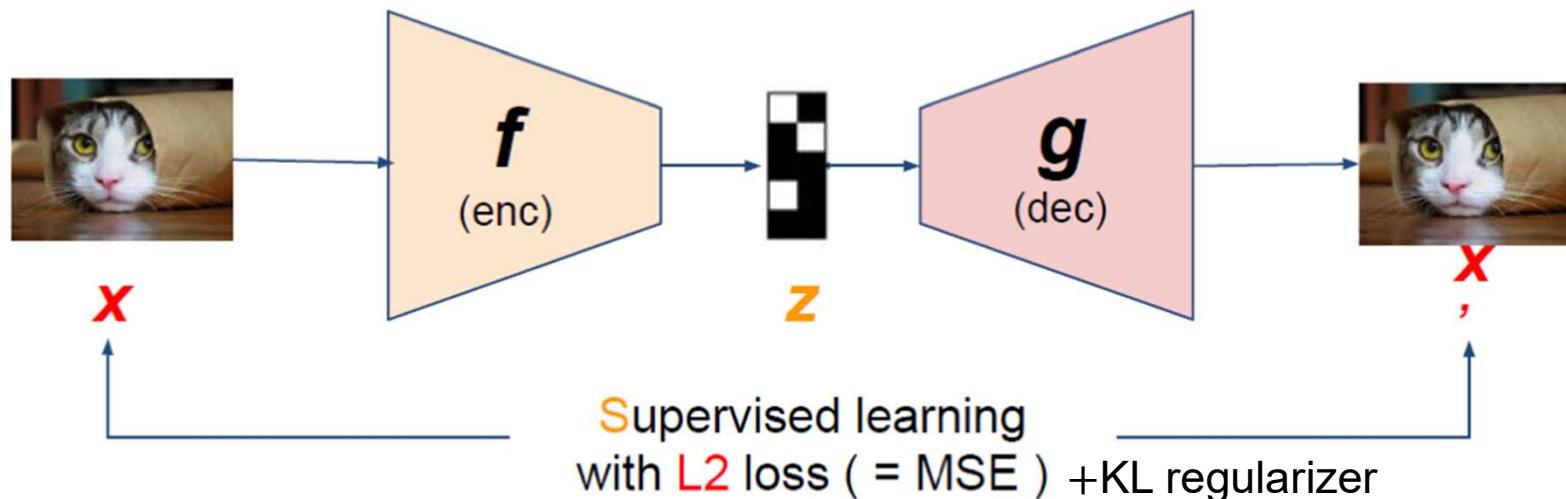
- Learnable orthogonal projections for Radon Transform  $\pi_\theta(x) = \theta^T x$  in a deep learning manner
- Differentiable 1D transport map  $\tau_\theta = (\pi_\theta^* F_Q)^{-1} \circ \pi_\theta^* F_P$ ,
- Dual form (non-linearity  $f$ ):
- Network training: updating orthogonal weights on Stiefel manifolds



# How AE based generative models work?

- Reconstruction with penalization on latent variables

$$\min_f f = \mathbb{E}_{q(z)} [\log p(x|z)] - KL(q(z)||p(z))$$



Images from Namju Kim

## Proposed Sliced Wasserstein Auto-Encoder (SWAE)

---

### Algorithm 1 The proposed primal SWD block

**Require:** Orthogonal matrix  $\mathbf{O}_\Theta = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_r] \in \mathbb{R}^{r \times r}$ , batch of latent codes  $\mathbf{M}_y = [\mathbf{y}_1, \dots, \mathbf{y}_b] \in \mathbb{R}^{r \times b}$ , batch of Gaussian noise  $\mathbf{M}_z = [\mathbf{z}_1, \dots, \mathbf{z}_b] \in \mathbb{R}^{r \times b}$ , and bin number  $l$

**Output:** Batch of transferred latent codes  $M_{\tilde{y}} = [\tilde{\mathbf{y}}_1, \dots, \tilde{\mathbf{y}}_b]$

**for**  $i \leftarrow 1, r$  **do**

$$\mathbf{y}'_i = \boldsymbol{\theta}_i^T \mathbf{M}_y, \mathbf{z}'_i = \boldsymbol{\theta}_i^T \mathbf{M}_z$$

$$\mathbf{y}''_i = \frac{\mathbf{y}'_i - \min_j \{y'_{i,j}\}}{\max_j \{y'_{i,j}\} - \min_j \{y'_{i,j}\}}, \mathbf{z}''_i = \frac{\mathbf{z}'_i - \min_j \{z'_{i,j}\}}{\max_j \{z'_{i,j}\} - \min_j \{z'_{i,j}\}},$$

$y'_{i,j}, z'_{i,j}$  are the  $j$ -th element of  $\mathbf{y}'_i, \mathbf{z}'_i$  respectively.

Compute soft PDF histogram  $p_{y''_i}, p_{z''_i}$  of  $\mathbf{y}''_i, \mathbf{z}''_i$  with  $l$  bins

Compute CDF  $F_{y''_i}, F_{z''_i}$  of  $p_{y''_i}, p_{z''_i}$

Compute  $F_{y''_i}(\mathbf{y}''_i)$  element-wise by linear interpolation

$$\hat{\mathbf{y}}_i = (\max_j \{z'_{i,j}\} - \min_j \{z'_{i,j}\})(F_{z''_i})^{-1} F_{y''_i}(\mathbf{y}''_i) + \min_j \{z'_{i,j}\}$$

**end for**

Compute  $\mathbf{M}_{\tilde{y}} = \mathbf{O}_\Theta \mathbf{M}_{\hat{y}}^T, \mathbf{M}_{\hat{y}} = [\hat{\mathbf{y}}_1^T, \dots, \hat{\mathbf{y}}_r^T]$

---



---

### Algorithm 2 The proposed SWAE

**Require:** Primal SWD block number  $m$ , batch size  $b$ , decoder  $G$  and encoder  $Q = S_{p,m} \circ \dots \circ S_{p,2} \circ S_{p,1} \circ E$ , training steps  $h$ , training hyperparameters, etc.

**for**  $t \leftarrow 1, h$  **do**

    Sample real data  $\mathbf{M}_x = [x_1, \dots, x_b]$  from  $P_X$

    Sample Gaussian noise  $\mathbf{M}_z = [z_1, \dots, z_b]$  from  $\mathcal{N}(0, 1)$

    Update the weights  $\mathbf{w}$  of  $Q$  and  $G$  by descending:

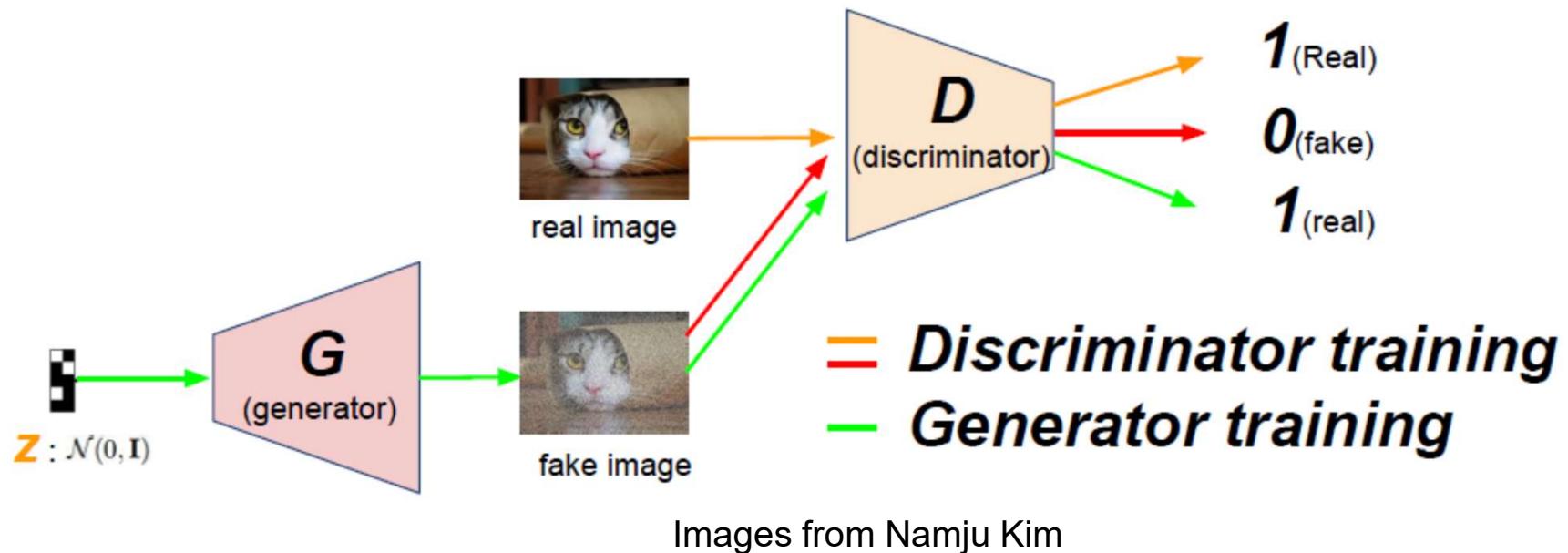
$$\mathbf{w} \leftarrow \text{Adam}(\nabla_{\mathbf{w}} (\frac{1}{b} \|\mathbf{M}_x - G(Q(\mathbf{M}_x, \mathbf{M}_z))\|_2^2), \mathbf{w})$$

**end for**

---

# How Generative Adversarial Nets (GANs) work?

- Two-player game (min-max objective function)
  - $\min_G \max_D(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x \sim p_z(z)}[\log(1 - D(G(z)))]$



# Proposed Sliced Wasserstein Generative Adversarial Nets (SWGAN)

---

### Algorithm 3 The proposed dual SWD block

**Require:** Orthogonal matrix  $\mathbf{O}_\Theta = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_r] \in \mathbb{R}^{r \times r}$  and batch of latent codes  $\mathbf{M}_y = [\mathbf{y}_1, \dots, \mathbf{y}_b] \in \mathbb{R}^{r \times b}$ .

**Output:** Batch of  $\tilde{\mathbf{y}}$  for dual SWD

**for**  $i \leftarrow 1, r$  **do**

    Compute  $\mathbf{y}'_i = \boldsymbol{\theta}_i^T \mathbf{M}_y$

    Compute  $\mathbf{y}''_i = F_i(\mathbf{y}'_i)$  element-wise, where  $F = (F_1, \dots, F_r)$  are one-dimensional functions to approximate the  $f$  in Eq. 10.

**end for**

$\tilde{\mathbf{y}} = [\mathbf{y}''_1, \dots, \mathbf{y}''_b]^T$

---

### Algorithm 4 The proposed SWGAN

**Require:** Number of dual SWD blocks  $m$ , batch size  $b$ , generator  $G$  and discriminator  $D = [S_{d,1} \circ E, \dots, S_{d,m} \circ E]^T$ , latent code dimension  $r$ , Lipschitz constant  $k$ , training steps  $h$ , training hyperparameters, etc.

**for**  $t \leftarrow 1, h$  **do**

    Sample real data  $\mathbf{M}_x = [\mathbf{x}_1, \dots, \mathbf{x}_b]$  from  $P_X$

    Sample Gaussian noise  $\mathbf{M}_z = [\mathbf{z}_1, \dots, \mathbf{z}_b]$  from  $\mathcal{N}(0, 1)$

    Sample two vectors  $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$  from uniform distribution  $U[0, 1]$  and for  $l = 1, \dots, b$  calculate the elements of  $\mathbf{M}_{\hat{x}}, \mathbf{M}_{\hat{y}}$ :

$$\hat{\mathbf{x}}_l = (1 - \mu_{1,l})\mathbf{x}_l + \mu_{1,l}G(\mathbf{z}_l)$$

$$\hat{\mathbf{y}}_l = (1 - \mu_{2,l})E(\mathbf{x}_l) + \mu_{2,l}E(G(\mathbf{z}_l))$$

    Update the weights  $\mathbf{w}_G$  of  $G$  by descending:

$$\mathbf{w}_G \leftarrow \text{Adam}(\nabla_{\mathbf{w}_G} (\frac{1}{b} \sum_{j,i=1}^{r \times m, b} D_{ji}(G(\mathbf{M}_z))), \mathbf{w}_G)$$

    Update the weights  $\mathbf{w}_D$  of  $D$  by descending:

$\mathbf{w}_D \leftarrow \text{Adam}(\nabla_{\mathbf{w}_D} (\frac{1}{b} \sum_{j,i=1}^{r \times m, b} (D_{ji}(\mathbf{M}_x) - D_{ji}(G(\mathbf{M}_z)) + \lambda_1 \|\nabla_{\mathbf{M}_{\hat{x}}} D(\mathbf{M}_{\hat{x}})\|_2^2 + \lambda_2 \|\nabla_{\mathbf{M}_{\hat{y}}} F(\mathbf{M}_{\hat{y}}) - k \cdot \mathbf{1}\|_2^2)), \mathbf{w}_D)$ , where we compute the gradients of  $F$  element-wise.

**end for**

---

WAE

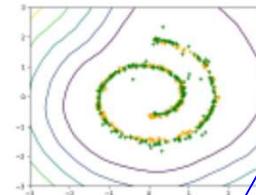
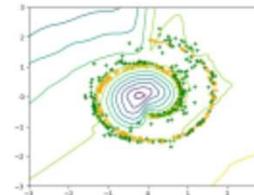
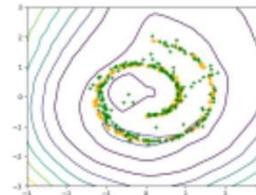
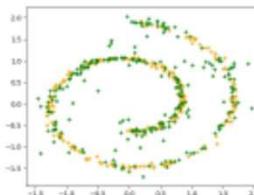
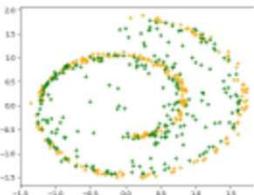
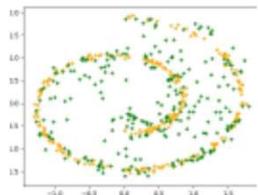
AE +100 IDT

SWAE

CT-GAN

SWG

SWGAN



Frechet Inception  
Distance (FID)

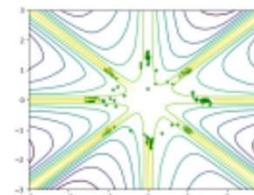
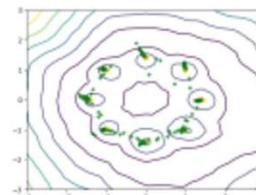
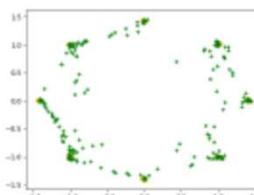
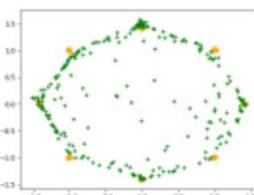
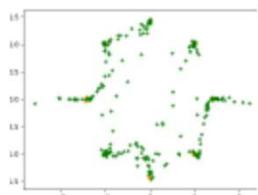
0.04 (0.06)

0.05 (0.06)

**0.04 (0.04)**

0.01

0.03

**0.01**

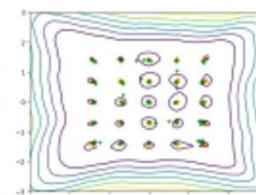
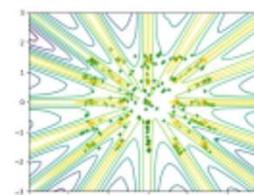
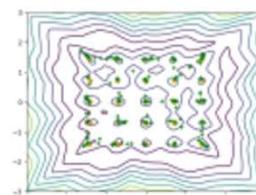
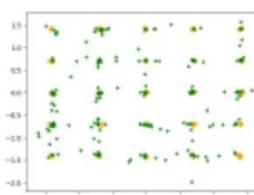
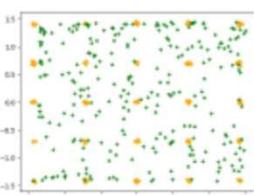
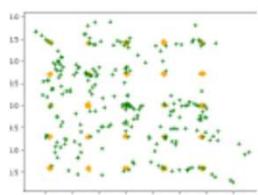
0.07 (0.04)

0.06 (0.06)

**0.03 (0.02)**

0.03

0.05

**0.02**

0.05 (0.03)

0.05 (0.06)

**0.04 (0.02)**

0.02

0.05

**0.01**

## PG-WGAN



7.5

## PG-SWGAN

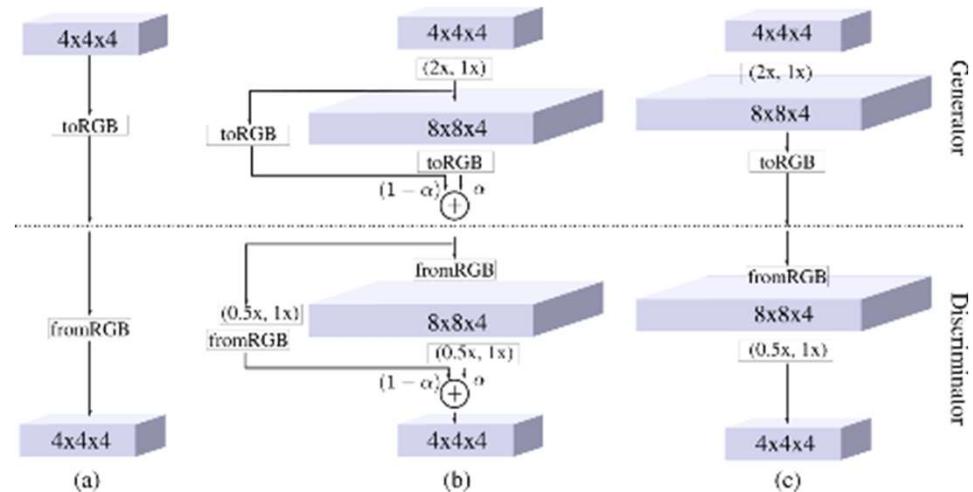
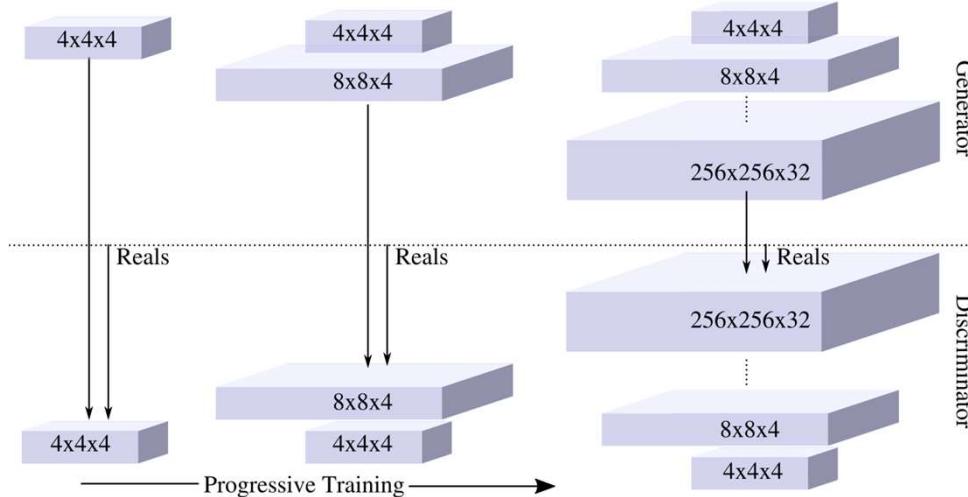


5.5

Frechet Inception  
Distance (FID)

	AMT Preference
PG-WGAN	0.45
PG-SWGAN	0.55

# Progressive Growing Technique for Video Generation





CVL  
Computer  
Vision  
Lab

## PG-WGAN-3D



462.6

Frechet Inception  
Distance (FID)

## PG-SWGAN-3D



404.1

AMT Preference
PG-WGAN
PG-SWGAN

The table compares the AMT preference scores for PG-WGAN and PG-SWGAN. PG-WGAN has a score of 0.46, while PG-SWGAN has a higher score of 0.54.

# Mixed-Perception Issue for Image Enhancement



Color Adjustment



Illumination Enhancement

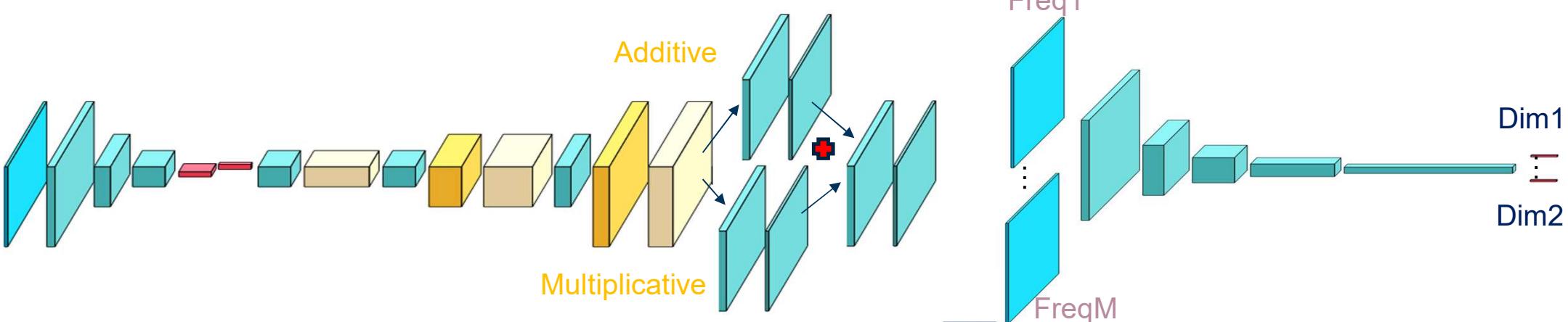
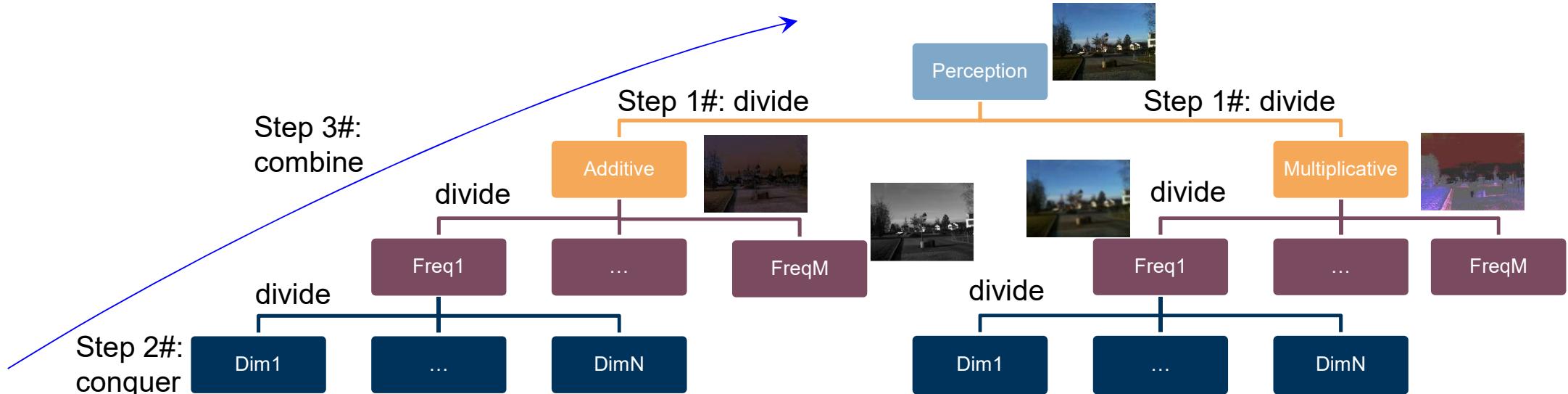


Texture Sharpening

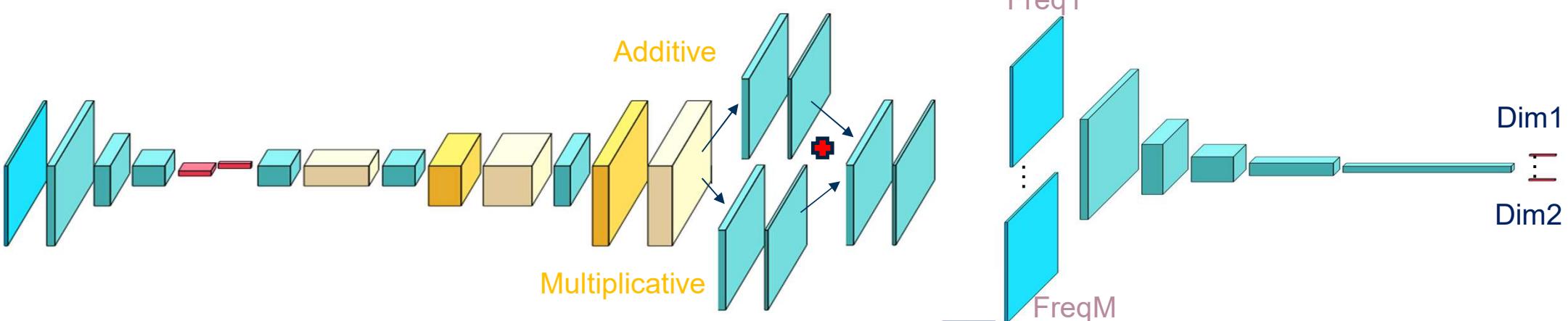
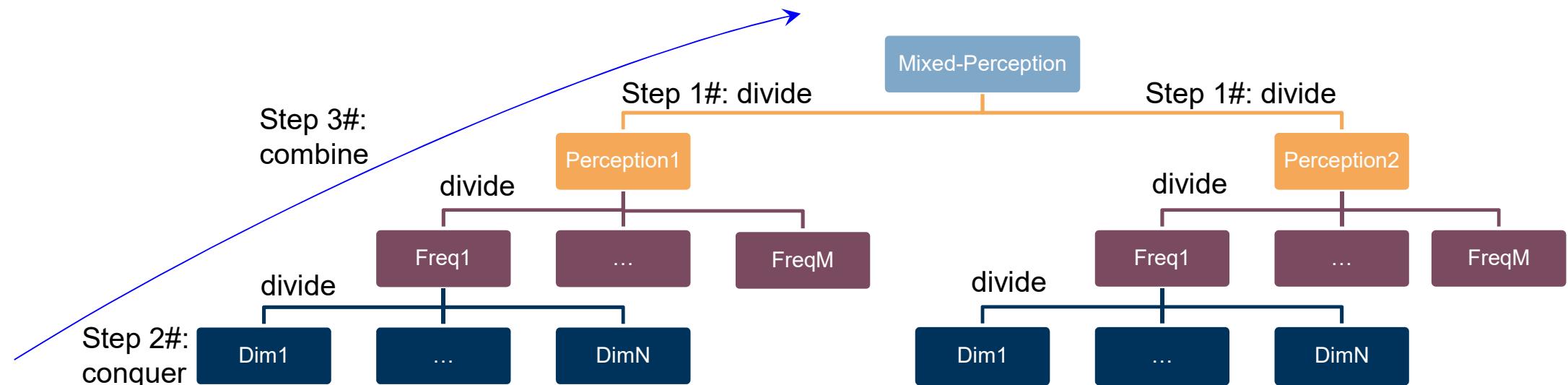
# Divide and Conquer



# Hierarchically Multisliced Methodology



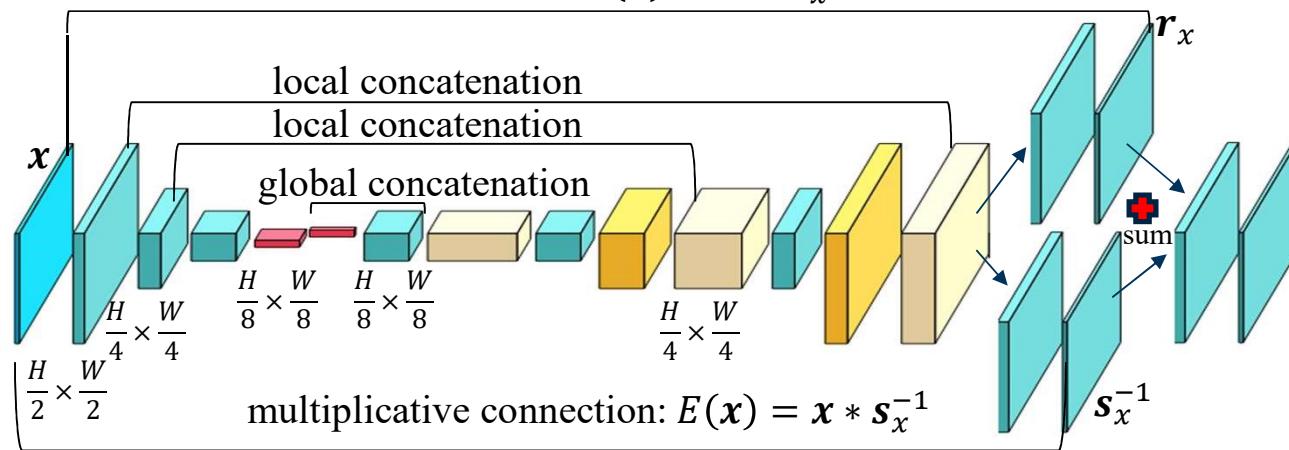
# Hierarchically Multisliced Methodology



# Network Design

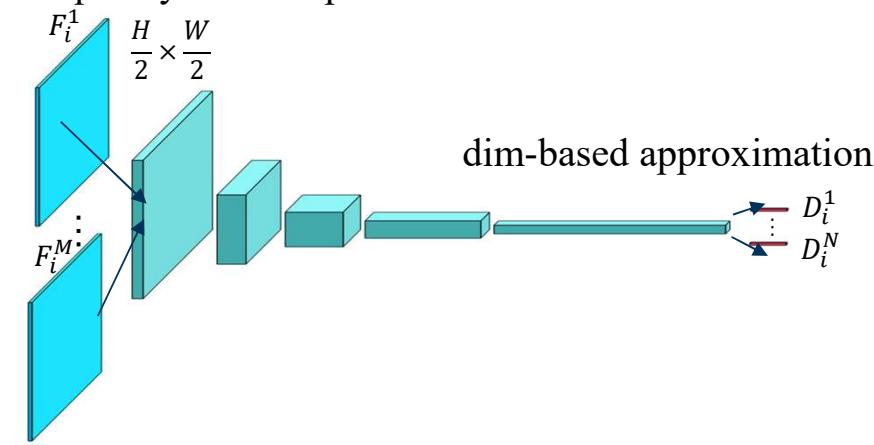


additive connection:  $E(x) = x + r_x$



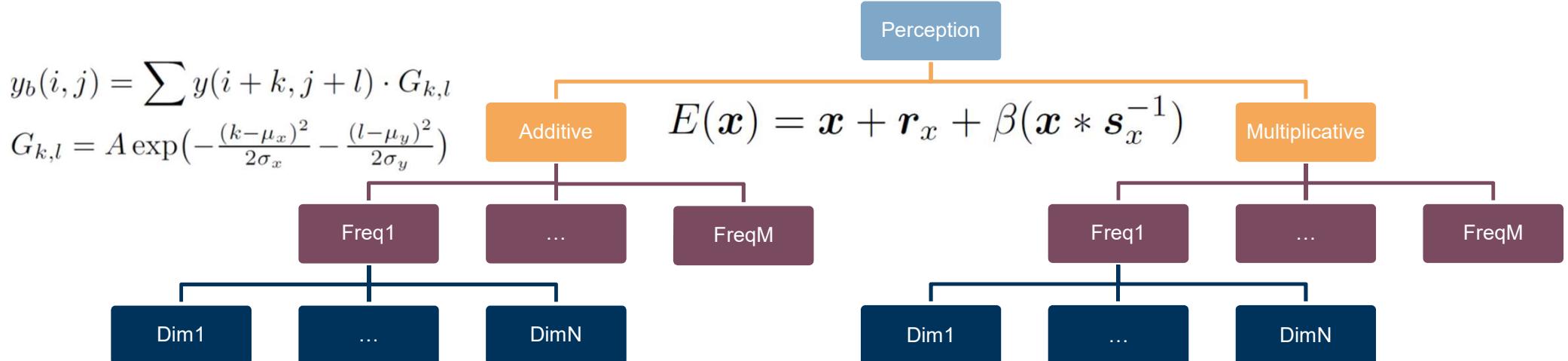
(a) Enhancer for Perception-based Division

frequency-based input



(b) Discriminator for Freq- and Dim-based Division

# Hierarchically Multisliced Methodology



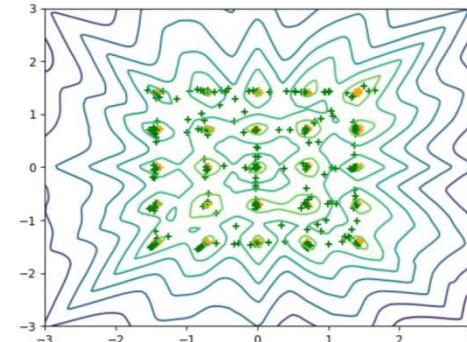
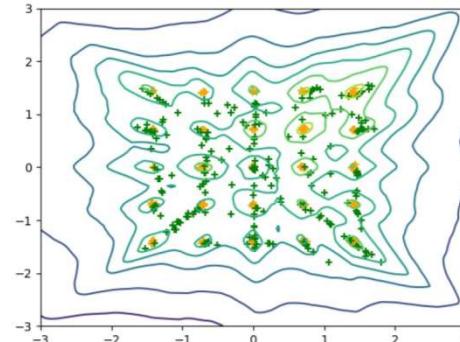
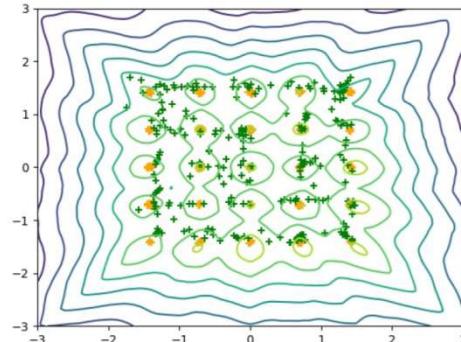
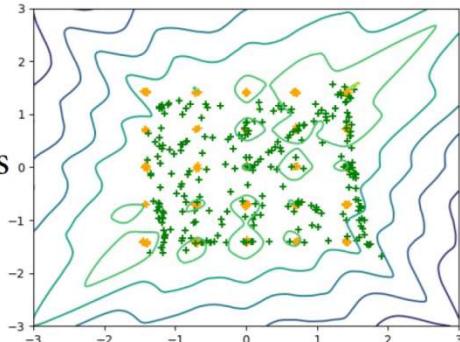
New SWGAN loss

$$\min_E \max_C \int_{\theta \in \mathbb{S}^{n-1}} \left( \mathbb{E}_{\mathbf{y} \sim P_y} [C(\mathbf{y})] - \mathbb{E}_{\hat{\mathbf{y}} \sim P_E} [C(E(\mathbf{x}))] \right) + \lambda \mathbb{E}_{\hat{\mathbf{y}} \sim P_{\hat{\mathbf{y}}}} [\max(0, \|\nabla_{\hat{\mathbf{y}}} C(\hat{\mathbf{y}})\|_2 - 1)]$$

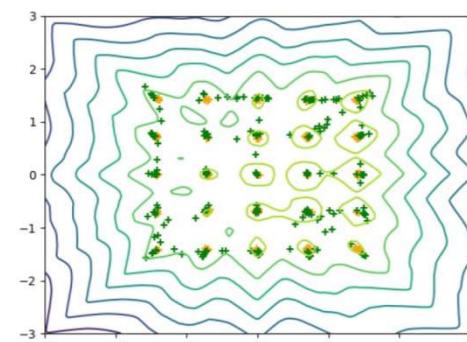
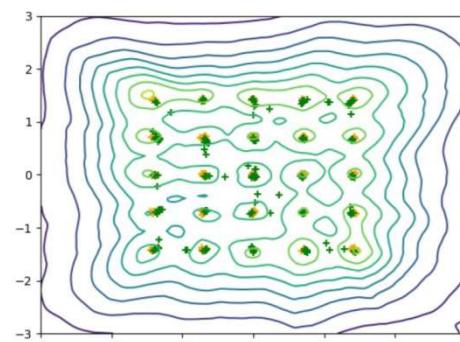
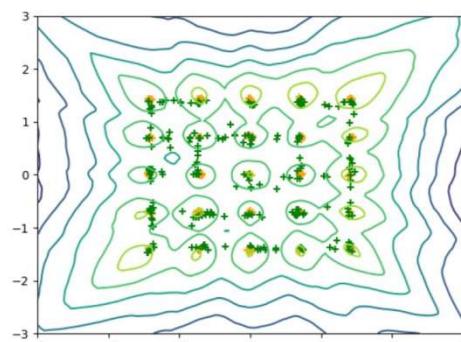
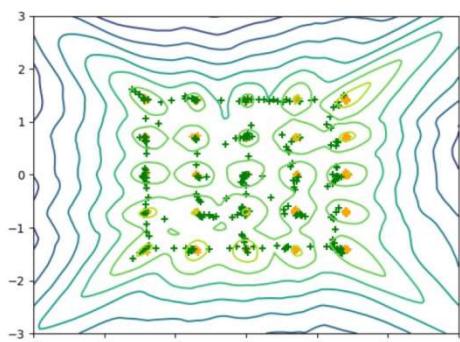
$$\overline{\nabla_{\hat{\mathbf{y}}} C(\hat{\mathbf{y}})} = \eta \overline{\nabla_{\hat{\mathbf{y}}} C(\hat{\mathbf{y}})} + (1 - \eta) \frac{\nabla_{\hat{\mathbf{y}}} C(\hat{\mathbf{y}})}{\lambda}$$

Adaptive Penalty

2.5k iterations



5k iterations



(a) WGAN

(b) AdaWGAN

(c) SWGAN

(d) Proposed AdaSWGAN

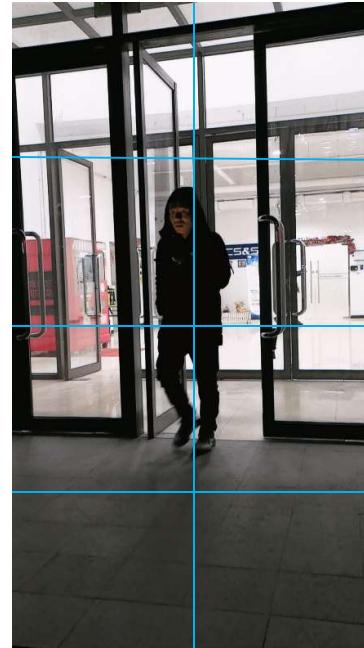
# High-resolution Issue for Image Enhancement



Input



Downscaling  
(low-res, noisy,  
blurry)  
*Deep Photo  
Enhancer (DPE)*  
*[Chen et al in  
CVPR'18]*



Patch-wise Enhancement  
(spatial inconsistency)  
*Weakly Supervised Photo  
Enhancer (WESPE) [our work  
in CVPR'18 workshop]*



Multi-scale Photo  
Enhancement (MUSPE)  
*Our current work*

coarse

fine

# Multi-scale Extension of DACAL for Image Enhancement

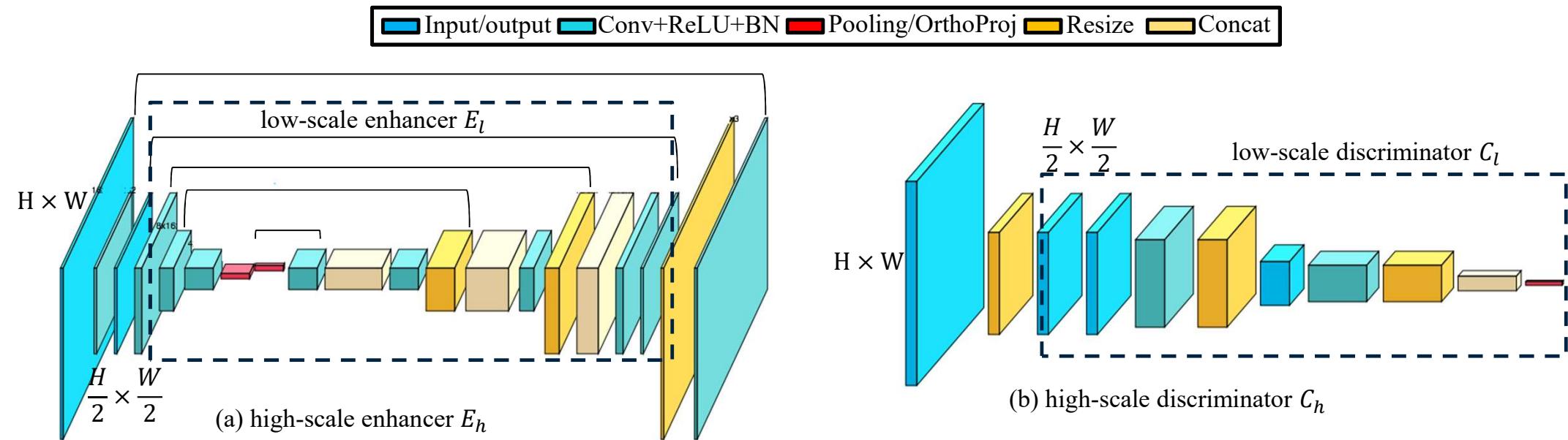


Table 1: PSNR and SSIM results for the MIT-Adobe FiveK [42] test images. Here, WB and DR indicate the White-Box and Distort-and-Recover methods, respectively.  $MUSPE_{l_1}$ ,  $MUSPE_{l_2}$ ,  $MUSPE_{l_3}$  and  $MUSPE_l$  represent the use of individual additive, individual multiplicative, multiplicative cascaded by additive, and our suggested parallel fusion (two-stream strategy), respectively.  $MUSPE_h$  is our higher-scale version.  $PSNR_d/SSIM_d$  and  $PSNR_f/SSIM_f$  indicate the results on downsampled images and full-resolution images, respectively.

	WB	DR	DPED	DPE	$MUSPE_{l_1}$	$MUSPE_{l_2}$	$MUSPE_{l_3}$	$MUSPE_l$	$MUSPE_h$
$PSNR_d$	18.86	21.64	21.05	22.10	22.73	22.99	23.01	23.52	<b>24.15</b>
$PSNR_f$	19.09	21.52	20.86	21.65	22.43	22.69	23.02	23.56	<b>24.07</b>
$SSIM_d$	0.928	0.936	0.922	0.947	0.958	0.942	0.949	0.959	<b>0.962</b>
$SSIM_f$	0.920	0.922	0.916	0.894	0.948	0.942	0.940	0.954	<b>0.956</b>

Table 2: PSNR and SSIM results for the DPED [14] test  $100 \times 100$  image patches. Here,  $l, f, d$  for MUSPE represent the use of our proposed sliced-perception, sliced-frequency and sliced-dimension learning respectively.  $MUSPE_h$  is our higher-scale version.

	WESPE	DPE	$MUSPE_l$	$MUSPE_{l+f}$	$MUSPE_{l+f+d}$	$MUSPE_h$
$PSNR_{100}$	17.45	18.53	19.62	20.01	20.43	<b>20.90</b>
$SSIM_{100}$	0.854	0.861	0.868	0.869	0.872	<b>0.874</b>



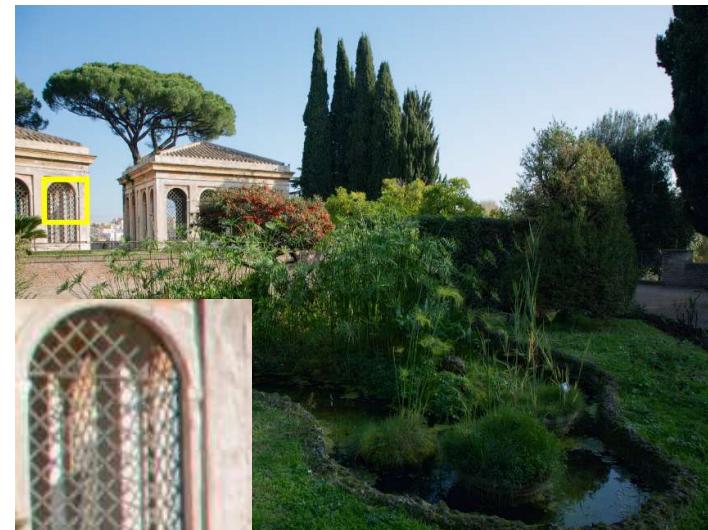
Input



DPE [Chen, CVPR'18]



WESPE [Ignatov, CVPRW'18]



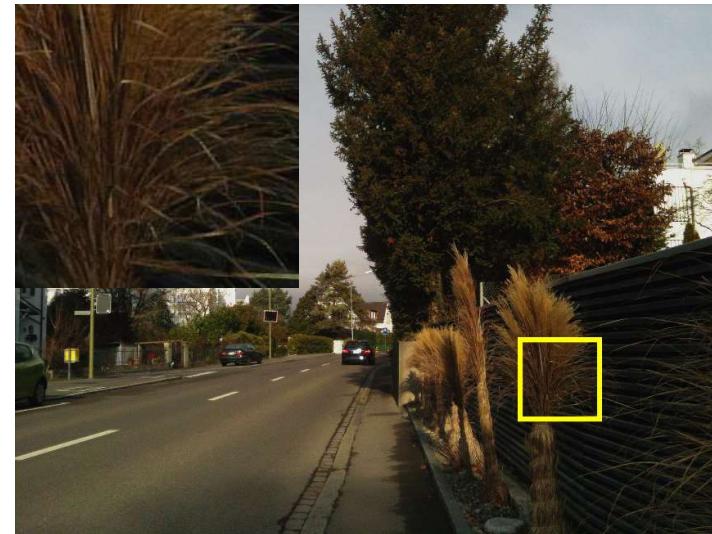
Proposed MUSPE [ICLR'20 submission]



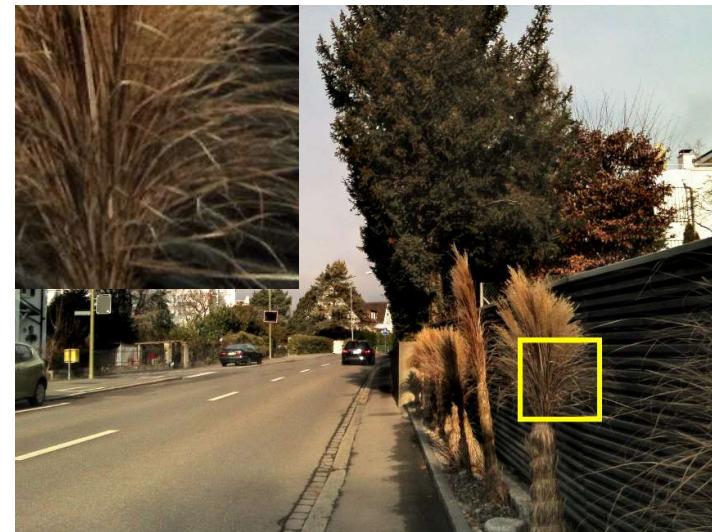
Input



DPE [Chen, CVPR'18]



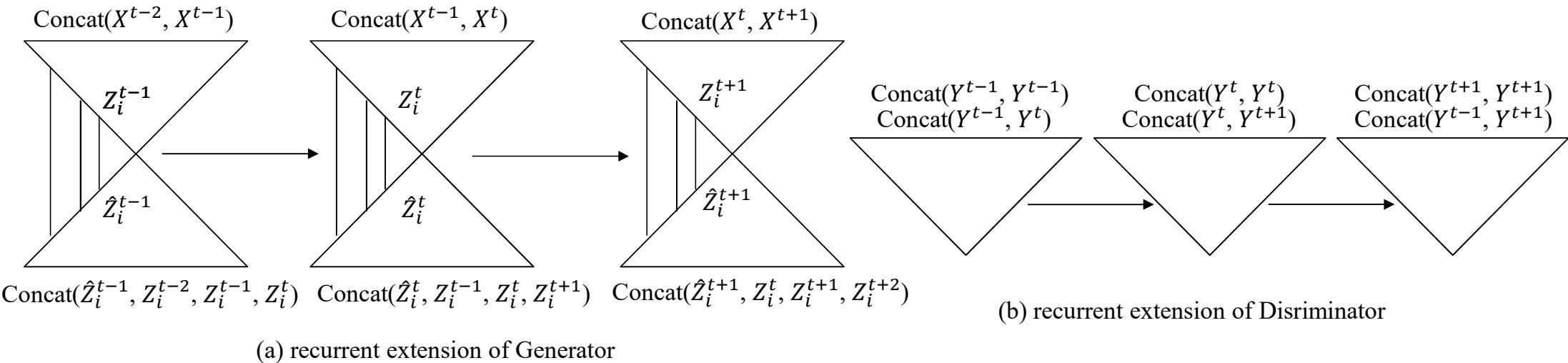
WESPE [Ignatov, CVPRW'18]



Proposed MUSPE [ICLR'20 submission]



## Recurrent Extension of DACAL for Video Enhancement





Perframe-DACAL



Recurrent-DACAL



Perframe-DACAL



Recurrent-DACAL (fine-tuned on Retouched&amp;DSLR images)



Perframe-DACAL



Recurrent-DACAL



Perframe-DACAL



Recurrent-DACAL

## Conclusion

- Sliced Wasserstein distance
  - Lower sample complexity
  - Lower projection complexity
- Sliced Wasserstein generative models for image & video generation
  - AE-based generative models
    - Penalization free on latent variables
    - Trained easier
  - GAN models
    - Easier dual form approximation
    - Reach state-of-the-art
- Divide-and-Conquer Adversarial Learning models for image & video enhancement
  - Hierarchical decomposition of complexity
    - Adaptive sliced Wasserstein distance learning



# Q&A

