Generative models

VAE, GANS, Diffusion models

Karel Zimmermann

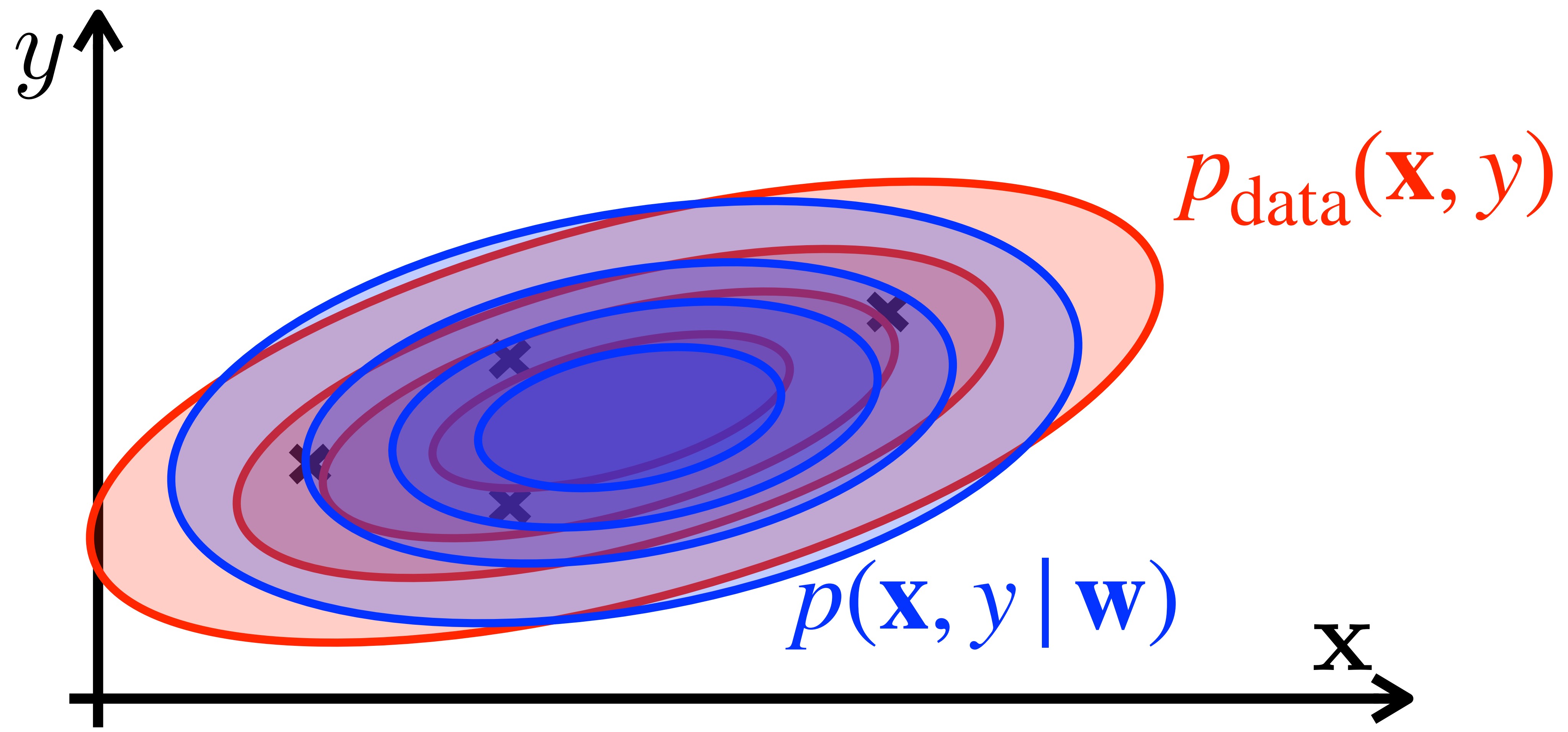
Prerequisites: Learning vs optimization

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big) = \int_{(\mathbf{x},y)} p_{\text{data}}(\mathbf{x}, y) \cdot \log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[\log \frac{p_{data}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}\right] = \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[-\log p(\mathbf{x}, y \,|\, \mathbf{w})\right]$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[\log p_{\text{data}}(\mathbf{x}, y) - \log p(y \,|\, \mathbf{x}, \mathbf{w}) p(\mathbf{x})\right]$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[-\log p(y \,|\, \mathbf{x}, \mathbf{w})\right] \approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} \left[-\log p(y_i \,|\, \mathbf{x}_i, \mathbf{w})\right]$$



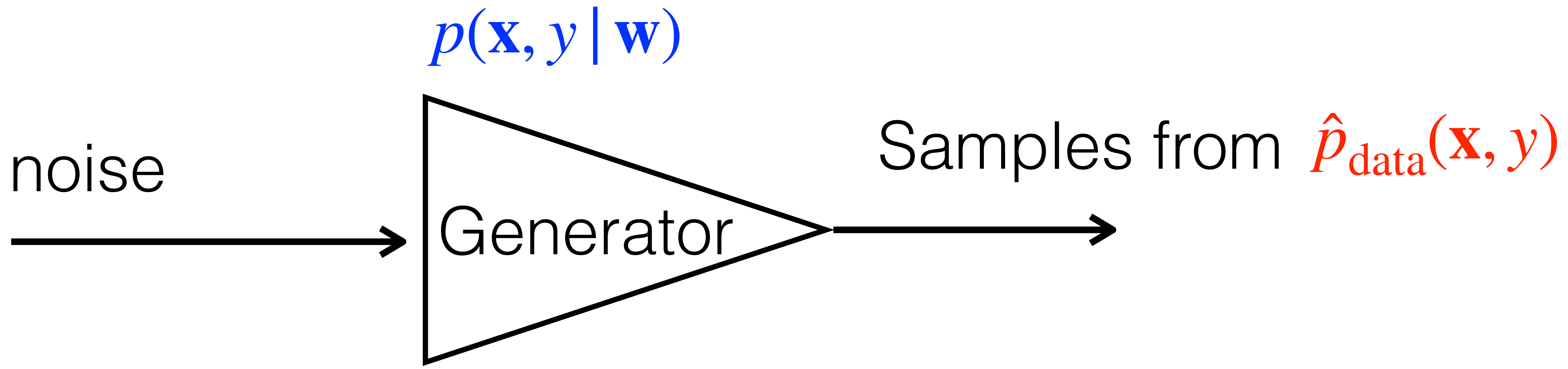
So-far we always modeled $p(y_i \,|\, \mathbf{x}_i, \mathbf{w})$

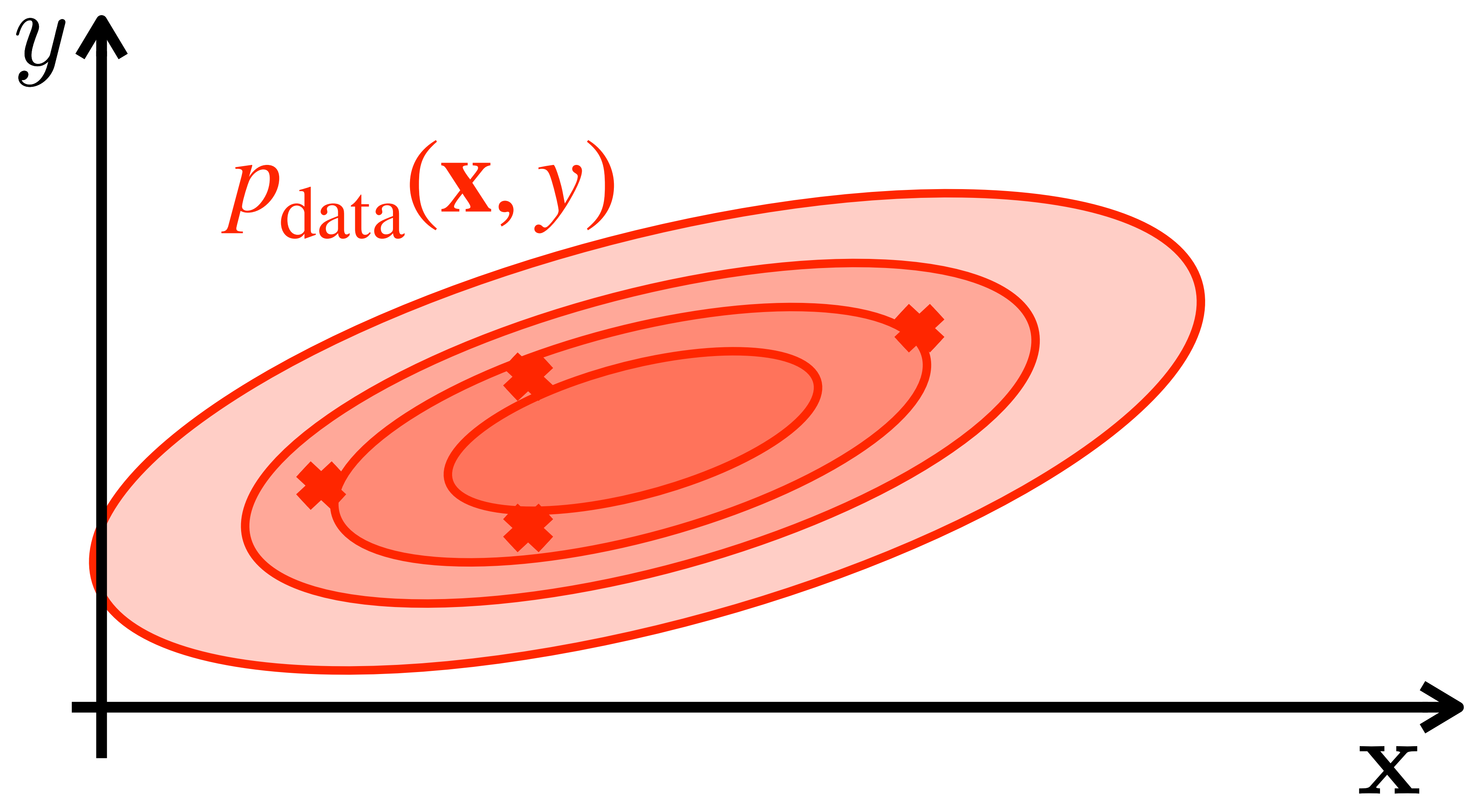
How comes?

What else can be modeled?

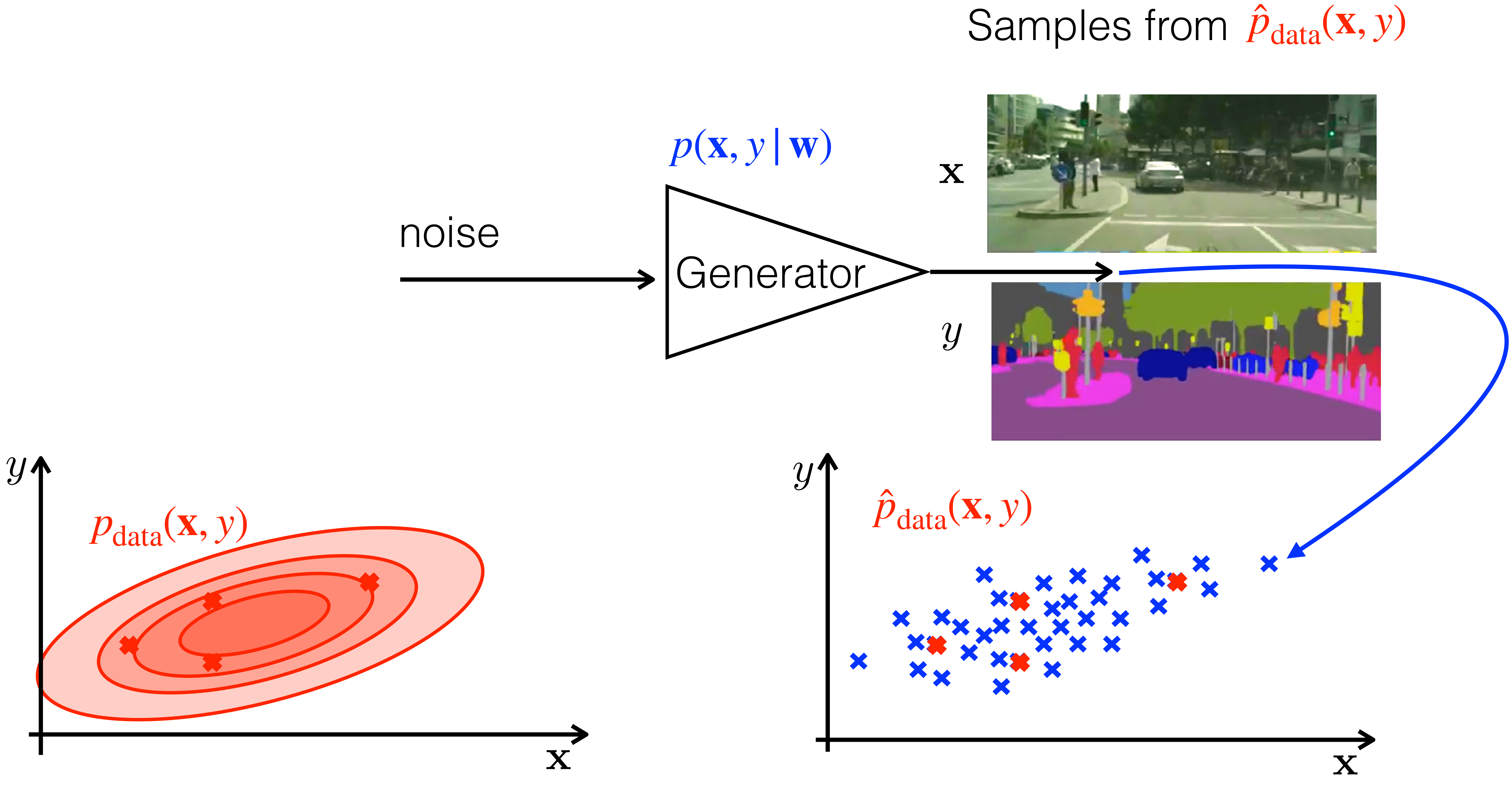
$p(\mathbf{x}, y \,|\, \mathbf{w})$ $\quad p(\mathbf{x} \,|\, y, \mathbf{w})$ $\quad p(\mathbf{x} \,|\, \mathbf{w})$

Dataset augmentation

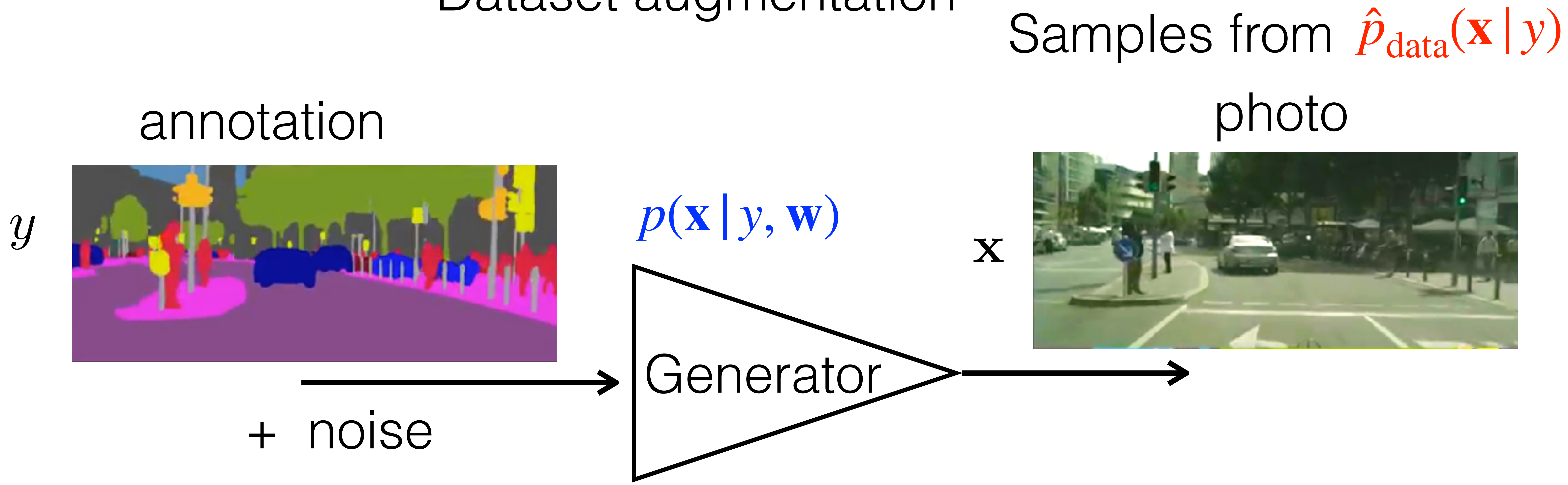




Dataset augmentation

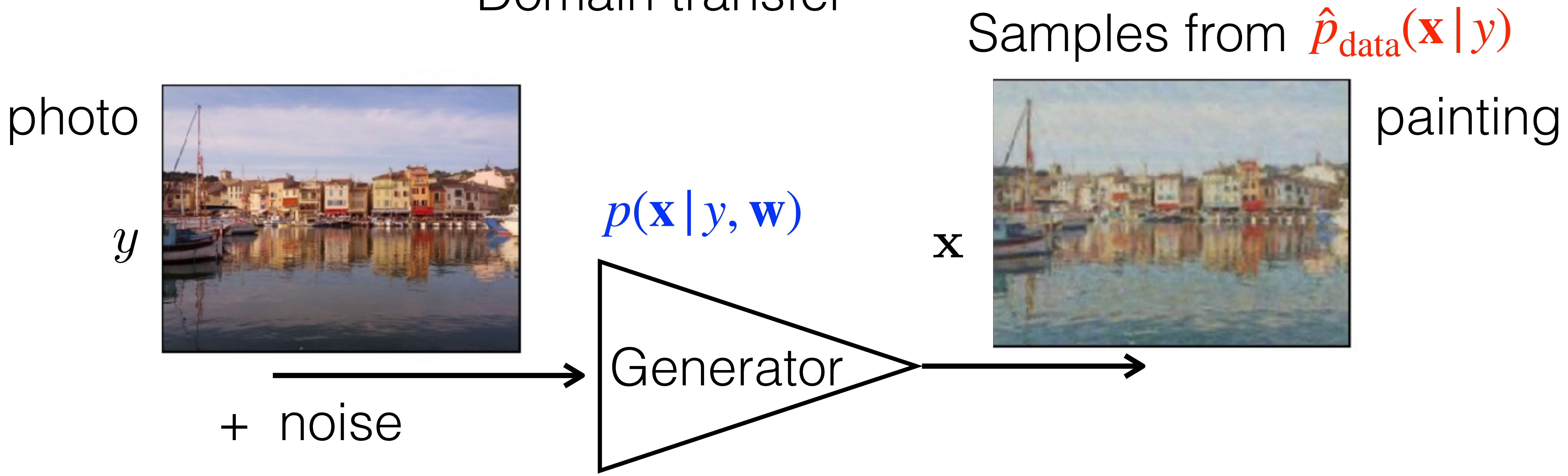


Dataset augmentation



Gives better control over generated data

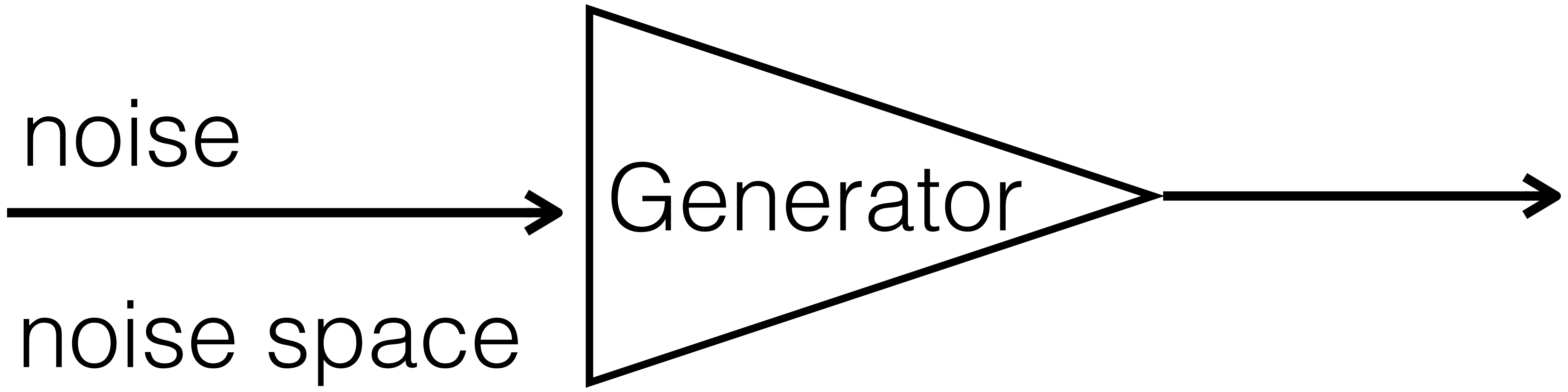
Domain transfer



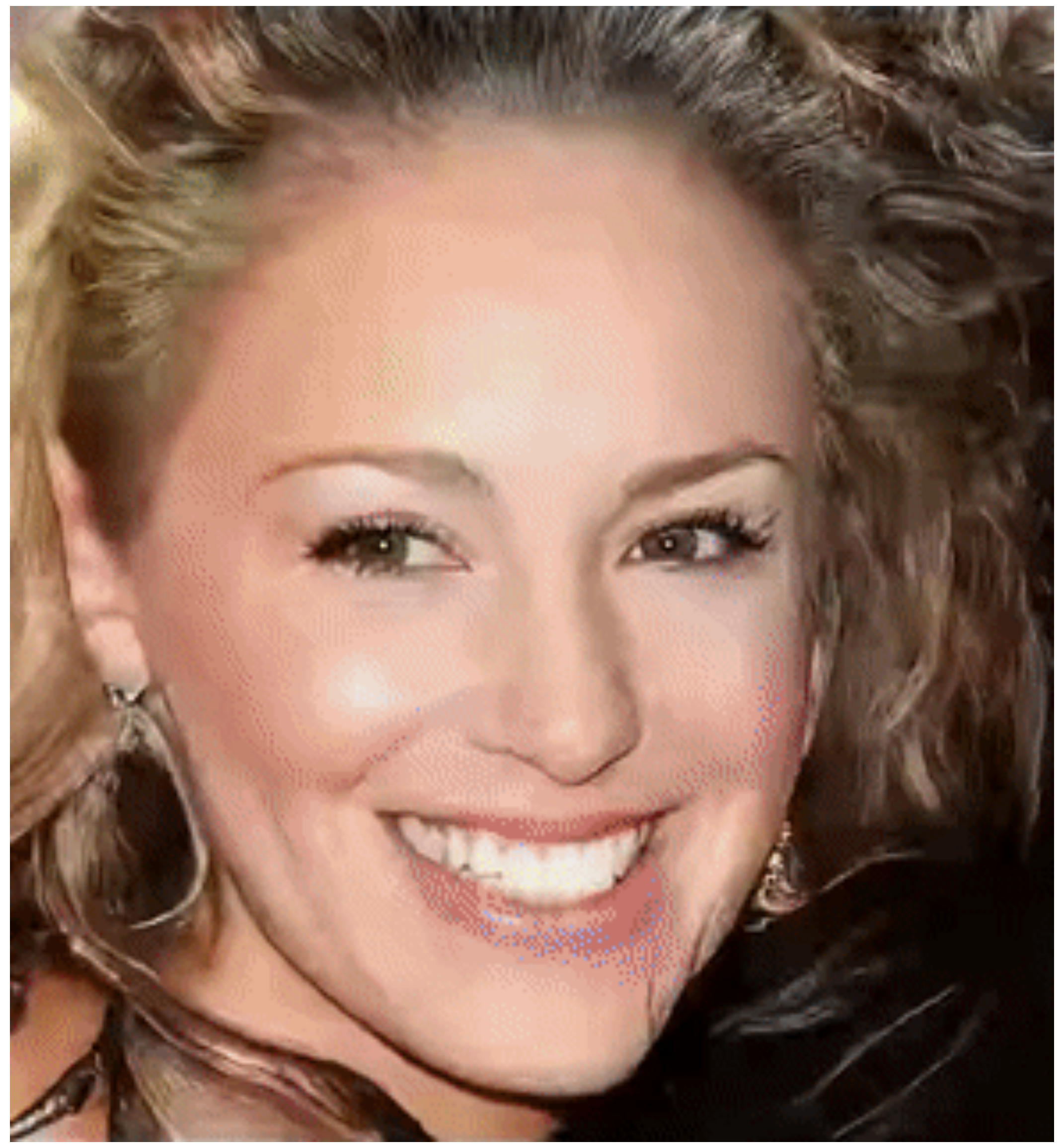
Classification dataset augmentation

Samples from $\hat{p}_{\text{data}}(\mathbf{x})$

$p(\mathbf{x} \mid \mathbf{w})$



random walk in noise space



[ProGAN]

How can I do it?

- Dataset augmentation
- Realistic simulator
- Generative networks

RGB images

Depth images

Stencil layer

Stencil layer - **cars**

Stencil layer - **humans**

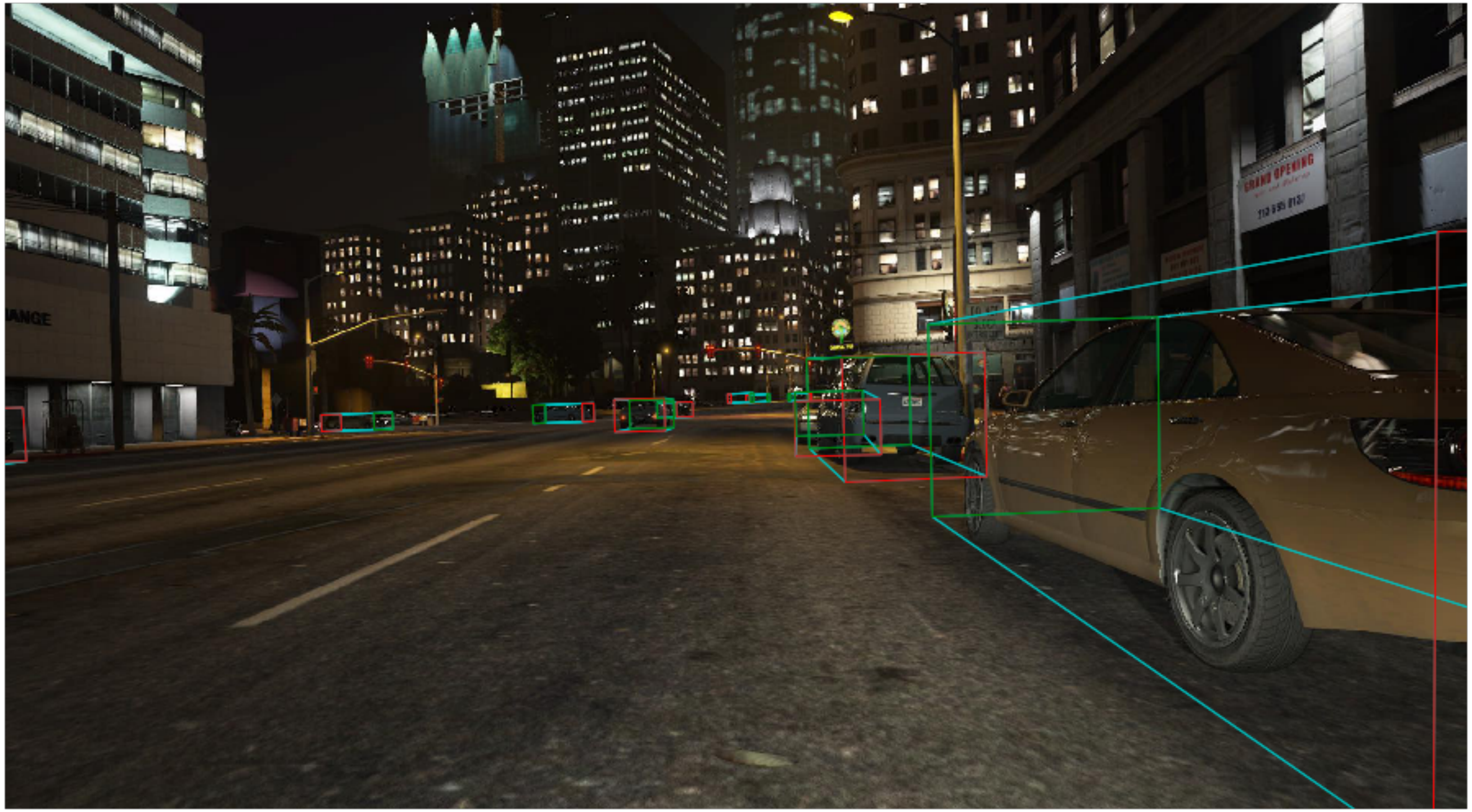
Stencil layer - **trees**

Stencil layer - **sky**

Stencil layer - **artificial light**

Stencil layer - **artificial light**

Other annotations for objects (e.g. cars, humans)



Other annotations for objects (e.g. cars, humans)

What can be controlled (night/day)

What can be controlled (night/day)

What can be controlled (weather): ExtraSunny

What can be controlled (weather): Clear

What can be controlled (weather): Foggy

What can be controlled (weather): OverCast

What can be controlled (weather): Raining

What can be controlled (weather): ThunderStorm

What can be controlled (weather): Clearing

What can be controlled (weather): SnowLight

What can be controlled (mods): Tsunami

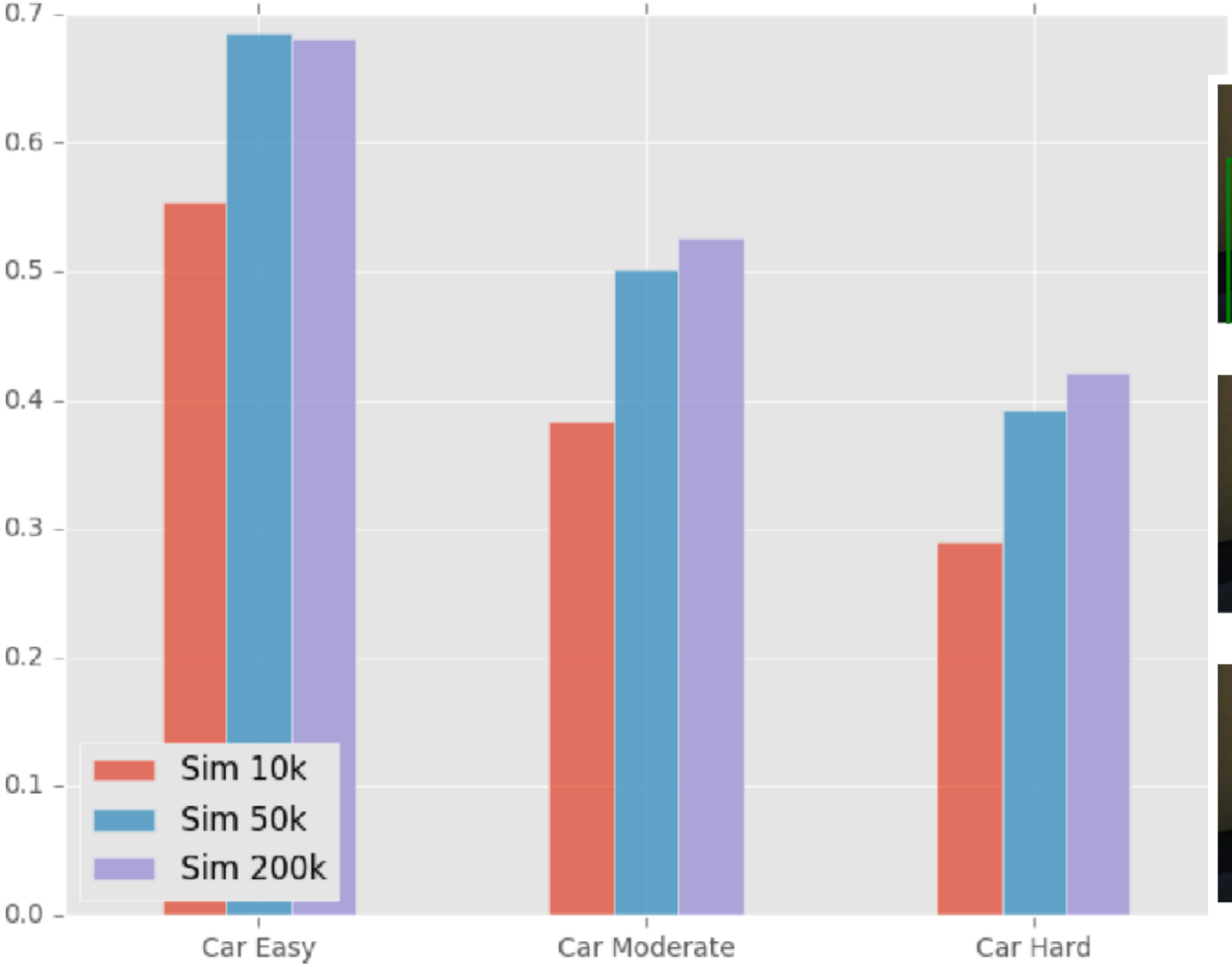
https://cs.gta5-mods.com/misc/tsunami-mod

What can be controlled (mods): Plane crashing mods

https://cs.gta5-mods.com/scripts/planes-hails

Driving in the matrix [Roberson ICRA 2017]
https://arxiv.org/abs/1610.01983

- Reverse engineering of GTA 5 (RAGE engine)



(a) 10k

easy

(b) 50k

moderate

(c) 200k

hard

How can I do it?

- Dataset augmentation
- Realistic simulator
- Generative networks

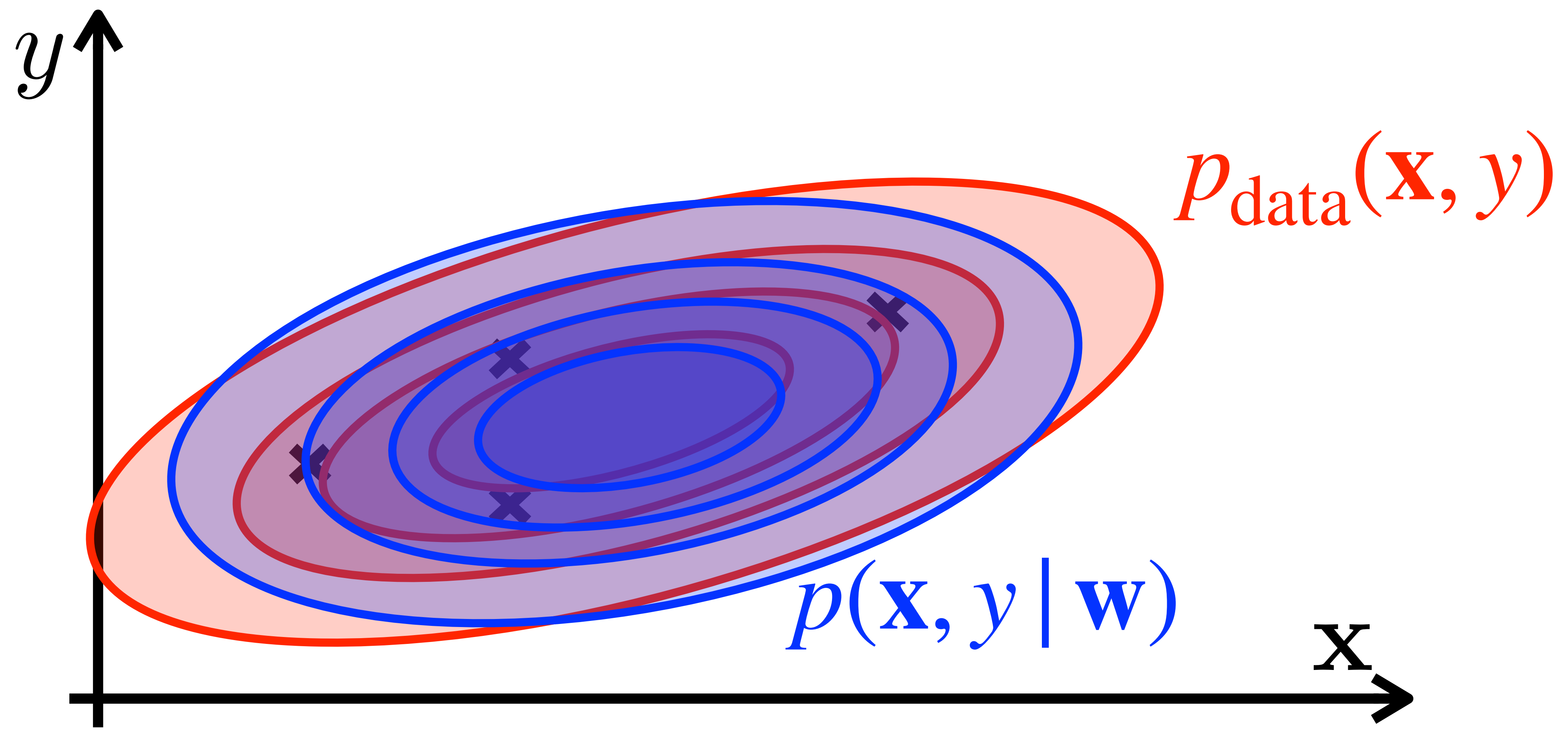
How can I do it?

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big) = \int_{(\mathbf{x},y)} p_{\text{data}}(\mathbf{x}, y) \cdot \log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[\log \frac{p_{data}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}\right] = \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[-\log p(\mathbf{x}, y \,|\, \mathbf{w})\right]$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[\log p_{\text{data}}(\mathbf{x}, y) - \log p(y \,|\, \mathbf{x}, \mathbf{w})p(\mathbf{x})\right]$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[-\log p(y \,|\, \mathbf{x}, \mathbf{w})\right] \approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i)\sim p_{\text{data}}(\mathbf{x}, y)} \left[-\log p(y_i \,|\, \mathbf{x}_i, \mathbf{w})\right]$$



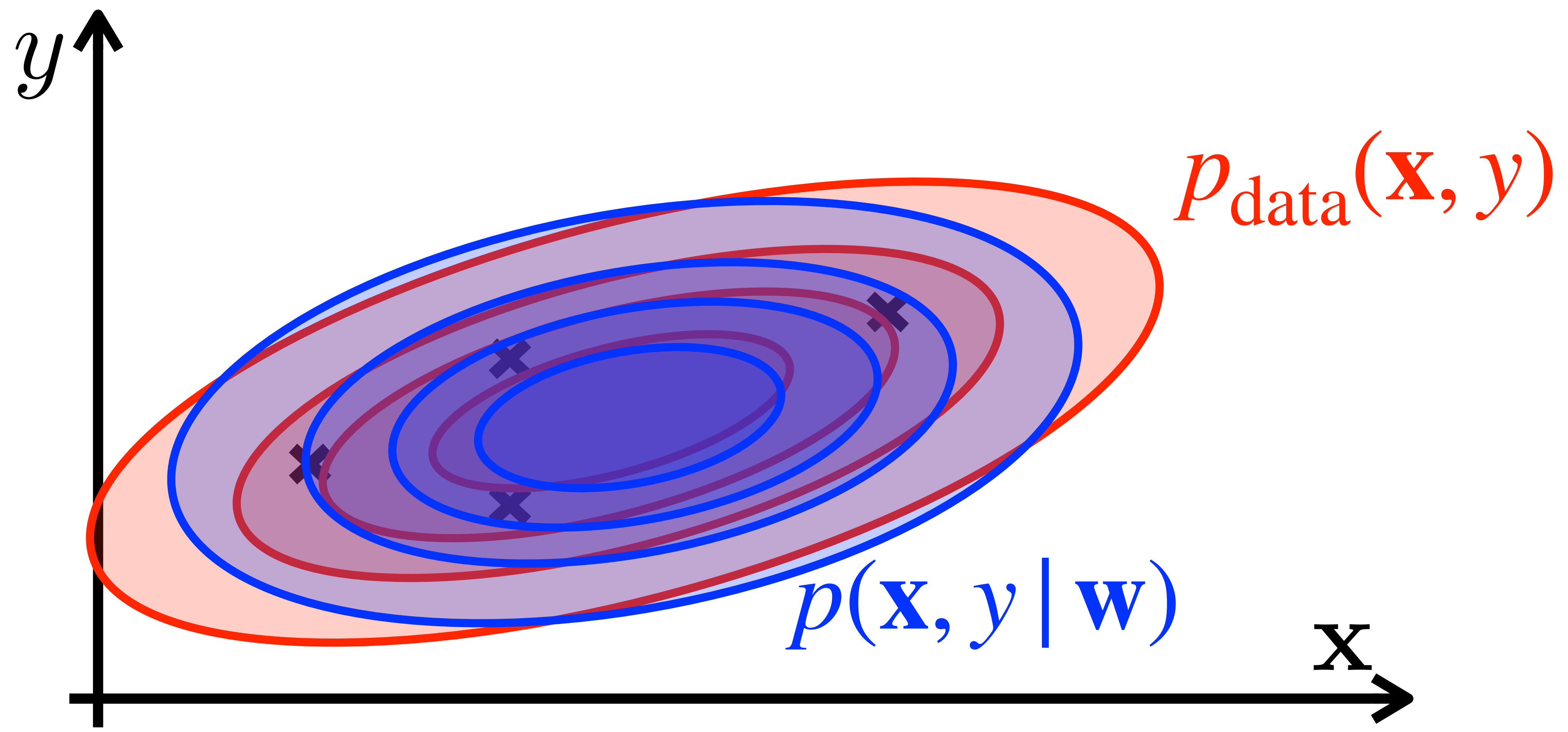
How can I do it?

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} D_{KL}\big(p_{\text{data}}(\mathbf{x}, y) \,\|\, p(\mathbf{x}, y \,|\, \mathbf{w})\big) = \int_{(\mathbf{x},y)} p_{\text{data}}(\mathbf{x}, y) \cdot \log \frac{p_{\text{data}}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}$$

$$= \arg\min_{\mathbf{w}} \; \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[\log \frac{p_{data}(\mathbf{x}, y)}{p(\mathbf{x}, y \,|\, \mathbf{w})}\right] = \arg\min_{\mathbf{w}} \; \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[-\log p(\mathbf{x}, y \,|\, \mathbf{w})\right]$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[\log p_{\text{data}}(\mathbf{x}, y) - \log p(\mathbf{x} \,|\, y, \mathbf{w}) p(y)\right]$$

$$= \arg\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[-\log p(\mathbf{x} \,|\, y, \mathbf{w})\right] \approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \sim p_{\text{data}}(\mathbf{x}, y)} \left[-\log p(\mathbf{x}_i \,|\, y_i, \mathbf{w})\right]$$



Dataset augmentation

Samples from $\hat{p}_{\text{data}}(\mathbf{x} \mid y)$

annotation

y

photo

$\mathbf{x}$

+ noise → Generator →

$$\approx \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i)} \left[-\log p(\mathbf{x}_i \mid y_i, \mathbf{w}) \right]$$

Is it that easy?

Photos corresponding to annotation comes from

- high-dimensional,
- intricate
- non-gaussian pdf

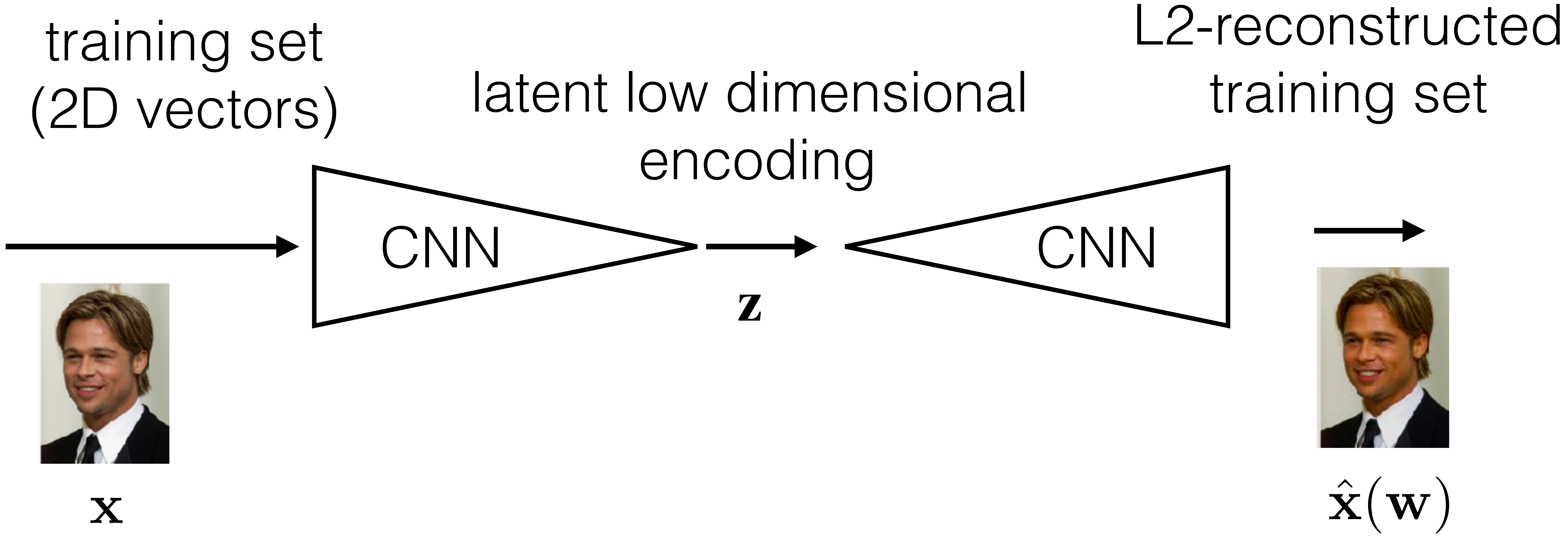
L2-loss is obviously wrong =>Where do I get the loss?

Deep Feature interpolations [Upchurch CVPR 2017]
https://arxiv.org/pdf/1611.05507.pdf

Generate samples by projecting images into low-dimensional representation

Generative models

You can do it even without any annotations just on collection of images



- Learning the self-reconstruction with L2 reconstruction loss

$$\arg\min_{\mathbf{w}} \|\mathbf{x} - \hat{\mathbf{x}}(\mathbf{w})\|_2^2$$

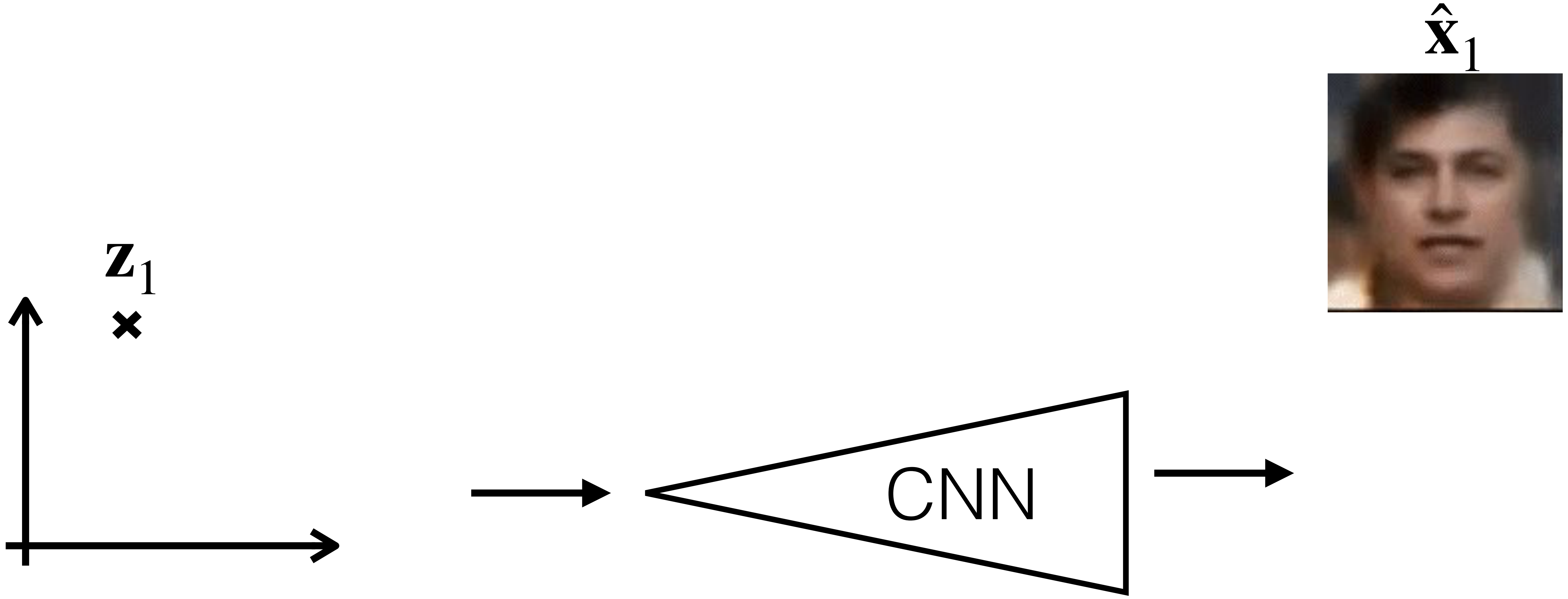
If CNN is pure linear function then:

- closed-form solution exists
- method is called PCA

If $\mathbf{z}$ is pushed towards gauusian distribution:

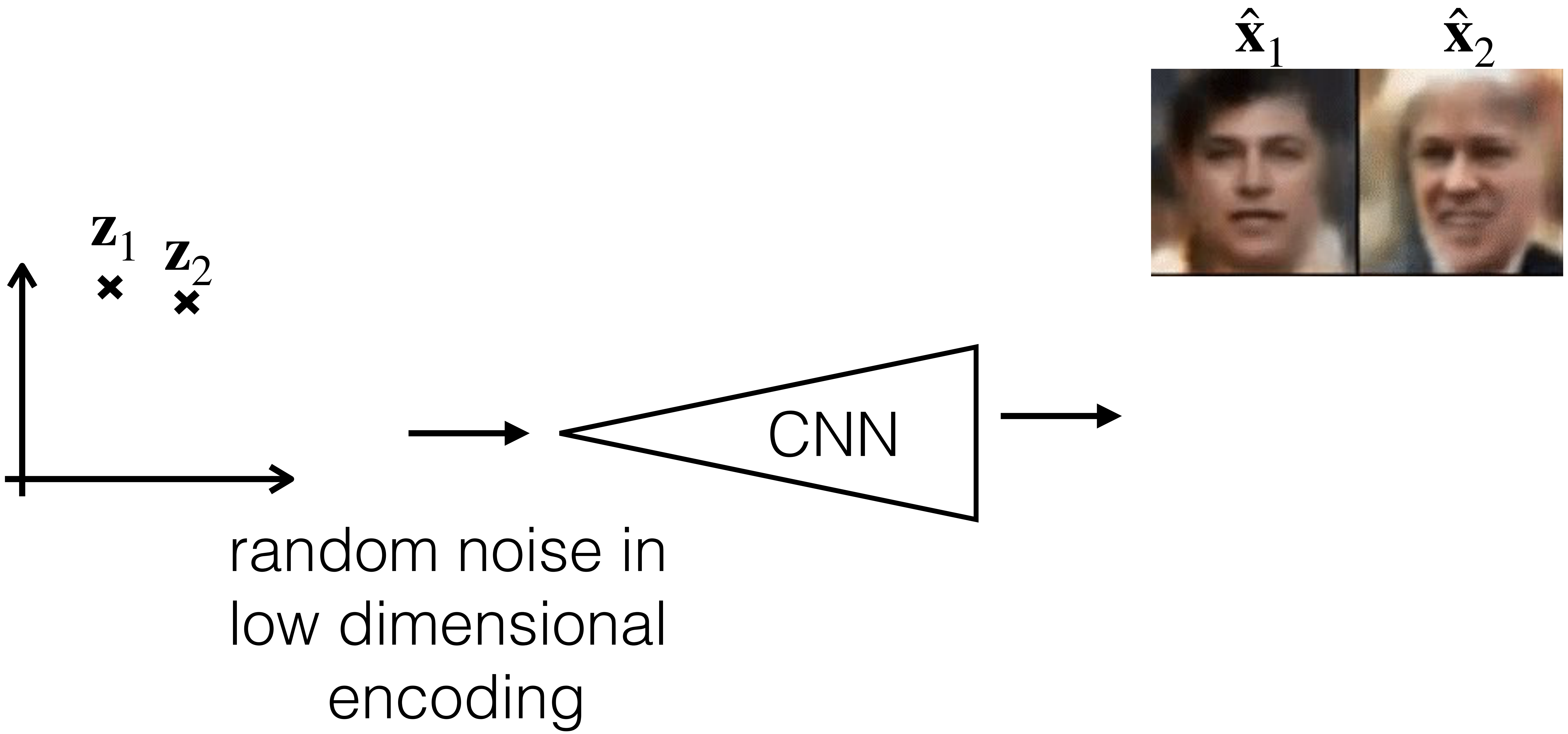
- method is referred as variational encoders

Generative models



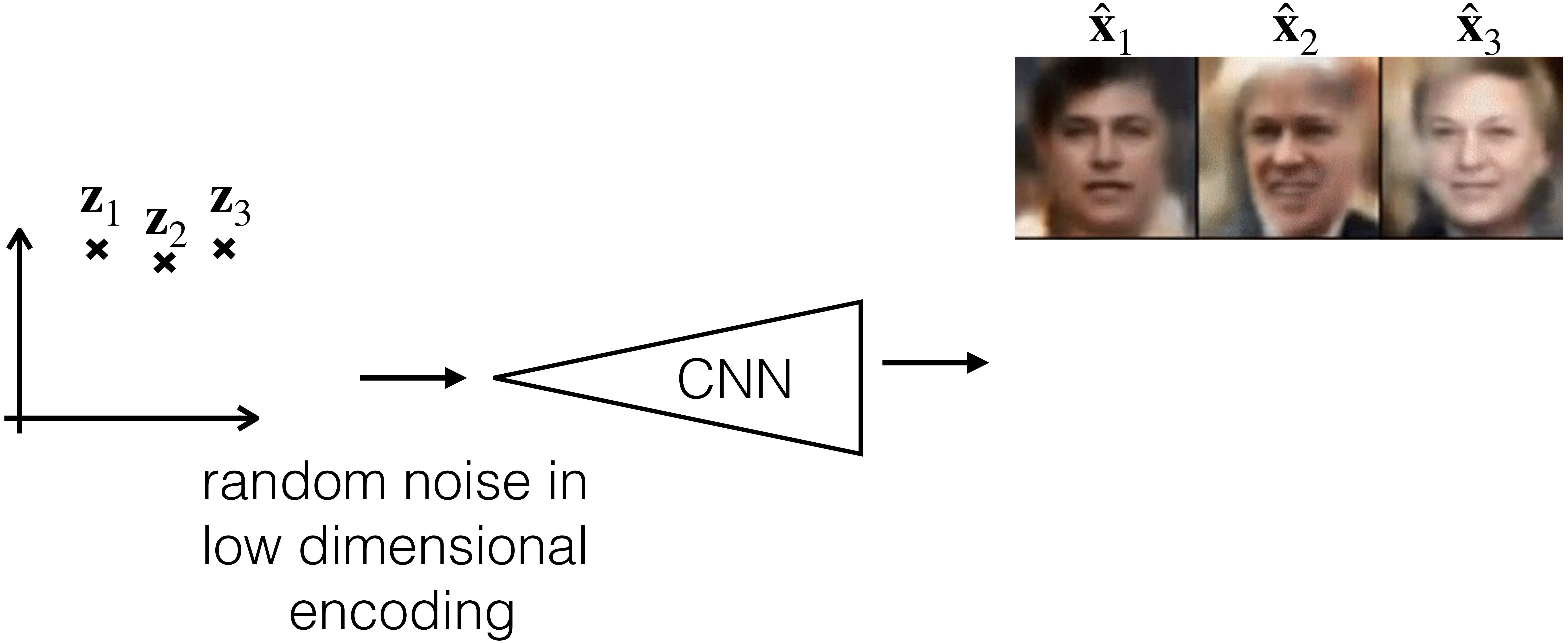
- New samples generated from random vectors in low-dimensional encoding.

Generative models



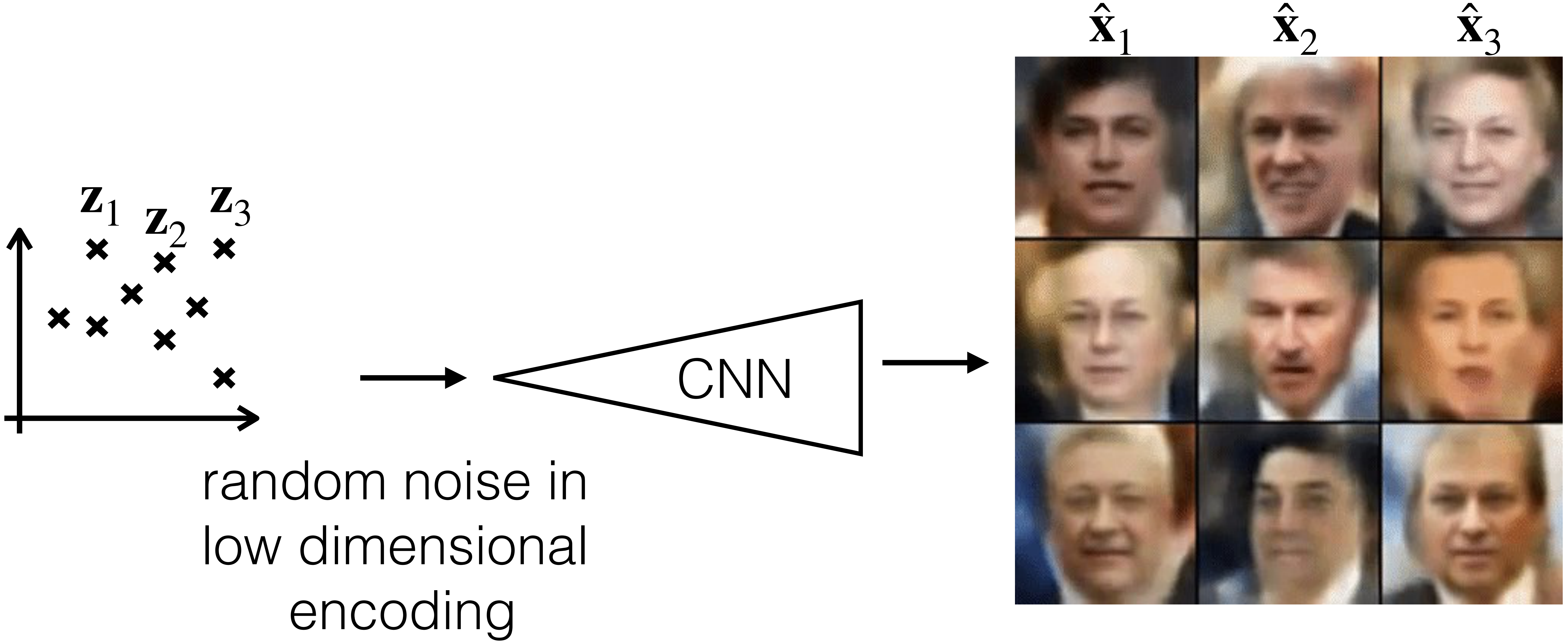
- New samples generated from random vectors in low-dimensional encoding.

Generative models



- New samples generated from random vectors in low-dimensional encoding.

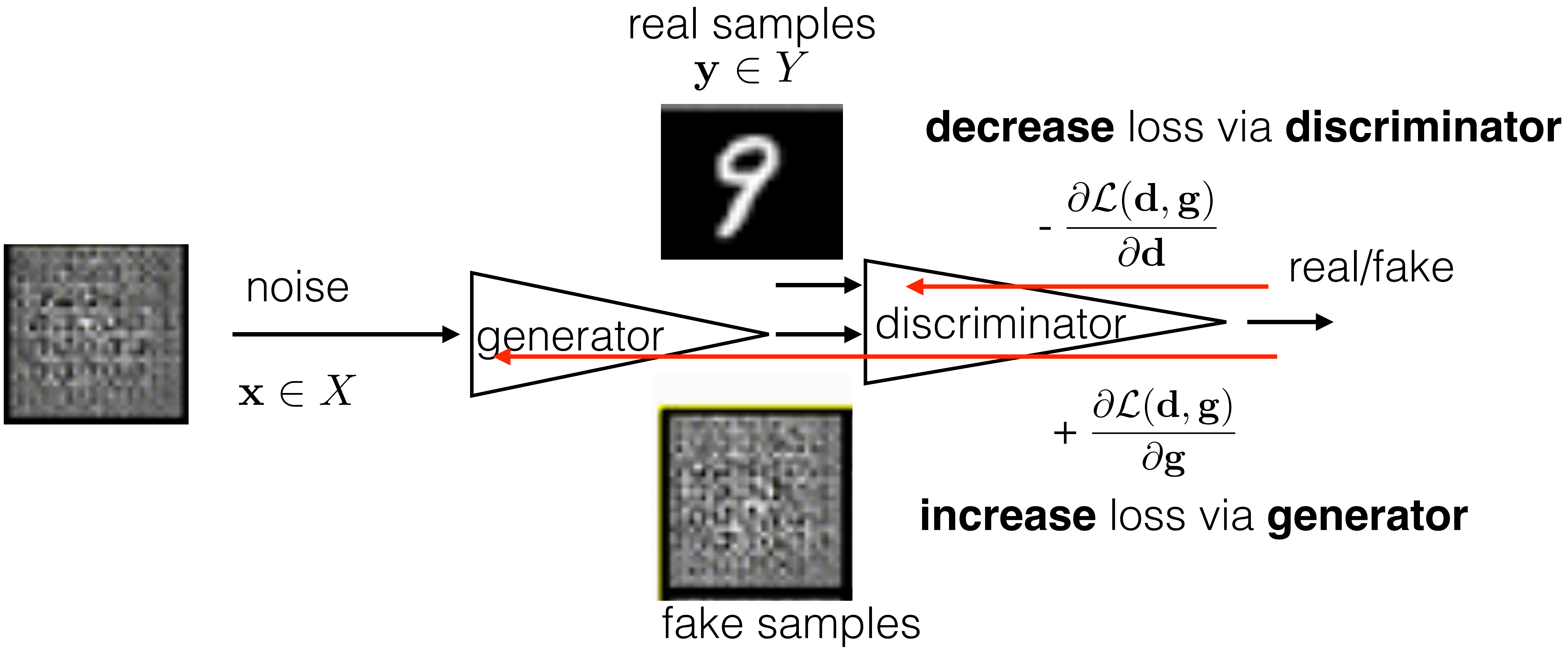
Generative models



- New samples generated from random vectors in low-dimensional encoding.

Generative Adversarial Nets [Goodfellow NIPS 2014]

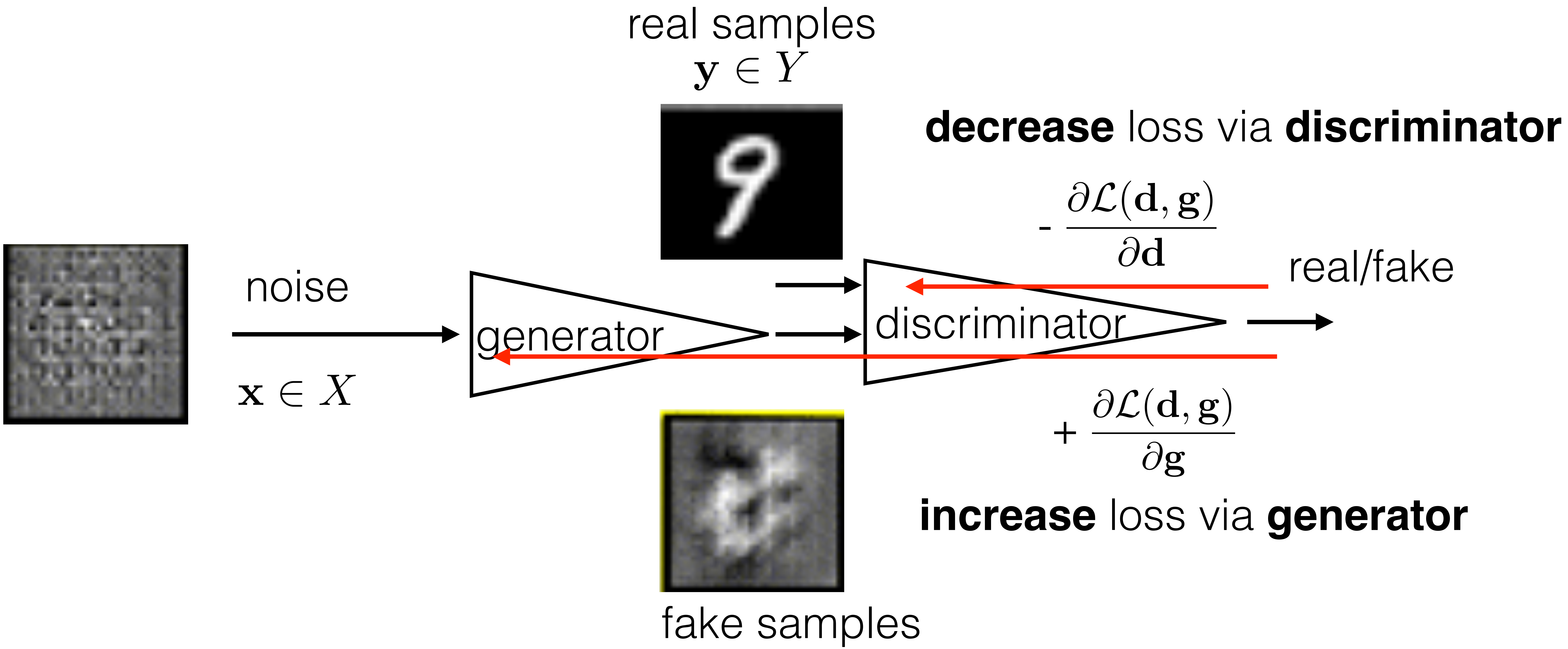
https://arxiv.org/abs/1406.2661



classification loss: $\mathcal{L}(\mathbf{d}, \mathbf{g}) = \sum_{\mathbf{x} \in X} -\log(\mathbf{d}(\mathbf{g}(\mathbf{x}))) + \sum_{\mathbf{y} \in Y} -\log(1 - \mathbf{d}(\mathbf{y}))$

Generative Adversarial Nets [Goodfellow NIPS 2014]

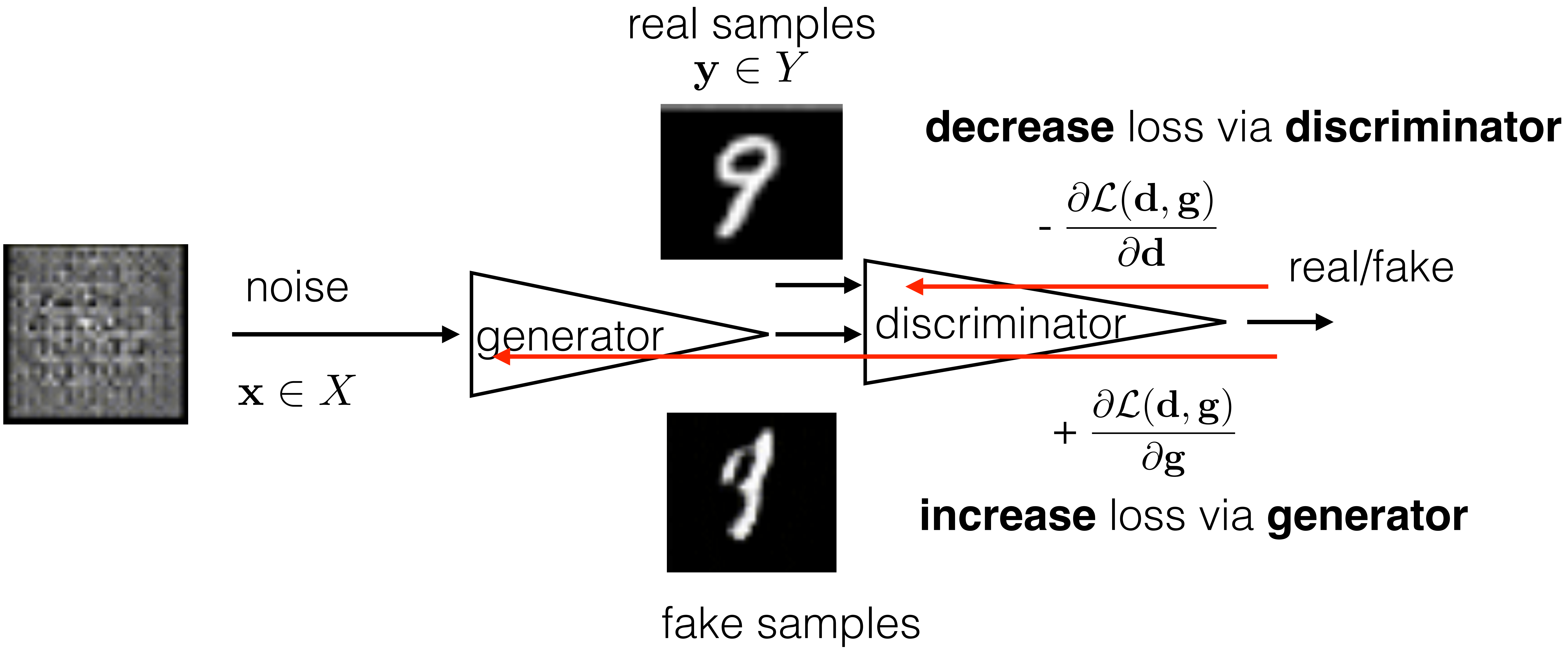
https://arxiv.org/abs/1406.2661



classification loss: $\mathcal{L}(\mathbf{d}, \mathbf{g}) = \sum_{\mathbf{x} \in X} - \log(\mathbf{d}(\mathbf{g}(\mathbf{x}))) + \sum_{\mathbf{y} \in Y} - \log(1 - \mathbf{d}(\mathbf{y}))$

Generative Adversarial Nets [Goodfellow NIPS 2014]

https://arxiv.org/abs/1406.2661



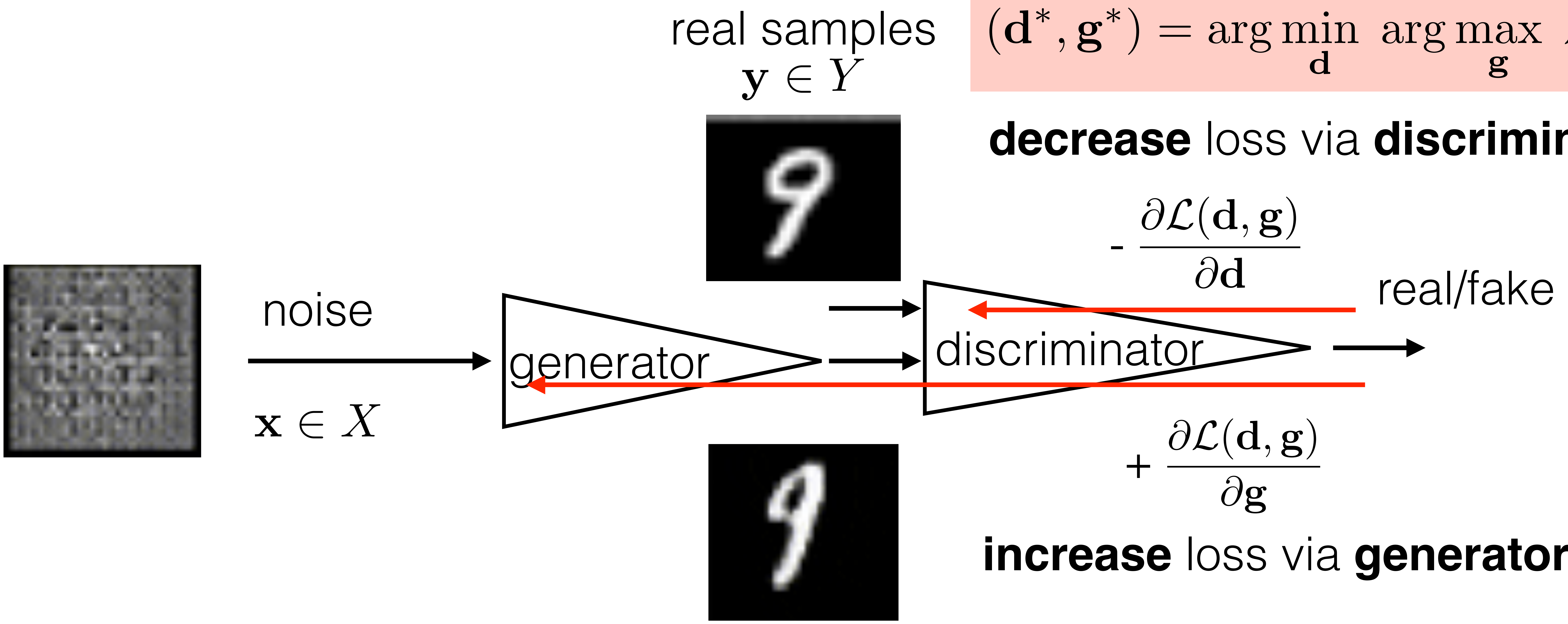
classification loss: $\mathcal{L}(\mathbf{d}, \mathbf{g}) = \sum_{\mathbf{x} \in X} -\log(\mathbf{d}(\mathbf{g}(\mathbf{x}))) + \sum_{\mathbf{y} \in Y} -\log(1 - \mathbf{d}(\mathbf{y}))$

Generative Adversarial Nets [Goodfellow NIPS 2014]

https://arxiv.org/abs/1406.2661

$$(\mathbf{d}^*, \mathbf{g}^*) = \arg\min_{\mathbf{d}} \ \arg\max_{\mathbf{g}} \ \mathcal{L}(\mathbf{d}, \mathbf{g})$$

decrease loss via **discriminator**



increase loss via **generator**

fake samples

classification loss: $\mathcal{L}(\mathbf{d}, \mathbf{g}) = \sum_{\mathbf{x} \in X} - \log(\mathbf{d}(\mathbf{g}(\mathbf{x}))) + \sum_{\mathbf{y} \in Y} - \log(1 - \mathbf{d}(\mathbf{y}))$

Generative Avdersarial Nets [Goodfellow NIPS 2014]

https://arxiv.org/abs/1406.2661

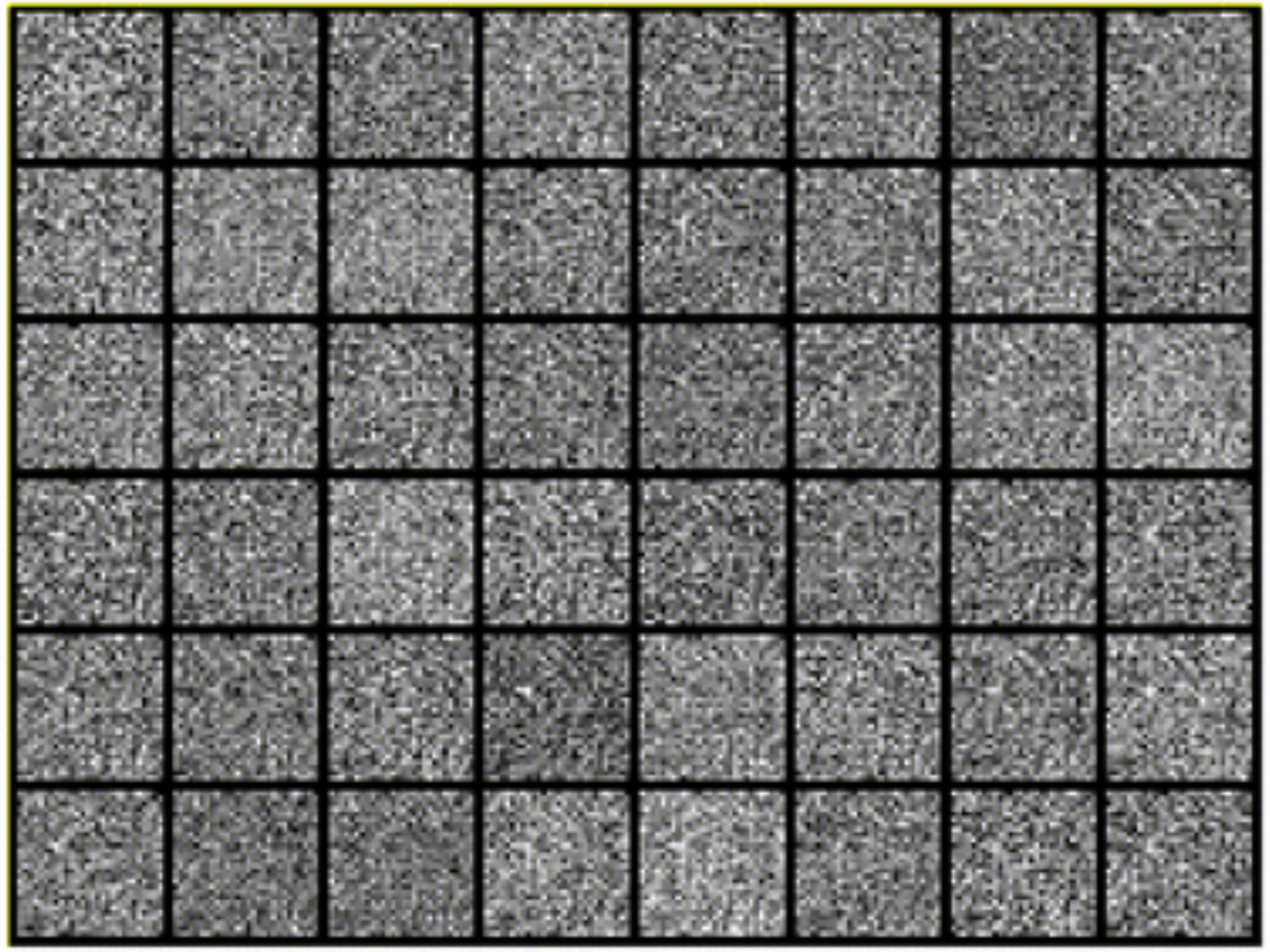
$$(\mathbf{d}^*, \mathbf{g}^*) = \arg\min_{\mathbf{d}} \; \arg\max_{\mathbf{g}} \; \mathcal{L}(\mathbf{d}, \mathbf{g})$$

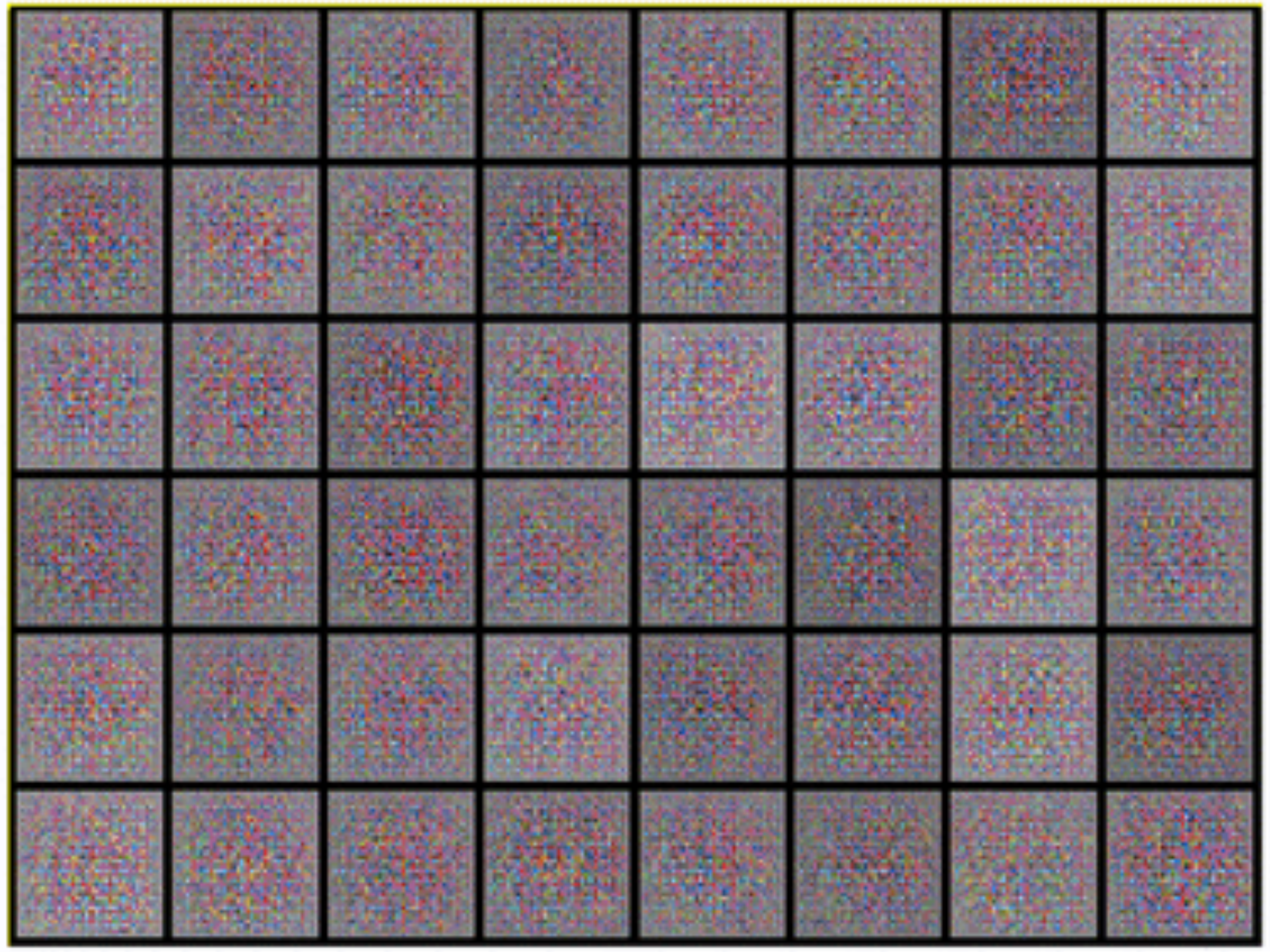
- Proof: Equilibrium in saddle point implies that generator generates samples from th real distribution (asymptotically consistent in contrast to VAE)

Generative Avdersarial Nets [Goodfellow NIPS 2014]
https://arxiv.org/abs/1406.2661

$$(\mathbf{d}^*, \mathbf{g}^*) = \arg\min_{\mathbf{d}} \ \arg\max_{\mathbf{g}} \ \mathcal{L}(\mathbf{d}, \mathbf{g})$$

- Proof: Equilibrium in saddle point implies that generator generates samples from th real distribution (asymptotically consistent in contrast to VAE)





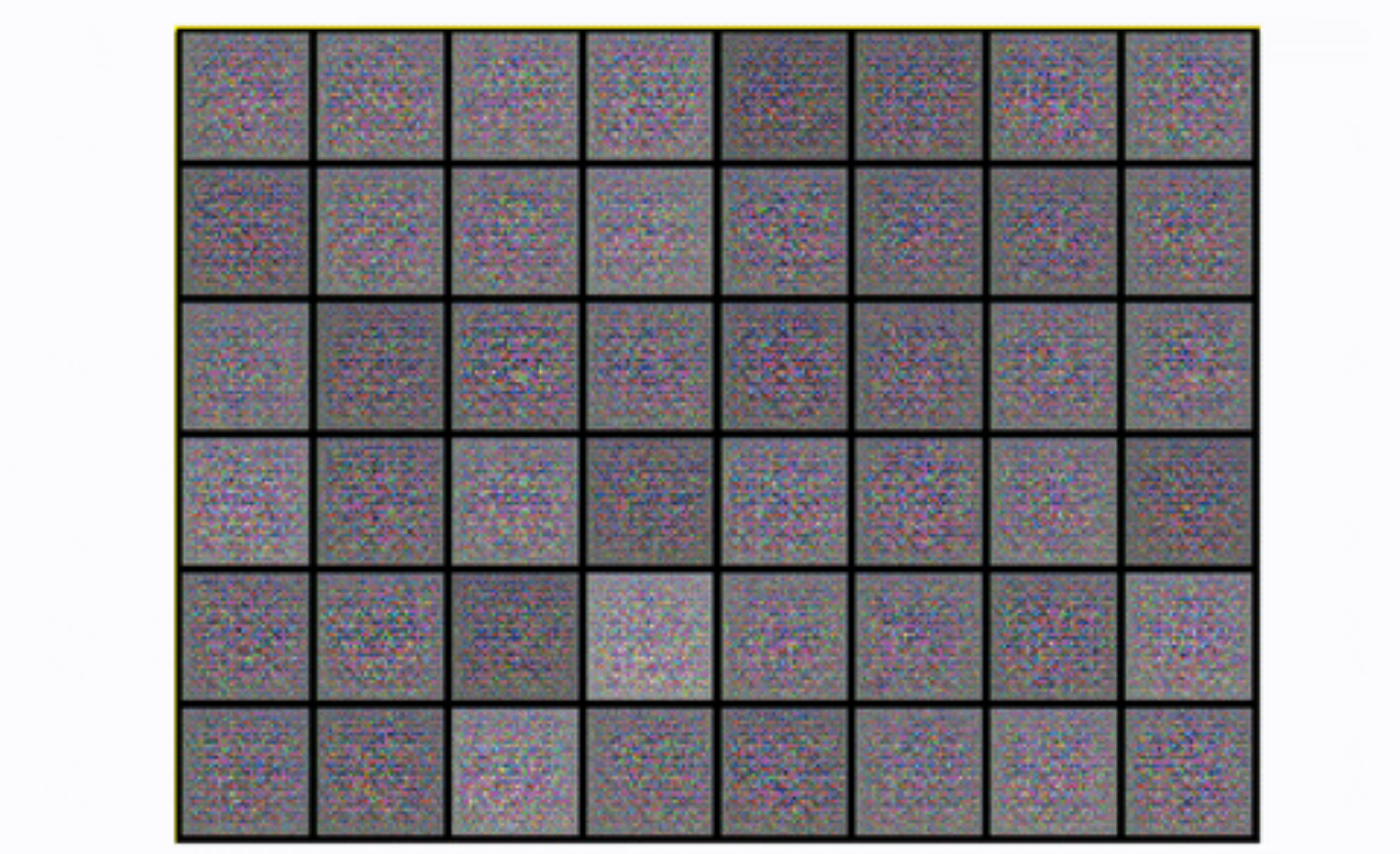
https://theaisummer.com/gan-computer-vision/

Generative Avdersarial Nets [Goodfellow NIPS 2014]

https://arxiv.org/abs/1406.2661

$$(\mathbf{d}^*, \mathbf{g}^*) = \arg\min_{\mathbf{d}} \ \arg\max_{\mathbf{g}} \ \mathcal{L}(\mathbf{d}, \mathbf{g})$$

- The learning is generally unstable and suffers from mode collapse

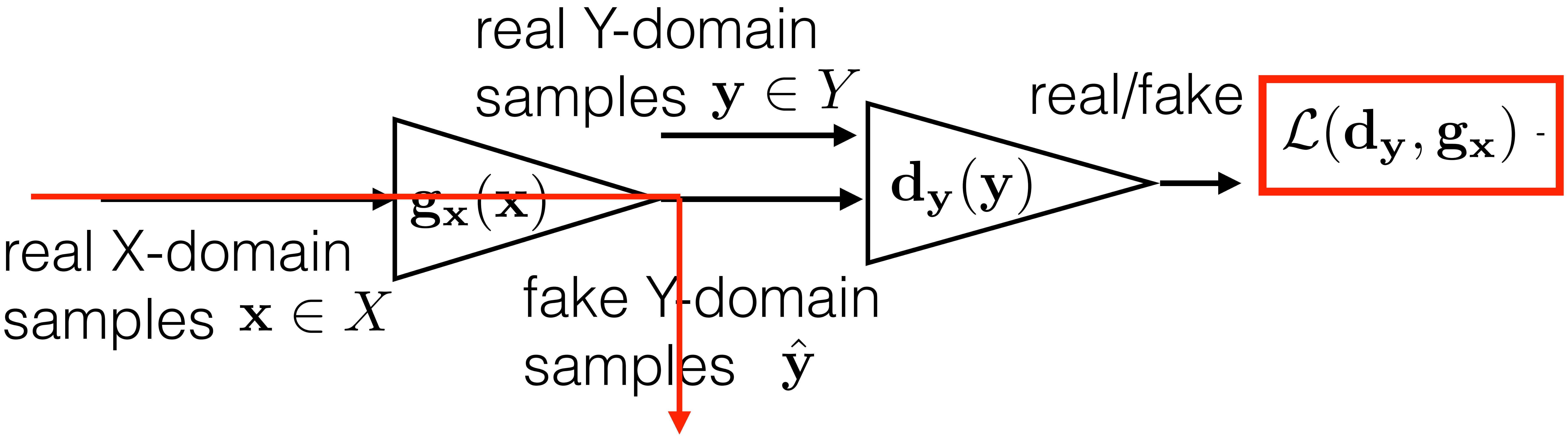


https://theaisummer.com/gan-computer-vision/

Domain transfer

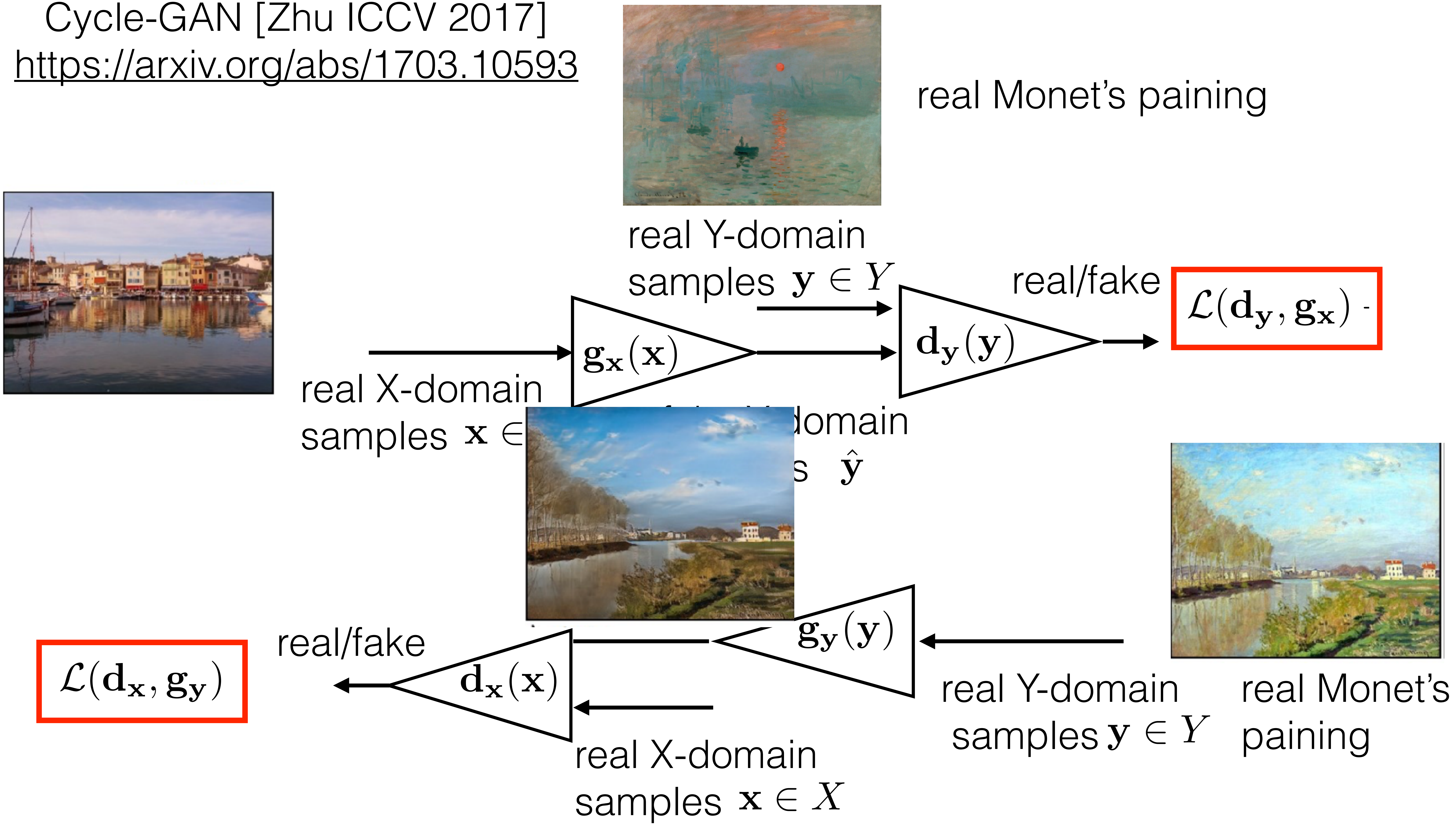
Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593

real Monet's paining



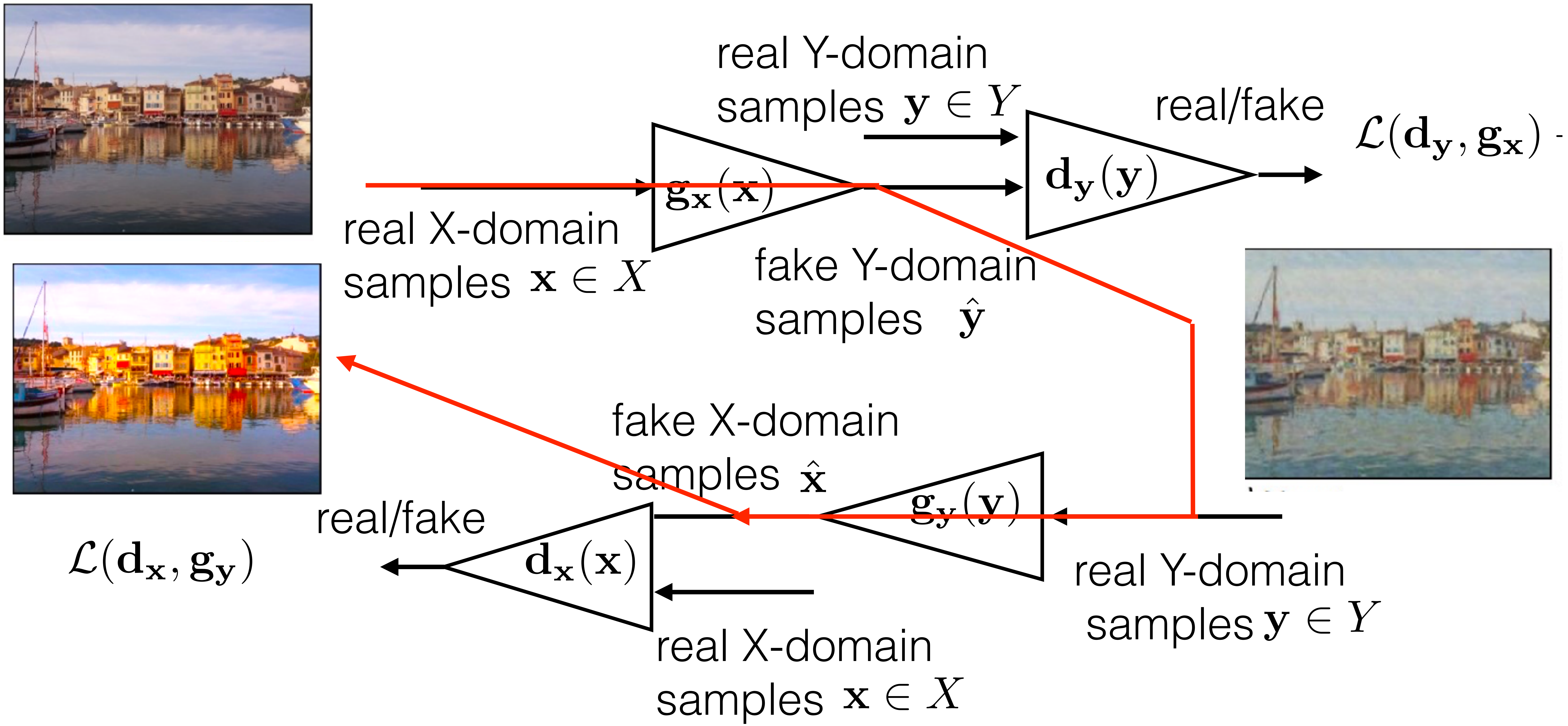
fake Monet's paining

Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593



Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593

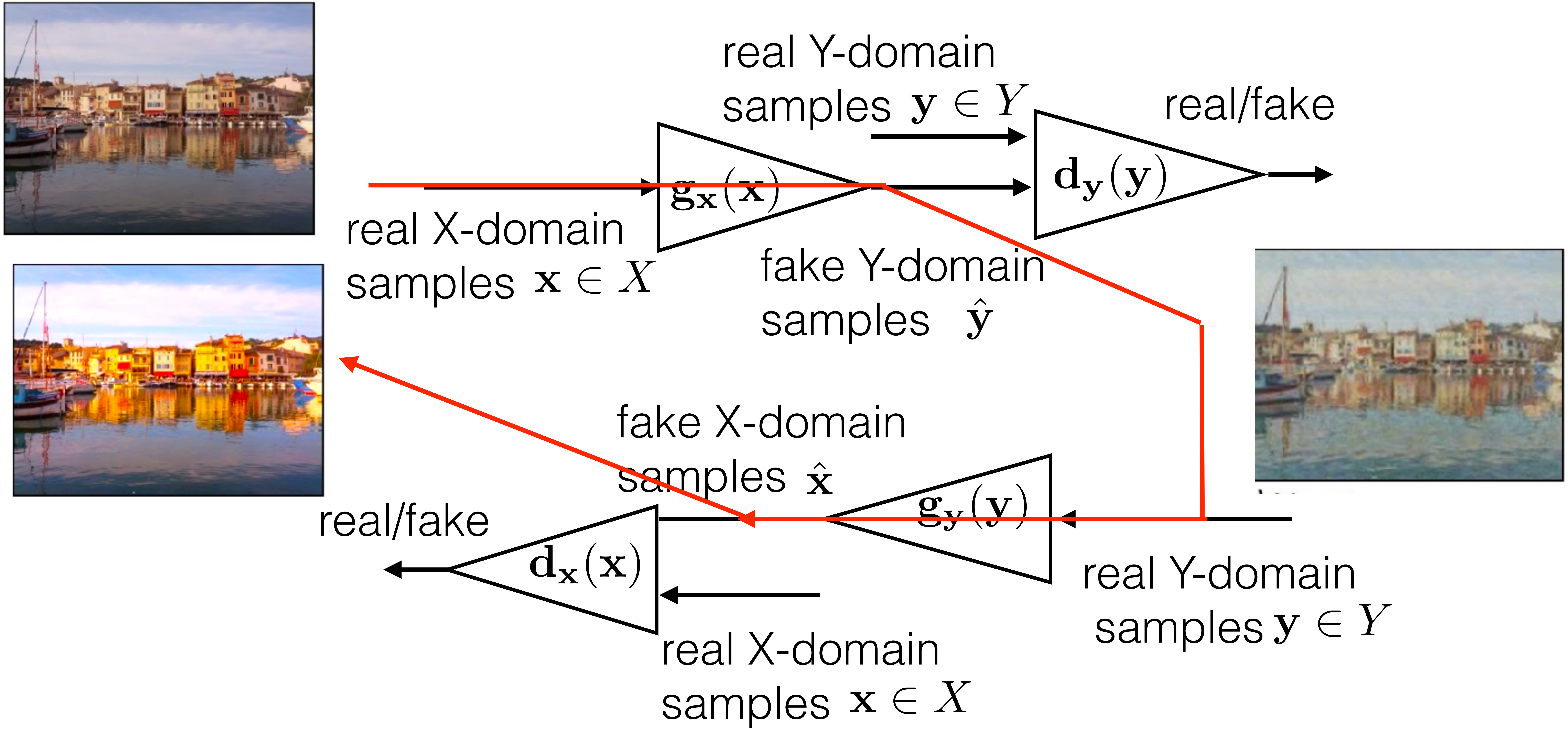
$$|\mathbf{g_y}(\mathbf{g_x}(\mathbf{x}) - \hat{\mathbf{x}}|$$



Cycle-GAN [Zhu ICCV 2017]

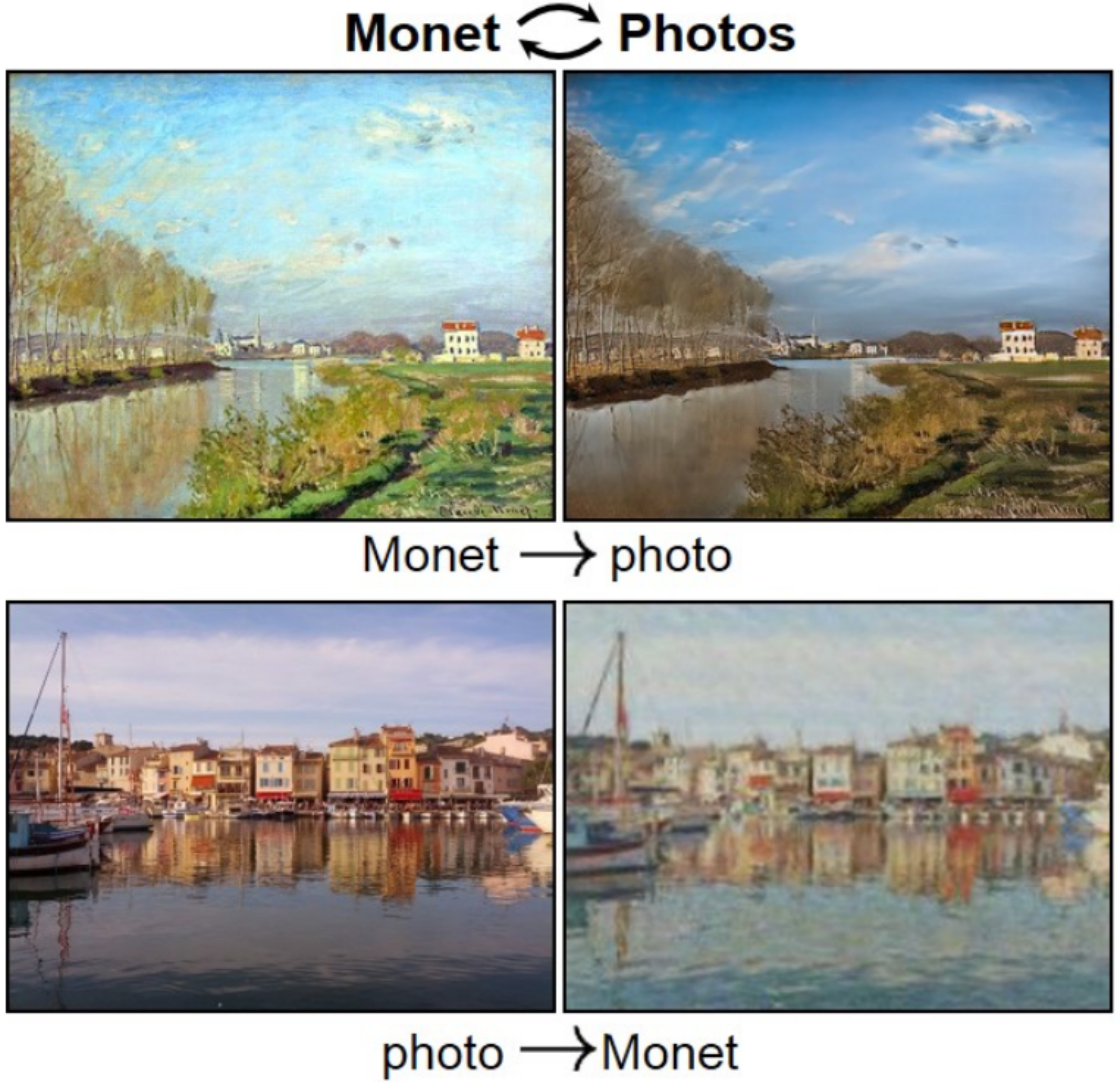
https://arxiv.org/abs/1703.10593

$$\mathcal{L}_{GAN}(\mathbf{d_x}, \mathbf{d_y}, \mathbf{g_x}, \mathbf{g_y}) = \mathcal{L}(\mathbf{d_x}, \mathbf{g_y}) + \mathcal{L}(\mathbf{d_y}, \mathbf{g_x}) + |\mathbf{g_y}(\mathbf{g_x}(\mathbf{x}) - \hat{\mathbf{x}}|$$



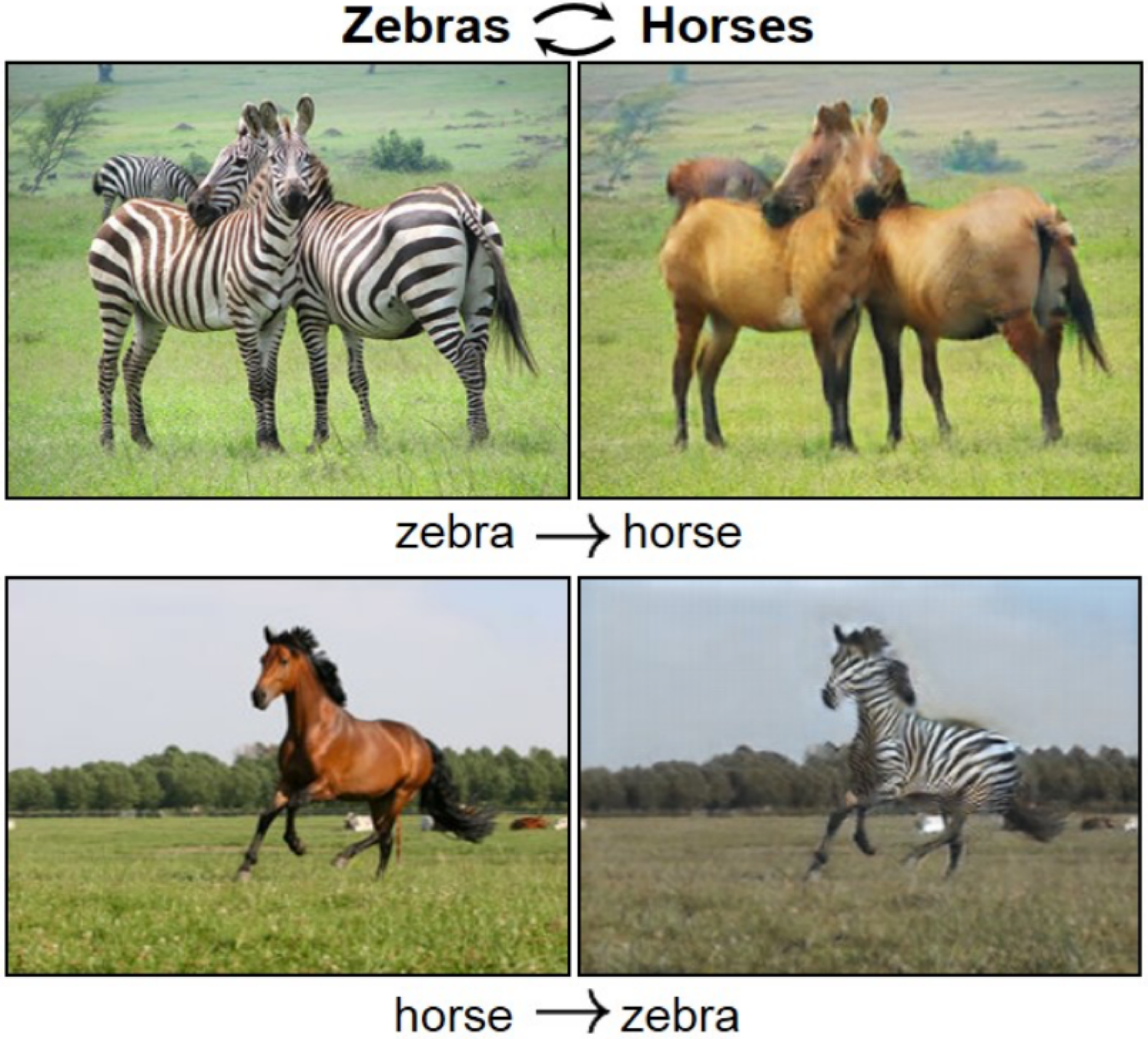
Cycle-GAN [Zhu ICCV 2017]

https://arxiv.org/abs/1703.10593



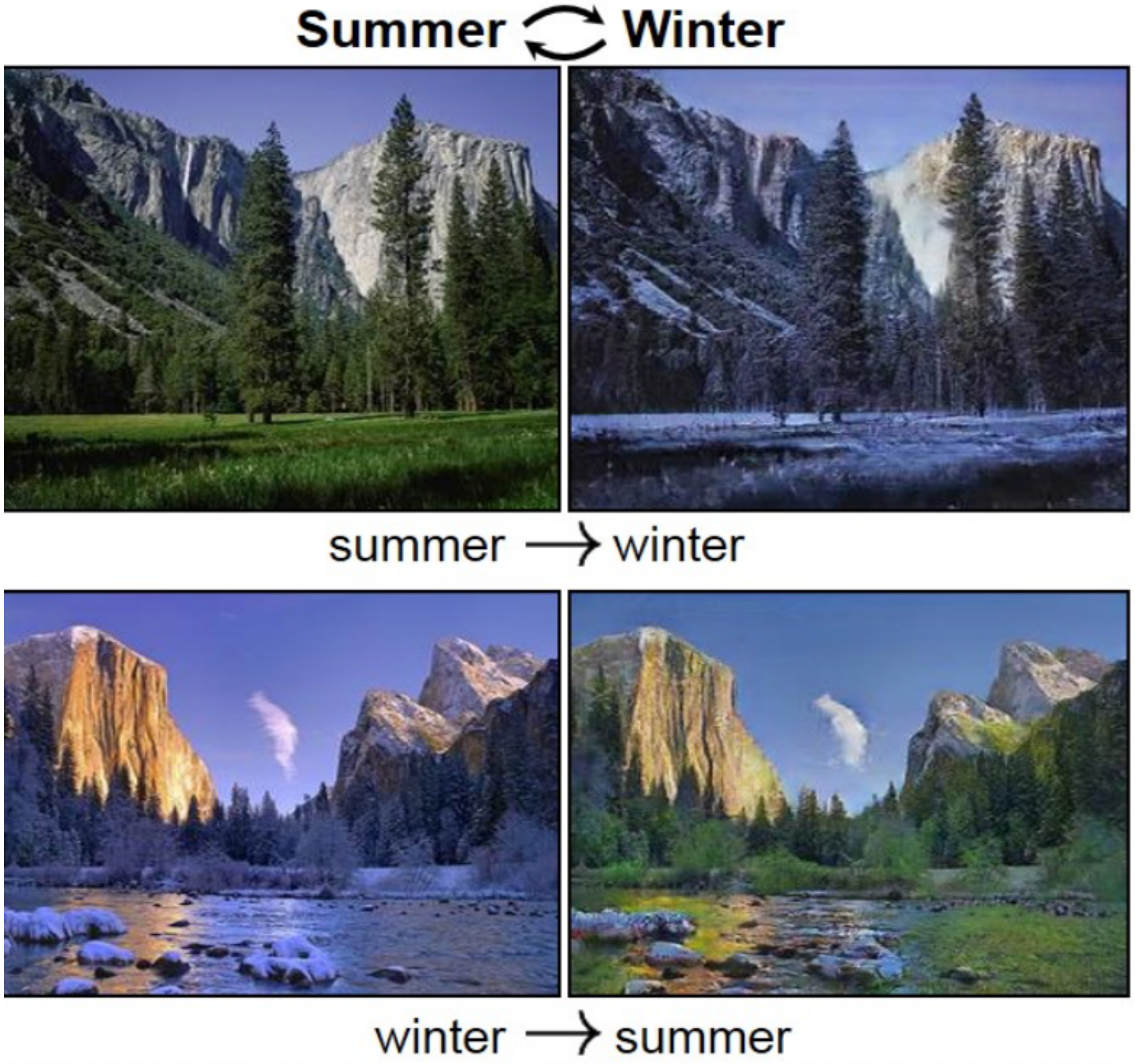
Cycle-GAN [Zhu ICCV 2017]

https://arxiv.org/abs/1703.10593



Cycle-GAN [Zhu ICCV 2017]

https://arxiv.org/abs/1703.10593

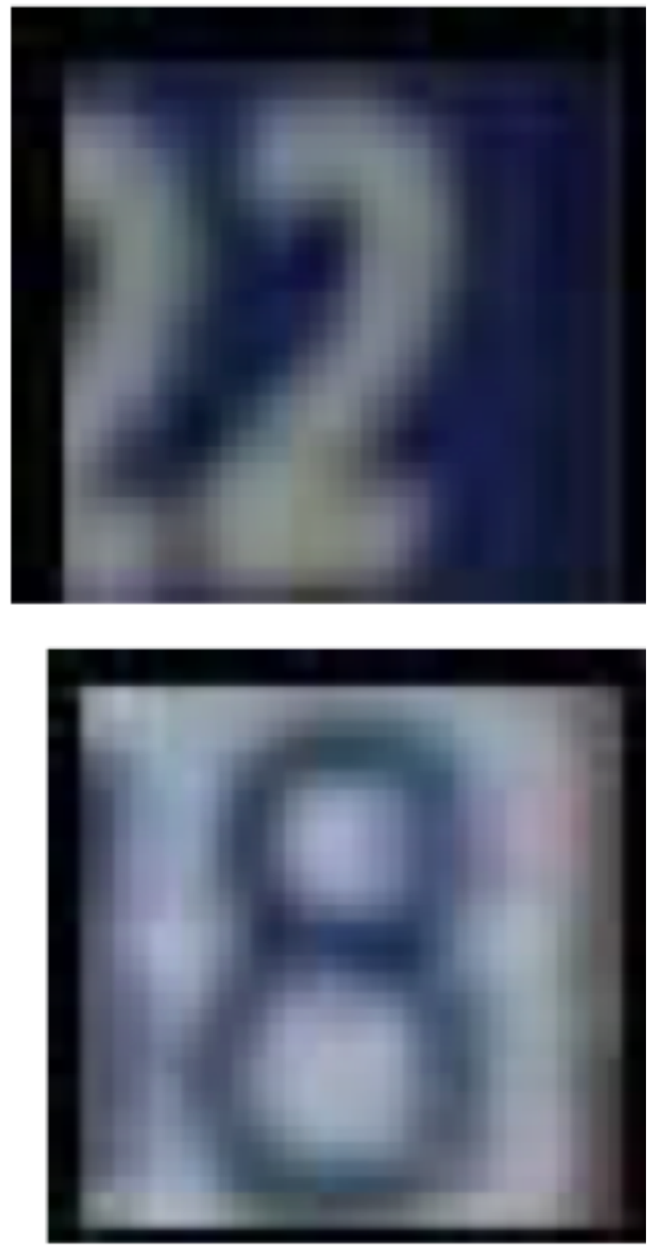


CyCaDa-GAN [Hoffman CVPR 2018]

https://arxiv.org/pdf/1711.03213.pdf

House numbers to MNIST transfer

$\mathbf{y}$



House numbers

$$\mathcal{L}_{GAN}(\mathbf{d_x}, \mathbf{d_y}, \mathbf{g_x}, \mathbf{g_y}) = \mathcal{L}(\mathbf{d_y}, \mathbf{g_x}) + \mathcal{L}(\mathbf{d_x}, \mathbf{g_y}) + \boxed{|\mathbf{g_x}(\mathbf{g_y}(\mathbf{y}) - \hat{\mathbf{y}}|}$$

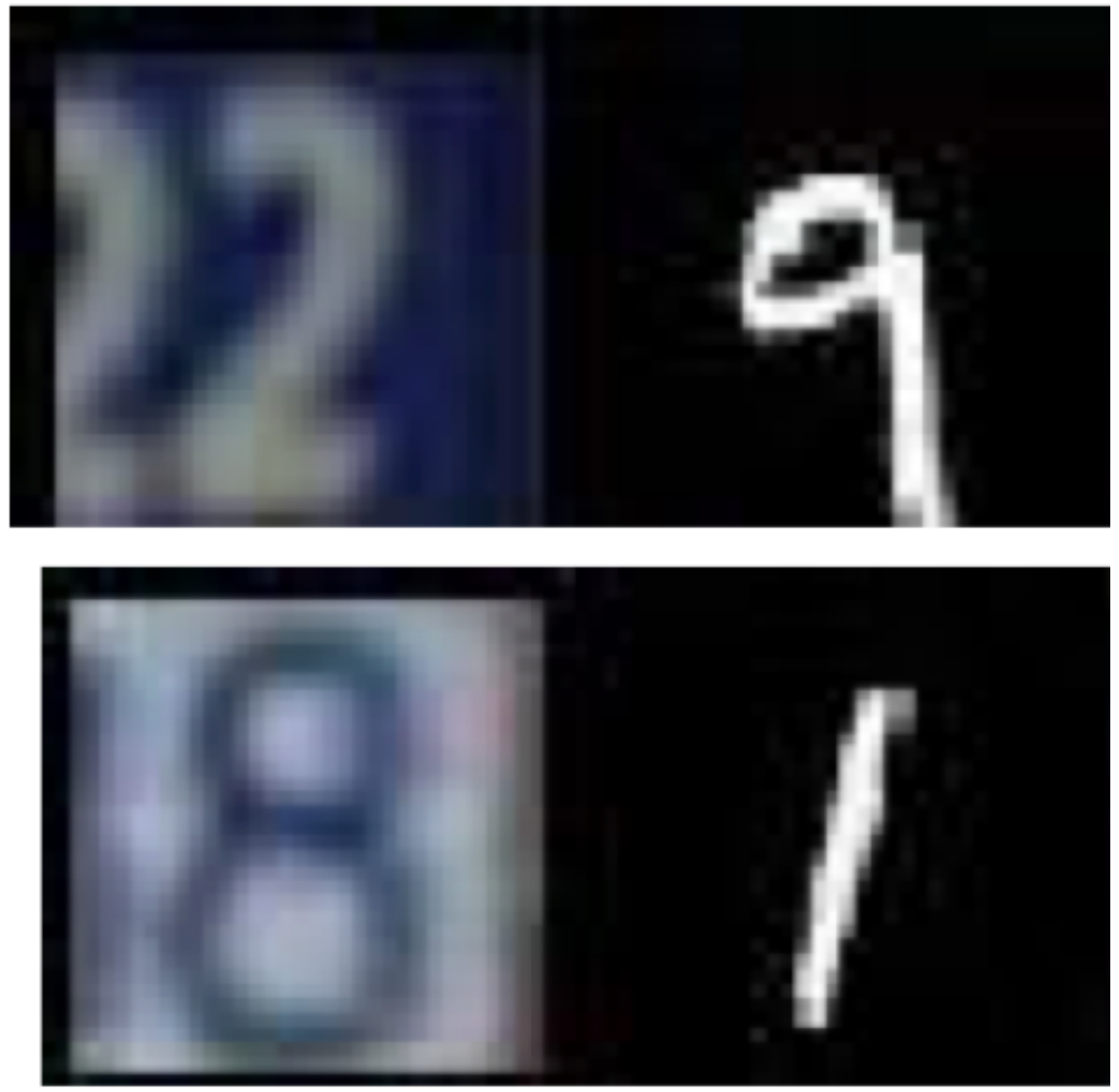
- Cycle consistency helps, but it still allows to learn totally semantically inconsistent transfer

CyCaDa-GAN [Hoffman CVPR 2018]

https://arxiv.org/pdf/1711.03213.pdf

House numbers to MNIST transfer

$\mathbf{y}$ $\mathbf{g_y}(\mathbf{y})$



House numbers MNIST

$$\mathcal{L}_{GAN}(\mathbf{d_x}, \mathbf{d_y}, \mathbf{g_x}, \mathbf{g_y}) = \mathcal{L}(\mathbf{d_y}, \mathbf{g_x}) + \mathcal{L}(\mathbf{d_x}, \mathbf{g_y}) + |\mathbf{g_x}(\mathbf{g_y}(\mathbf{y}) - \hat{\mathbf{y}}|$$

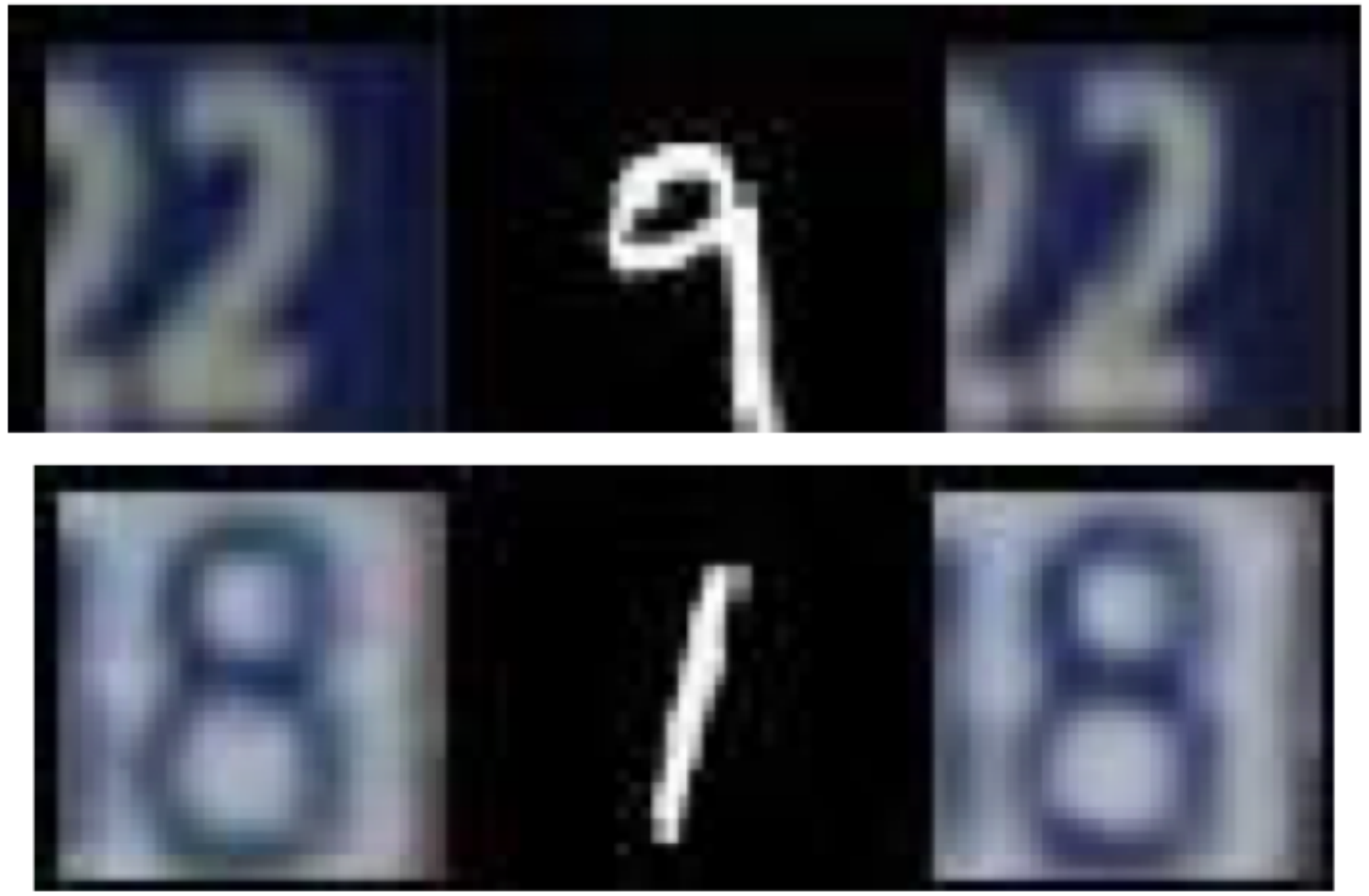
- Cycle consistency helps, but it still allows to learn totally semantically inconsistent transfer

CyCaDa-GAN [Hoffman CVPR 2018]

https://arxiv.org/pdf/1711.03213.pdf

House numbers to MNIST transfer

$\mathbf{y}$ $\mathbf{g_y}(\mathbf{y})$ $\mathbf{g_x}(\mathbf{g_y}(\mathbf{y}))$



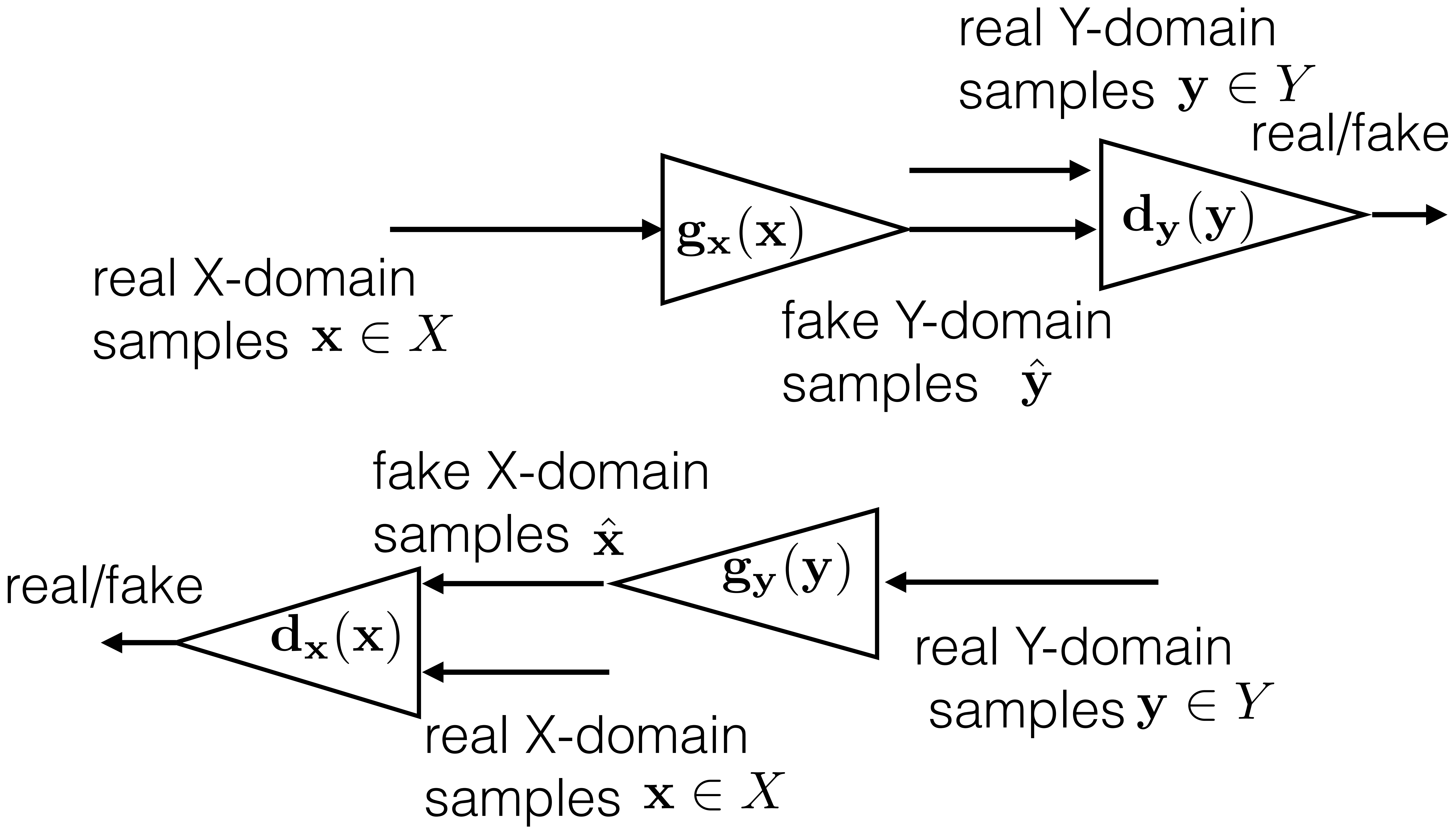
House numbers MNIST House numbers

$$\mathcal{L}_{GAN}(\mathbf{d_x}, \mathbf{d_y}, \mathbf{g_x}, \mathbf{g_y}) = \mathcal{L}(\mathbf{d_y}, \mathbf{g_x}) + \mathcal{L}(\mathbf{d_x}, \mathbf{g_y}) + |\mathbf{g_x}(\mathbf{g_y}(\mathbf{y}) - \hat{\mathbf{y}}|$$

- Cycle consistency helps, but it still allows to learn totally semantically inconsistent transfer

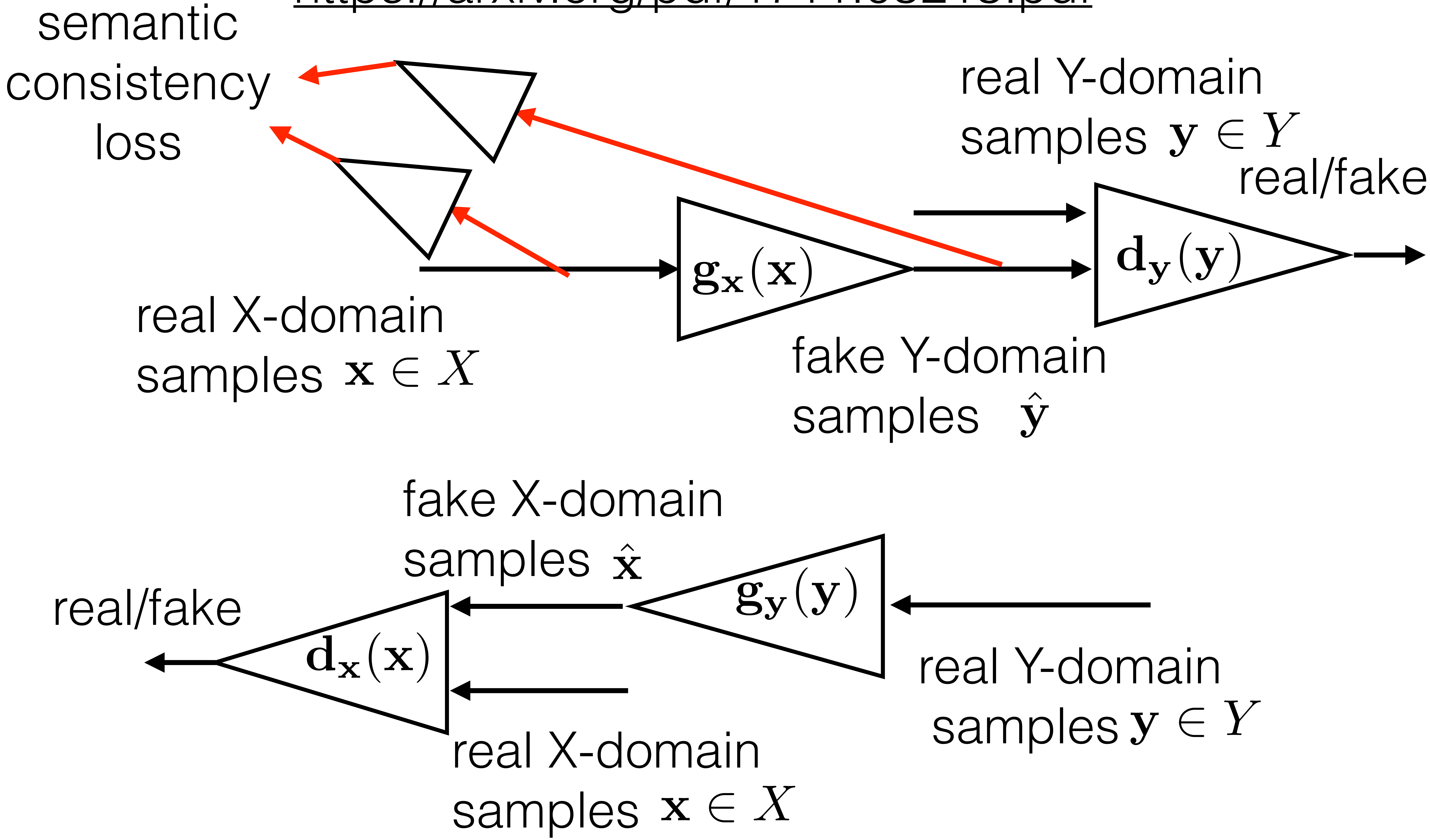
CyCaDa-GAN [Hoffman CVPR 2018]

https://arxiv.org/pdf/1711.03213.pdf



CyCaDa-GAN [Hoffman CVPR 2018]

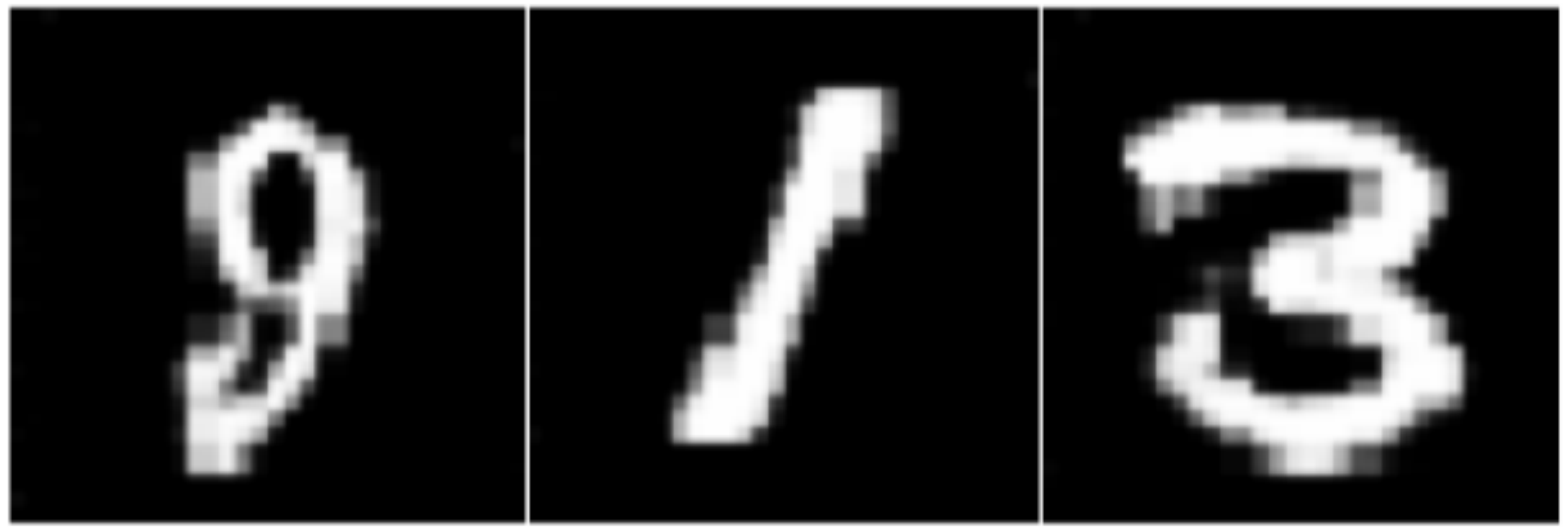
https://arxiv.org/pdf/1711.03213.pdf



CyCaDa-GAN [Hoffman CVPR 2018]
https://arxiv.org/pdf/1711.03213.pdf

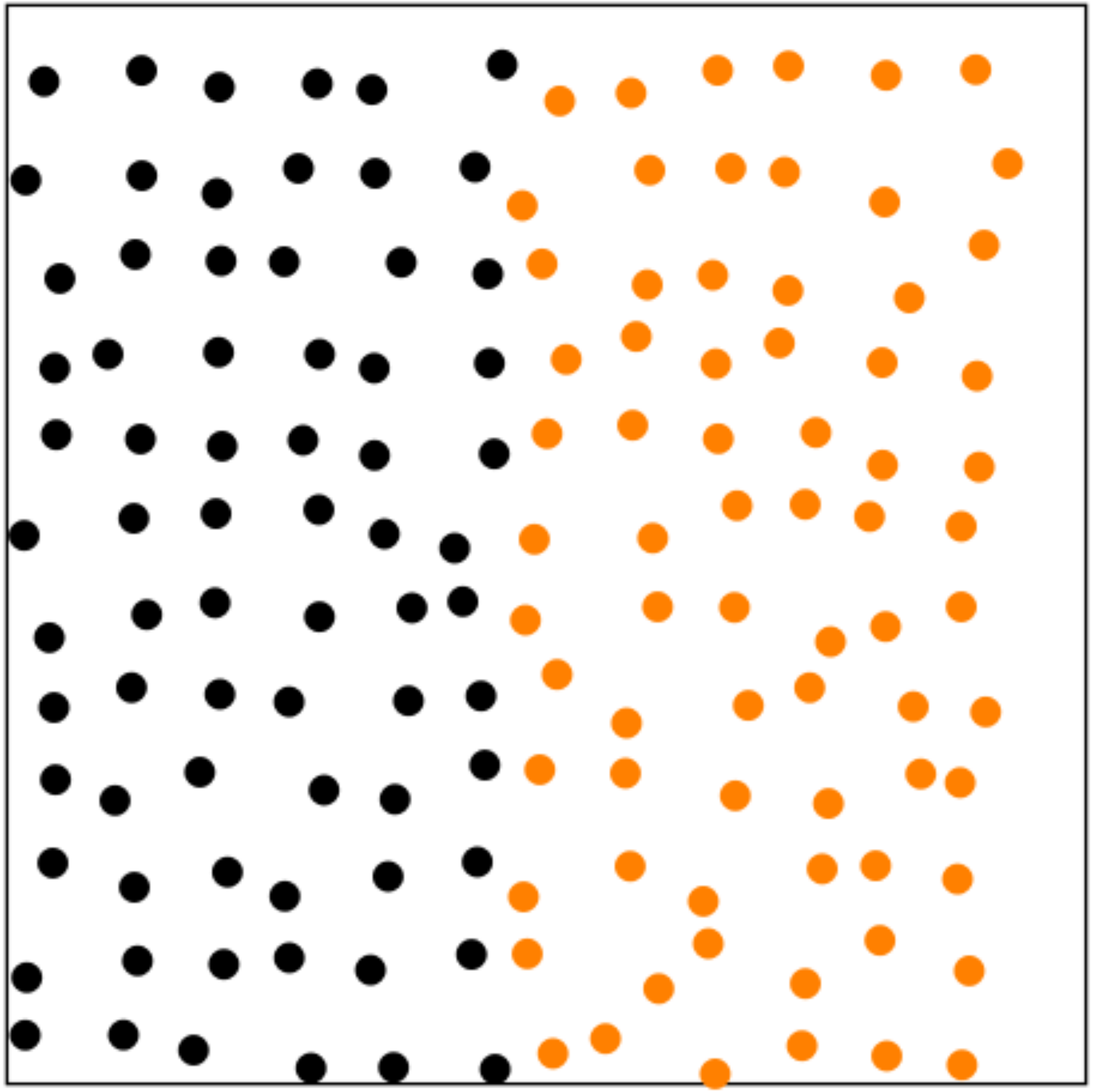
- Semantic consistency enforce transformation to be semantically consistent

house numbers



MNIST

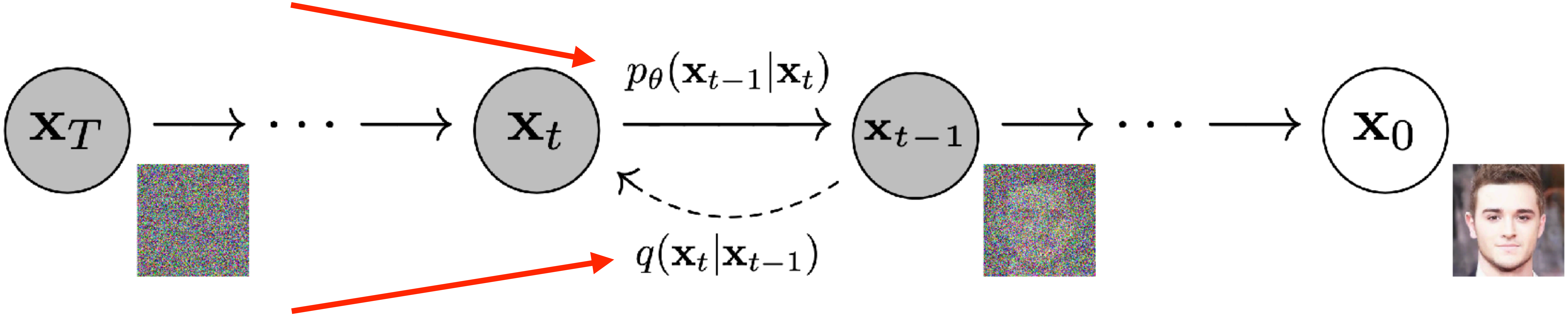
Diffusion models



sciencing.com

Diffusion models
[Hu NeurIPS 2020]

Reverse of the diffusion process to generate original data from the noise



Markov chain of diffusion steps in which we slowly and randomly add noise t

If noise is small the backward step has also “almost” gaussian distribution

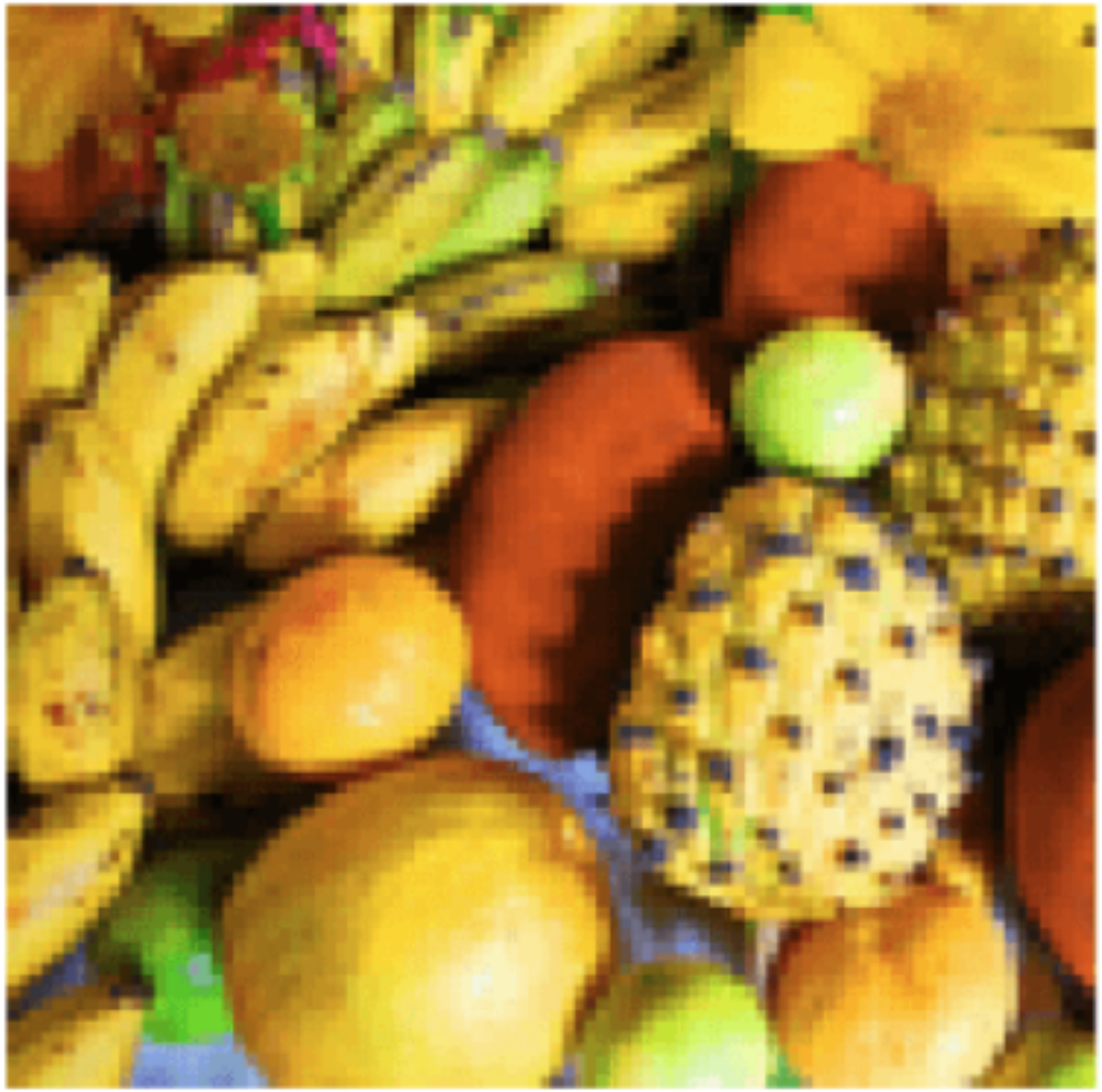
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

=> learn de-noising networks through L2-norm

Diffusion networks

Super resolution

input



output

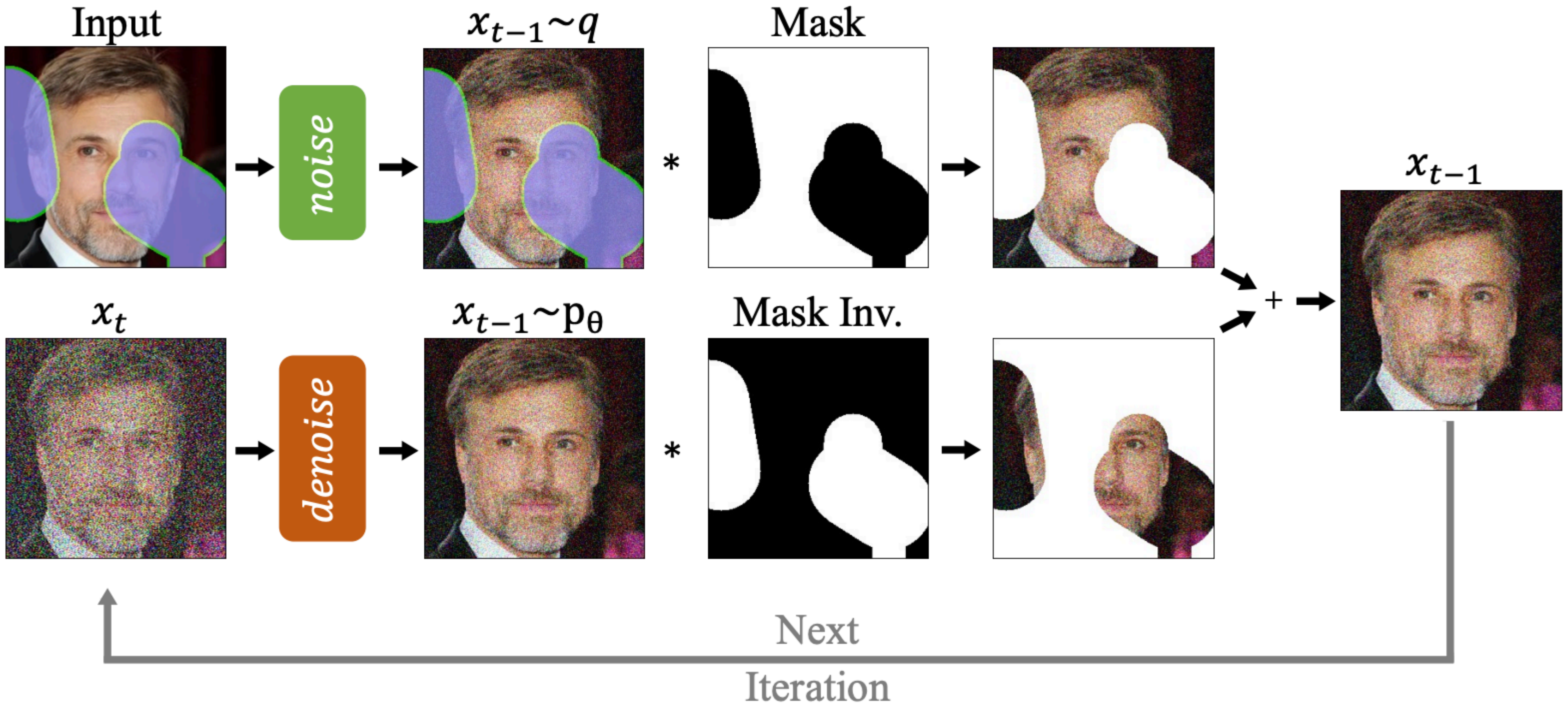
v7labs.com

Inpainting

[Repaint, CVPR 2022]

https://openaccess.thecvf.com/content/CVPR2022/papers/Lugmayr_RePaint_Inpainting_Using_Denoising_Diffusion_Probabilistic_Models_CVPR_2022_paper.pdf

Inpainting



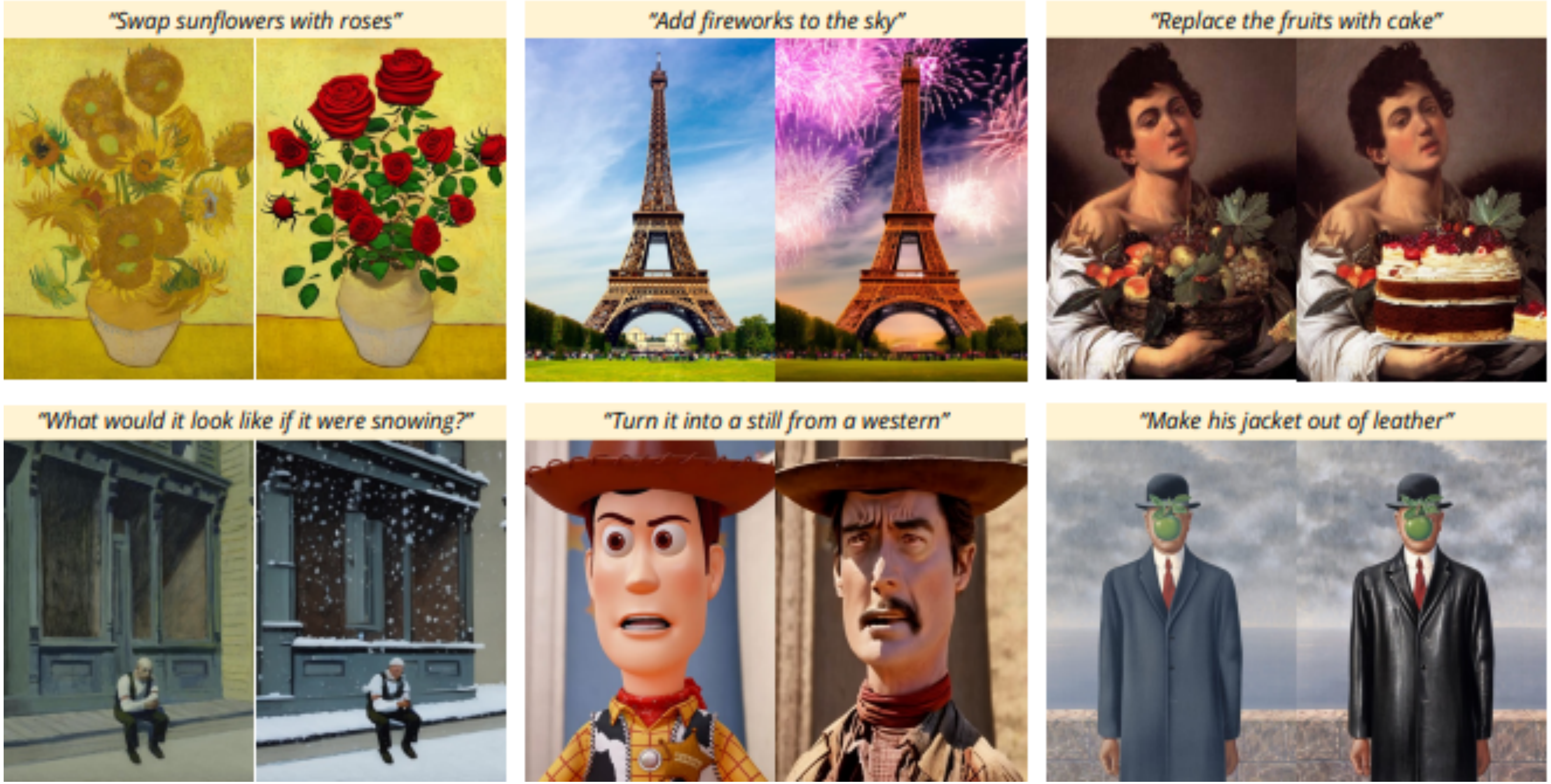
[Repaint, CVPR 2022]

https://openaccess.thecvf.com/content/CVPR2022/papers/Lugmayr_RePaint_Inpainting_Using_Denoising_Diffusion_Probabilistic_Models_CVPR_2022_paper.pdf

Domain transfer/Stylization

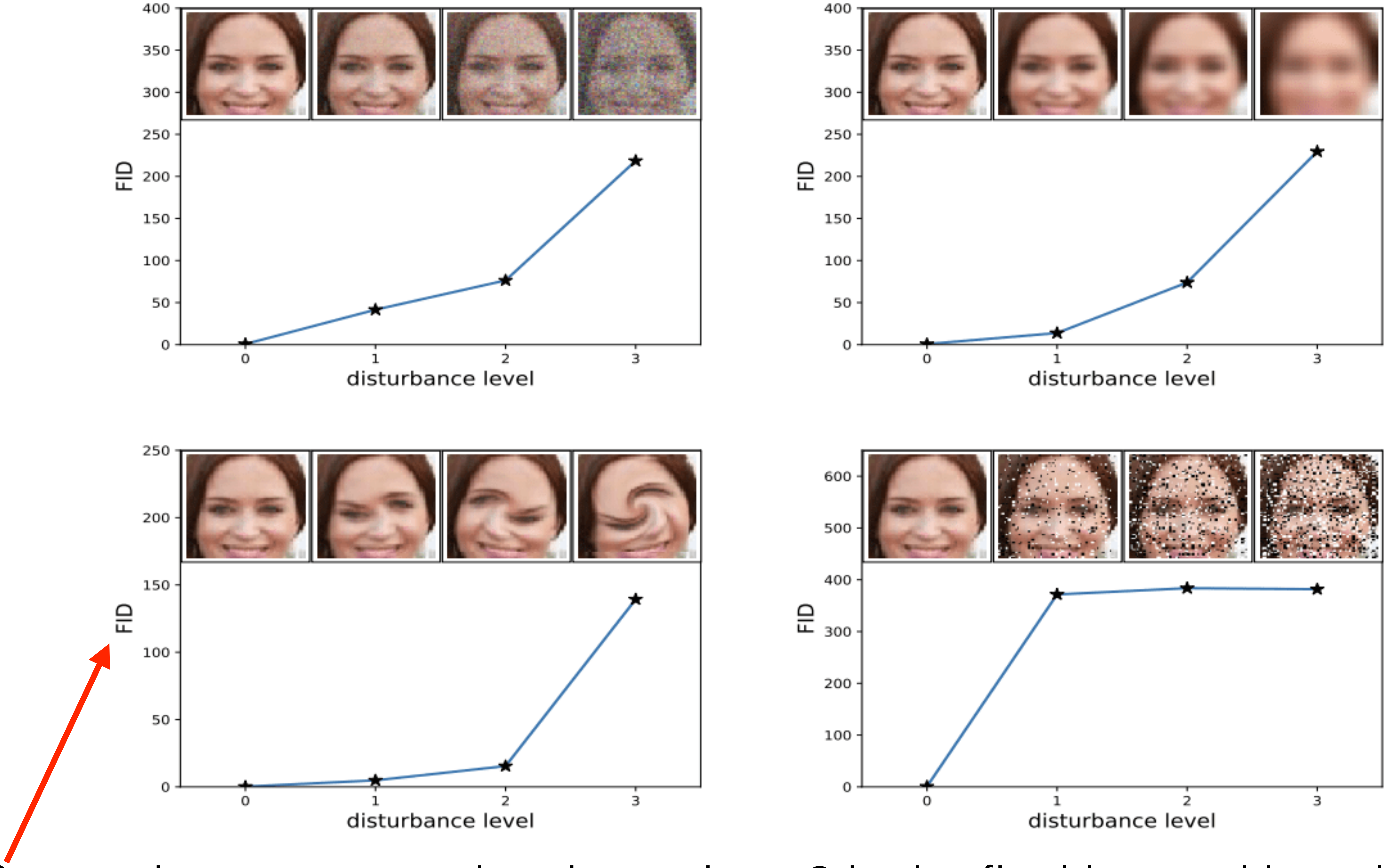
v7labs.com

Description driven image manipulations



Brooks, Tim, Aleksander Holynski, and Alexei A. Efros. "Instructpix2pix: Learning to follow image editing instructions." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

Measuring quality of generated images???



Comparing mean + std on inception v3 in the final layer with real images

Data bias

input

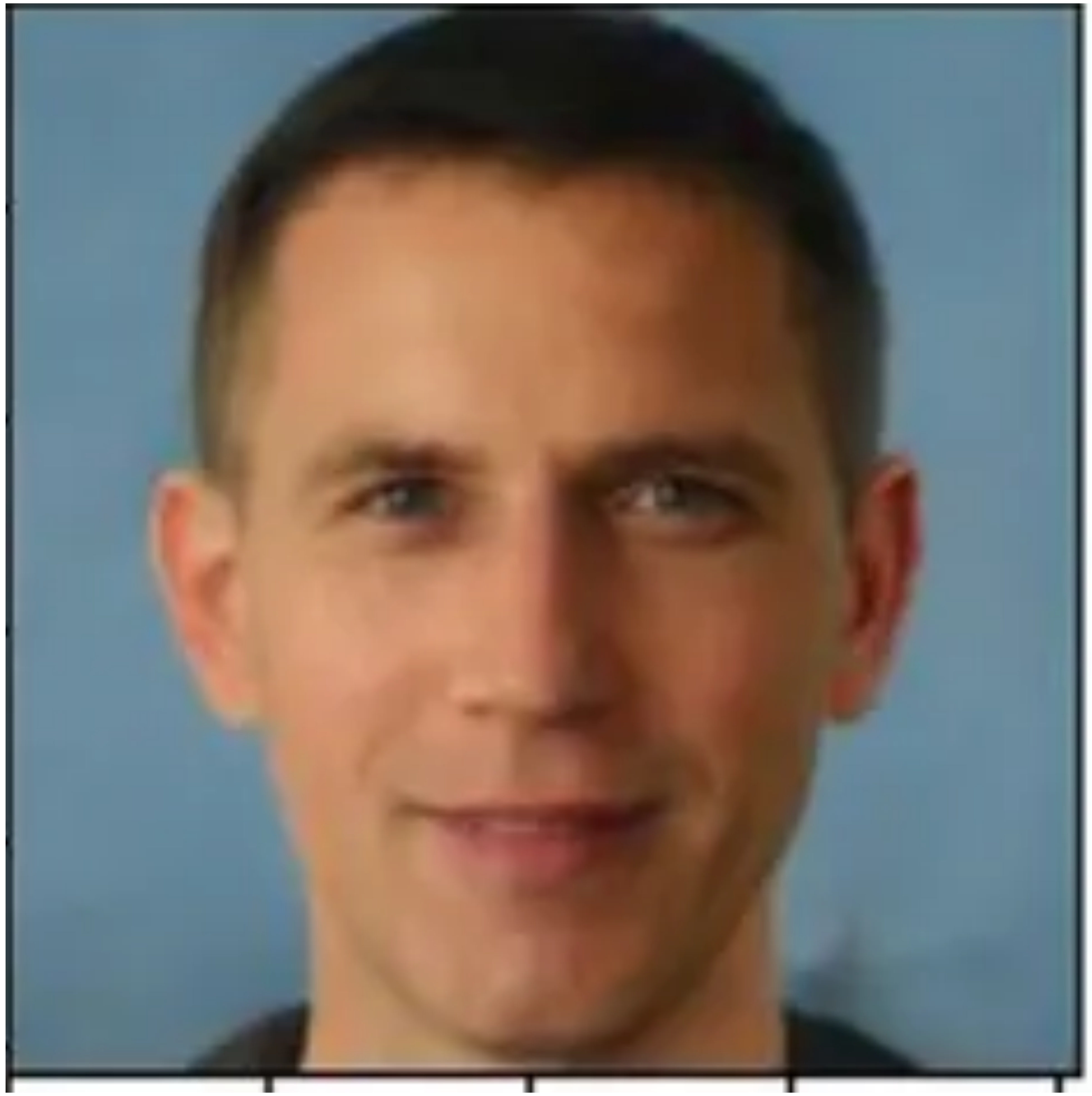
output

For the last in this course: How is it?

Data bias

input

output



For the last in this course: How is it?