

Generative models

Outline

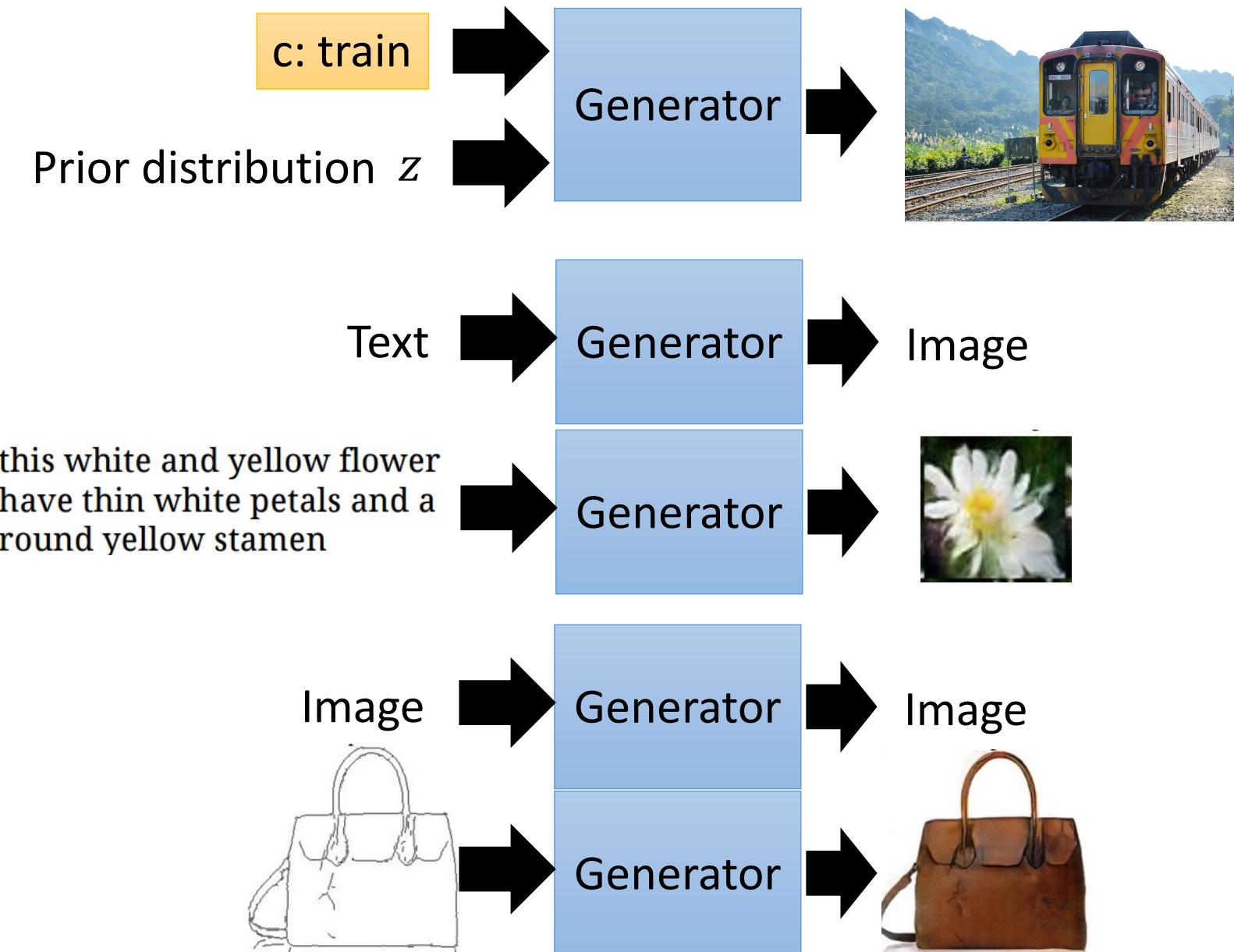
1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
3. GAN architectures
4. Editing
- 5. Conditional GANs**

Generative models

Outline

1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
3. GAN architectures
4. Editing
5. **Conditional GANs**
 1. Principle

Motivation



Conditional GAN

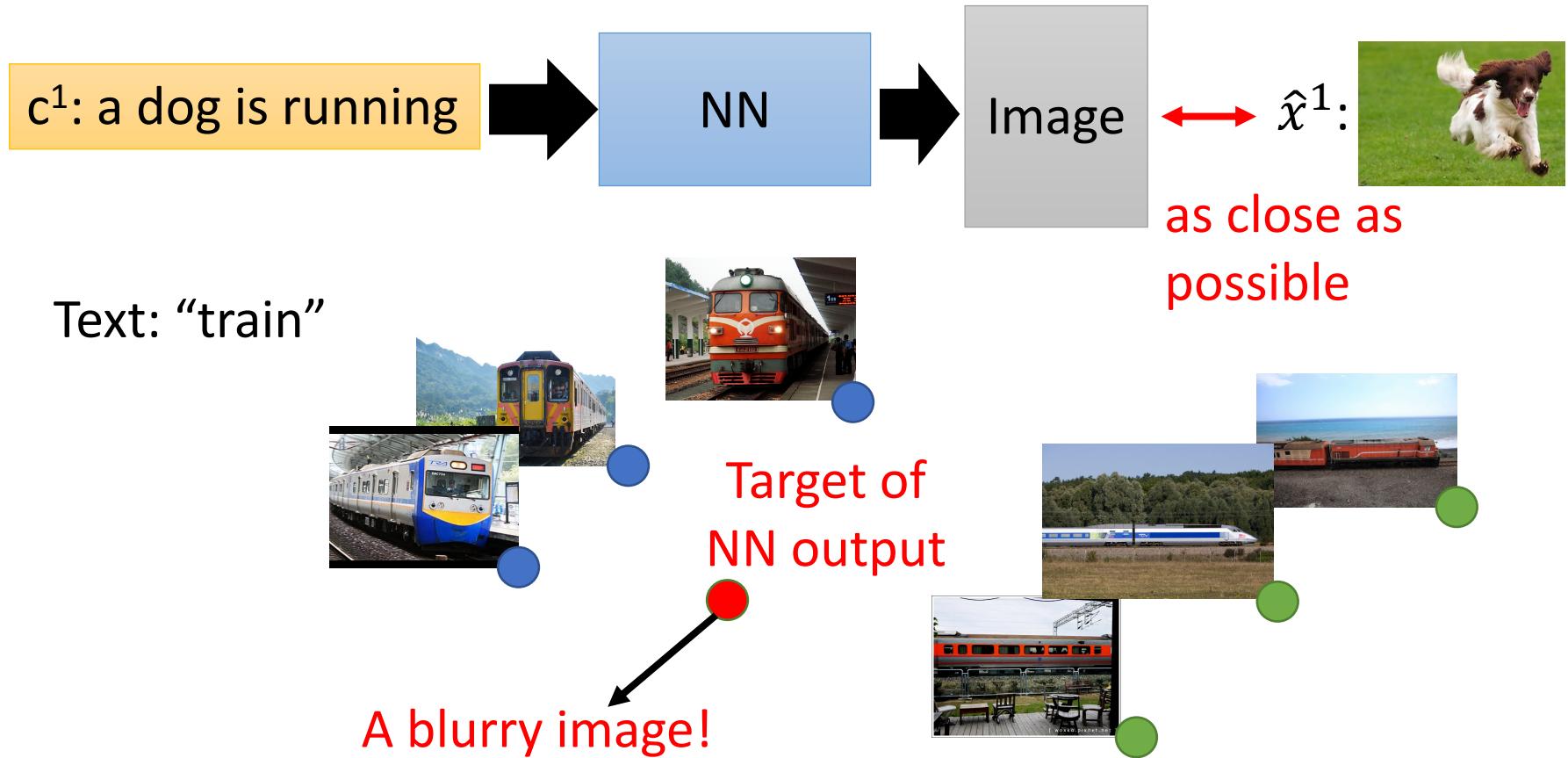
c^1 : a dog is running \hat{x}^1 :



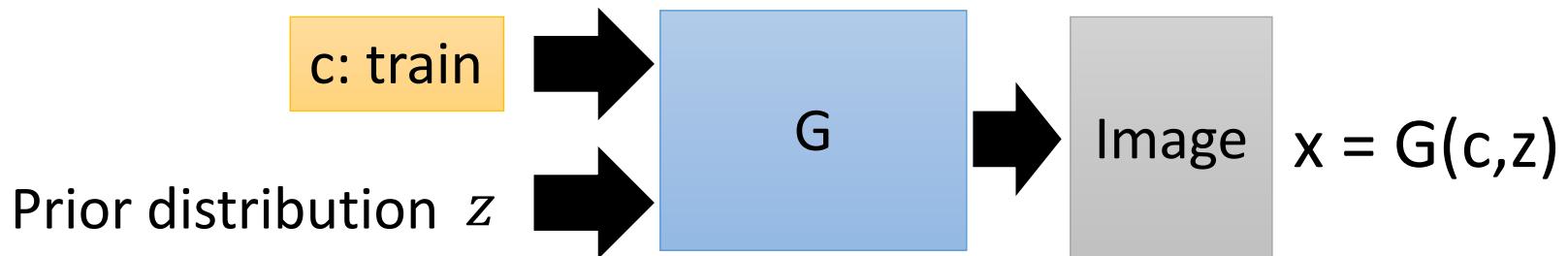
c^2 : a bird is flying \hat{x}^2 :



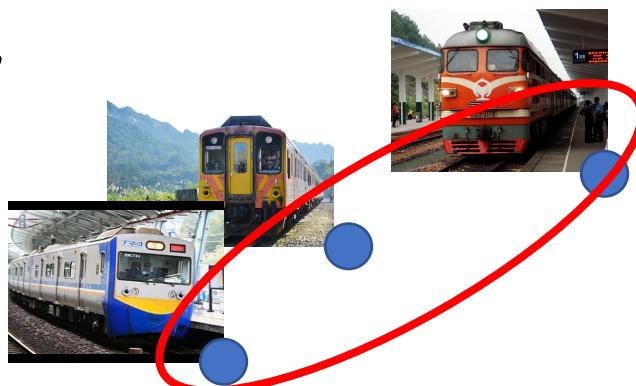
- **Text to image** by traditional supervised learning



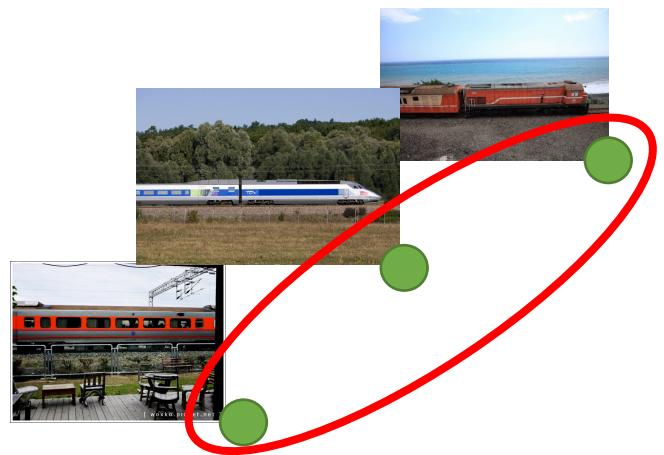
Conditional GAN



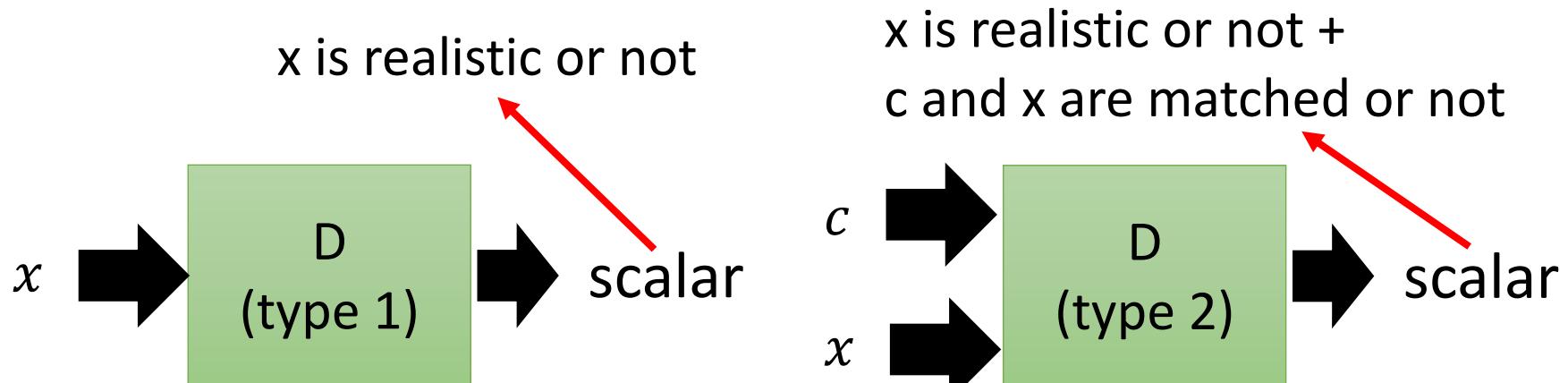
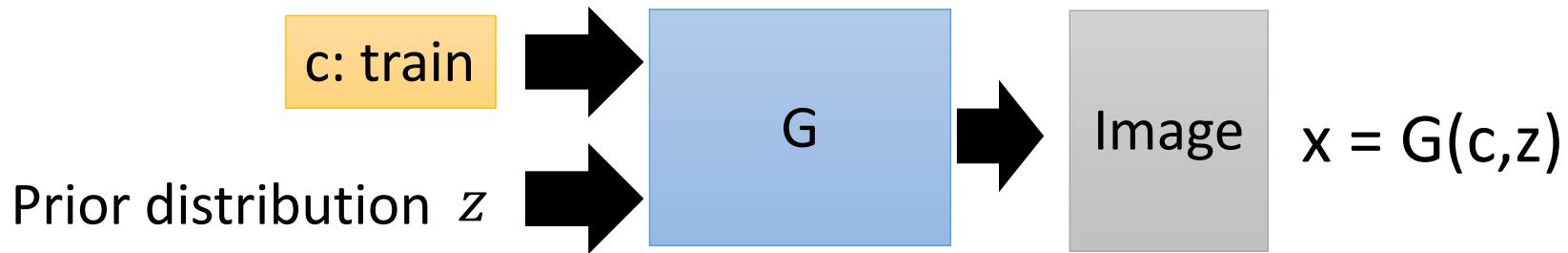
Text: “train”



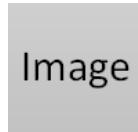
It is a distribution
Approximate the
distribution of real data

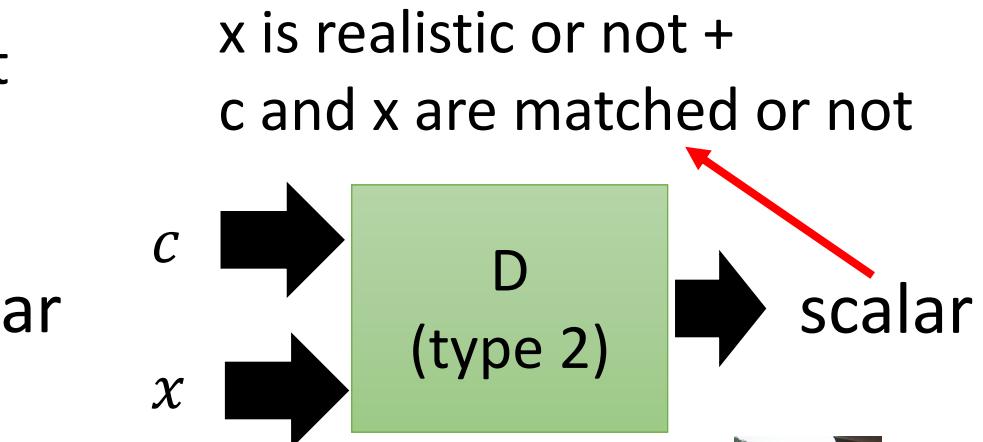


Conditional GAN



Positive example: 

Negative example: 

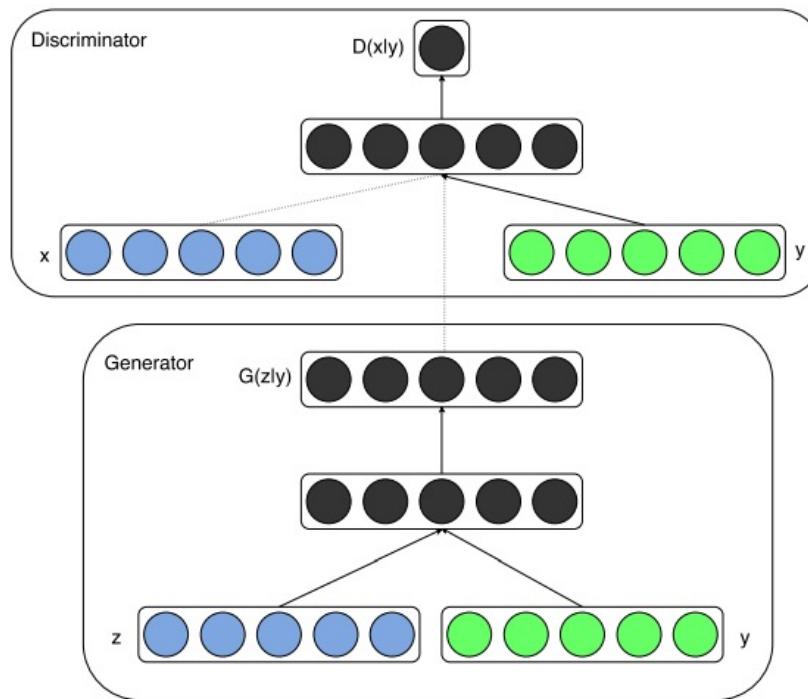


Positive example: (train , )

Negative example: (train , Extra neg )
(cat , )

Conditional GAN (CGAN model)

$$\min_G \max_D \left(\mathbb{E}_{\mathbf{x}, \mathbf{y} \sim p_{\text{data}}(\mathbf{x}, \mathbf{y})} [\log D(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{y} \sim p_y, \mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z}, \mathbf{y}), \mathbf{y}))] \right)$$



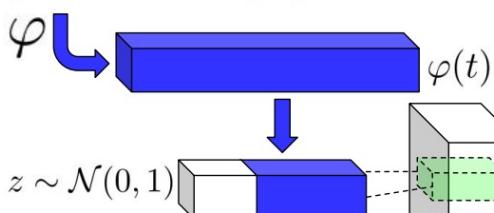
Generative models

Outline

1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
3. GAN architectures
4. Editing
5. Conditional GANs
 1. Principle
 2. **Text2Image**

Text2Image: architecture example

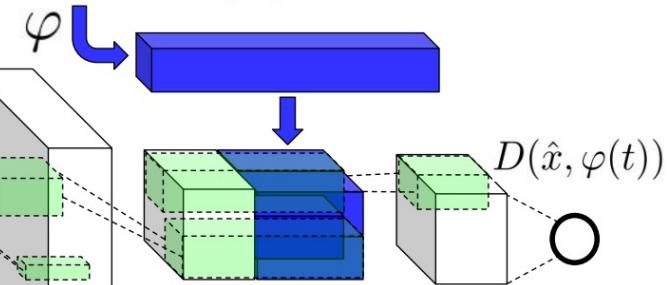
This flower has small, round violet petals with a dark purple center



$$\hat{x} := G(z, \varphi(t))$$

Generator Network

This flower has small, round violet petals with a dark purple center



Discriminator Network

- Positive samples:
 - real image + right texts
- Negative samples:
 - fake image + right texts
 - Real image + wrong texts

Text2Image results

this small bird has a pink breast and crown, and black primaries and secondaries.



this magnificent fellow is almost all black with a red crest, and white cheek patch.



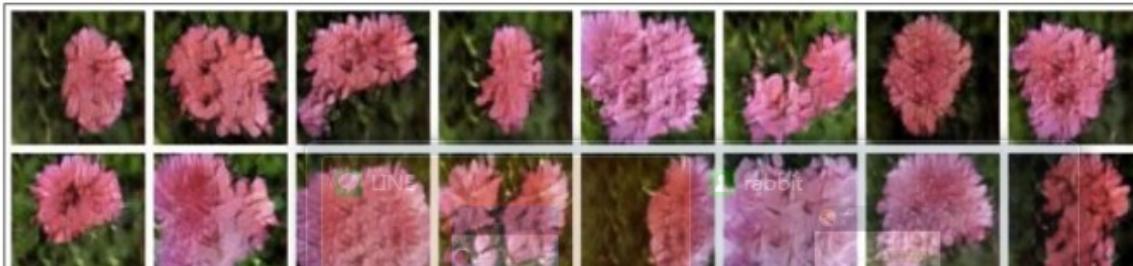
the flower has petals that are bright pinkish purple with white stigma



this white and yellow flower have thin white petals and a round yellow stamen



Text2Image results

Caption	Image
this flower has white petals and a yellow stamen	
the center is yellow surrounded by wavy dark purple petals	
this flower has lots of small round pink petals	

Text2Image: architecture example (2)

StackGAN: similar idea with LapGan to generate higher resolution images

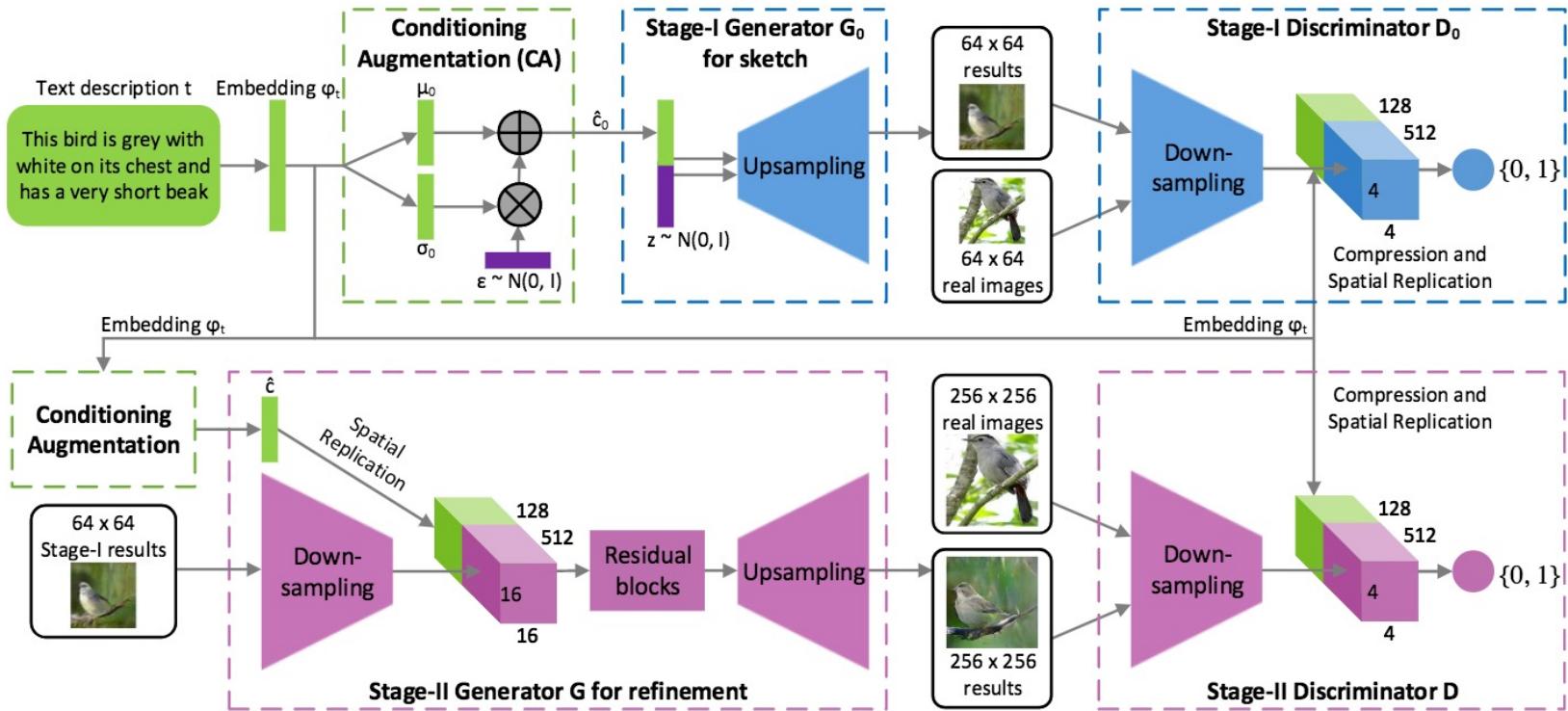
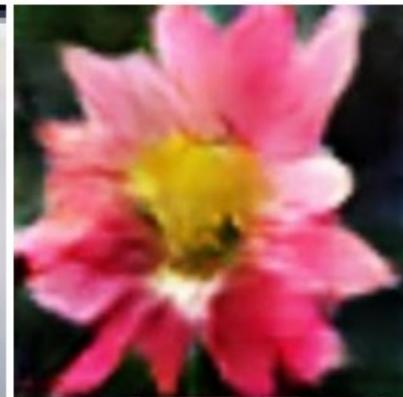
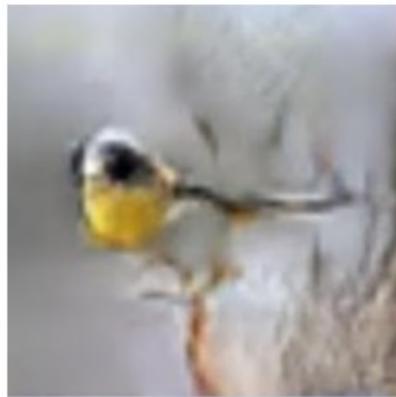


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects and adds compelling details into Stage-I results, yielding a more realistic high-resolution image.

StackGAN results

(a) Stage-I
images



(b) Stage-II
images



[[Zhang et al. 2016](#)]

This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face

This bird is white with some black on its head and wings, and has a long orange beak

This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments

Text2Image results

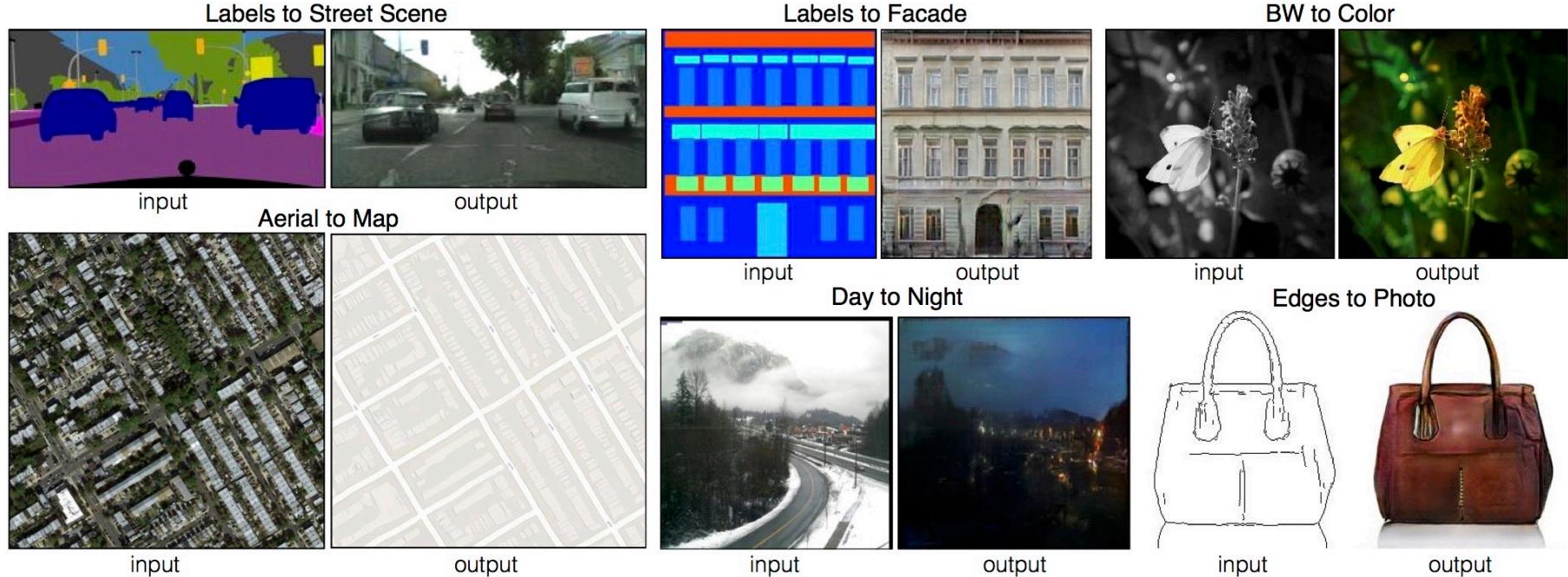
Caption	Image
a pitcher is about to throw the ball to the batter	
a group of people on skis stand in the snow	
a man in a wet suit riding a surfboard on a wave	

Generative models

Outline

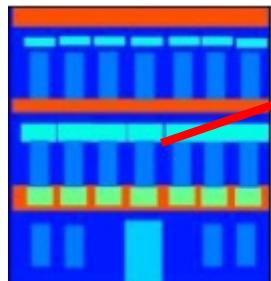
1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
3. GAN architectures
4. Editing
5. Conditional GANs
 1. Principle
 2. Text2Image
 3. **Image2Image**

Image-to-Image Translation pix2pix



- Conditioned on an image of different modality
- No need to specify the loss function

Image-to-image pix2pix



c
 z

G

$$x = G(z | c)$$



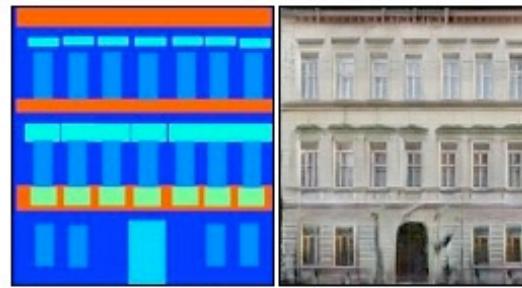
Labels to Street Scene



input

output

Labels to Facade



input

output

BW to Color



input

output

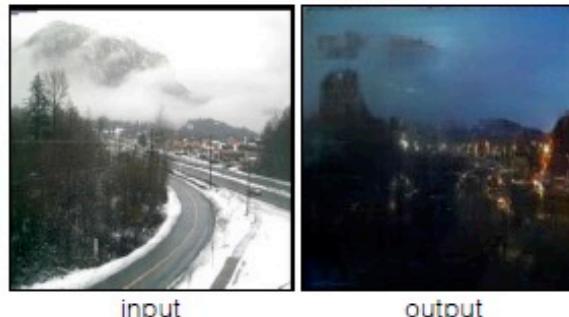
Aerial to Map



input

output

Day to Night



input

output

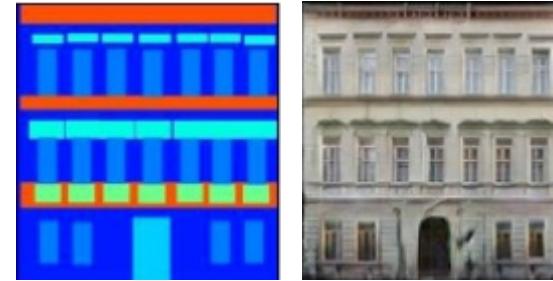
Edges to Photo



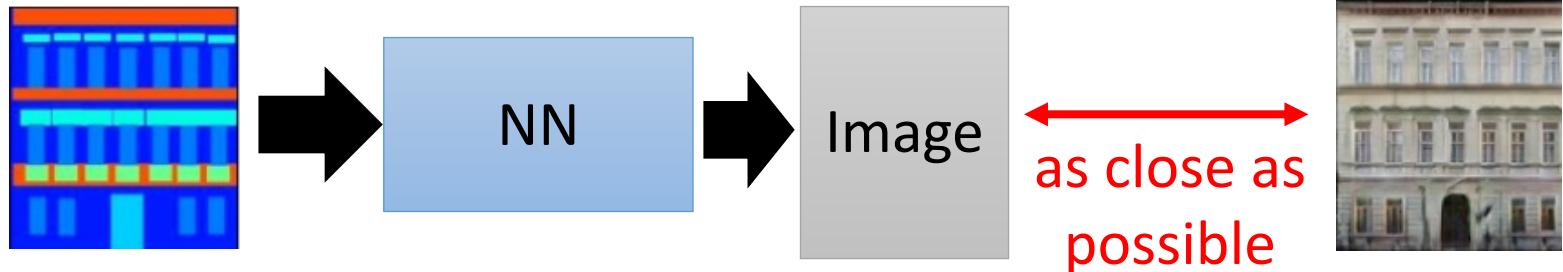
input

output

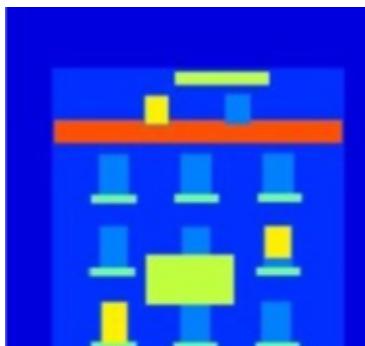
Image-to-image pix2pix



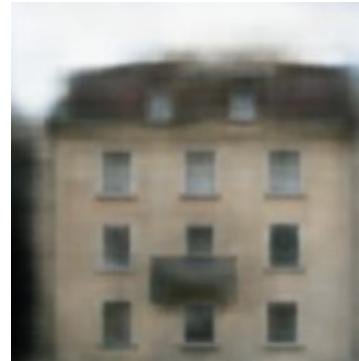
- Traditional supervised approach



Testing:



input



close

It is blurry
because it is
the average of
several images.

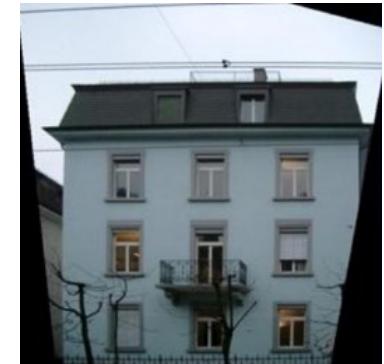
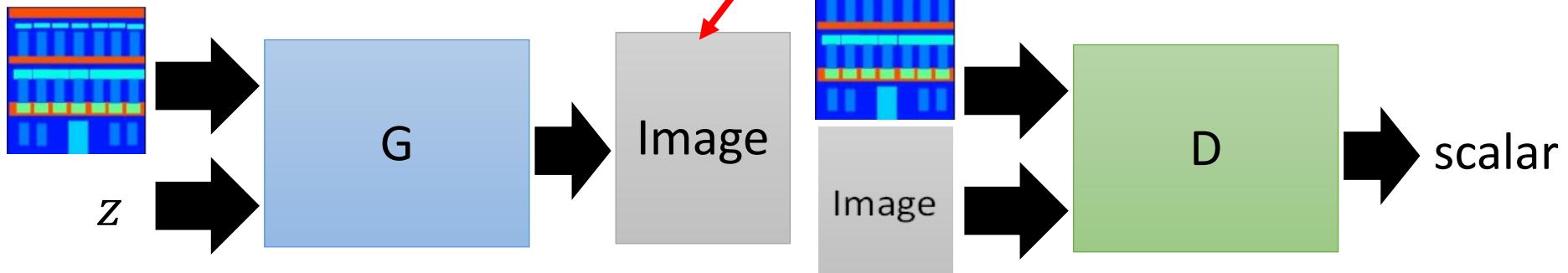
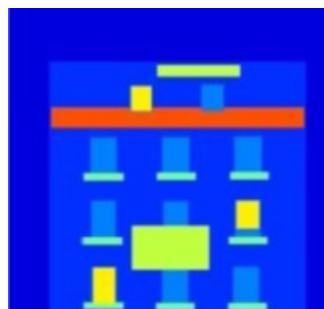


Image-to-image

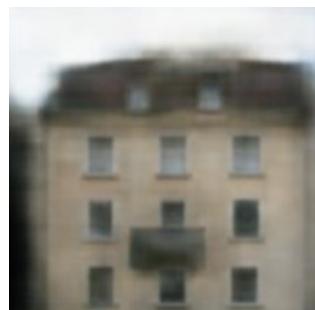
- Conditional GAN



Testing:



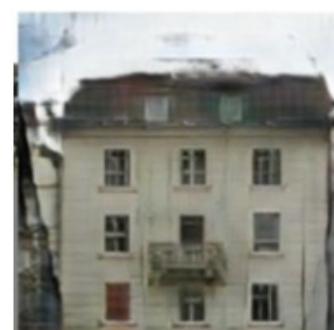
input



close



GAN



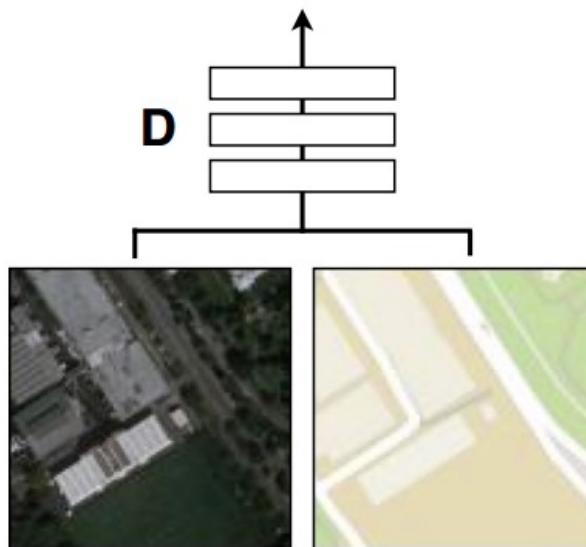
GAN + close



GT

Positive examples

Real or fake pair?

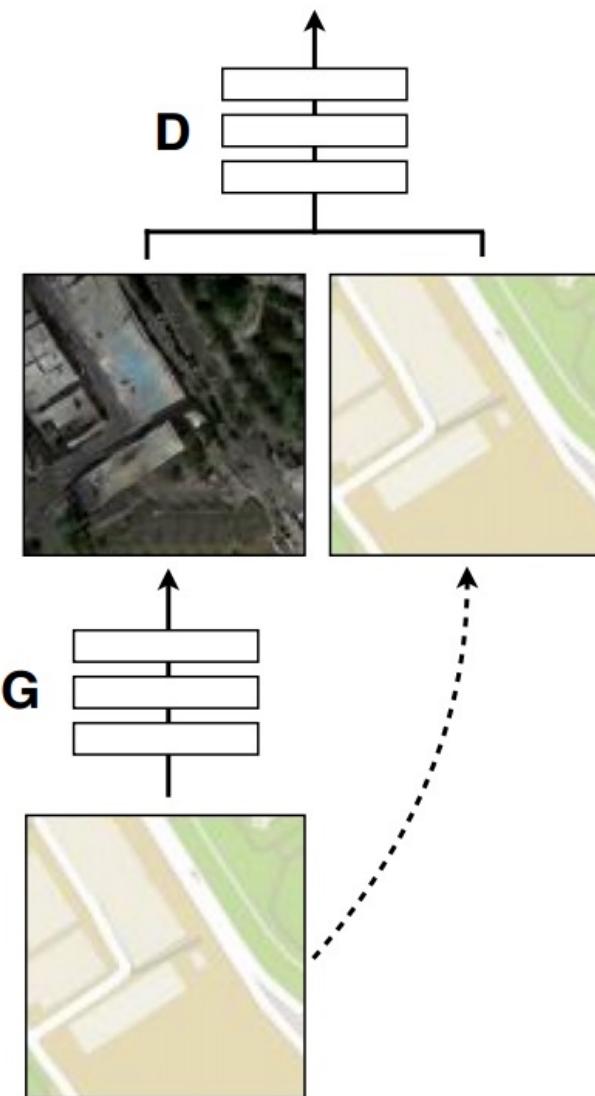


G tries to synthesize fake images that fool **D**

D tries to identify the fakes

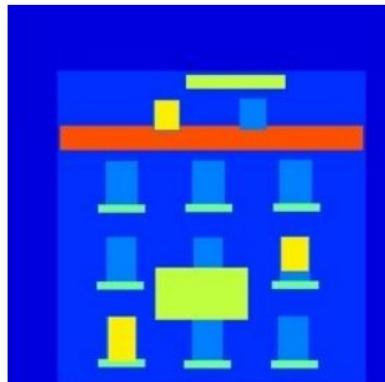
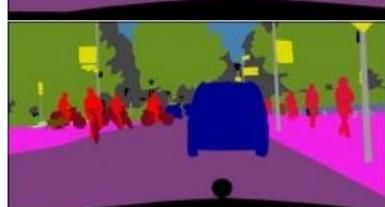
Negative examples

Real or fake pair?



Label2Image

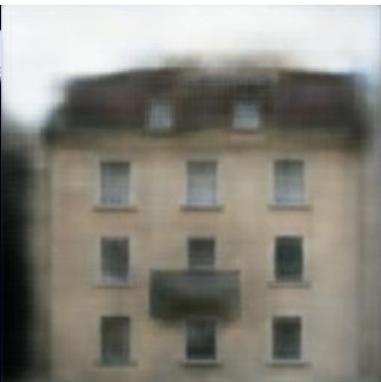
Input



Ground truth



L1



cGAN



L1 + cGAN



Edges2Image



Pix2pixHD [CVPR 2018]

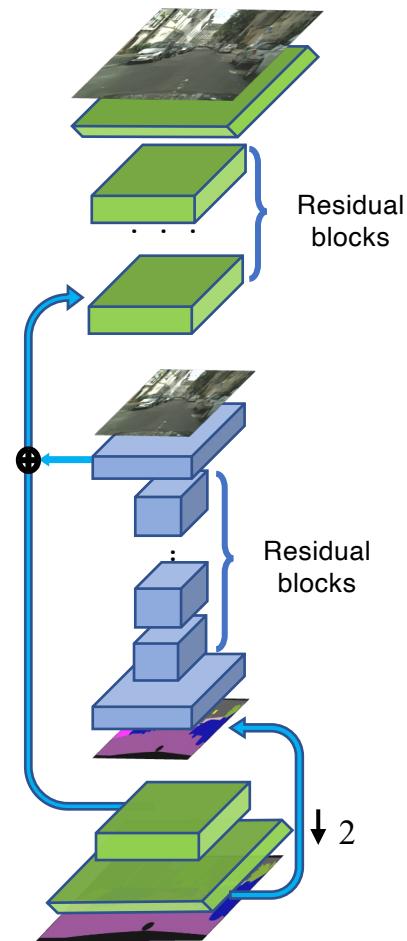
High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, Bryan Catanzaro

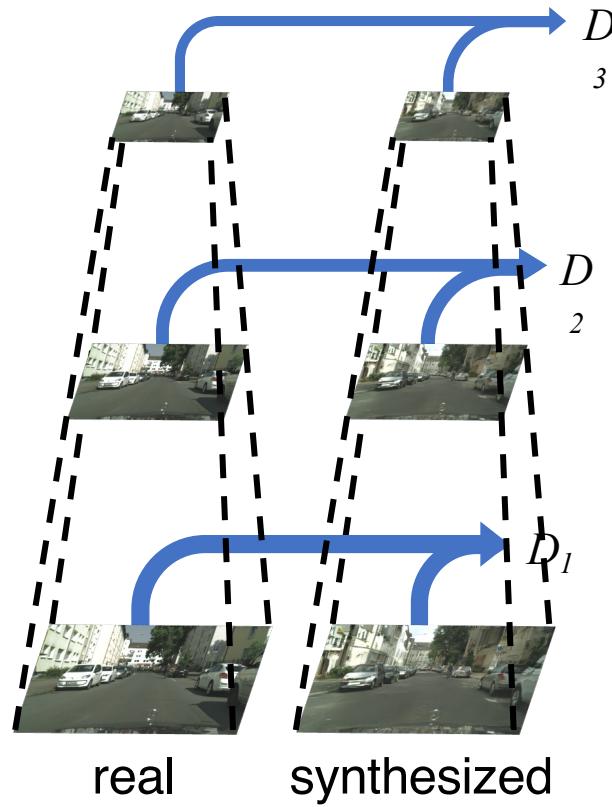


Pix2pixHD [CVPR 2018]

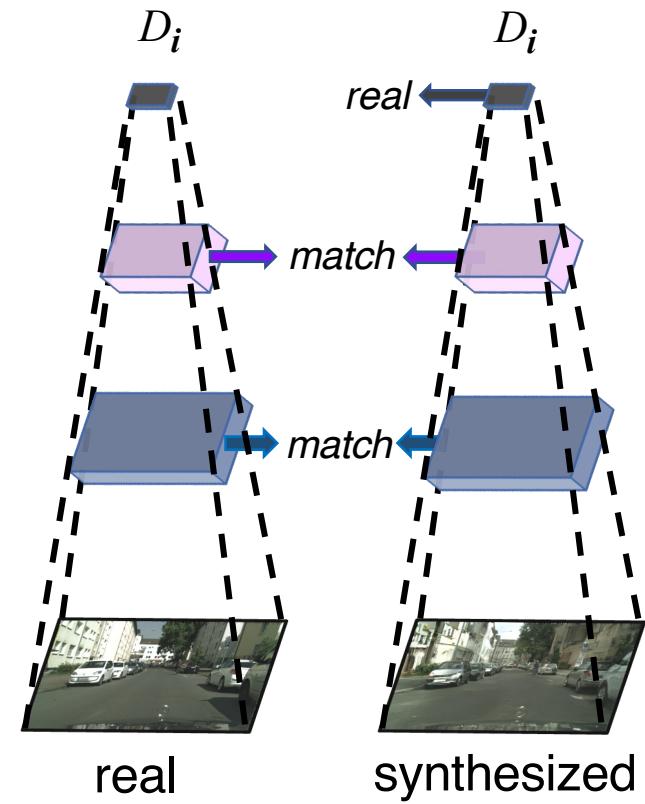
Coarse-to-fine Generator



Multi-scale Discriminators



Robust Objective





Semantic Map



pix2pix



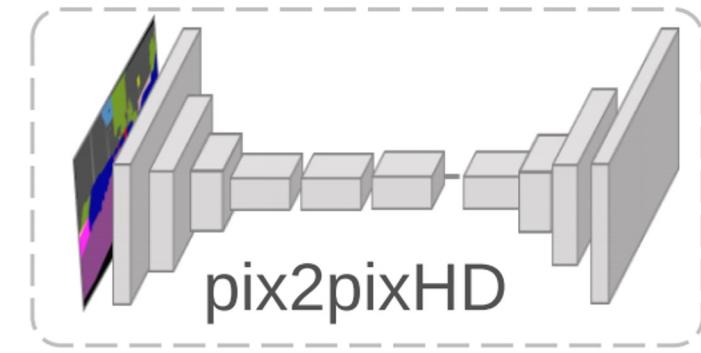
CRN



Ours

Improving Segmentation2Image strategy?

Limitation of the approach:

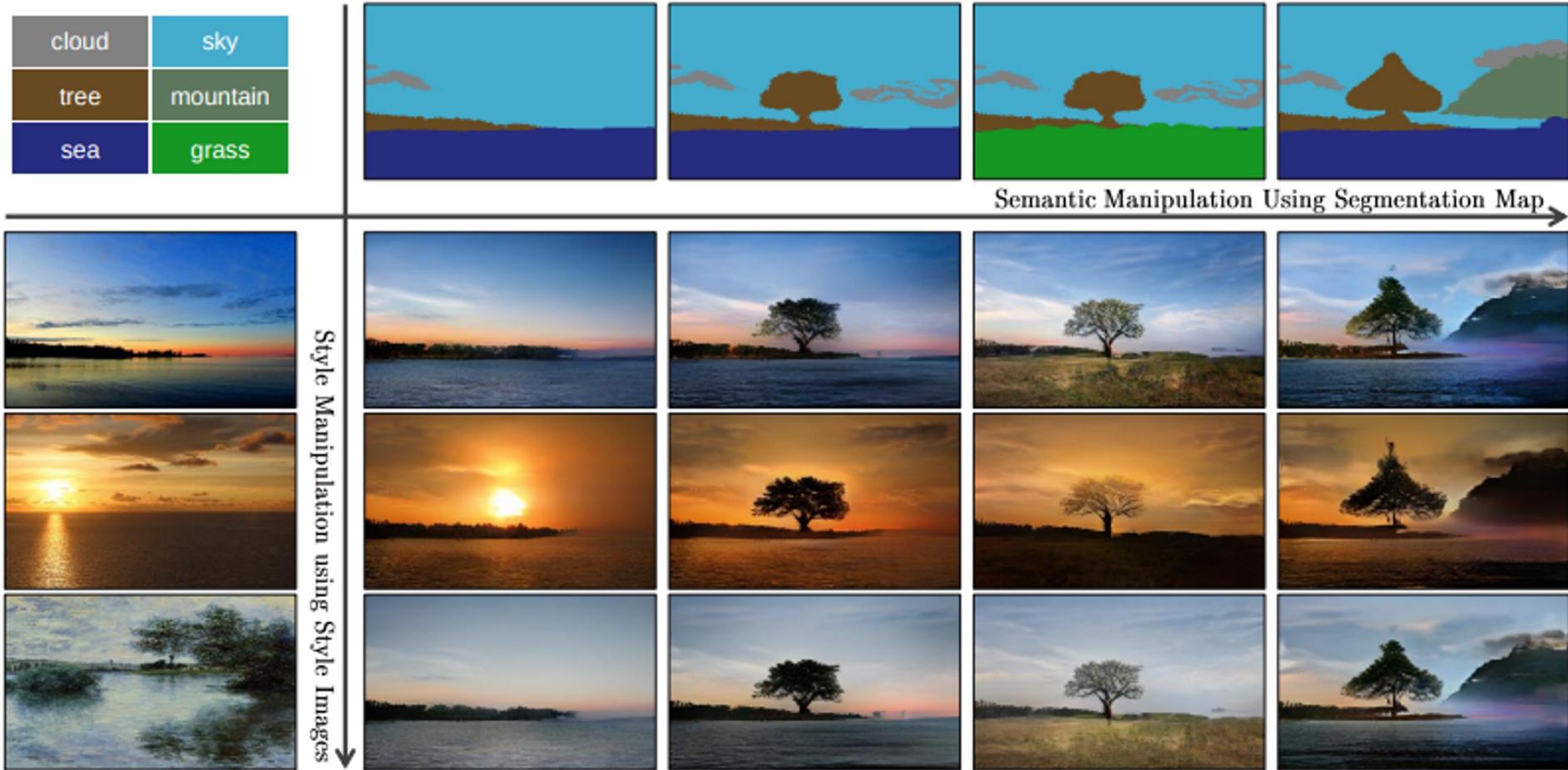


Directly feed the semantic layout as input to the deep network, which is processed through stacks of convolution, normalization, and nonlinearity layers.

However, this is suboptimal as the normalization layers tend to “wash away” semantic information in input semantic segmentation masks.

Improving Segmentation2Image strategy

[SPADE: Semantic Image Synthesis with Spatially-Adaptive Normalization, CVPR19]

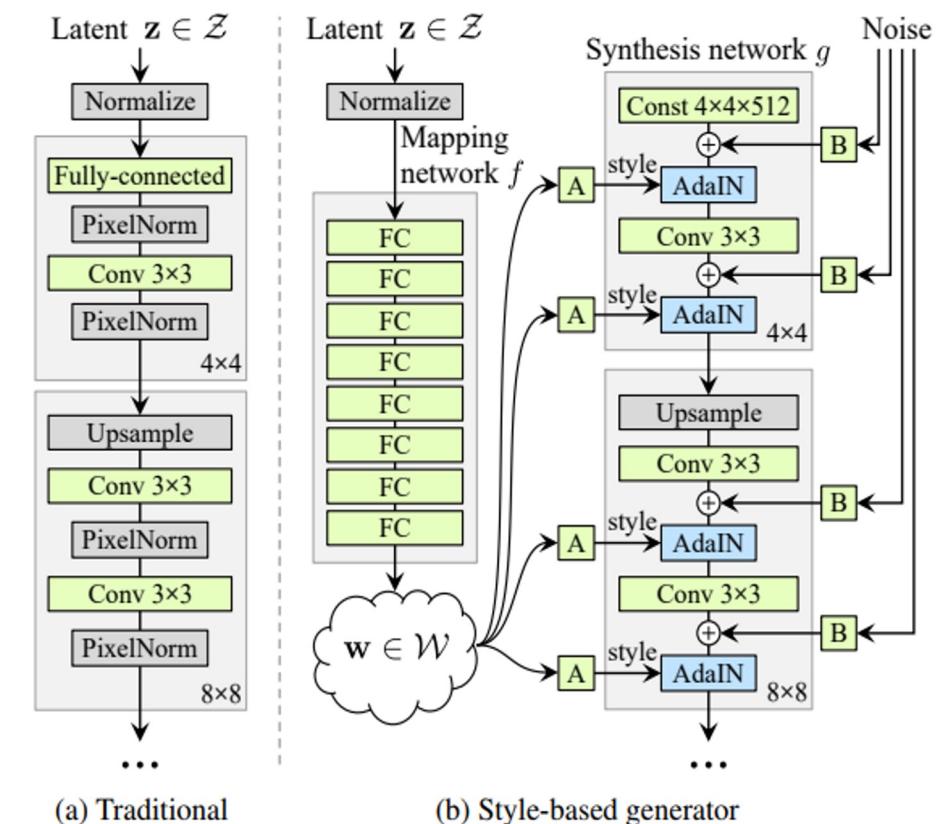


Improving Segmentation2Image strategy

Proven effective for recent generative adversarial networks such as **StyleGAN**

Can we do the same for conditional GAN?

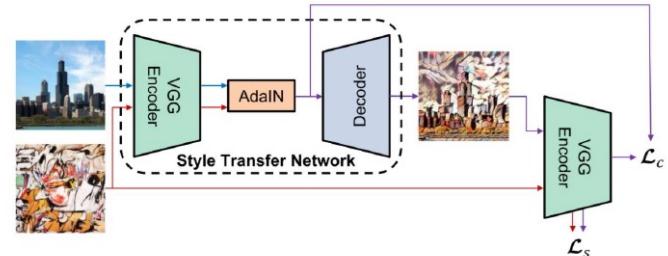
Conditional Normalization Layers?



Improving Segmentation2Image strategy

Recall: Adaptive instance normalization

$$\text{AdaIN}(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y)$$

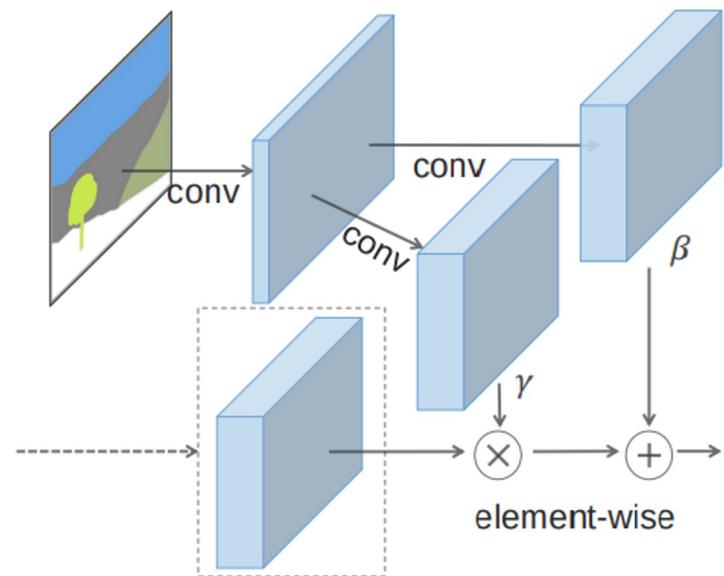


SPADE block= spatially-adaptive denormalization:
Same idea but per class c over each channel i (N=batch size)

$$\gamma_{c,y,x}^i(\mathbf{m}) \frac{h_{n,c,y,x}^i - \mu_c^i}{\sigma_c^i} + \beta_{c,y,x}^i(\mathbf{m})$$

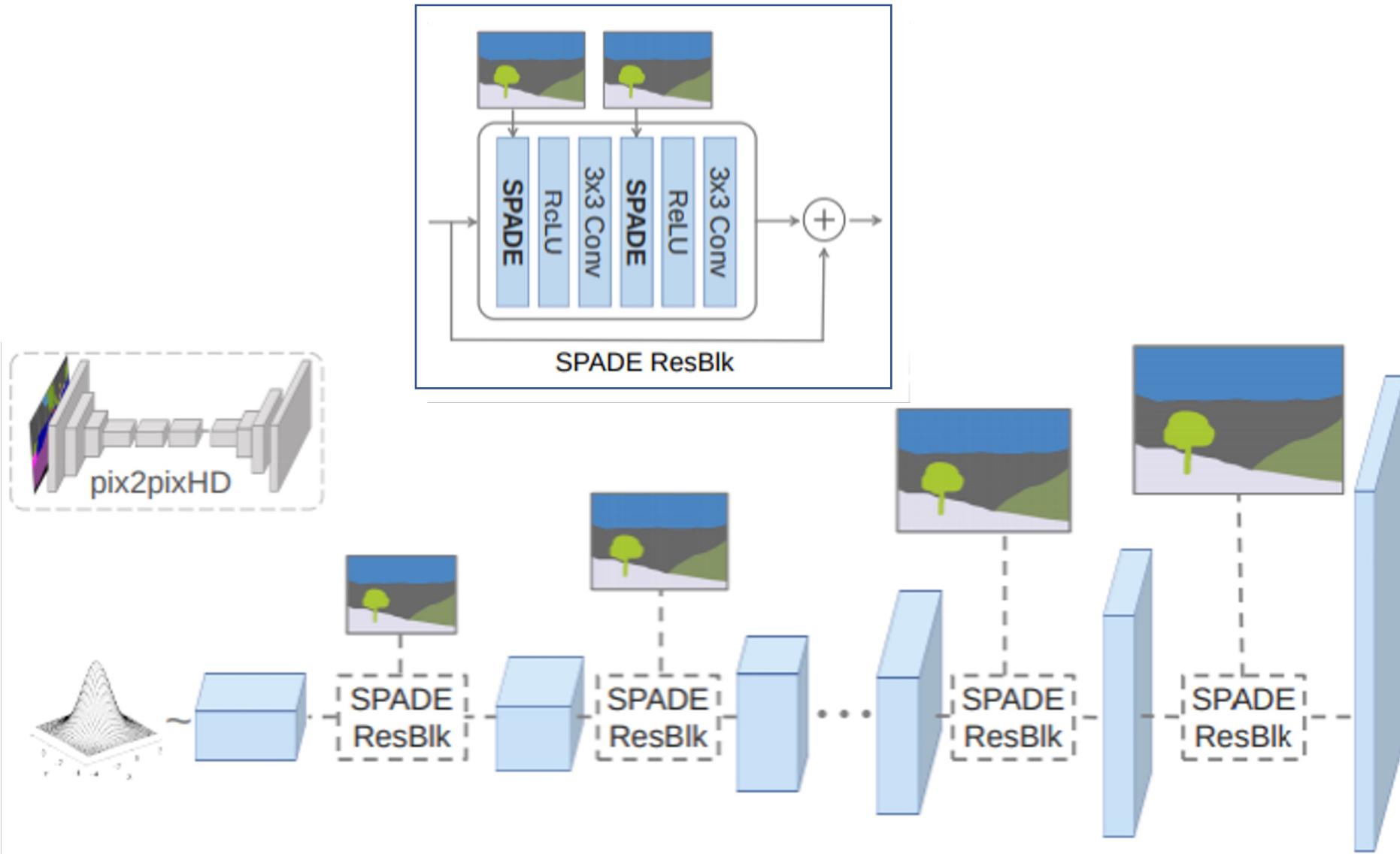
$$\mu_c^i = \frac{1}{NH^iW^i} \sum_{n,y,x} h_{n,c,y,x}^i$$

$$\sigma_c^i = \sqrt{\frac{1}{NH^iW^i} \sum_{n,y,x} (h_{n,c,y,x}^i)^2 - (\mu_c^i)^2}$$

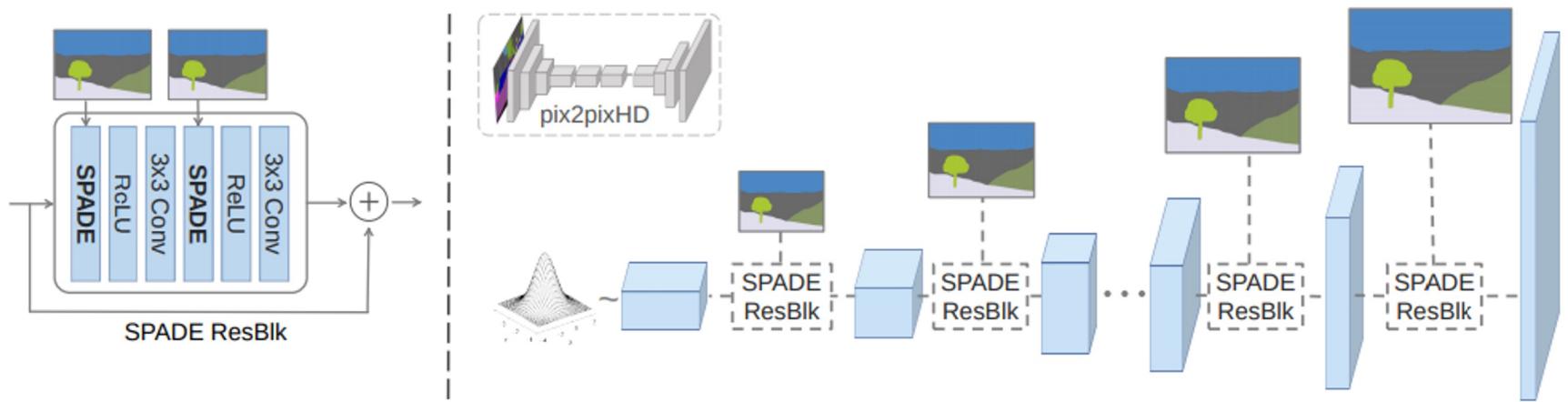


SPADE paper = [Semantic Image Synthesis with Spatially-Adaptive Normalization CVPR 2019]

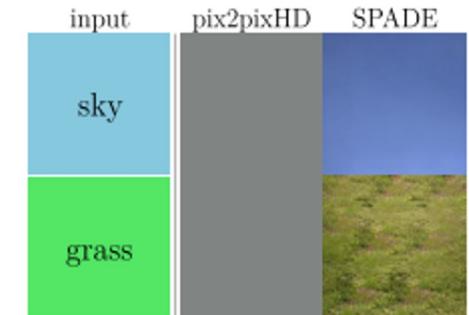
SPADE Generator



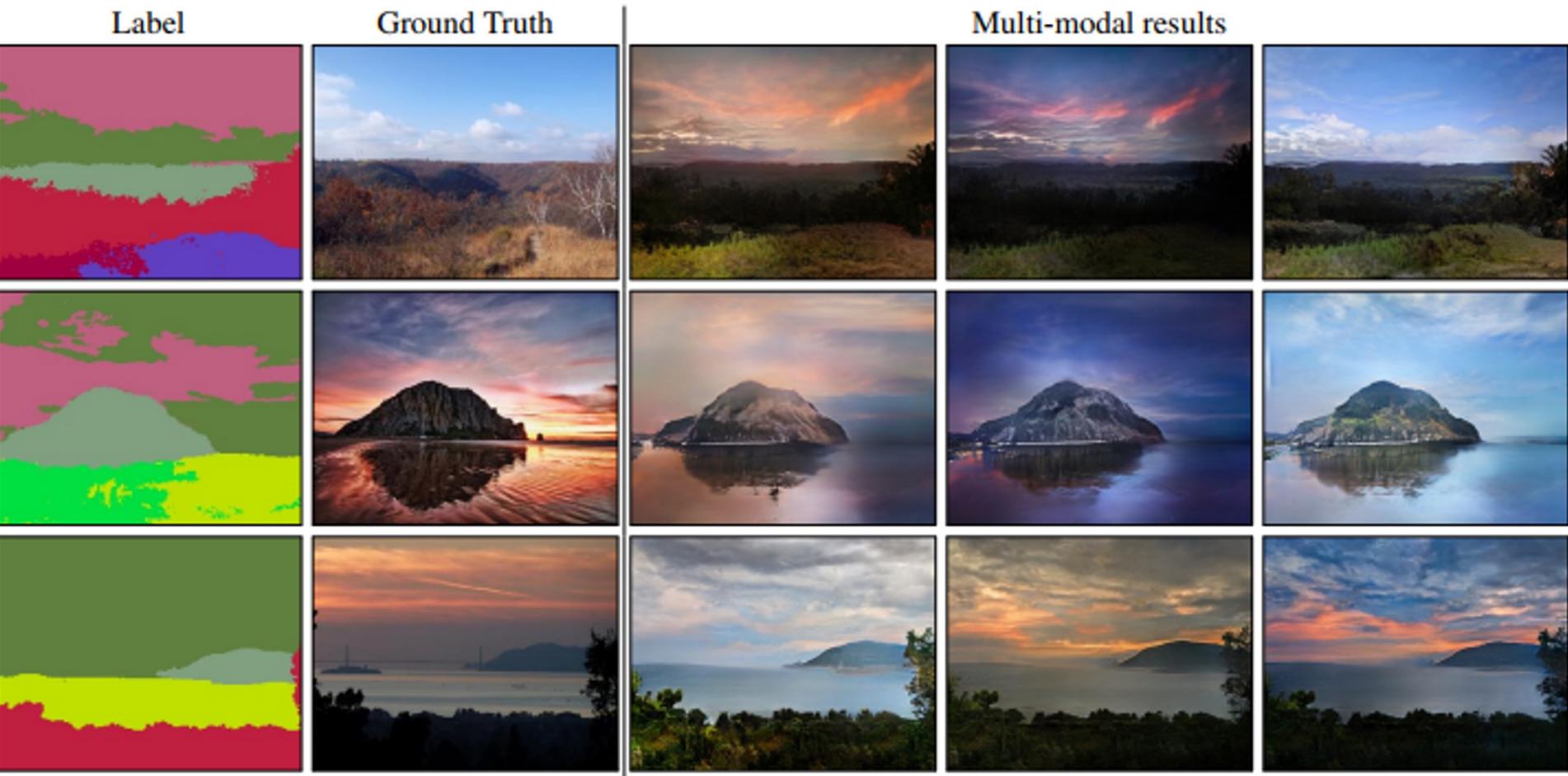
SPADE Generator



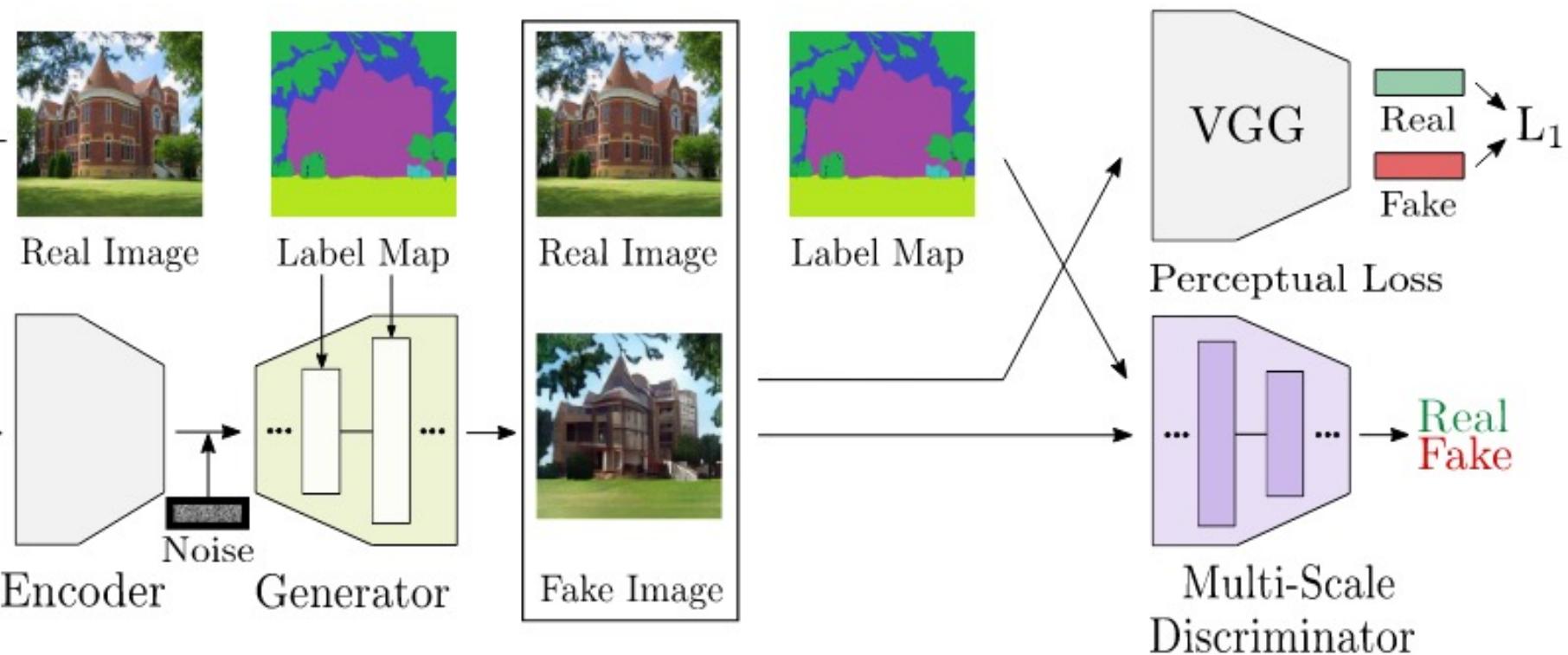
Better preserve semantic information
against common normalization layers



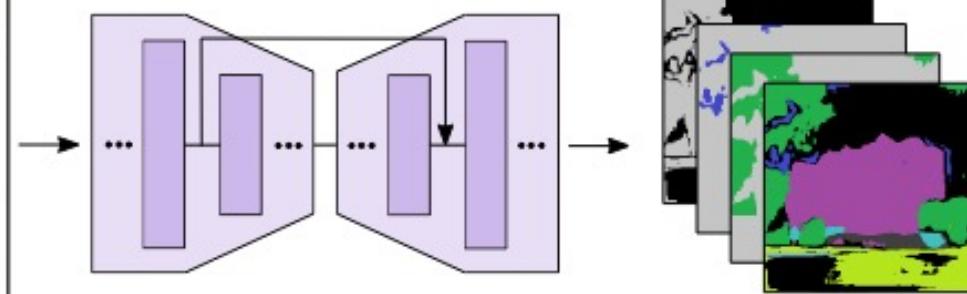
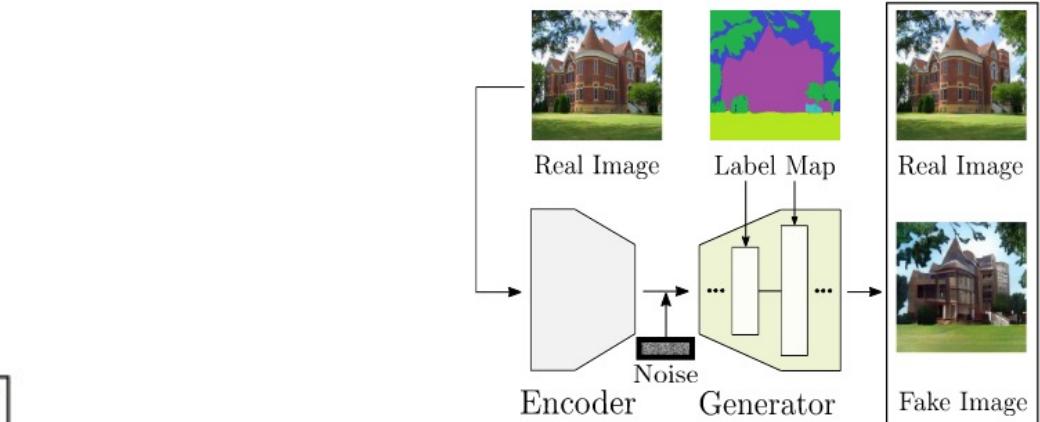
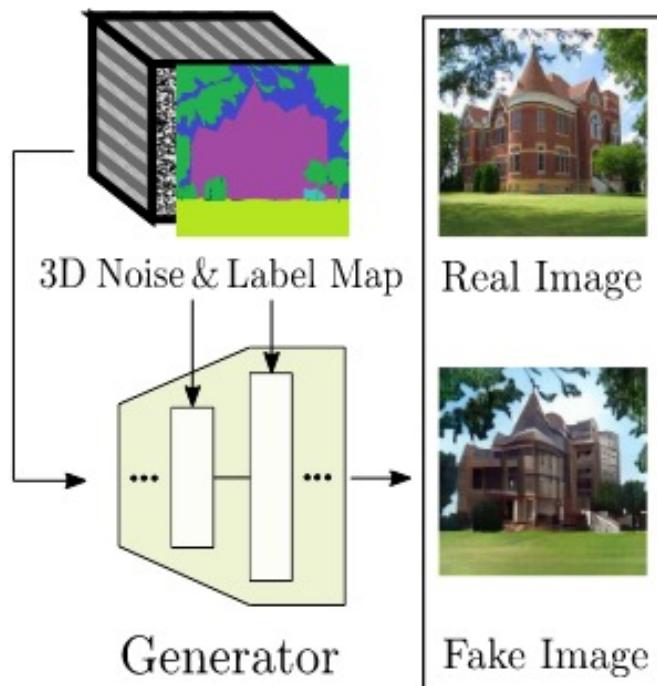
SPADE results



SPADE with real image:



[OASIS iclr 2021] (follow-up paper of SPADE) with real image:

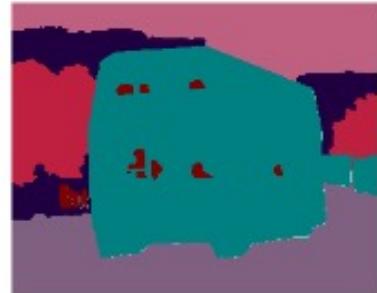


Segmentation-Based
Discriminator

N **Real** /
1 **Fake** Class

[OASIS iclr 2021] (follow-up paper of SPADE) with real image:

Label map



Ground truth



SPADE



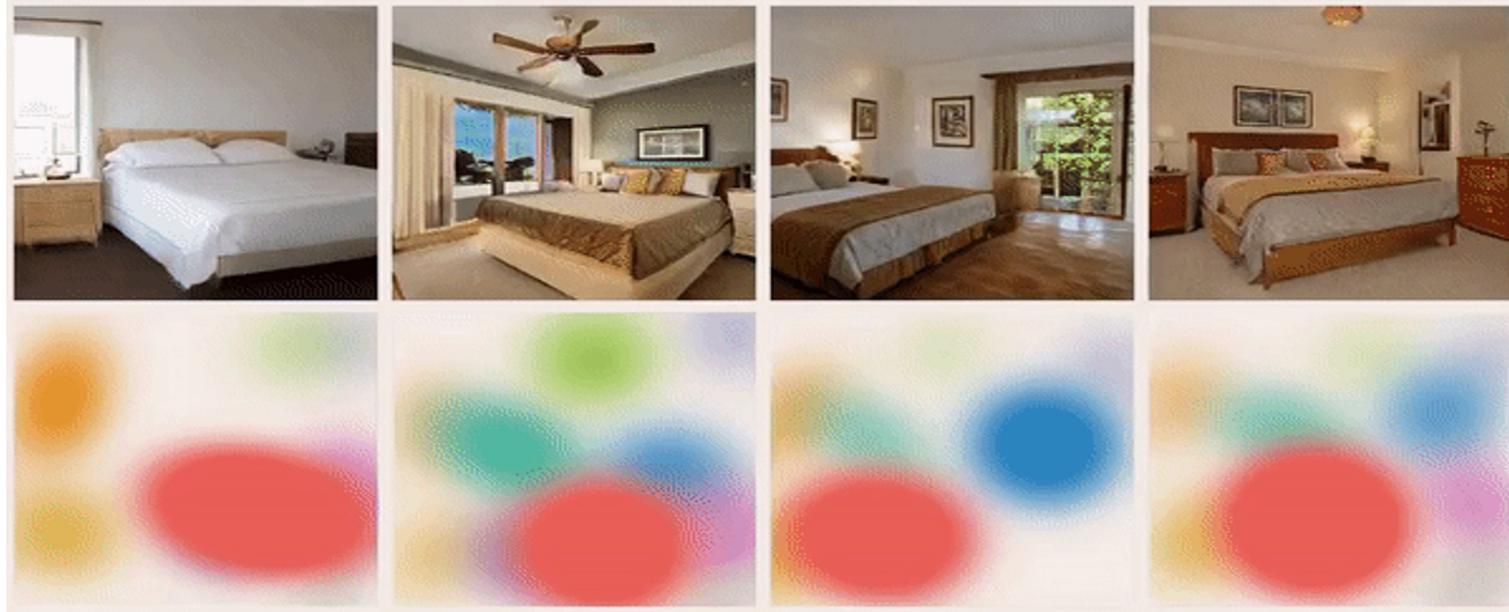
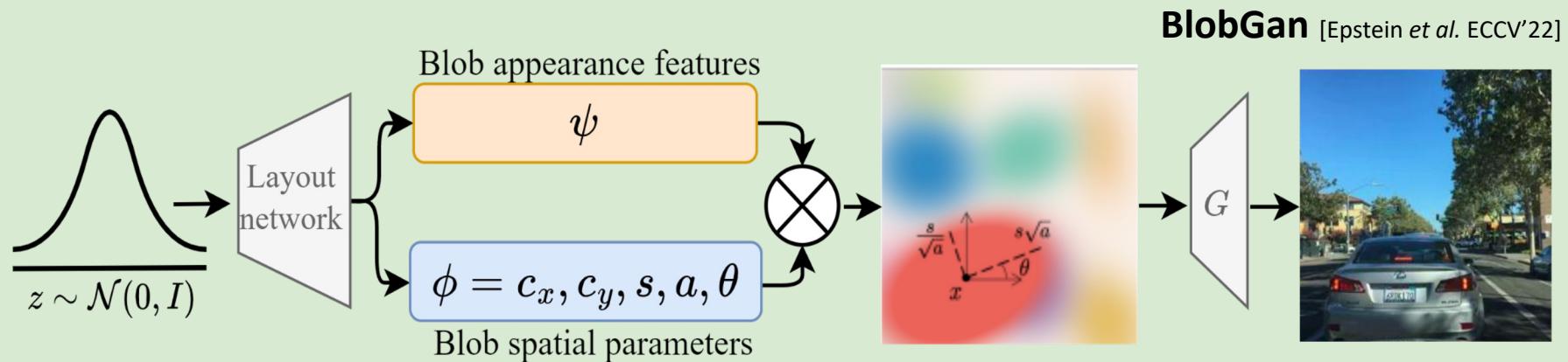
CC-FPSE



OASIS



Editing with *conditional-or-structured-latent* GANs



Editing with *conditional-or-structured-latent* GANs

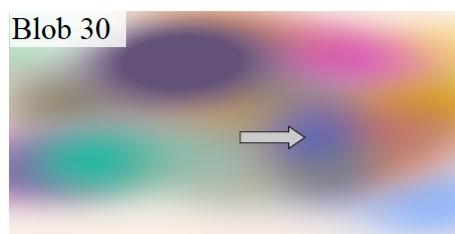
Original blobs



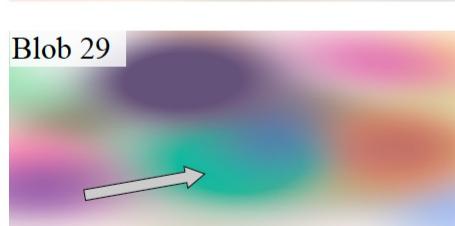
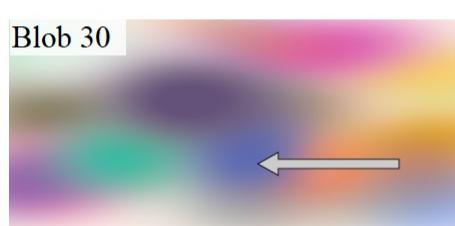
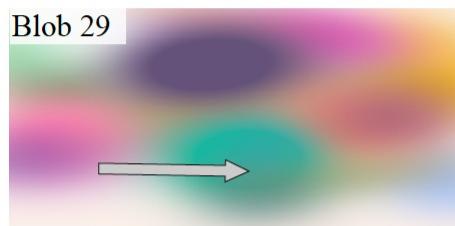
Original image



Edited blobs



Edited image



Editing with *conditional-or-structured-latent* GANs

Example of Counterfactual optimization for editing



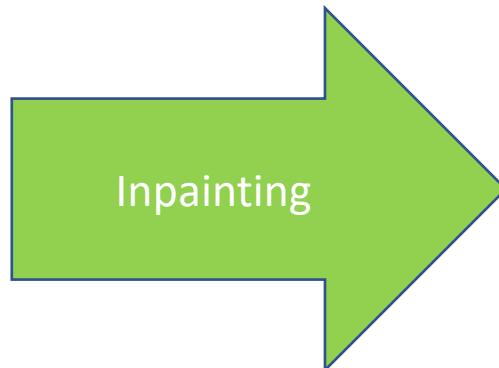
Generative models

Outline

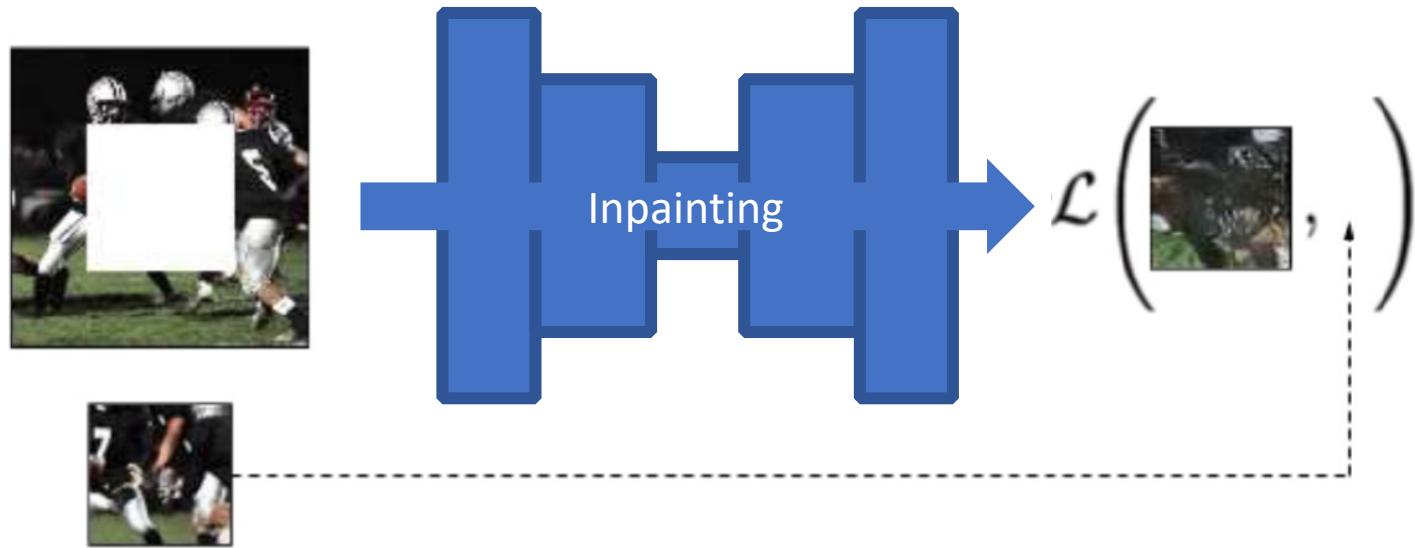
1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
3. GAN architectures
4. Editing
5. Conditional GANs
 1. Principle
 2. Text2Image
 3. Image2Image
 4. **Inpainting and general missing data encoder**

Inpainting task

- Complete the missing part



Inpainting as unsupervised learning with GAN loss

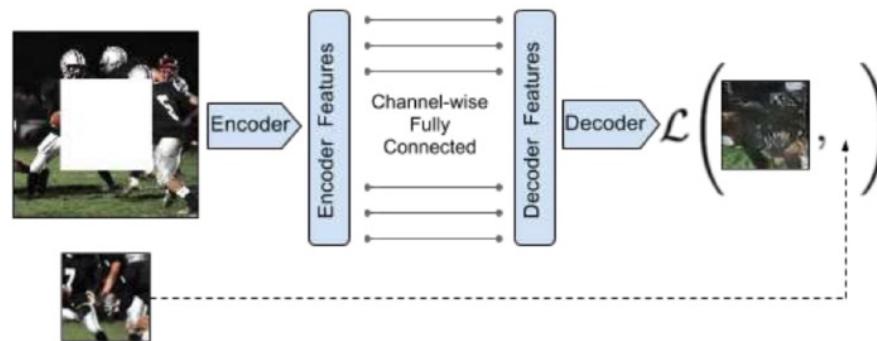


Reconstruct missing pixels by decoding using context

Loss defined on the predicted patch and the real one (known at training time)

First proposition -- Architecture

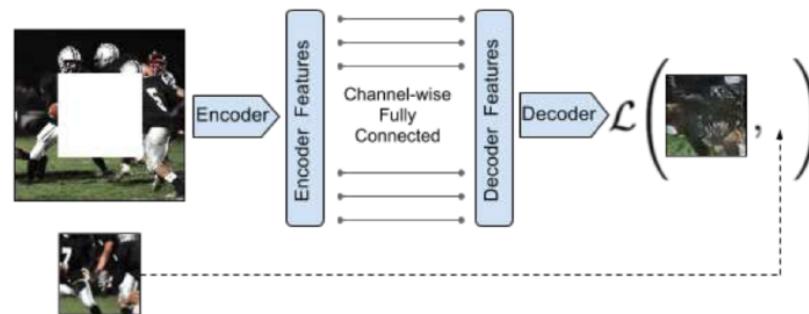
- Architecture: Encoder/Fully connected/Decoder



- DC-GAN for inpainting task
- **Input:** $227 \times 227 \times 3$ image
- **Output:** encoder context features ($6 \times 6 \times 256$)

Channel-wise fully-connected layer

- **Input / output:** $6 \times 6 \times 256$ channels
- **First layer:** Channel-wise fully-connected
(each 6×6 input connected to the corresponding 6×6 output)
- **Second layer:** Stride 1 convolution to mix channels

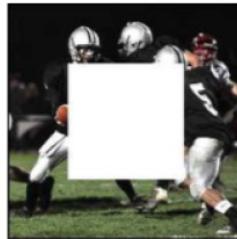


Decoder

- **Architecture:** Same as DC-GAN: 5 up-convolutional layers (“deconv” + ReLU)
- **Input:** decoder context features $6 \times 6 \times 256$
- **Output:** $227 \times 227 \times 3$ image

Training: Masking the images

- **How to define the mask ?**
 - ▶ Center region of the image
 - ▶ Random regions (chosen solution)
 - ▶ Random segmentation mask from VOC (said to be equivalent to random regions)
- **Formal definition:** Defined by a mask $\hat{M} \in \{0, 1\}^{227 \times 227}$ with 1 if the pixel should be masked



Training: Loss - Overview

- Trained completely from scratch to fill-up the masked areas
- **Problem:** multiple plausible solutions
- **Solution:** combining 2 losses:
 - ▶ \mathcal{L}_{rec} **L2 reconstruction loss:** learn the structure of the missing region (average multiple modes in prediction)
 - ▶ \mathcal{L}_{adv} **Adversarial loss:** make it look real (pick a mode from the distribution)

$$\min_F \mathcal{L} = \lambda_{rec} \mathcal{L}_{rec} + \lambda_{adv} \mathcal{L}_{adv}$$

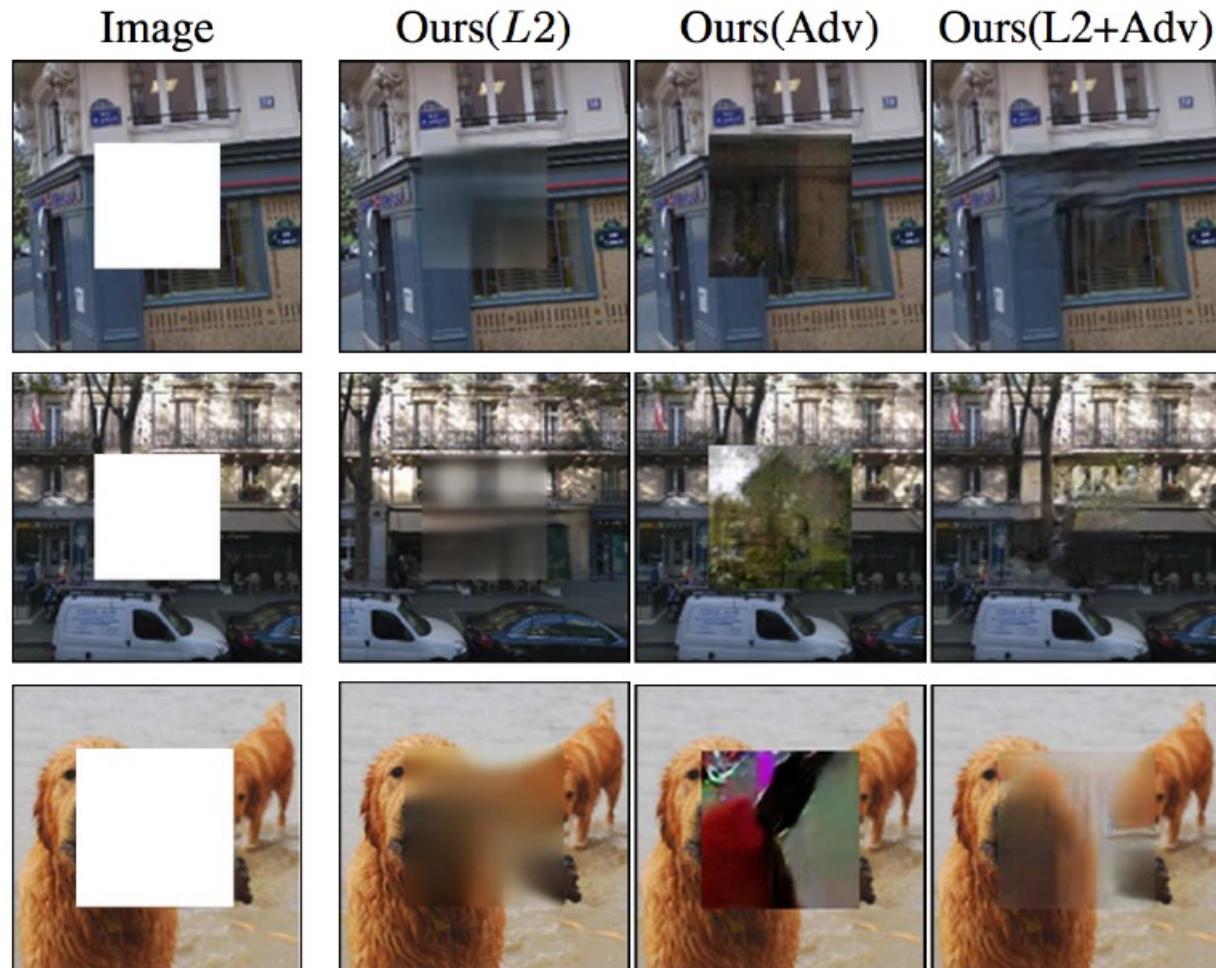
$$\mathcal{L}_{rec}(x) = \left\| \hat{M} \odot \left(x - F((1 - \hat{M}) \odot x) \right) \right\|_2$$

$$\mathcal{L}_{adv} = \max_D \mathbb{E}_{x \in \mathcal{X}} \left[\log(D(x)) + \log \left(1 - D(F((1 - \hat{M}) \odot x)) \right) \right]$$

- Rq: The encoder-decoder is the generator, D is a CNN

Results

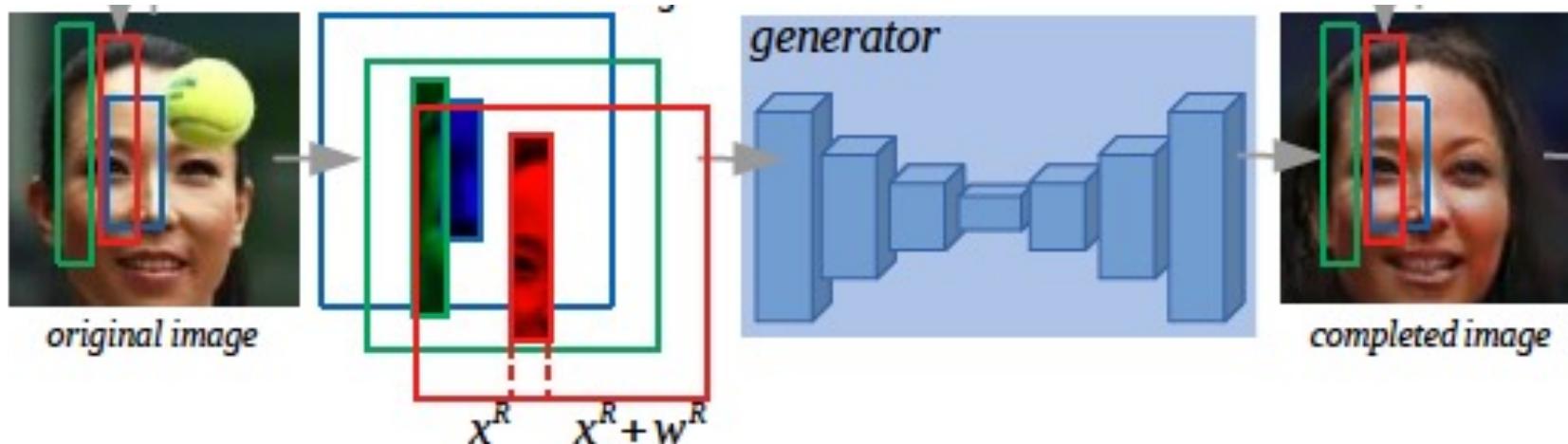
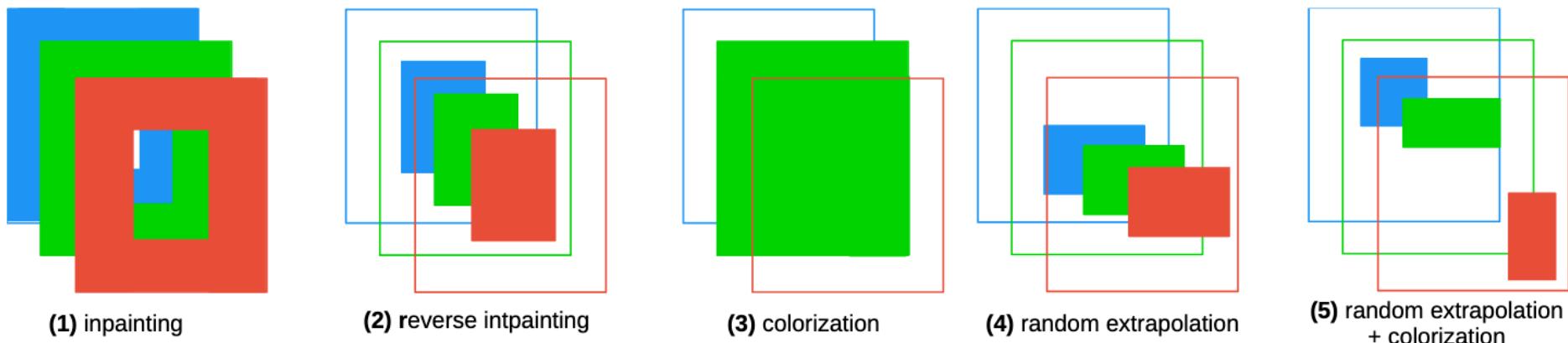
Dataset: StreetView Paris and ImageNet



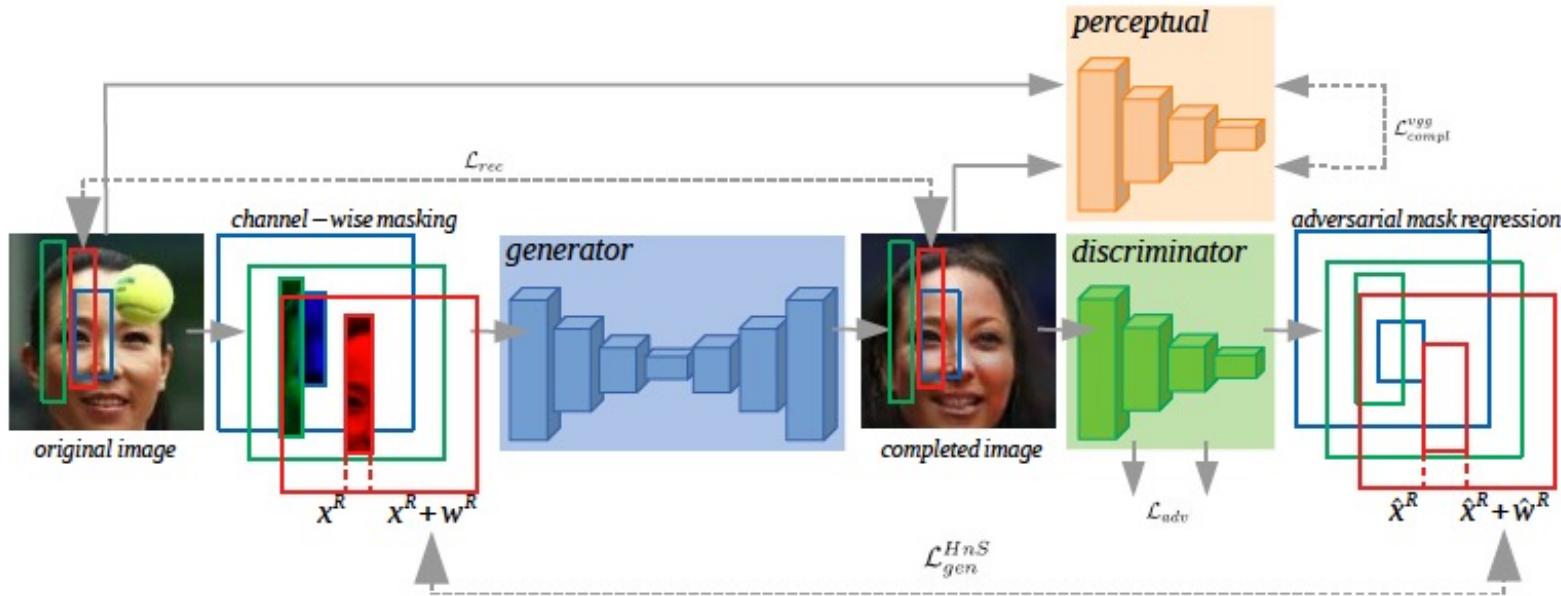
Semantic inpainting - Qualitative results



Generalizing inpainting: missing data encoder



Adding perceptual loss, BB regression loss



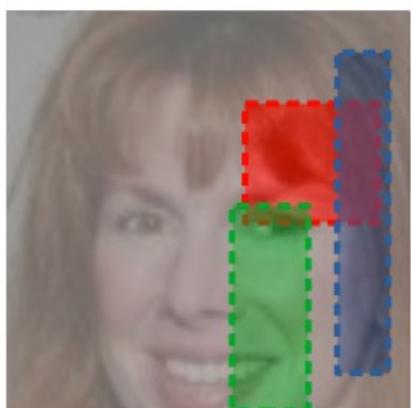
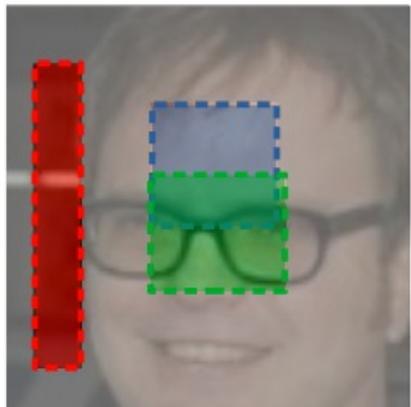
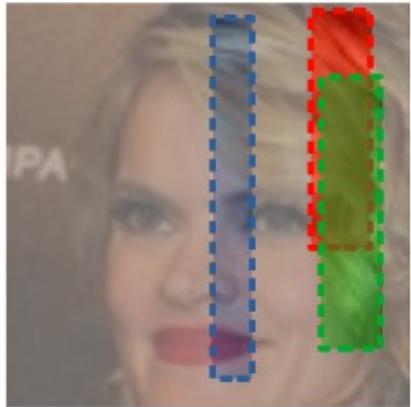
$$r^c = (x^c/W, y^c/H, (x^c + w^c)/W, (y^c + h^c)/H)$$

$$\mathcal{L}_{disc}^{HnS}(\theta_d) = \frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C \| r_i^c - \hat{r}_i^c(\theta_g, \theta_d) \|$$

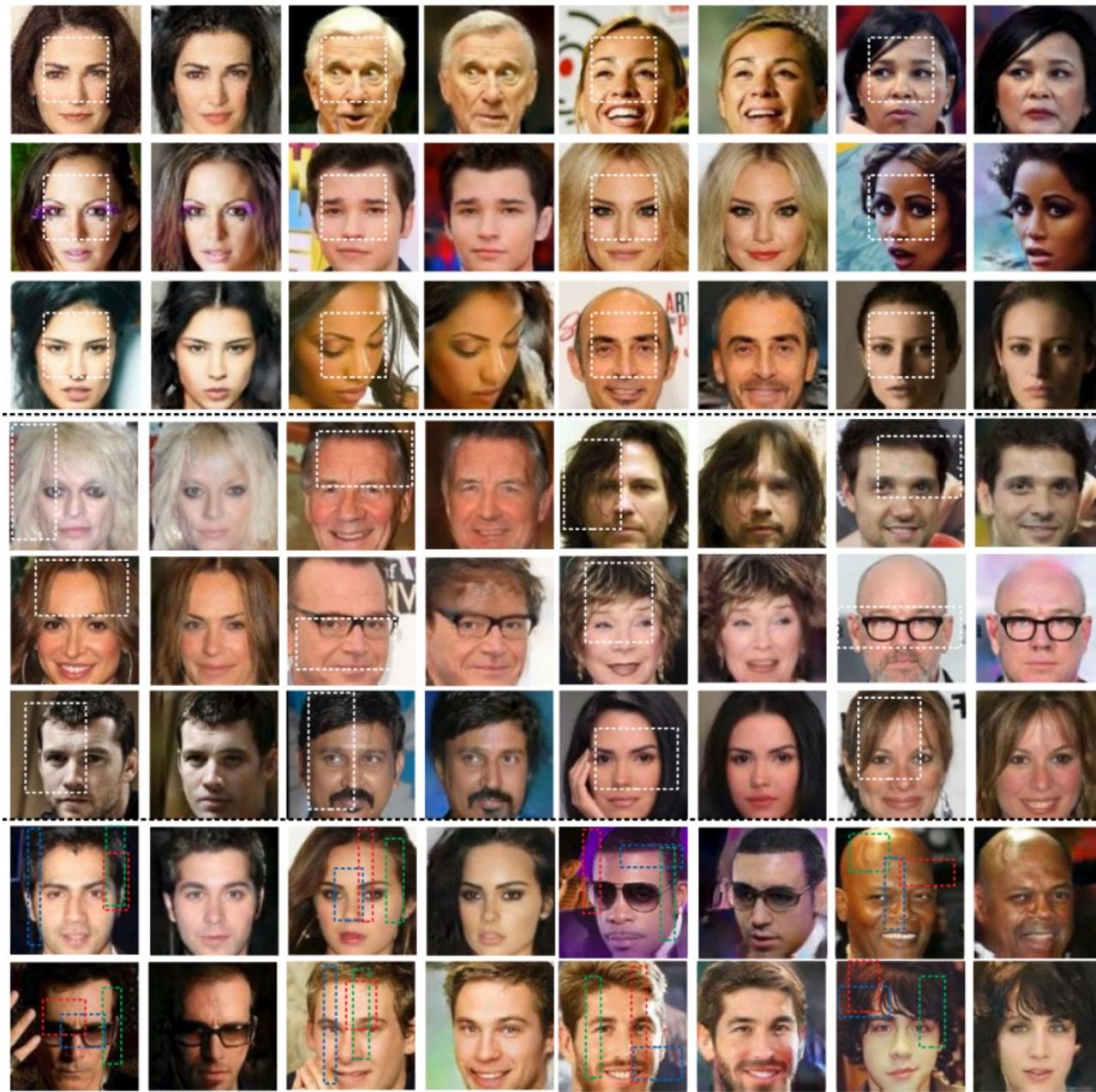
$$\mathcal{L}_{gen}^{HnS}(\theta_g) = \frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C \| q_i^c - \hat{r}_i^c(\theta_g, \theta_d) \|$$

$$\mathcal{L}_{tot}(\theta_g, \theta_d) = \mathcal{L}_{rec}(\theta_g) + \lambda_{compl} \mathcal{L}_{compl}^{vgg}(\theta_g) + \lambda_{adv} \mathcal{L}_{adv}(\theta_g, \theta_d) + \lambda_{HnS} \mathcal{L}_{coord}^{HnS}(\theta_g, \theta_d)$$

Results



Results



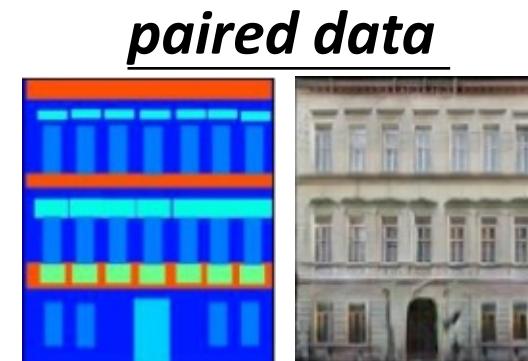
Generative models

Outline

1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
3. GAN architectures
4. Editing
5. Conditional GANs
 1. Principle
 2. Text2Image
 3. Image2Image
 4. Inpainting and general missing data encoder
 5. **Learning unpaired Transformation**

Unpaired Transformation

- Cycle GAN, Disco GAN



Transform an object from one domain to another ***without paired data***



photo



van Gogh



Domain X



Domain Y



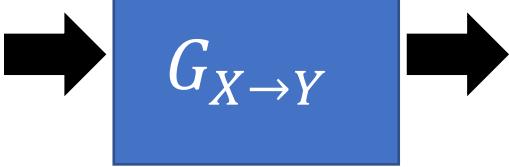
Cycle GAN

<https://arxiv.org/abs/1703.10593>

<https://junyanz.github.io/CycleGAN/>



Domain X

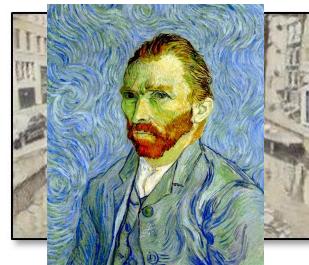


ignore input

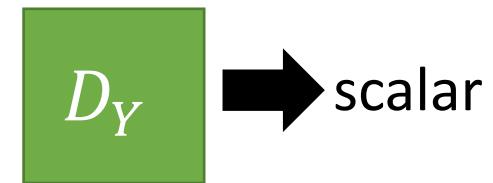


Domain Y

Become similar
to domain Y



Not what we want



scalar



Input image
belongs to
domain Y or not

Cycle GAN

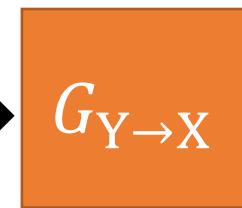
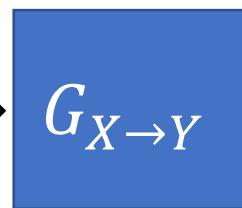
Domain X



Domain Y



as close as possible



Lack of information
for reconstruction



$G_{Y \rightarrow X}$

D_Y

scalar

Input image
belongs to
domain Y or not



Domain Y

Cycle GAN

Domain X



Domain Y



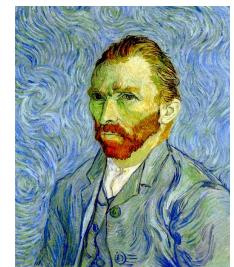
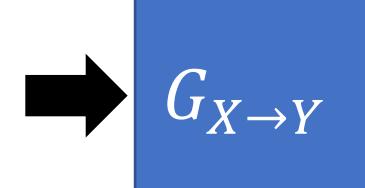
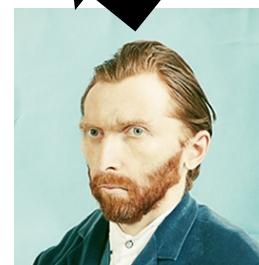
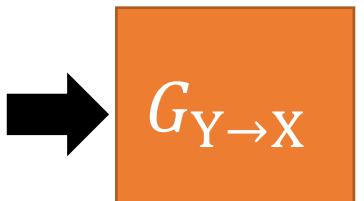
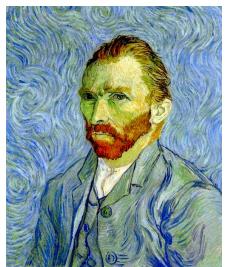
as close as possible



scalar: belongs to domain X or not



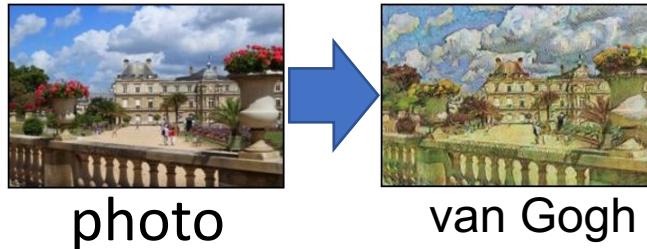
scalar: belongs to domain Y or not



as close as possible



Results -- Cycle GAN



Monet ↪ Photos



Monet → photo

Zebras ↪ Horses



horse → zebra

Summer ↪ Winter



summer → winter

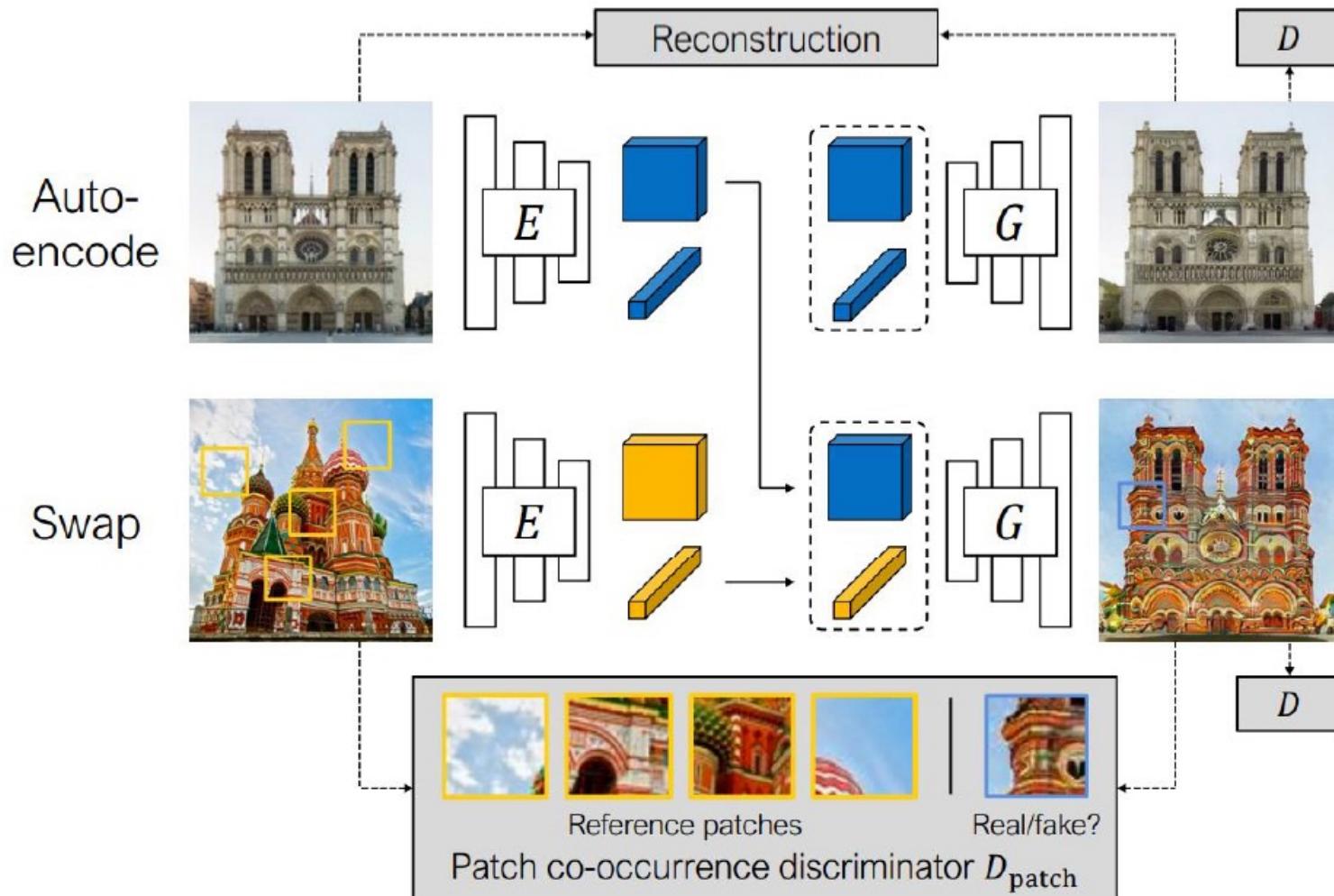
photo → Monet



winter → summer

GANs: works in progress

A lot of things to better understand, to use, adapt, ...

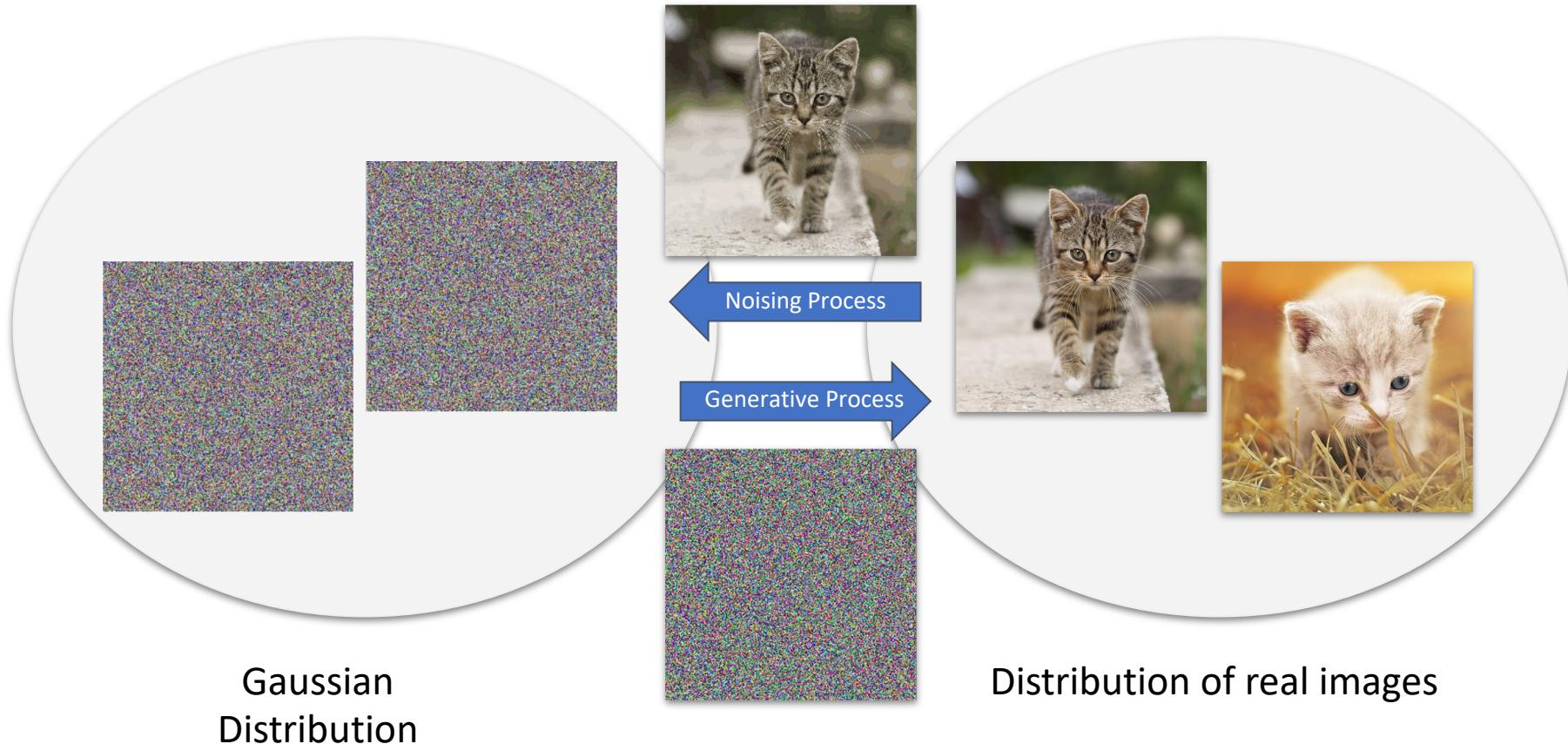


Generative models

Outline

1. Preview: Auto-Encoders, VAE
2. Generative models with GAN
3. GAN architectures
4. Editing
5. Conditional GANs
- 6. Diffusion models**

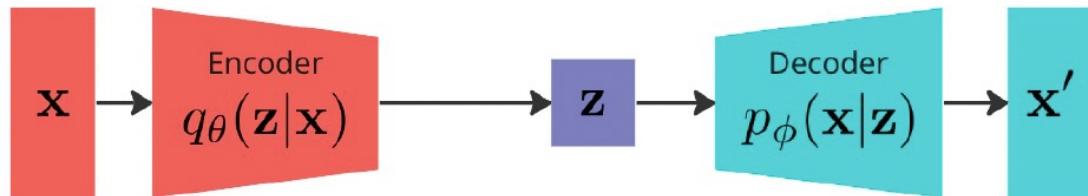
Generative Modelling with Diffusion models



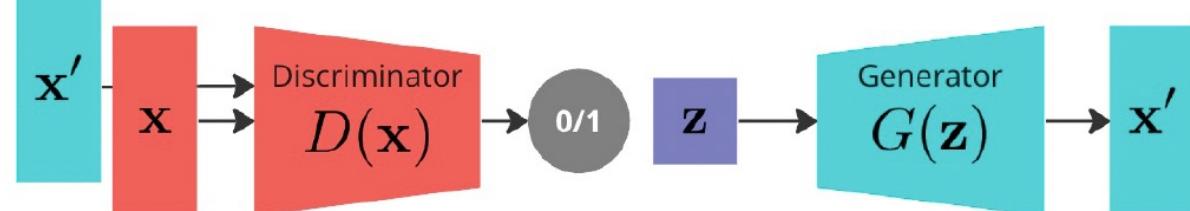
Generative Modelling with Diffusion models

*DDPM: Denoising Diffusion Probabilistic Models
In context with other generative Models:*

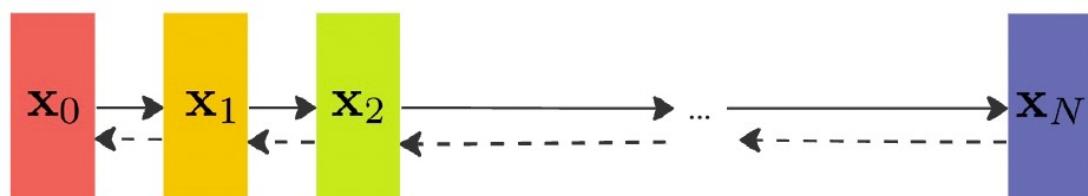
VAE : Maximizes the variational lower bound



GAN: Adversarial training



DDPM: Gradually add noise and learn to denoise



Generative Modelling with Diffusion models

DDPM: Denoising Diffusion Probabilistic Models

$$\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \epsilon$$



$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \epsilon$$



Training:

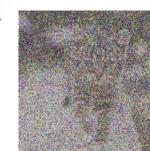
The reverse process refers to learning a model, ϵ_θ , which approximates the noise added at a given timestep t :

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|_2^2$$

Sampling:

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\alpha_t}} (\mathbf{x}_t - \sqrt{1 - \alpha_t} \cdot \epsilon_\theta(\mathbf{x}_t, t)),$$

$$\mathbf{x}_{t-1} = \frac{(\alpha_{t-1} - \alpha_t) \sqrt{\alpha_{t-1}}}{\alpha_{t-1}(1 - \alpha_t)} \hat{\mathbf{x}}_0 + \frac{(1 - \alpha_{t-1}) \sqrt{\alpha_t}}{(1 - \alpha_t) \sqrt{\alpha_{t-1}}} \mathbf{x}_t + \sigma_t \mathbf{z},$$



Generative Modelling with Diffusion models

DDPM: Denoising Diffusion Probabilistic Models

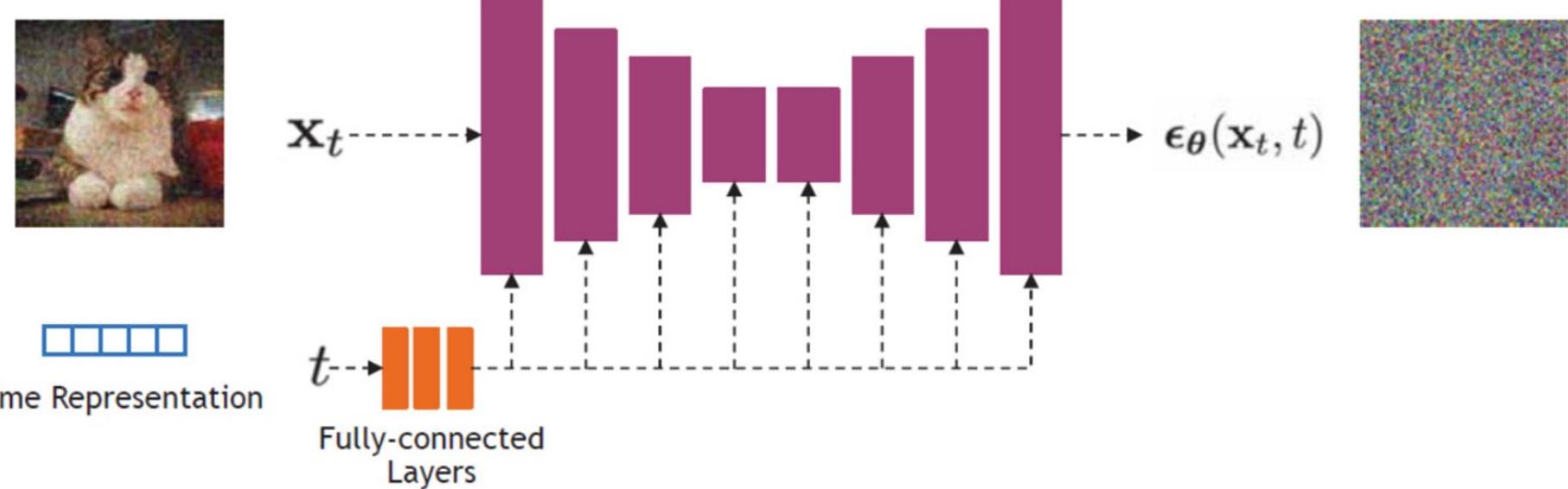
Algorithm 1 Training

```
1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
        $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2$ 
6: until converged
```

Algorithm 2 Sampling

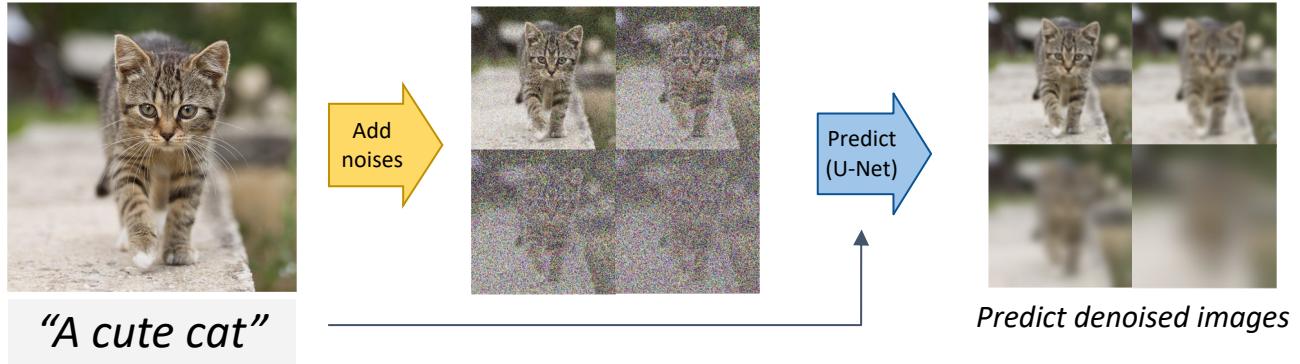
```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left( \mathbf{x}_t - \frac{1 - \bar{\alpha}_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

U-Net for Diffusion models

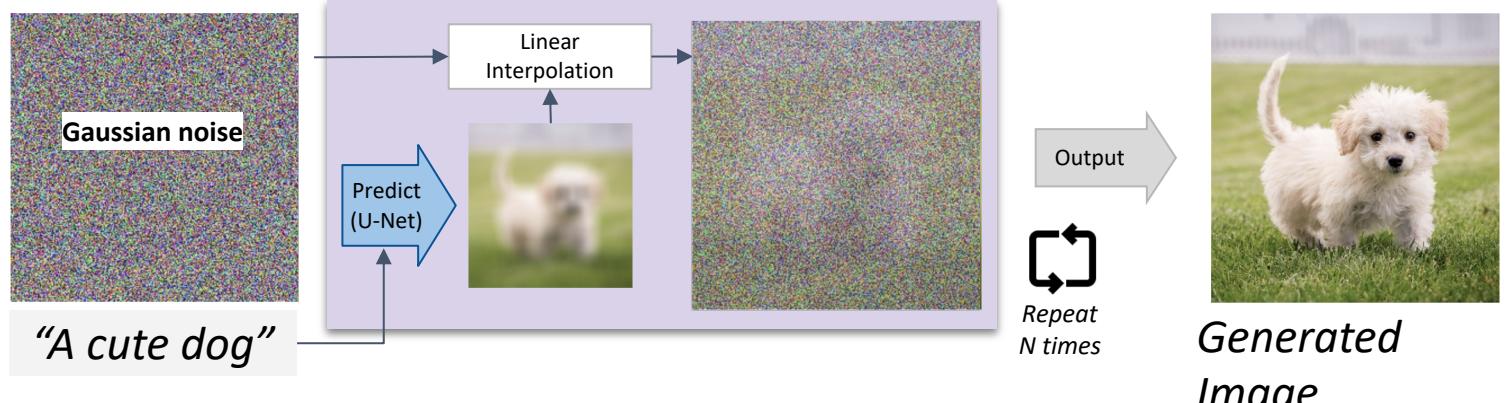


Text-conditioned Diffusion models

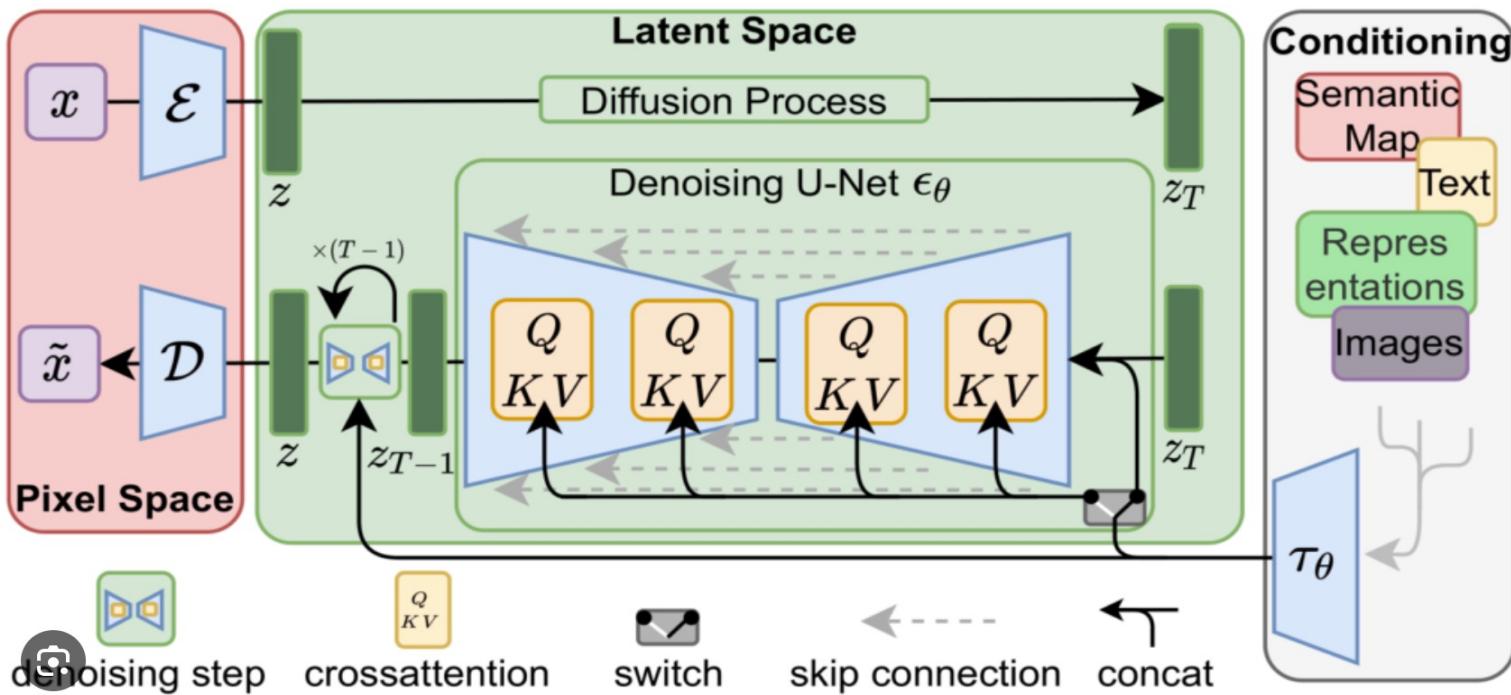
● Training:



● Sampling:



Text-conditional Diffusion models



Text-conditional Diffusion models

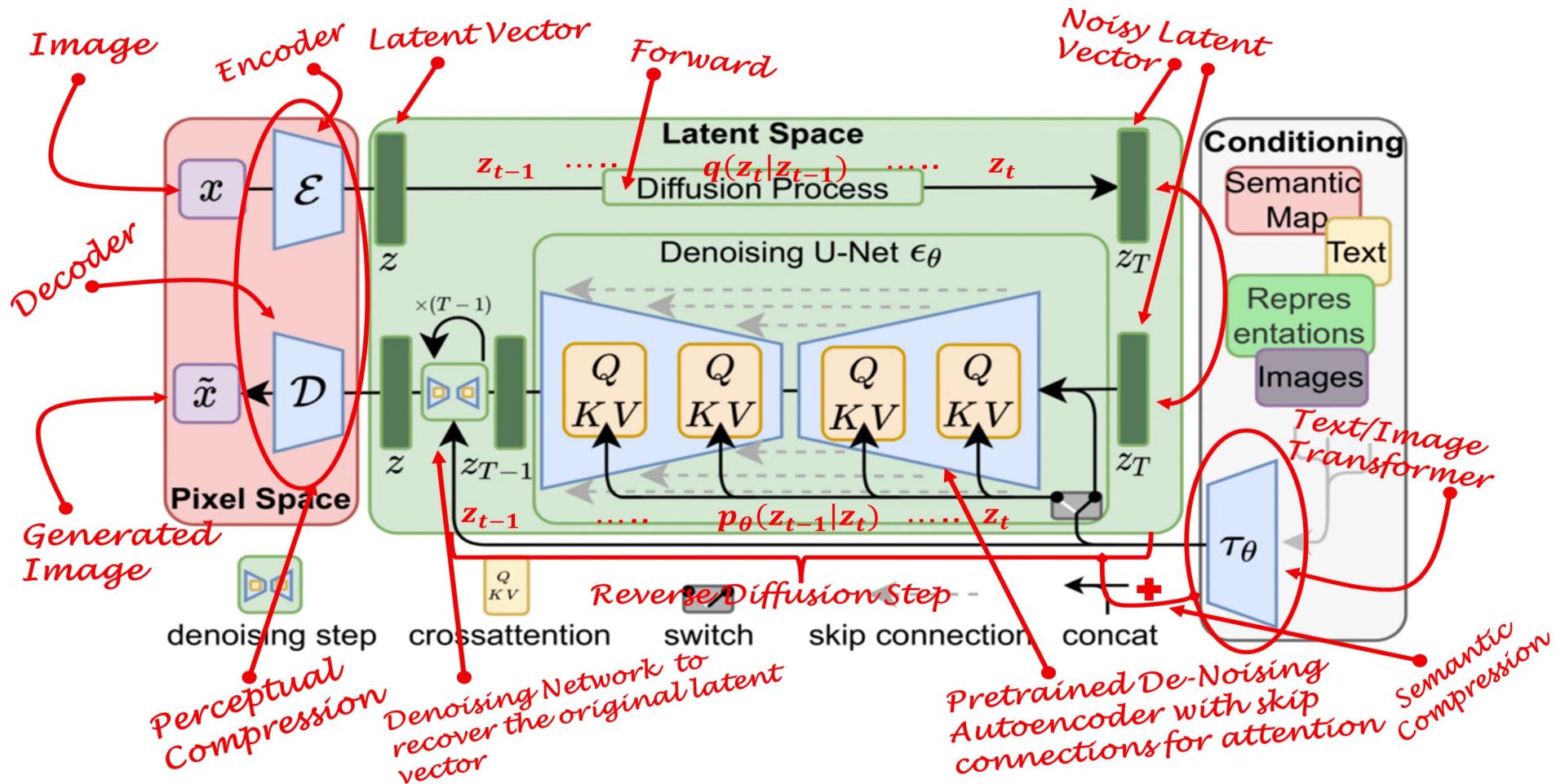


Image Generation using Diffusion Models



Sprouts in the shape of text 'Imagen' coming out of a fairytale book



A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.



A high contrast portrait of a very happy fuzzy panda dressed as a chef in a high end kitchen making dough. There is a painting of flowers on the wall behind him.



A cute corgi lives in a house made out of sushi.

Generation using Diffusion Models

And the next big thing is ...

Text to Video



Animation



Transition



Transition



Prediction