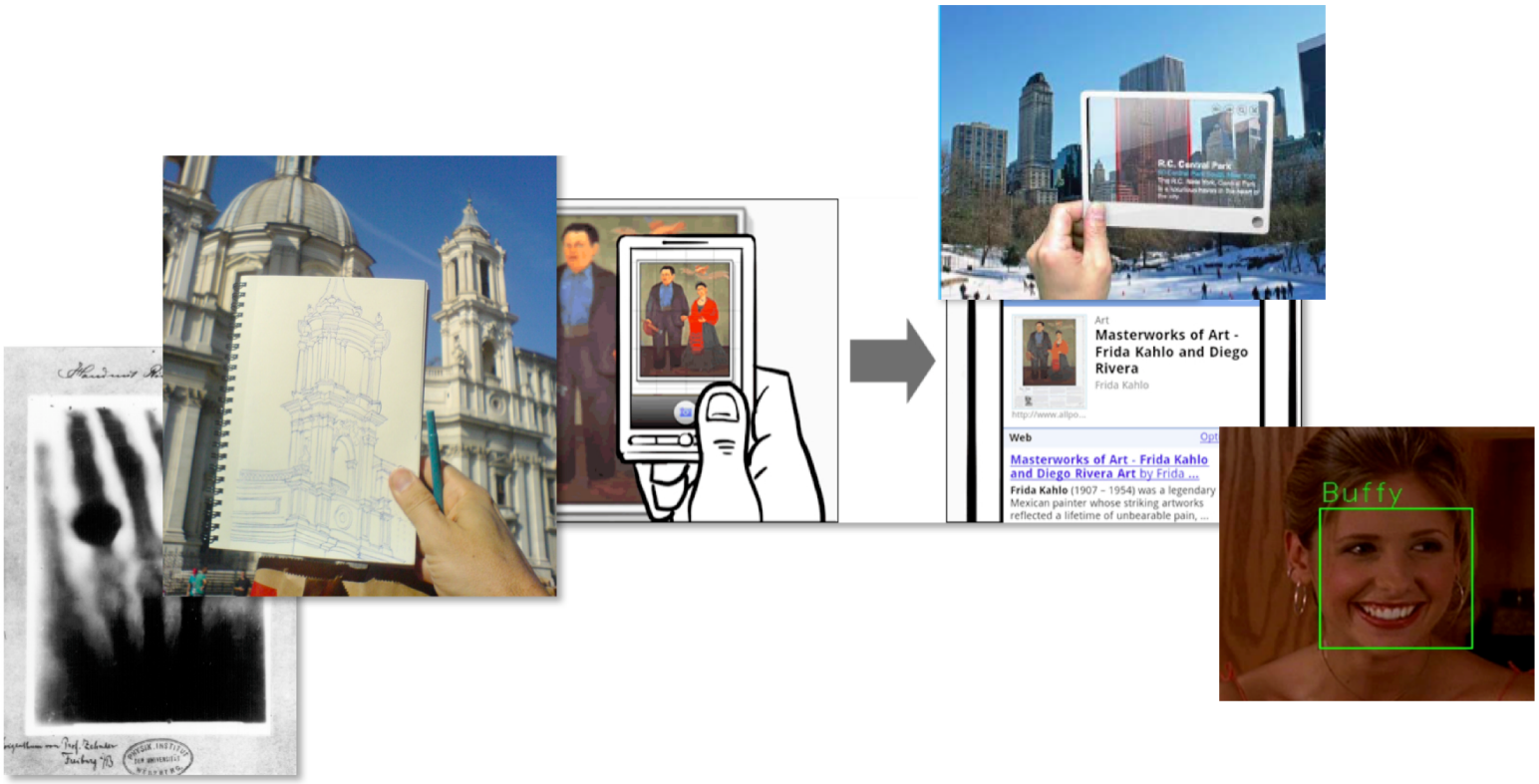


Deep (6): generative models

Matthieu Cord —LIP6 / UPMC Paris 6

Generative models

Outline

1. Preview: Auto-Encoders, VAE
2. Generative models with GAN

   Few insights about KL divergence in Max Likelihood vs. Jensen Shanon Divergence in traditional GANs

3. GAN architectures for image generation

Which measure to evaluate how $P_G(x;\theta)$ is close to $P_{data}(x)$ in Maximum Likelihood optimization?

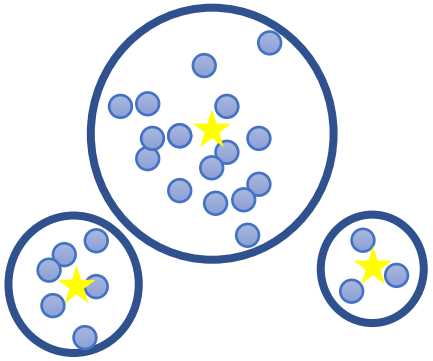
- Given a data distribution $P_{data}(x)$
- We have a distribution $P_G(x;\theta)$ parameterized by θ
  - E.g. $P_G(x;\theta)$ is a Gaussian Mixture Model, θ are means and variances of the Gaussians
  - We want to find θ such that $P_G(x;\theta)$ close to $P_{data}(x)$

Sample $\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$

We can compute $P_G(x^i;\theta)$

Likelihood of generating the samples

$$L = \prod_{i=1}^{m} P_G(x^i;\theta)$$



Find θ^* maximizing the likelihood

Which measure to evaluate how $P_G(x;\theta)$ is close to $P_{data}(x)$ in Maximum Likelihood optimization?

$$\theta^* = arg\max_\theta \prod_{i=1}^{m} P_G(x^i;\theta) = arg\max_\theta log \prod_{i=1}^{m} P_G(x^i;\theta)$$

$$= arg\max_\theta \sum_{i=1}^{m} logP_G(x^i;\theta)$$

$\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$

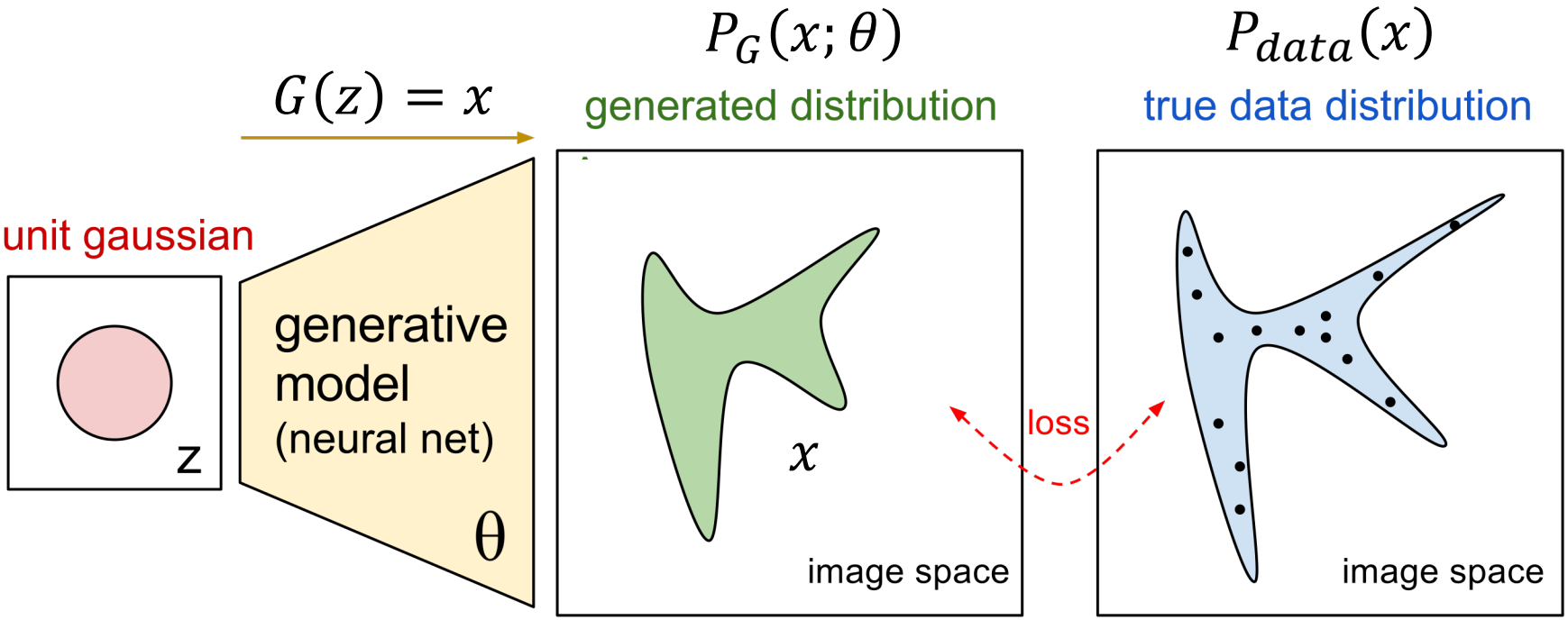
$$\approx arg\max_\theta E_{x\sim P_{data}}[logP_G(x;\theta)]$$

$$= arg\max_\theta \int_x P_{data}(x)logP_G(x;\theta)dx - \int_x P_{data}(x)logP_{data}(x)dx$$

$$= arg\min_\theta KL\left(P_{data}(x)||P_G(x;\theta)\right)$$

In Maximum Likelihood it is a KLD Kullback Leibler Divergence

Now $P_G(x;\theta)$ is a NN



$$P_G(x) = \int_z P_{prior}(z) I_{[G(z)=x]} dz$$

It is difficult to compute the likelihood.

Credits: https://blog.openai.com/generative-models/

Basic Idea of GAN

- Generator G

  Hard to learn by maximum likelihood

  - G is a function, input z, output x
  - Given a prior distribution $P_{prior}(z)$, a probability distribution $P_G(x)$ is defined by function G
- Discriminator D
  - D is a function, input x, output scalar
  - Evaluate the "difference" between $P_G(x)$ and $P_{data}(x)$
- There is a function V(G,D).

$$G^* = arg \min_G \max_D V(G,D)$$

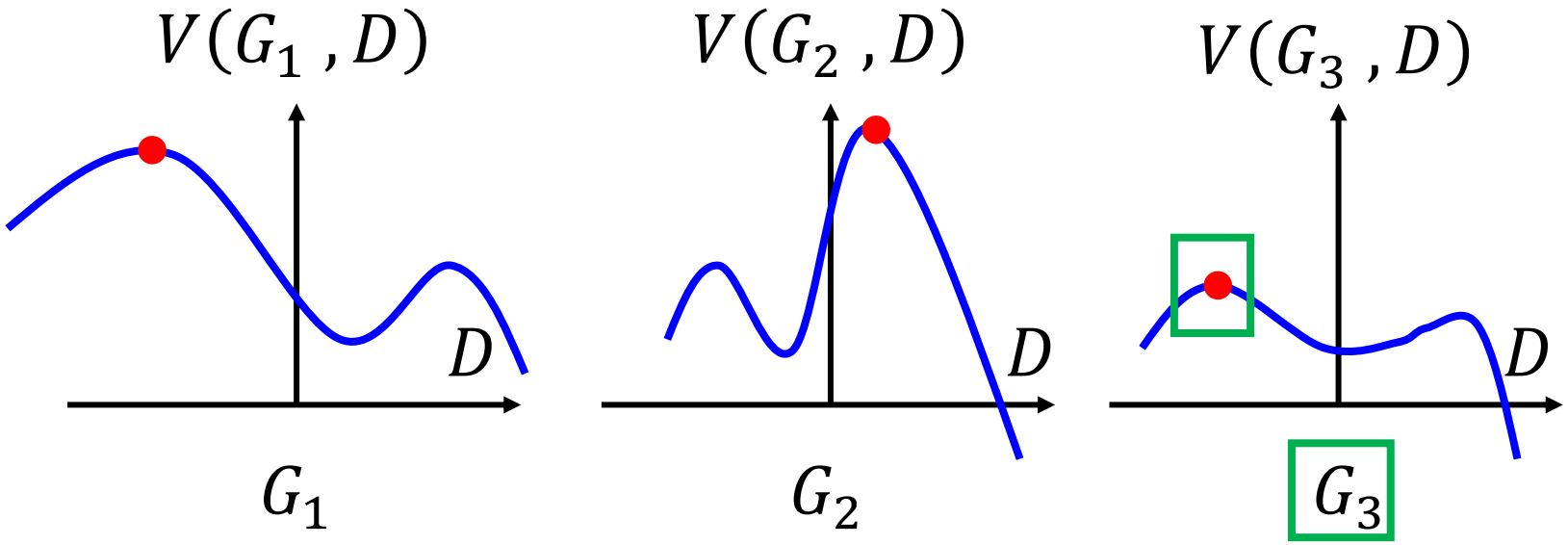
Basic Idea

$$G^* = arg \min_G \max_D V(G,D)$$

$$V = E_{x \sim P_{data}}[logD(x)] + E_{x \sim P_G}[log(1 - D(x))]$$

Given a generator G, $\max_D V(G,D)$ evaluate the "difference" between P_G and P_{data}

Pick the G defining P_G most similar to P_{data}



$$\max_D V(G,D) \qquad G^* = arg \min_G \max_D V(G,D)$$

- Given G, what is the optimal D* maximizing

$$V = E_{x\sim P_{data}}[logD(x)] + E_{x\sim P_G}\left[log\left(1-D(x)\right)\right]$$

$$= \int_x P_{data}(x)logD(x)\,dx + \int_x P_G(x)log\left(1-D(x)\right)dx$$

$$= \int_x \left[P_{data}(x)logD(x) + P_G(x)log\left(1-D(x)\right)\right]dx$$

Assume that D(x) can have any value here

- Given x, the optimal D* maximizing

$$P_{data}(x)logD(x) + P_G(x)log\left(1-D(x)\right)$$

$$\max_D V(G,D) \qquad G^* = arg\min_G \max_D V(G,D)$$

- Given x, the optimal D* maximizing

$$P_{data}(x)logD(x) + P_G(x)log\big(1-D(x)\big)$$

a D b D

- Find D* maximizing: $\mathrm{f}(D) = \mathrm{a}log(D) + blog(1-D)$

$$\frac{d\mathrm{f}(D)}{dD} = a\times\frac{1}{D} + b\times\frac{1}{1-D}\times(-1) \quad = 0$$

$$a\times\frac{1}{D^*} = b\times\frac{1}{1-D^*} \qquad a\times(1-D^*) = b\times D^*$$

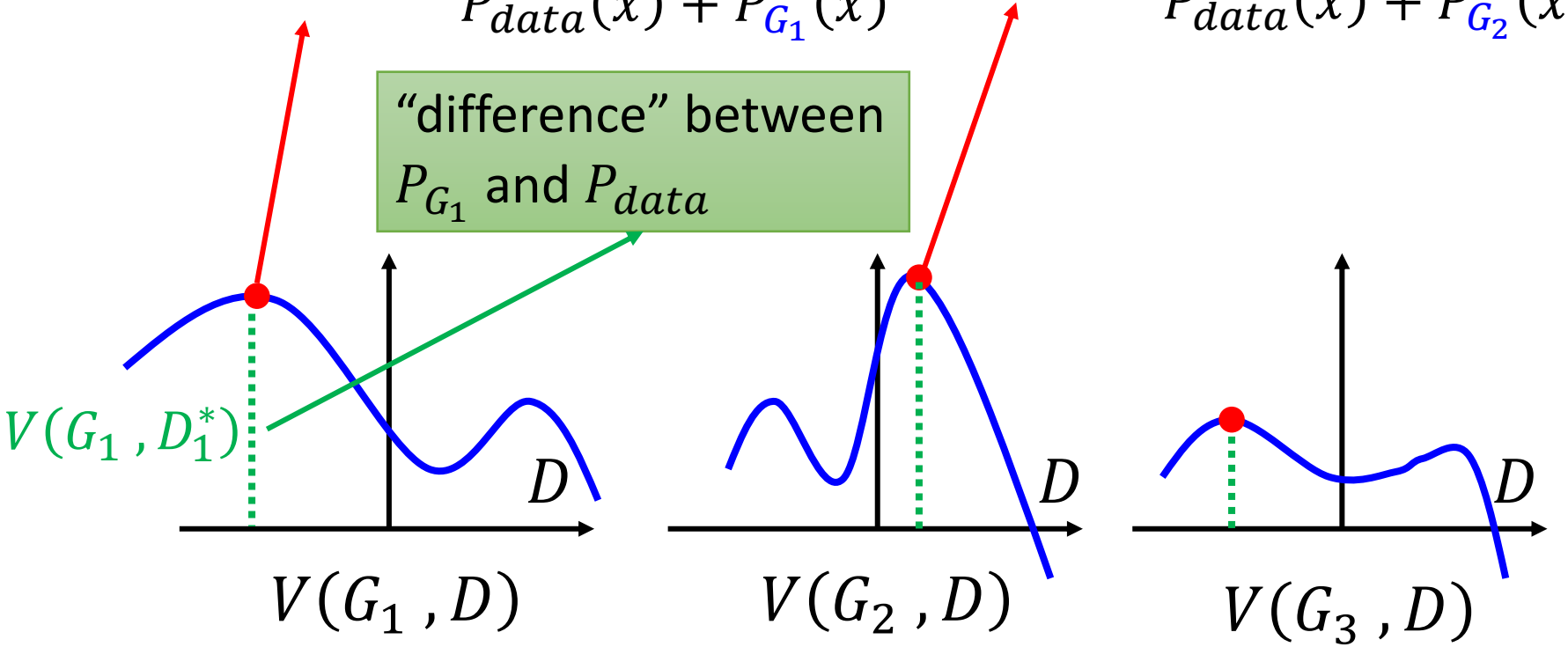
$$a - aD^* = bD^*$$

$$D^* = \frac{a}{a+b} \Rightarrow \qquad D^*(x) = \frac{P_{data}(x)}{P_{data}(x)+P_G(x)}$$

$0 <$ < 1

$$\max_D V(G,D) \qquad G^* = arg\min_G \max_D V(G,D)$$

$$D_1^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_{G_1}(x)} \qquad D_2^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_{G_2}(x)}$$



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

$$V = E_{x\sim P_{data}}[logD(x)] + E_{x\sim P_G}[log(1-D(x))]$$

$$\max_D V(G,D) = V(G,D^*) \qquad D^*(x) = \frac{P_{data}(x)}{P_{data}(x)+P_G(x)}$$

$$= E_{x\sim P_{data}}\left[log\frac{P_{data}(x)}{P_{data}(x)+P_G(x)}\right] + E_{x\sim P_G}\left[log\frac{P_G(x)}{P_{data}(x)+P_G(x)}\right]$$

$$= \int_x P_{data}(x) log \frac{\frac{1}{2}P_{data}(x)}{\frac{P_{data}(x)+P_G(x)}{2}} dx + \int_x P_G(x) log \frac{\frac{1}{2}P_G(x)}{\frac{P_{data}(x)+P_G(x)}{2}} dx$$

$$+2log\frac{1}{2} \quad -2log2$$

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

$$\mathrm{JSD}(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M)$$

$$M = \frac{1}{2}(P+Q)$$

$$\max_D V(G,D) = V(G,D^*) \qquad D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

$$= -2log2 + \int_x P_{data}(x) log \frac{P_{data}(x)}{\big(P_{data}(x) + P_G(x)\big)/2} dx$$

$$+ \int_x P_G(x) log \frac{P_G(x)}{\big(P_{data}(x) + P_G(x)\big)/2} dx$$

$$= -2\log2 + \mathrm{KL}\left(\mathrm{P_{data}(x)} || \frac{\mathrm{P_{data}(x) + P_G(x)}}{2}\right)$$

$$+\mathrm{KL}\left(\mathrm{P_G(x)} || \frac{\mathrm{P_{data}(x) + P_G(x)}}{2}\right)$$

$$= -2log2 + 2JSD\big(P_{data}(x) || P_G(x)\big)$$ Jensen-Shannon divergence

In the end ……

$$V = E_{x\sim P_{data}}[logD(x)] + E_{x\sim P_G}[log(1-D(x))]$$

- Generator G, Discriminator D
- Looking for G* such that

$$G^* = arg \min_G \max_D V(G,D)$$

- Given G, $\max_D V(G,D)$ 0 < < log 2

$$= -2log2 + 2JSD(P_{data}(x)||P_G(x))$$

- What is the optimal G?

$$P_G(x) = P_{data}(x)$$

Generative models

Outline

1. Preview: Auto-Encoders, VAE
2. Generative models with GAN

3. **GAN architectures for image generation**

Recall Algo GAN

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.

Which architectures for G and D?

Basic Archi for G and D and expe

Models

G and D fully connected nets
or convolutional for D, (de)convolutional for G (as seen for segmentation nets)
ReLU and/or sigmoids, dropout

Datasets

MNIST, Toronto Face Database, CIFAR-10

GAN - Evaluation

- Approximate p_g by fitting a Gaussian Parzen window on the generated images.
- Cross-validate σ to maximize likelihood of validation set
- Compute the likelihood of the test set

Evaluation not trivial, can be done using generated images as inputs for deep nets => inception scores

GAN - Qualitative results 1/2

Figure: Right col nearest from dataset. a) MNIST, b) TFD, c) CIFAR-10 (fully connected), d) CIFAR-10 (convolutional D, deconvolutional G)

GAN - Qualitative results 2/2

Figure: Linear interpolation between 2 points in z space

- **Advantages:**
  - Computational advantages (no complex likelihood inference)
  - Can represent sharper distributions
- **Disadvantages:**
  - G and D must be well synchronized for the algorithm to converge correctly

GAN architectures

- How to improve result quality?
  - Spatial resolution

    $\Rightarrow$Cascade of GAN
  - Object quality

    => Progressive growing of spatial resolution in G

Laplacian Pyramid GANs (LAPGANs)

- GANs do not work well for complex / high level / natural images.
- Idea: decompose the generation in successive tasks using Laplacian Pyramid (of GANs)

Let $d(I)$ and $u(I)$ be down-sampling and up-sampling operations.
Gaussian pyramid:

$$\mathcal{G}(I) = [I_0, I_1, ..., I_K], I_k = d^{(k)}(I)$$

Laplacian pyramid:

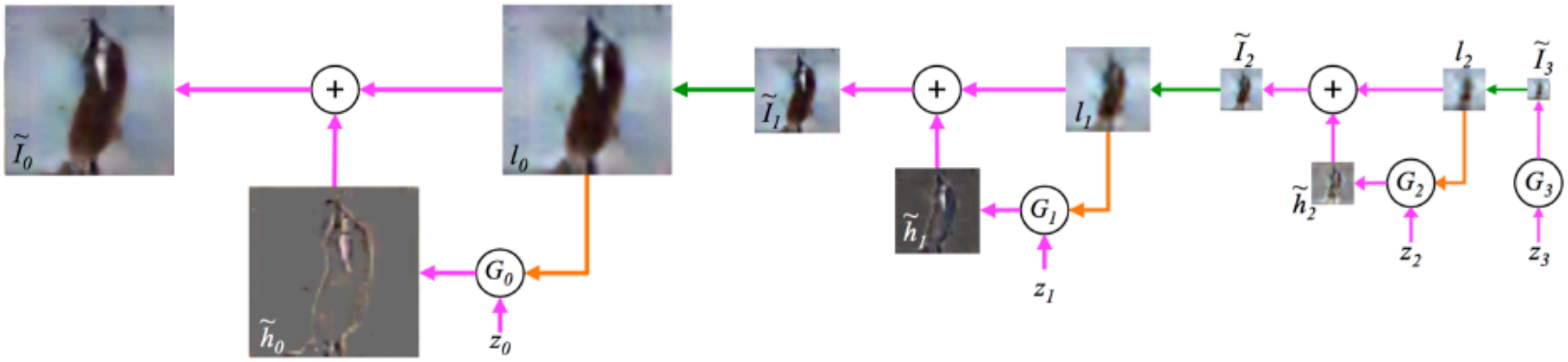
$$h_k = \mathcal{L}_k(I) = I_k - u(I_{k+1})$$

Reconstruction:

$$I_k = u(I_{k+1}) + h_k$$

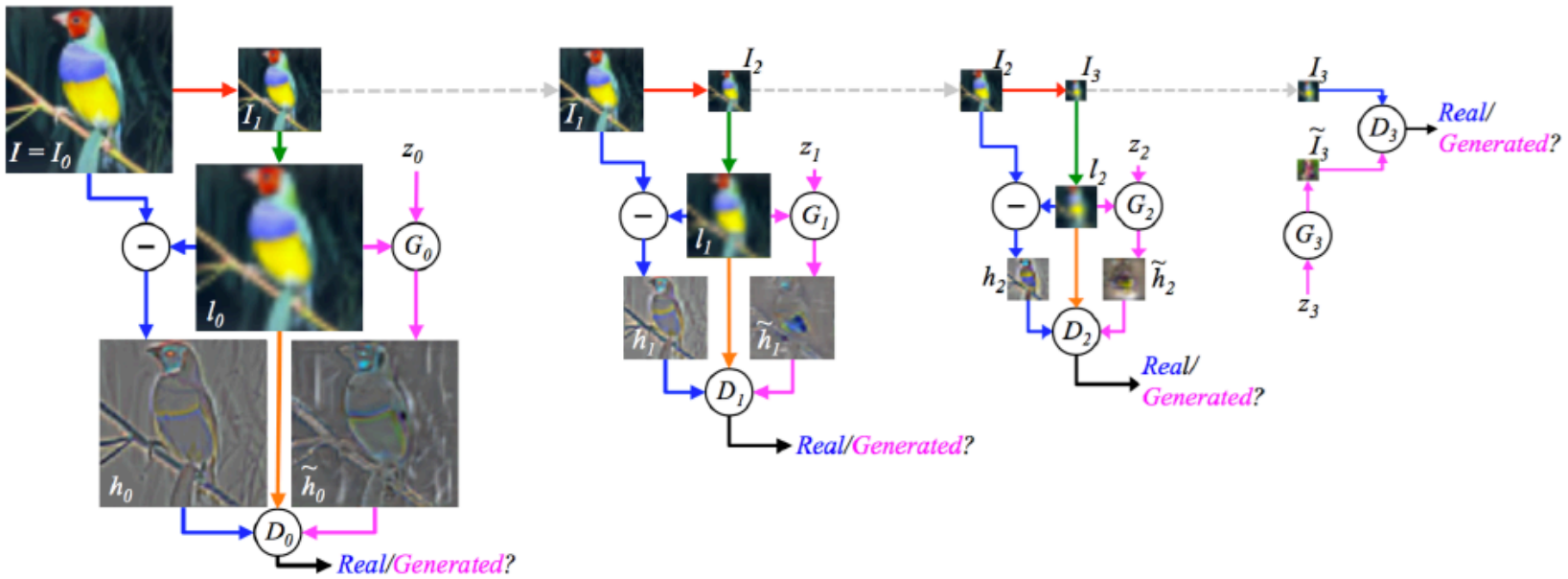
LAPGAN model - sampling

- Set of generative convnets: $G_0, \ldots, G_K$
- Generated details: $\tilde{h}_k = G_k(z_k, u(\tilde{I}_{k+1}))$
- Reconstructed image: $\tilde{I}_k = u(\tilde{I}_{k+1}) + \tilde{h}_k \ (\tilde{I}_{K+1} = 0)$



LAPGAN model - training

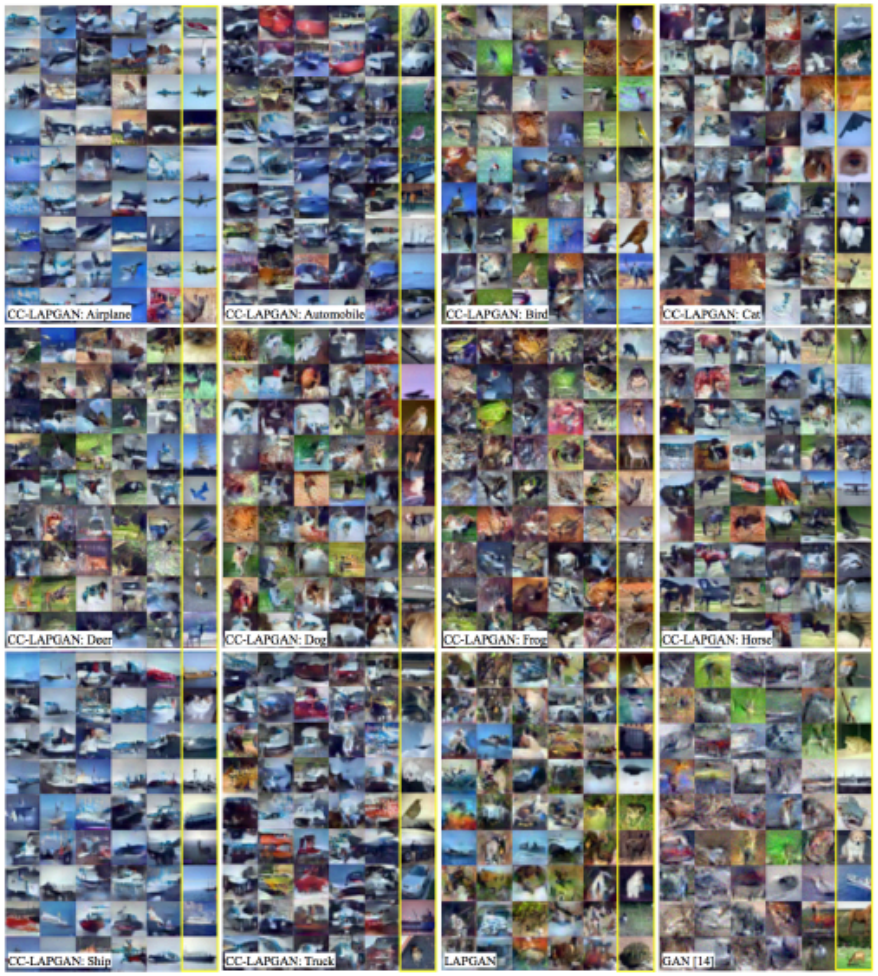
- Low-pass version of l_0: $l_0 = u(d(l_0))$
- Either:
  - High-pass version of l_0: $h_0 = l_0 - l_0$
  - Generate $\tilde{h}_0 = G_0(z_0, l_0)$
- Forward $D_0(l_0 + h_0$ or $\tilde{h}_0)$
- Backward D_0 and G_0
- G_K and D_K are trained as a simple GAN



LAPGAN model - Experiments

- **Datasets:** CIFAR-10, STL
- **Initial scale:**
  - G_K and D_K have 2 hidden layers and ReLU
  - $z_K \sim U_{[-1,1]^{100}}$
  - Trained as GAN
- **Subsequent scales:**
  - G_k and D_k convnets with 3 and 2 layers
  - $z_k \sim U_{[-1,1]^{|I_k|}}$ ("color" layer)
  - Trained as CGAN

LAPGAN model - Results - CIFAR

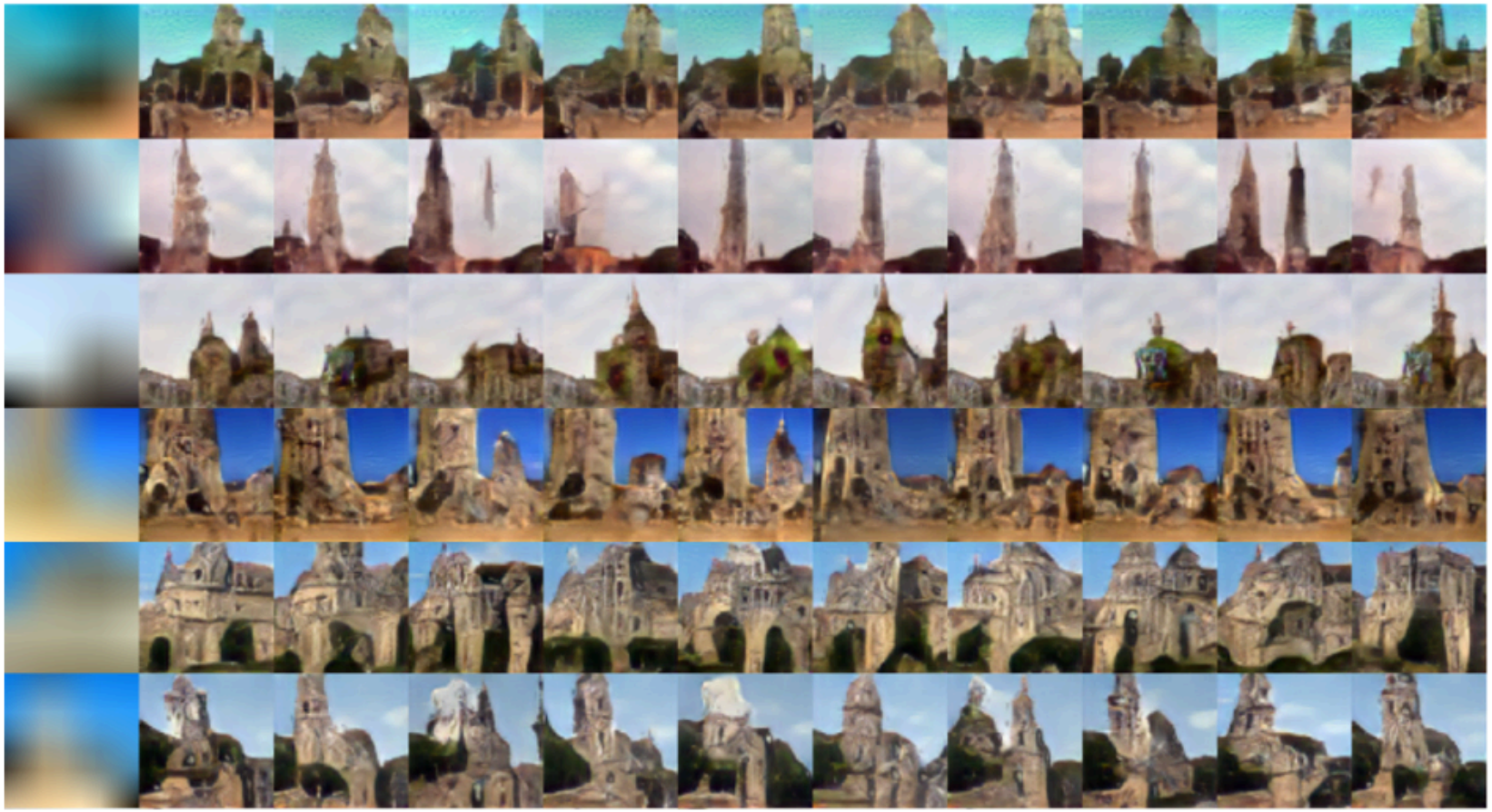


LAPGAN model - Results - STL

(a) (b)

Figure 4: STL samples: **(a)** Random 96x96 samples from our LAPGAN model. **(b)** Coarse-to-fine generation chain.

LAPGAN model - Results - LSUN



LAPGAN

- Good idea but Genetor structure too weak
- Recent improvement: Pix2PixHD described later

Progressive growing of spatial resolution in G

Deep Convolutional GANs (DCGANs)

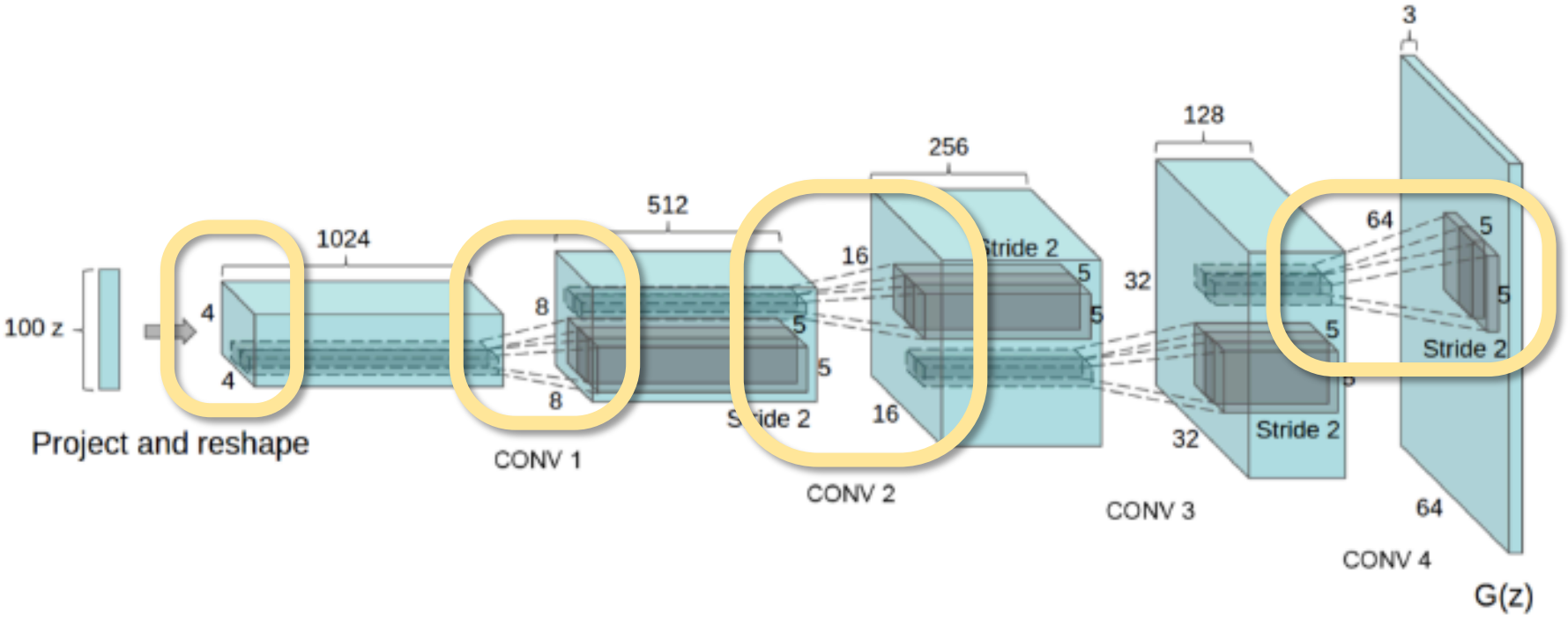
GANs are hard to scale => Identify a family of architectures that gives stable training

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator)
- Use batchnorm in both the generator and the discriminator
- Remove fully connected hidden layers for deeper architectures
- Use ReLU activation in generator for all layers except for the output, which uses Tanh
- Use LeakyReLU activation in the discriminator for all layers

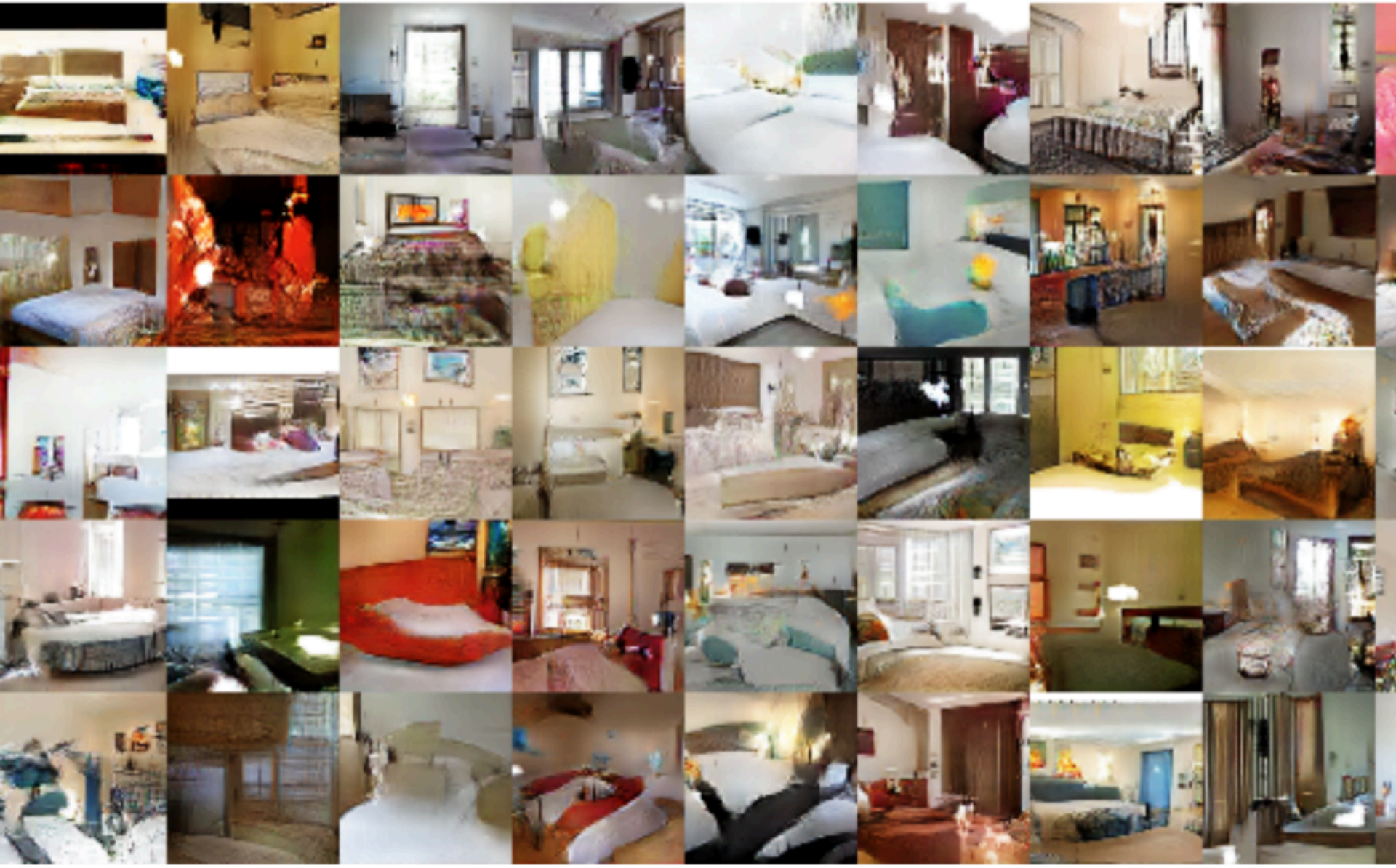
Progressive growing of spatial resolution in G: DCGAN

Upsampling step by step

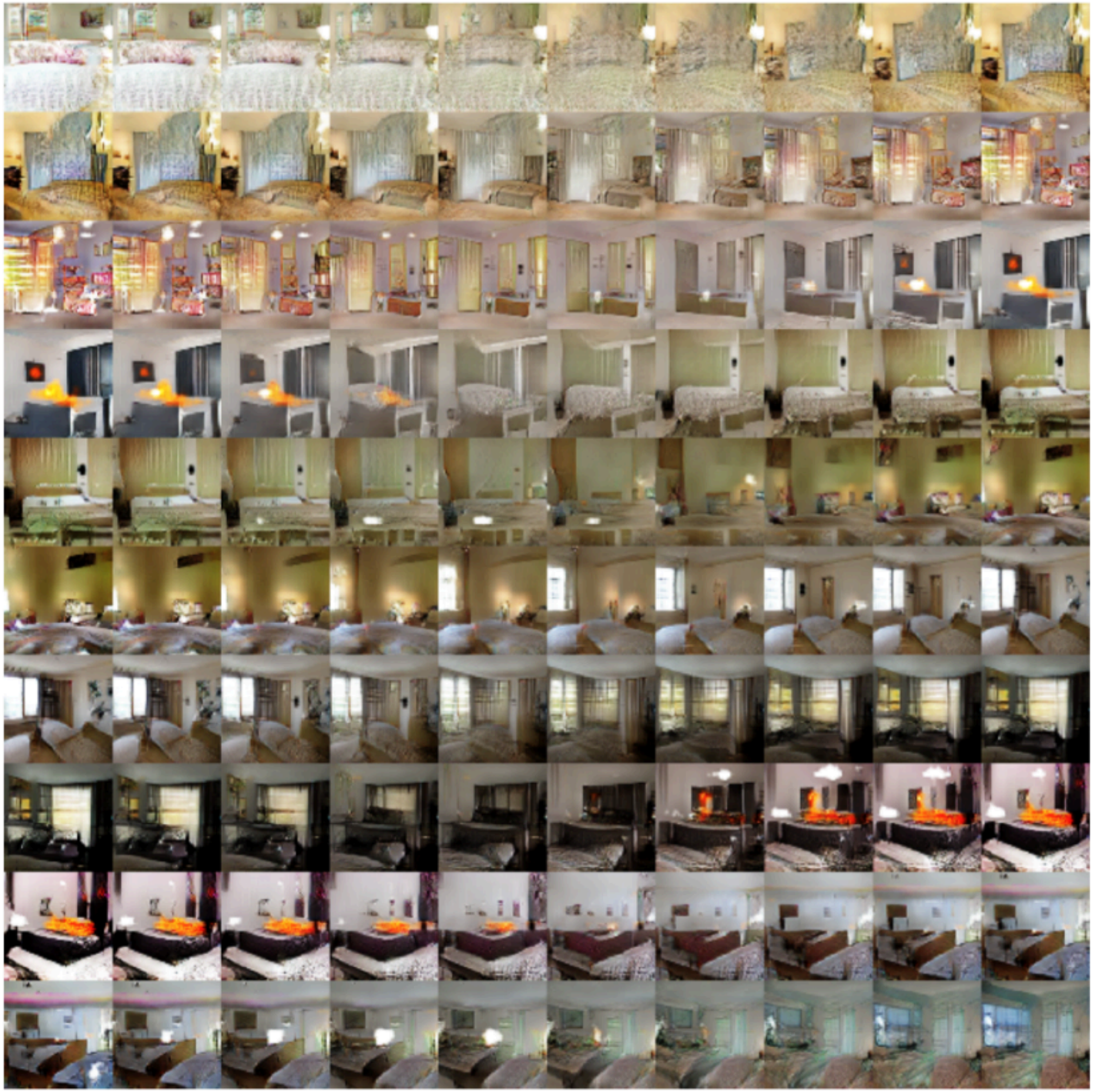
Combine with convolutional layers



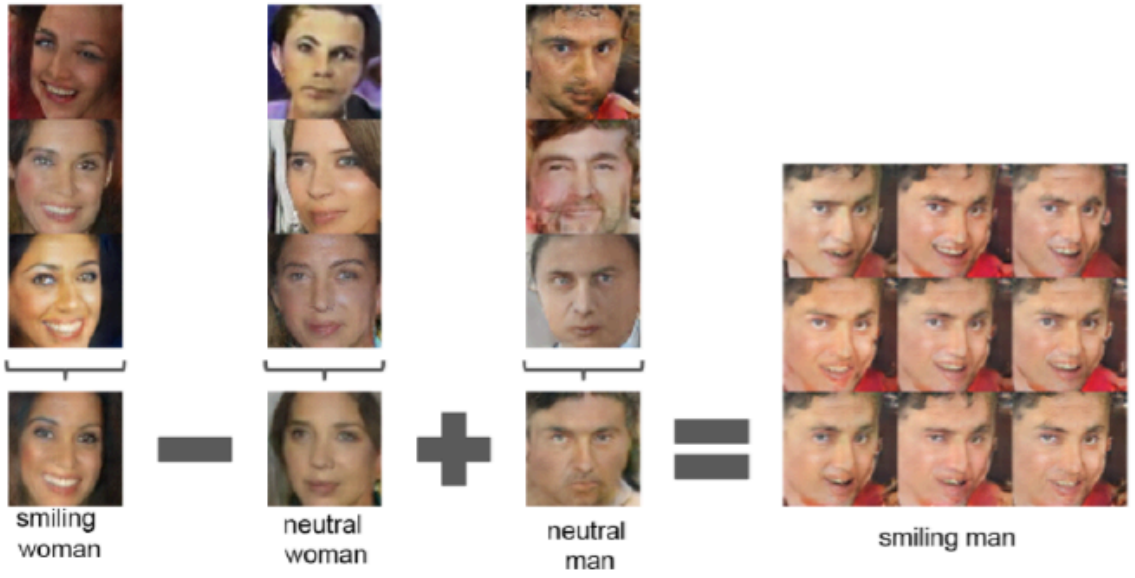
DCGAN - Results - generated bedrooms

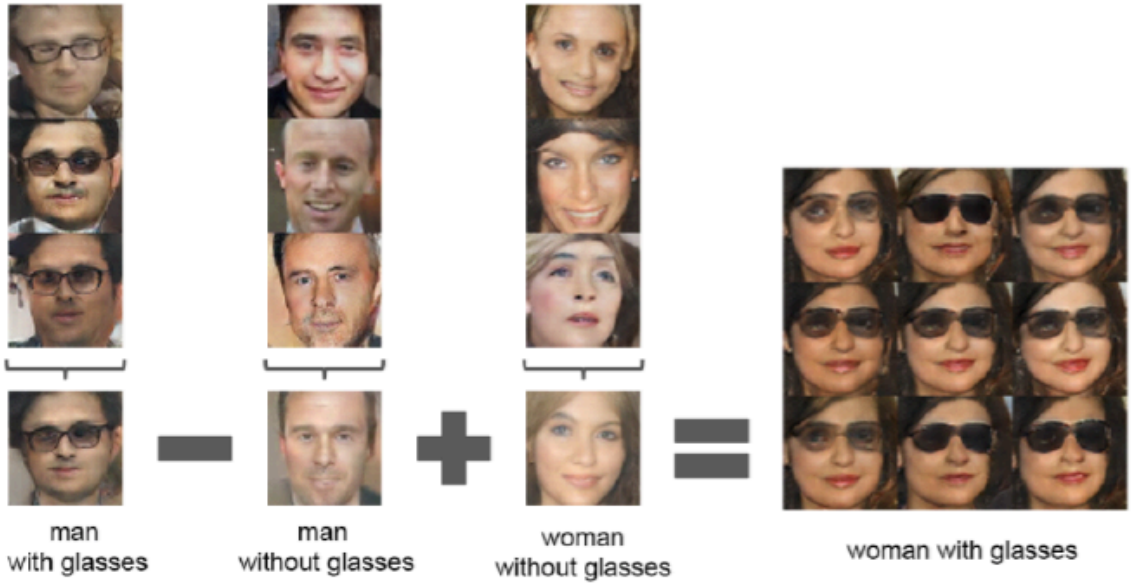


DCGAN Walking in latent



DCGAN - Arithmetics in Z space





Conditional GAN

Motivation

c: train

Prior distribution z

Generator

Text

Generator

Image

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

Generator

Image

Generator

Image

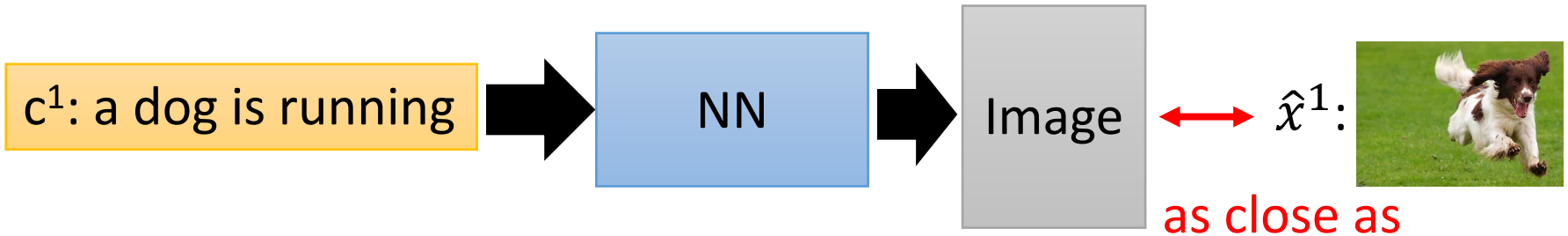
Generator

Conditional GAN

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

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

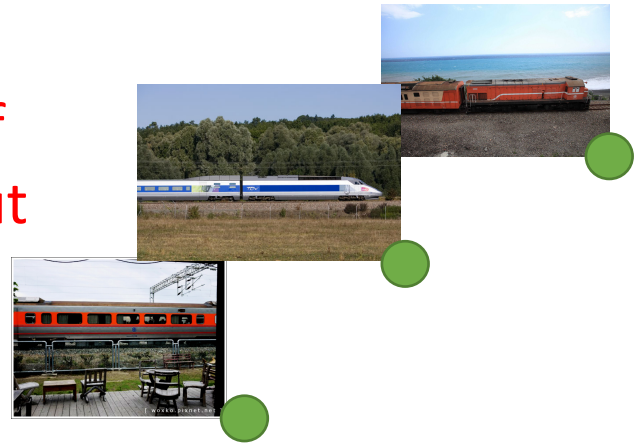
- ***Text to image*** by traditional supervised learning



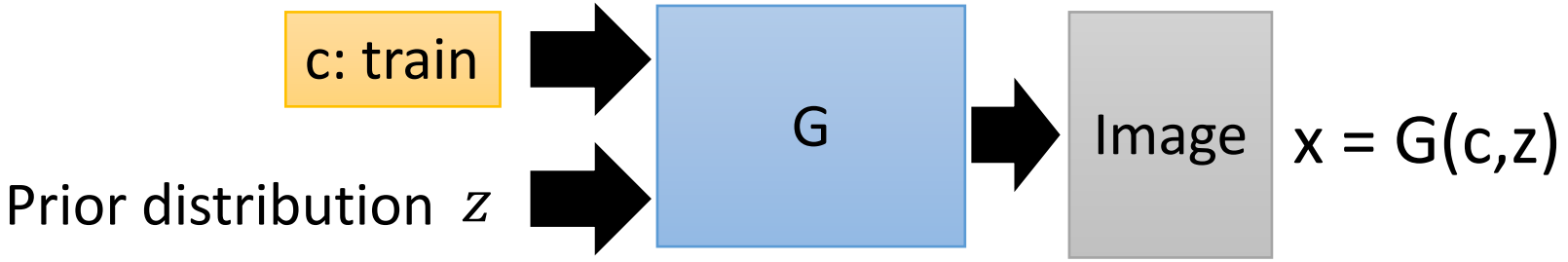
Text: "train"

Target of NN output

A blurry image!



Conditional GAN



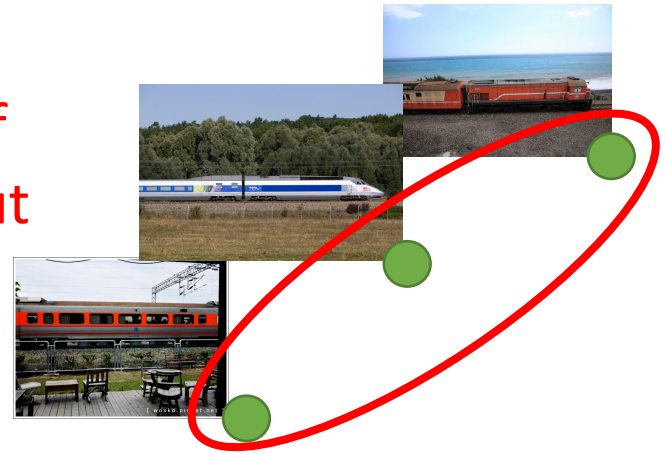
It is a distribution.

Approximate the distribution below

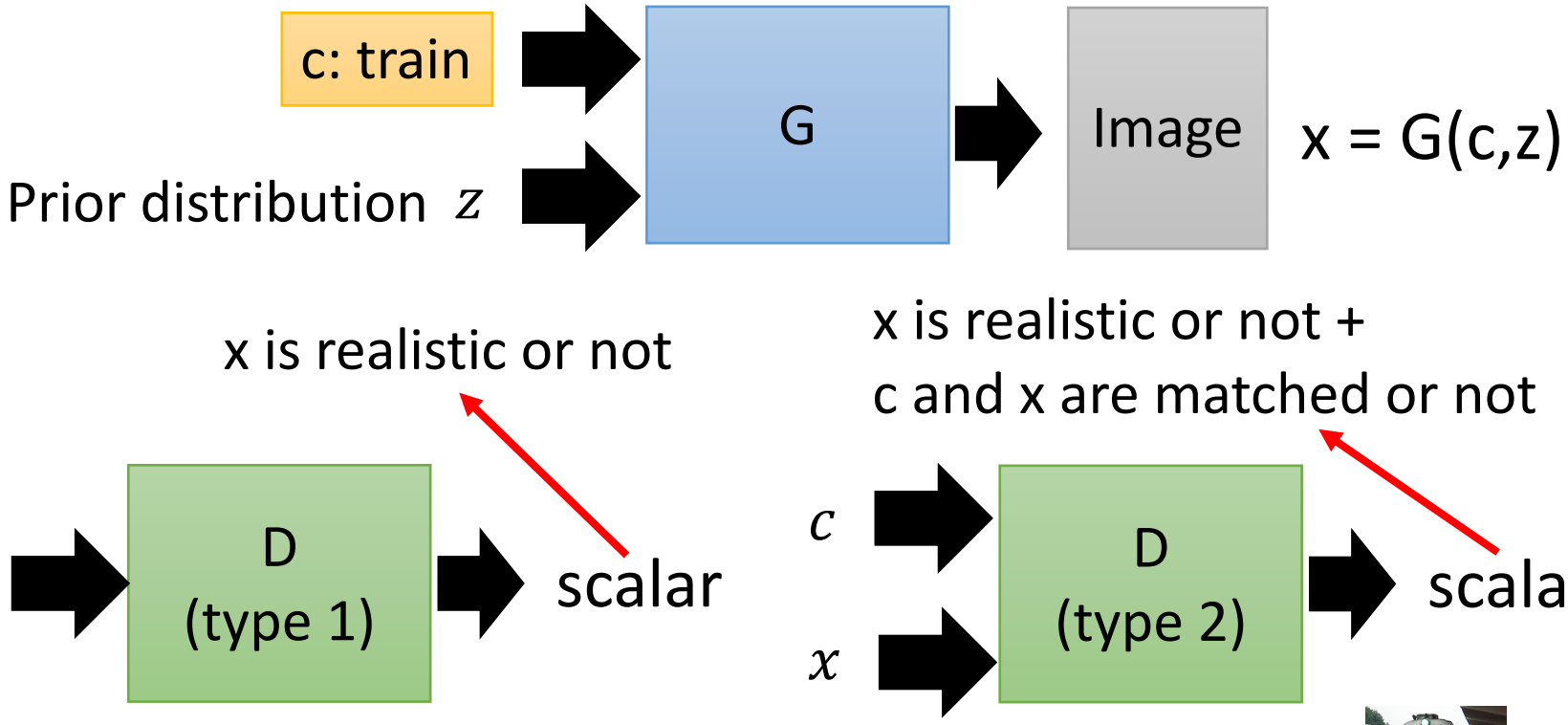
Text: "train"

Target of NN output

A blurry image!

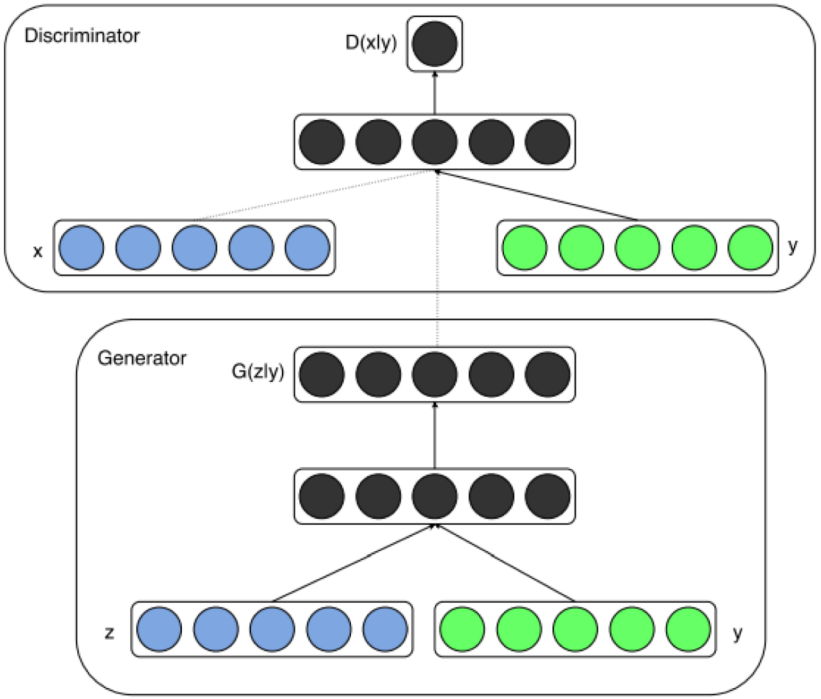


Conditional GAN



Conditional GAN (CGAN model)

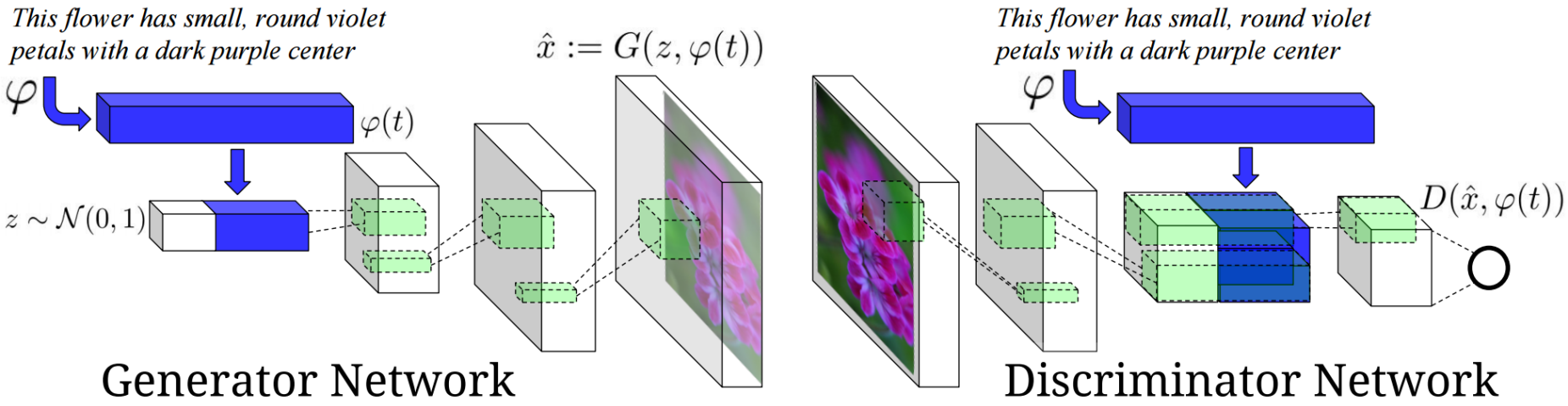
$$\min_G \max_D \left(\mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}(\mathbf{x},\mathbf{y})} \left[\log D(\mathbf{x},\mathbf{y}) \right] + \mathbb{E}_{\mathbf{y}\sim p_{\mathbf{y}},\, \mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})} \left[\log(1 - D(G(\mathbf{z},\mathbf{y}),\mathbf{y})) \right] \right)$$



Applications

- Text2Im
- Im2Im
- Unpaired transformation
- …

Text2Image



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

[Reed et al. ICML 2016]

Text2Image

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

[Reed et al. ICML 2016]

Text2Image

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	

StackGAN: similar idea with LapGan

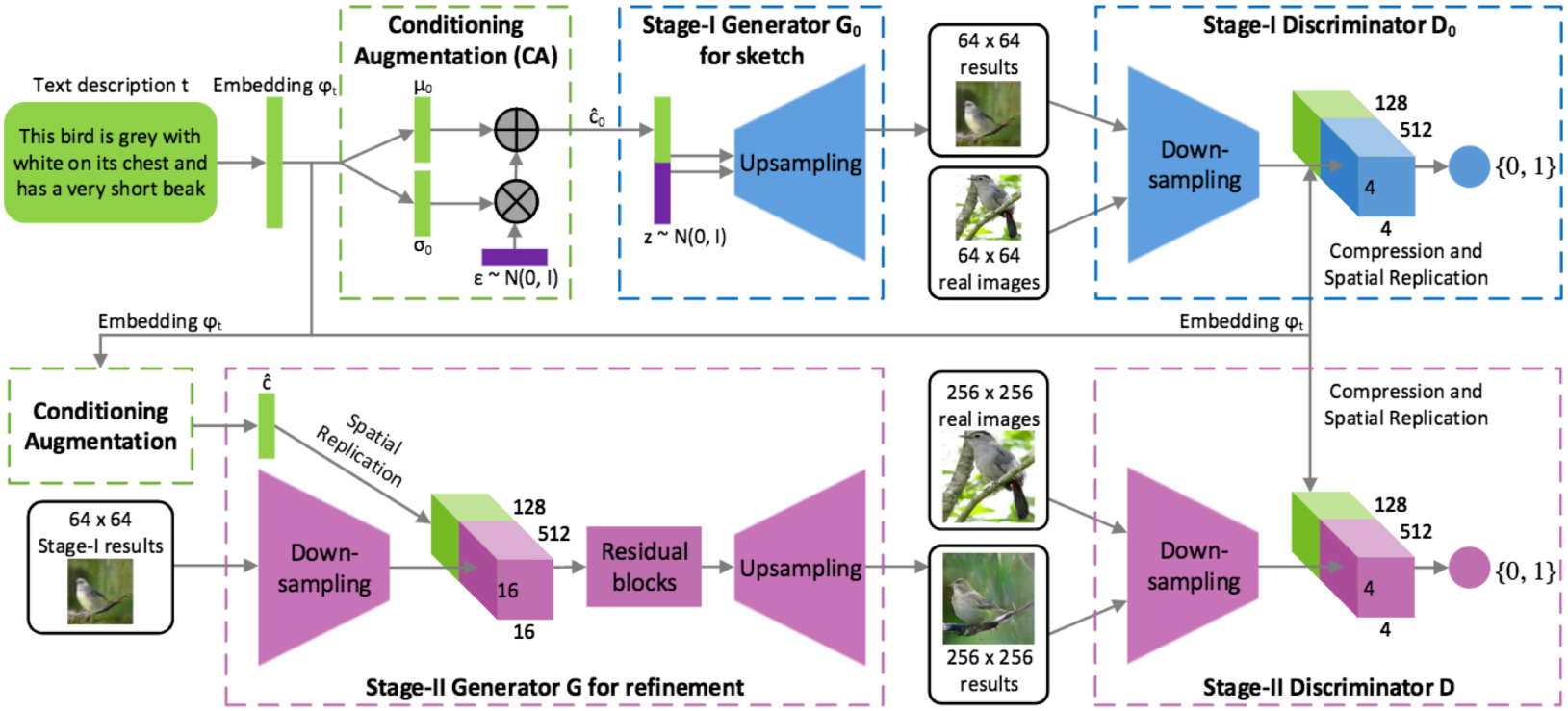
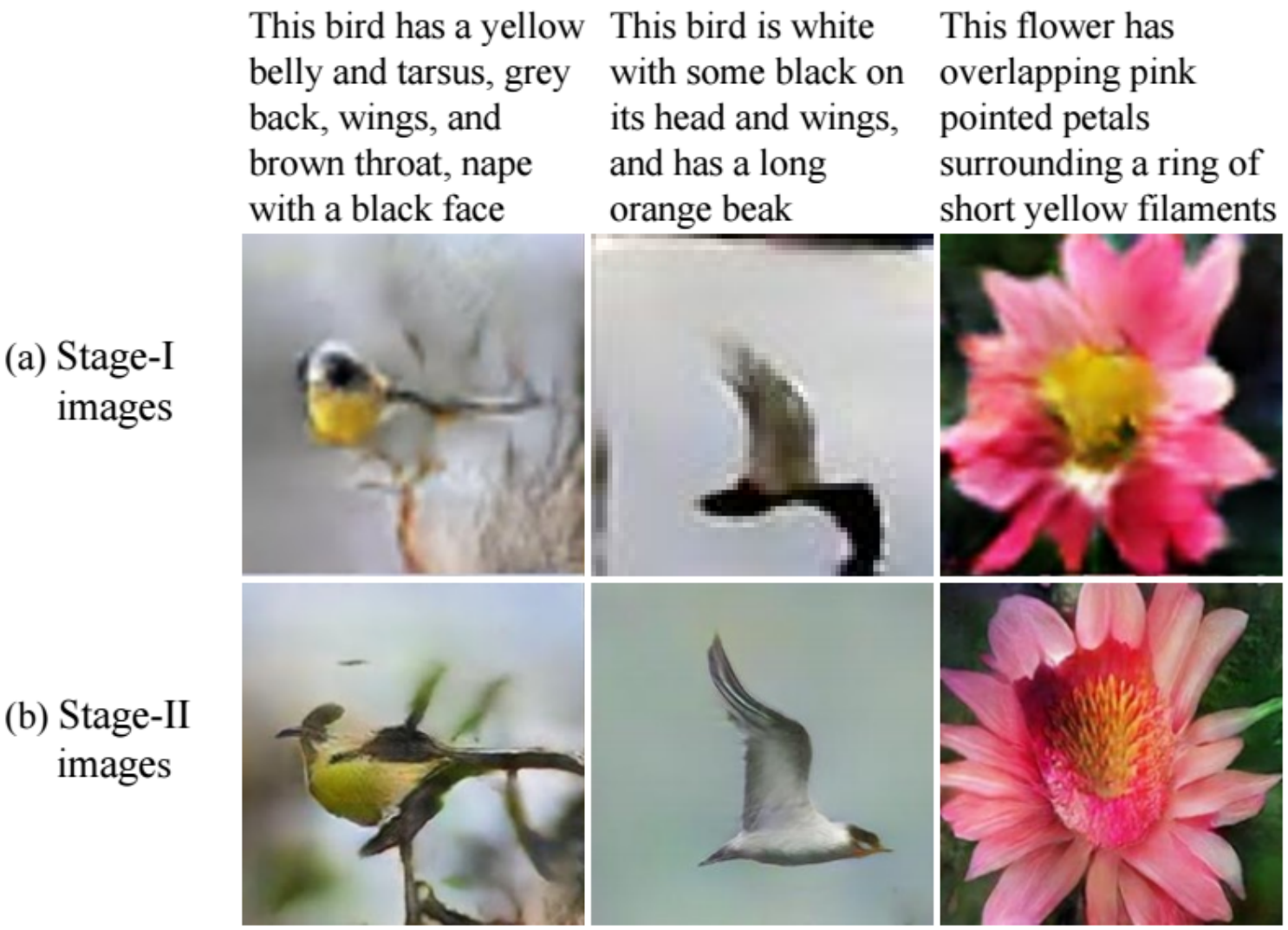


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects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

[Zhang et al. 2016]

StackGAN

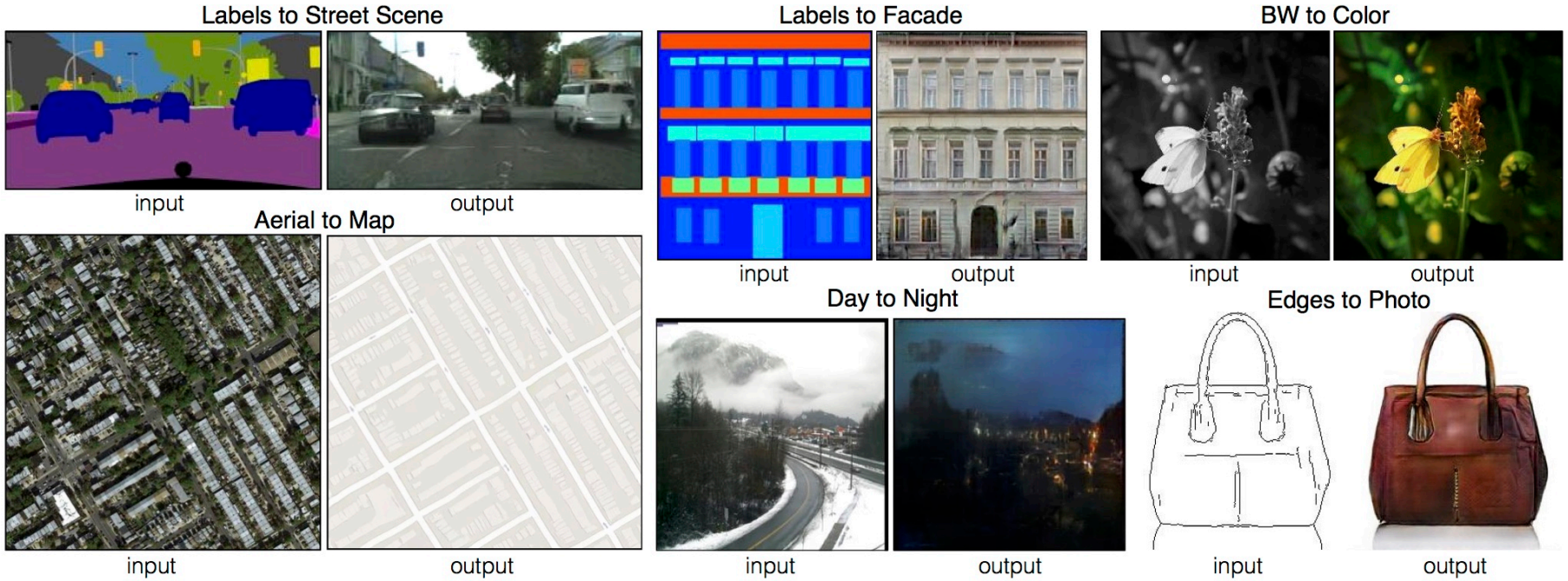


[Zhang et al. 2016]

Text2Image

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	

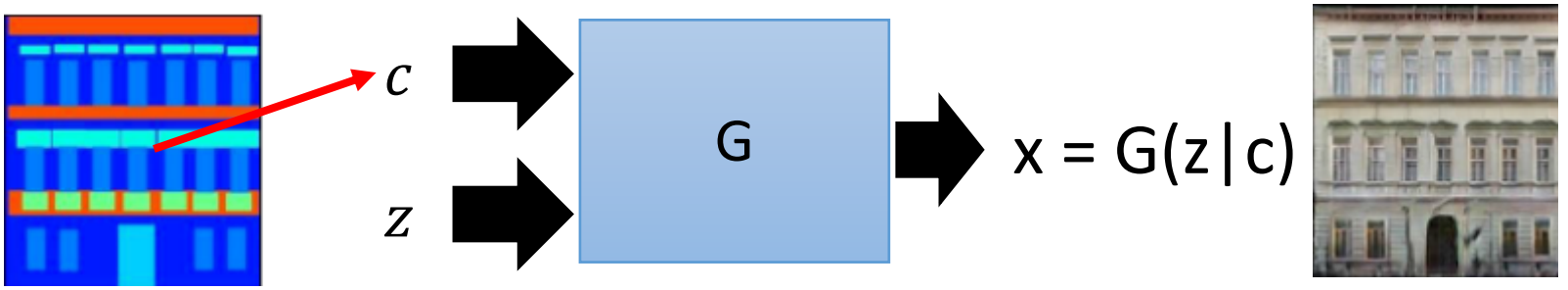
Image-to-Image Translation pix2pix

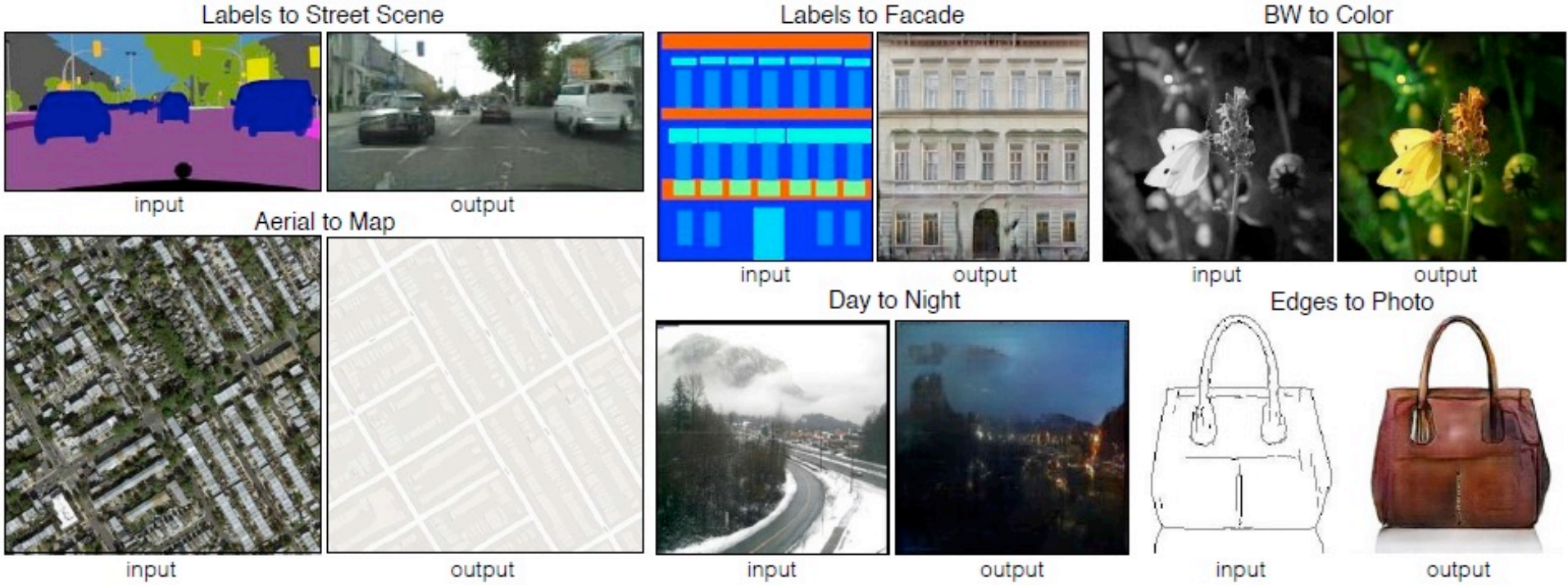


- Conditioned on an image of different modality
- No need to specify the loss function

[Isola et al. CVPR 2017]

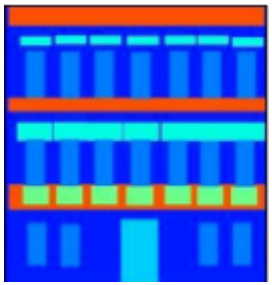
Image-to-image pix2pix



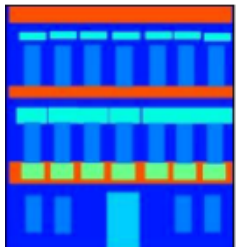


https://arxiv.org/pdf/1611.07004

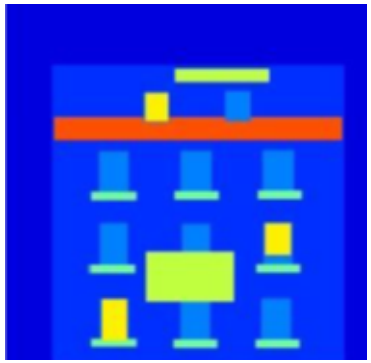
Image-to-image pix2pix

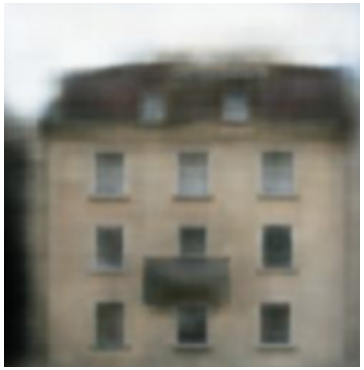



- Traditional supervised approach



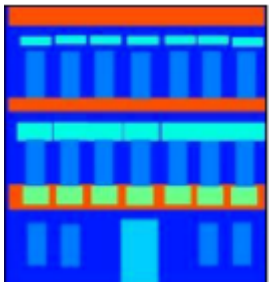
Testing:


input


close

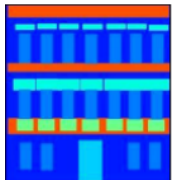
It is blurry because it is the average of several images.

Image-to-image



- Conditional GAN

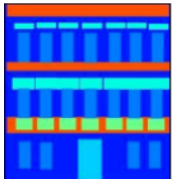
Supplementary loss (close)



z

G

Image

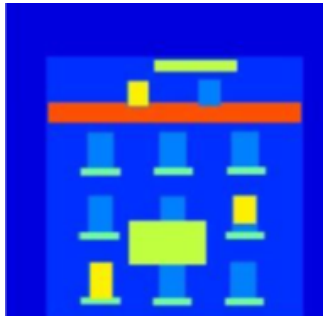


Image

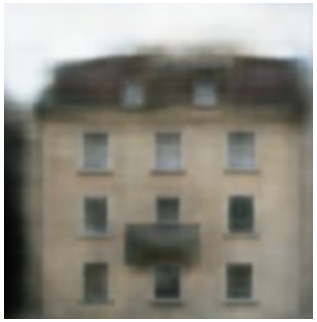
D

scalar

Testing:



input



close

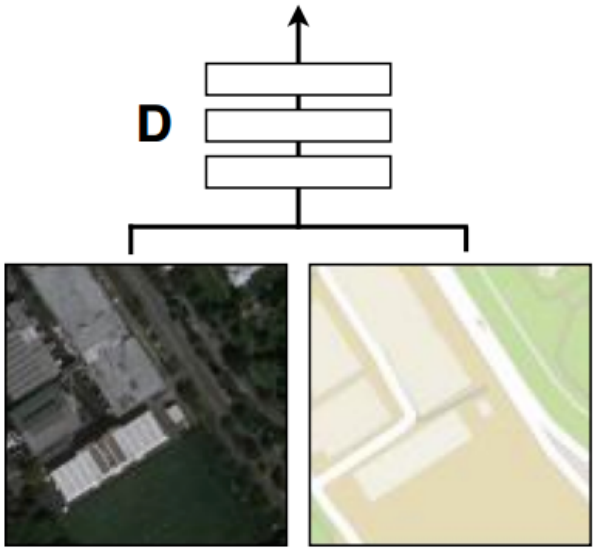
GAN

GAN + close

GT

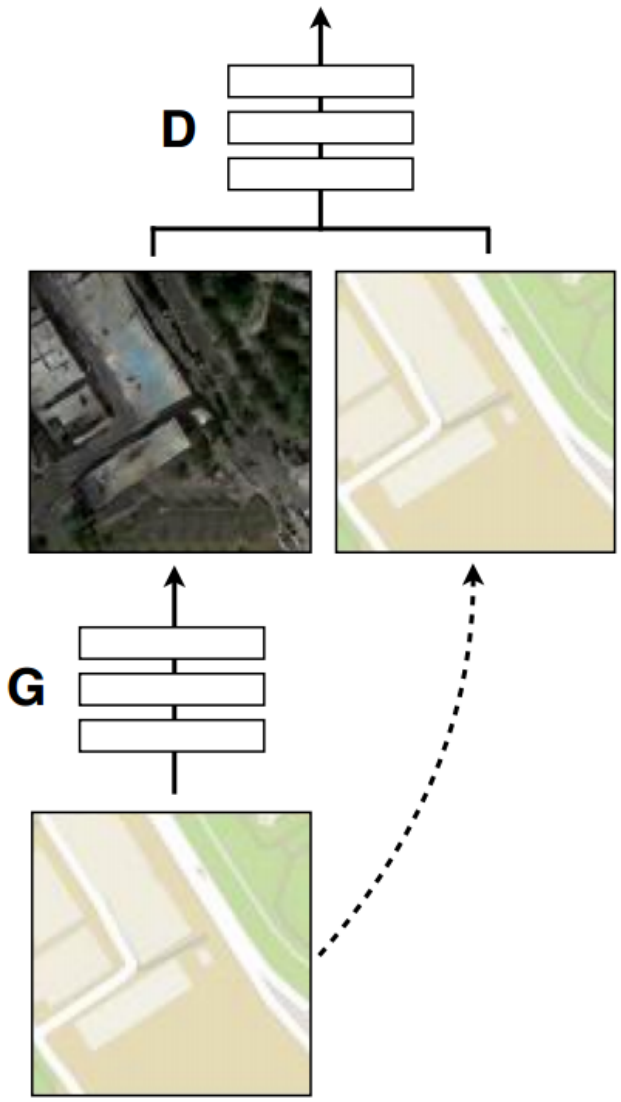
Positive examples

Real or fake pair?



Negative examples

Real or fake pair?

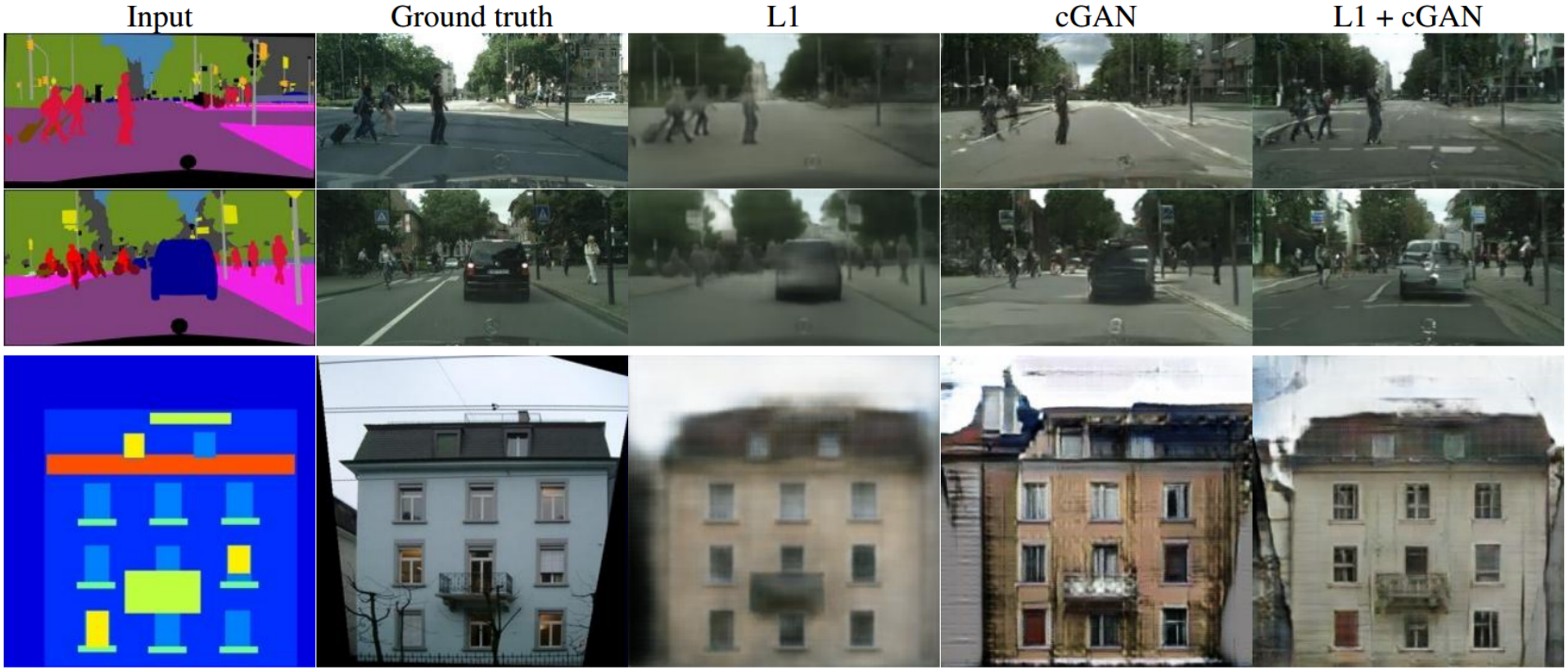


G tries to synthesize fake images that fool **D**

D tries to identify the fakes

[Isola et al. CVPR 2017]

Label2Image



[Isola et al. CVPR 2017]

Edges2Image

[Isola et al. CVPR 2017]

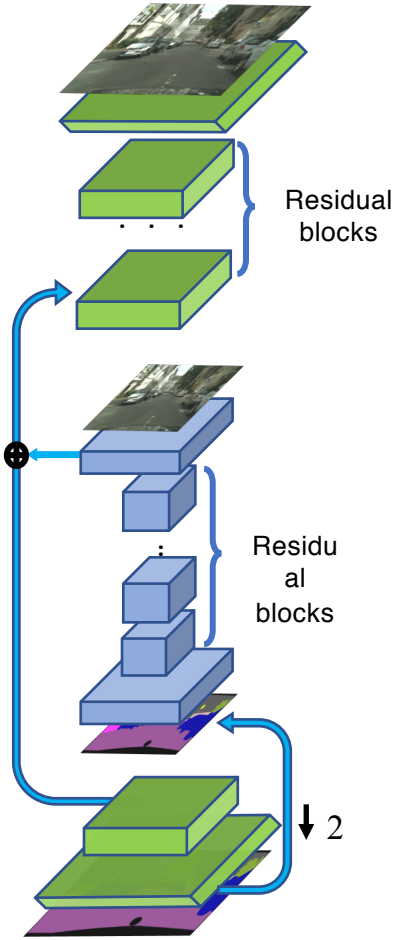
Pix2pixHD [CVPR 2018]

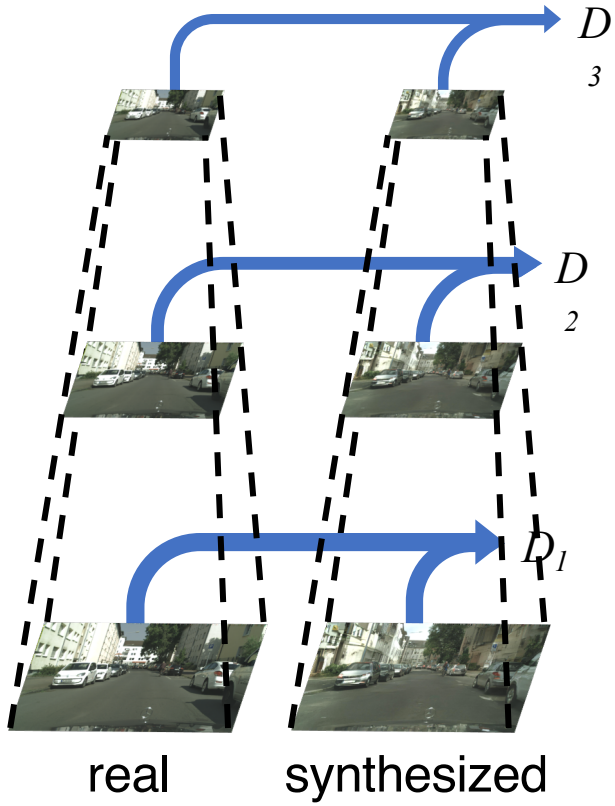
High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs

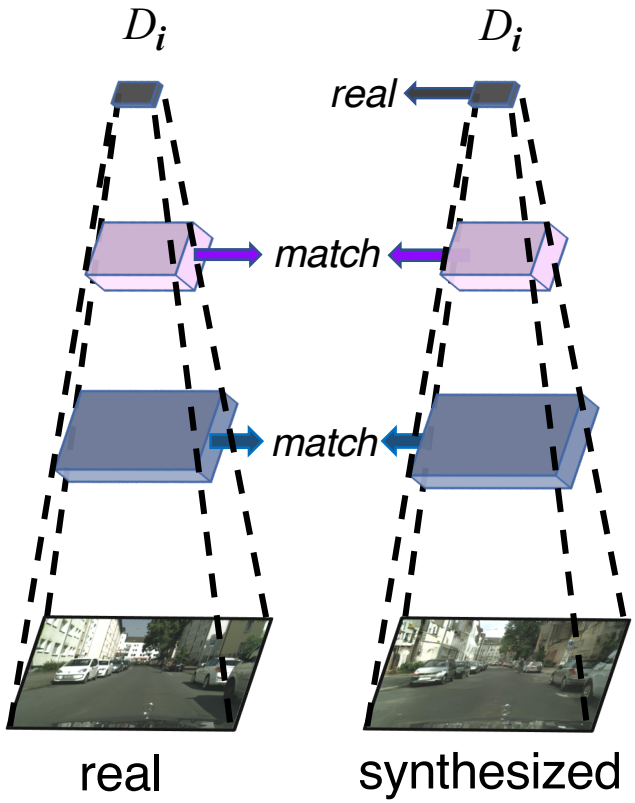
Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, Bryan Catanzaro

Pix2pixHD [CVPR 2018]

Coarse-to-fine Generator ***Multi-scale Discriminators*** ***Robust Objective***



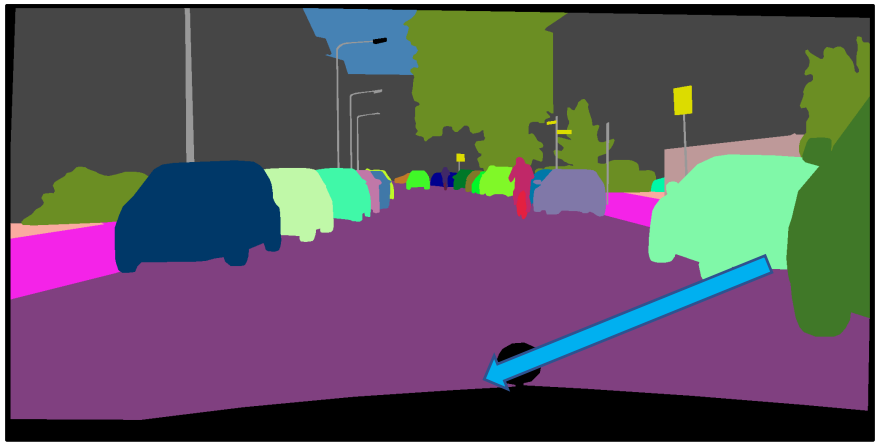


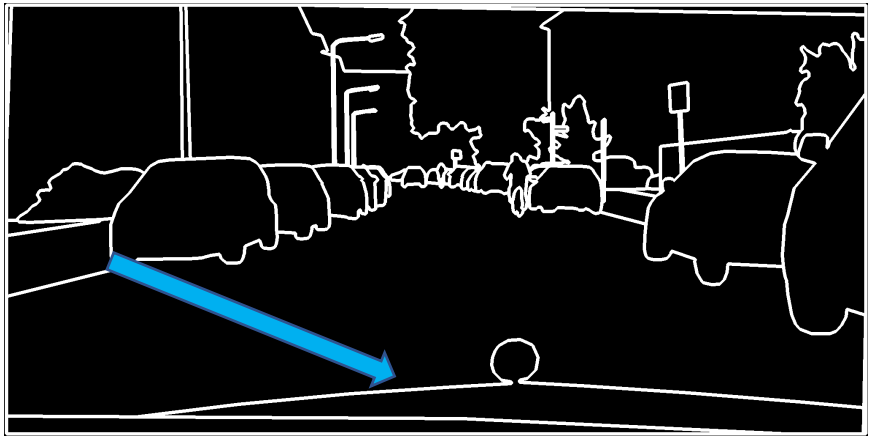


Pix2pixHD [CVPR 2018]

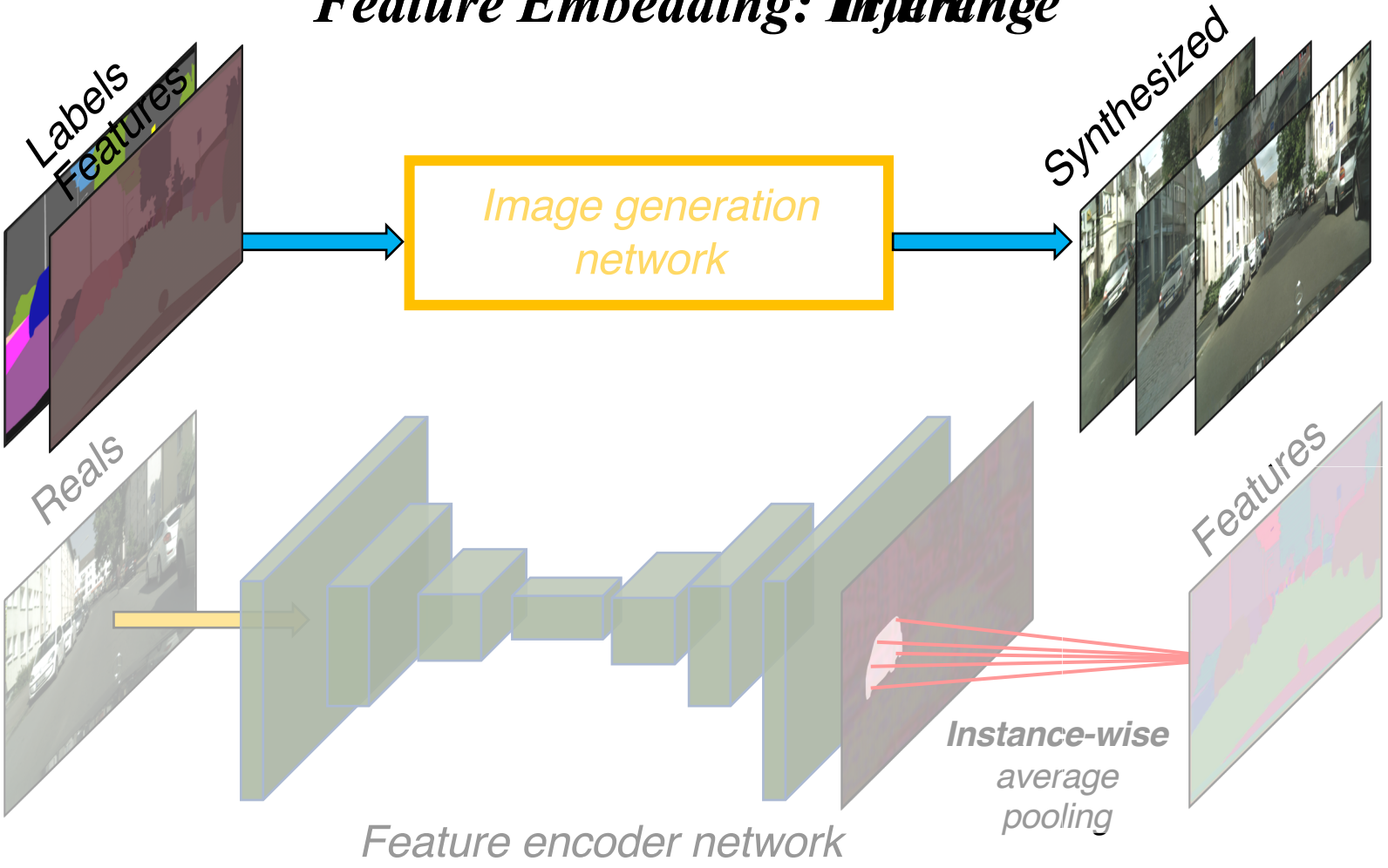
Our Method

- Boundary improvement





Feature Embedding: Training



Semantic Map

pix2pix

CRN

Ours

Unpaired Transformation

Unpaired Transformation - Cycle GAN, Disco GAN

paired data

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/

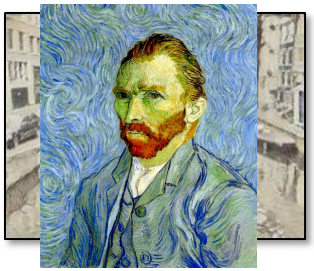
Domain X

Domain Y

Domain X

$G_{X \to Y}$

Become similar to domain Y



Not what we want

ignore input

D_Y → scalar

Input image belongs to domain Y or not

Domain Y

Cycle GAN

Domain X

Domain Y

as close as possible

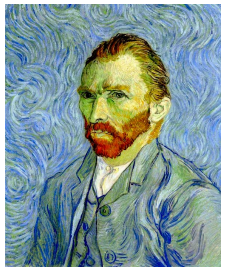
$G_{X \to Y}$

$G_{Y \to X}$

Lack of information for reconstruction

D_Y

scalar



Input image belongs to domain Y or not

Domain Y

Cycle GAN

as close as possible

$G_{X \to Y}$

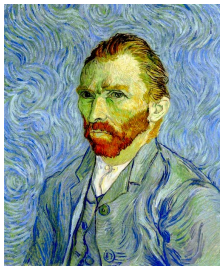
$G_{\mathrm{Y} \to \mathrm{X}}$

scalar: belongs to domain X or not

D_X

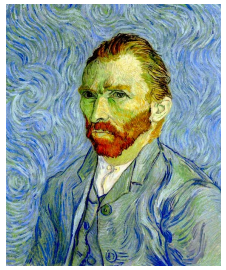
D_Y

scalar: belongs to domain Y or not



$G_{\mathrm{Y} \to \mathrm{X}}$

$G_{X \to Y}$



as close as possible

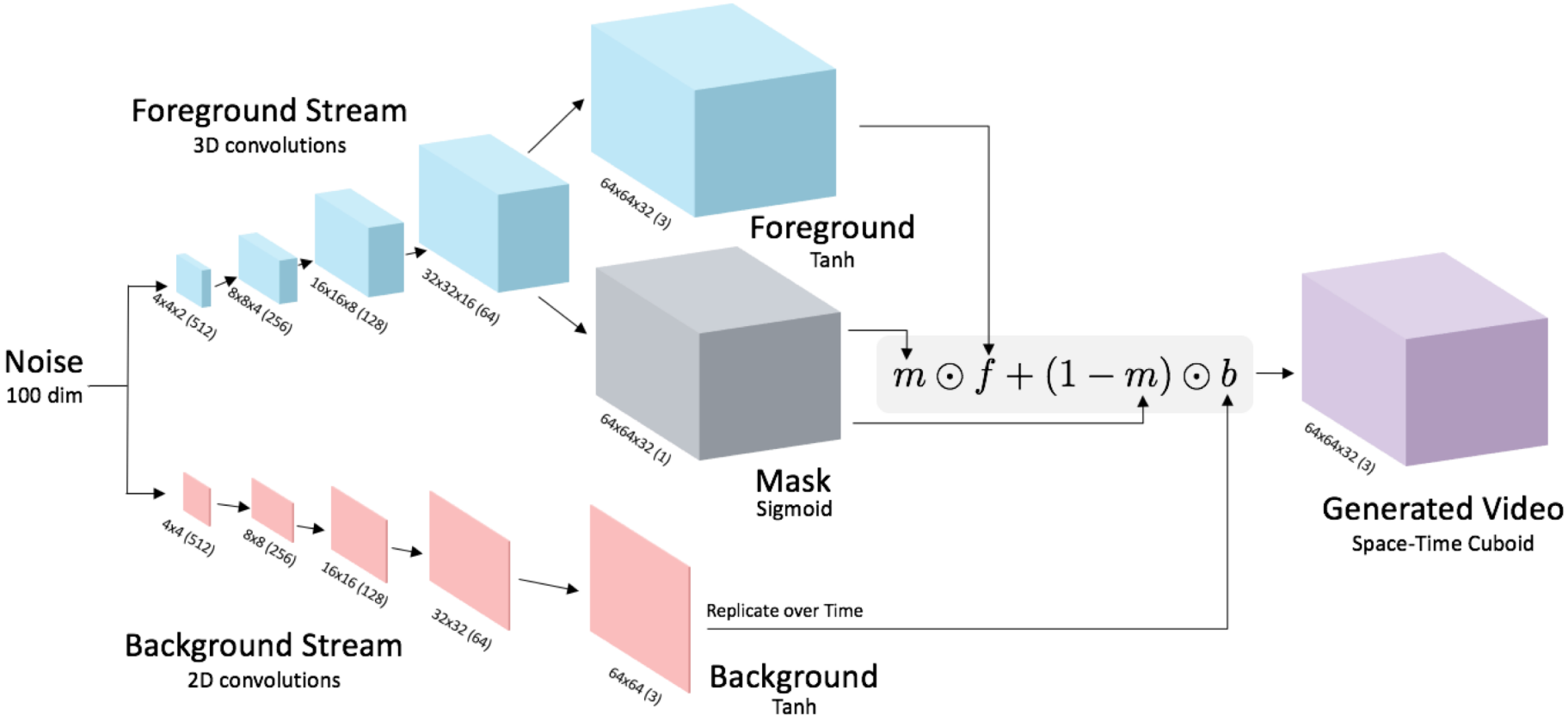
Results -- Cycle GAN

photo van Gogh

Domain X

Domain Y

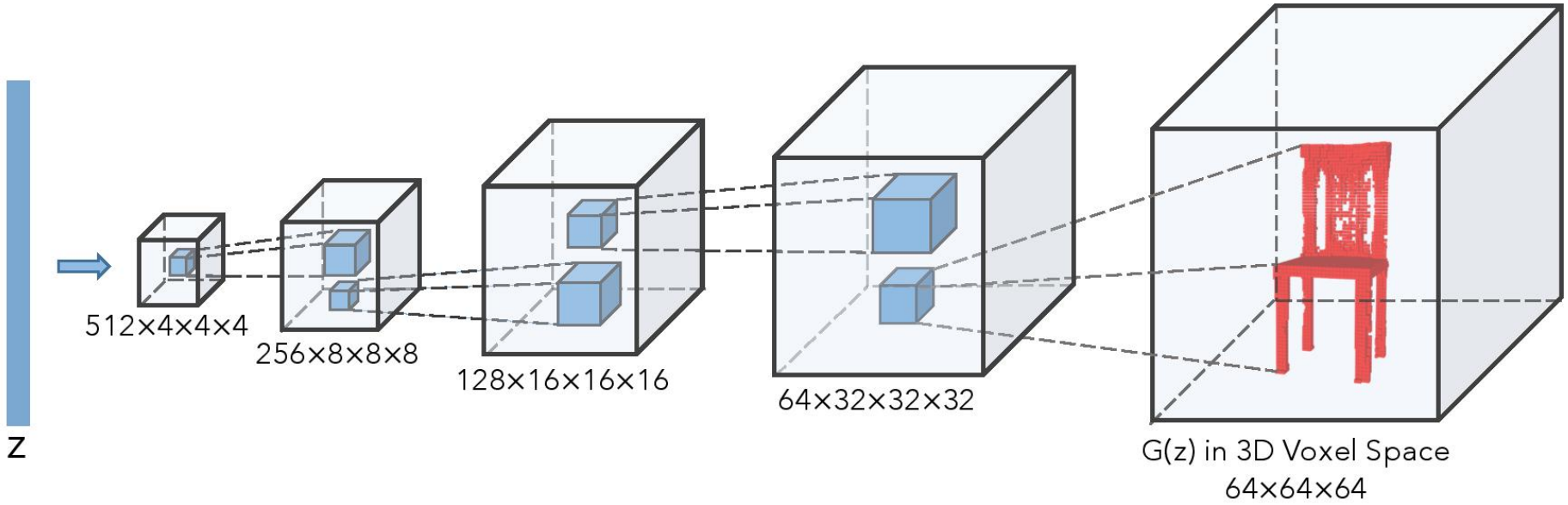
Video GAN



- Videos http://web.mit.edu/vondrick/tinyvideo/

[Vondrick et al. NIPS 2016]

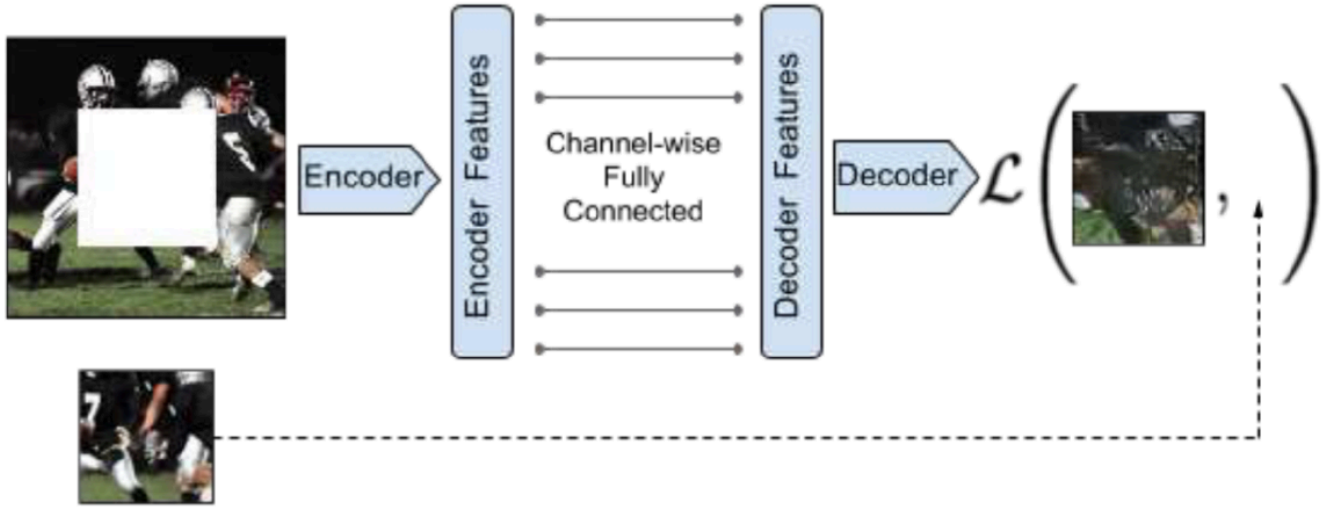
Shape modeling using 3D Generative Adversarial Network



[Wu et al. NIPS 2016]

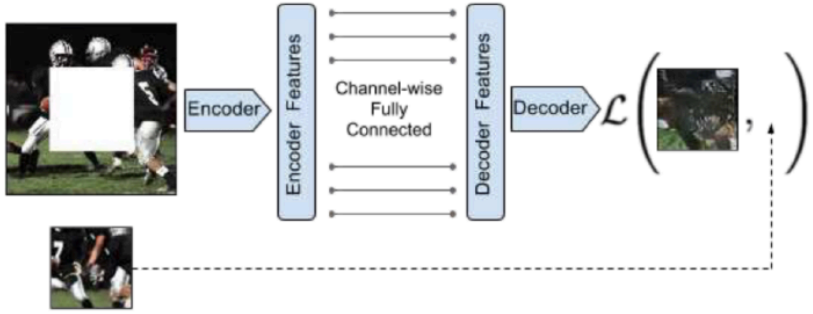
GAN for Inpainting

Use of GANs: Inpainting



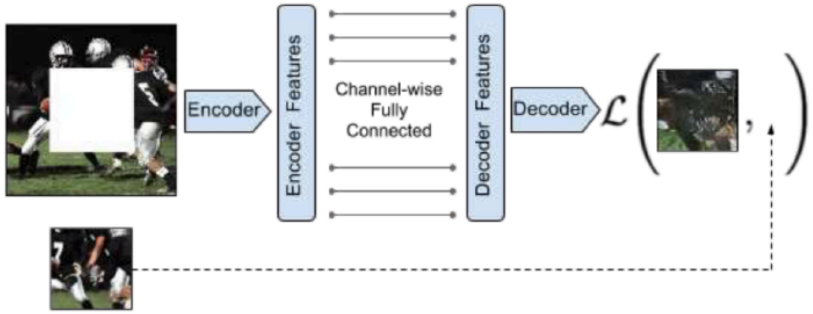
- Unsupervised learning (only uses the images) with GAN approach
- **Task:** Reconstructing masked pixels (inpainting) by decoding the *context features*
- Other Task (not analysed here) in [Context Encoders: Feature Learning by Inpainting, Pathak D., Krahenbuhl P., Donahue J., Darrell T., Efros AA., CVPR 2016]

Encoder



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

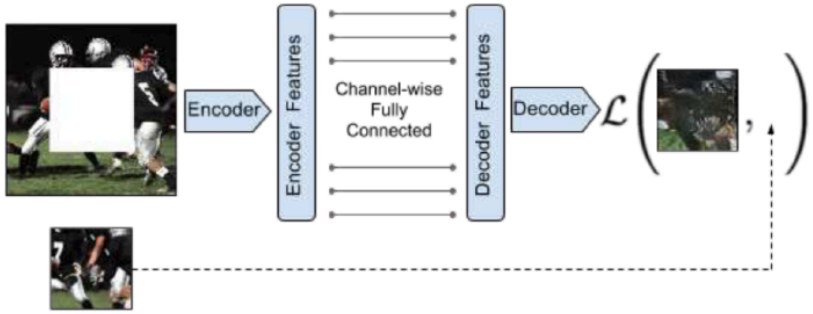
Channel-wise fully-connected layer



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

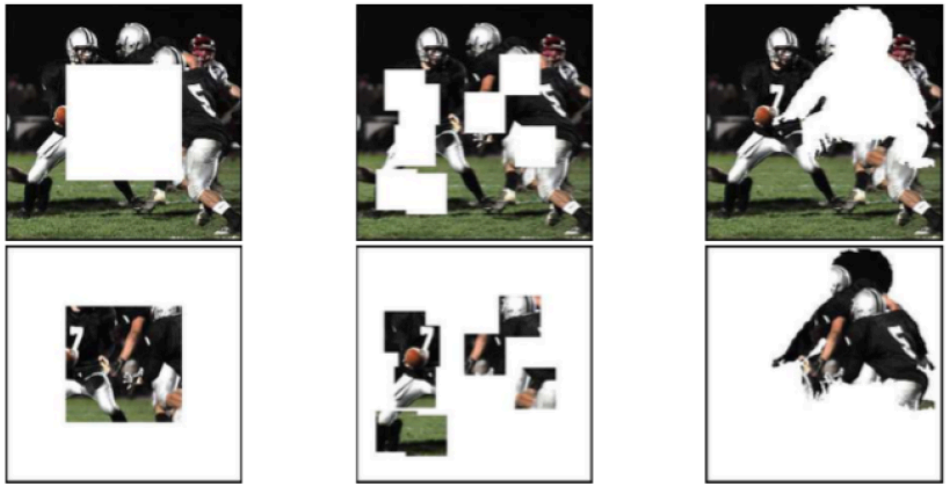


Decoder

- **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)

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

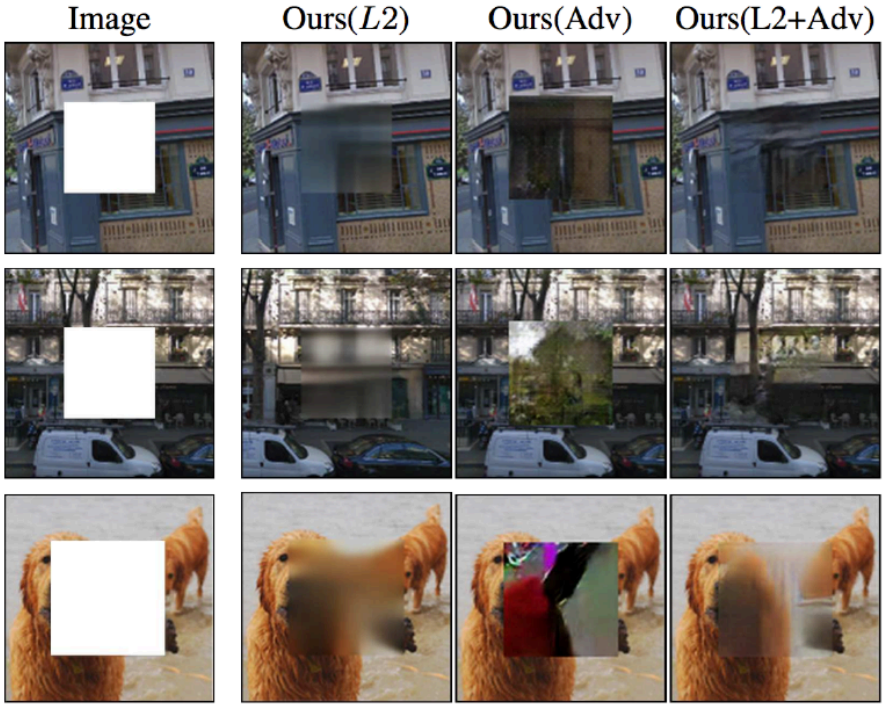
$$\mathcal{L}_{rec}(x) = \left\| \hat{M} \odot \left(x - F\left((1 - \hat{M}) \odot x\right) \right) \right\|_2$$

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

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

Semantic inpainting - Task: Fill masked areas in the image

- **Learning:**
  - $\lambda_{rec} = 0.999$, $\lambda_{adv} = 0.001$
  - $lr_F = 10 \times lr_D$
  - Also apply $L2$ loss to 7px around region to paint
  - Adam optimizer
- **Dataset:** StreetView Paris and ImageNet



Semantic inpainting - Qualitative results

Missing data encoder