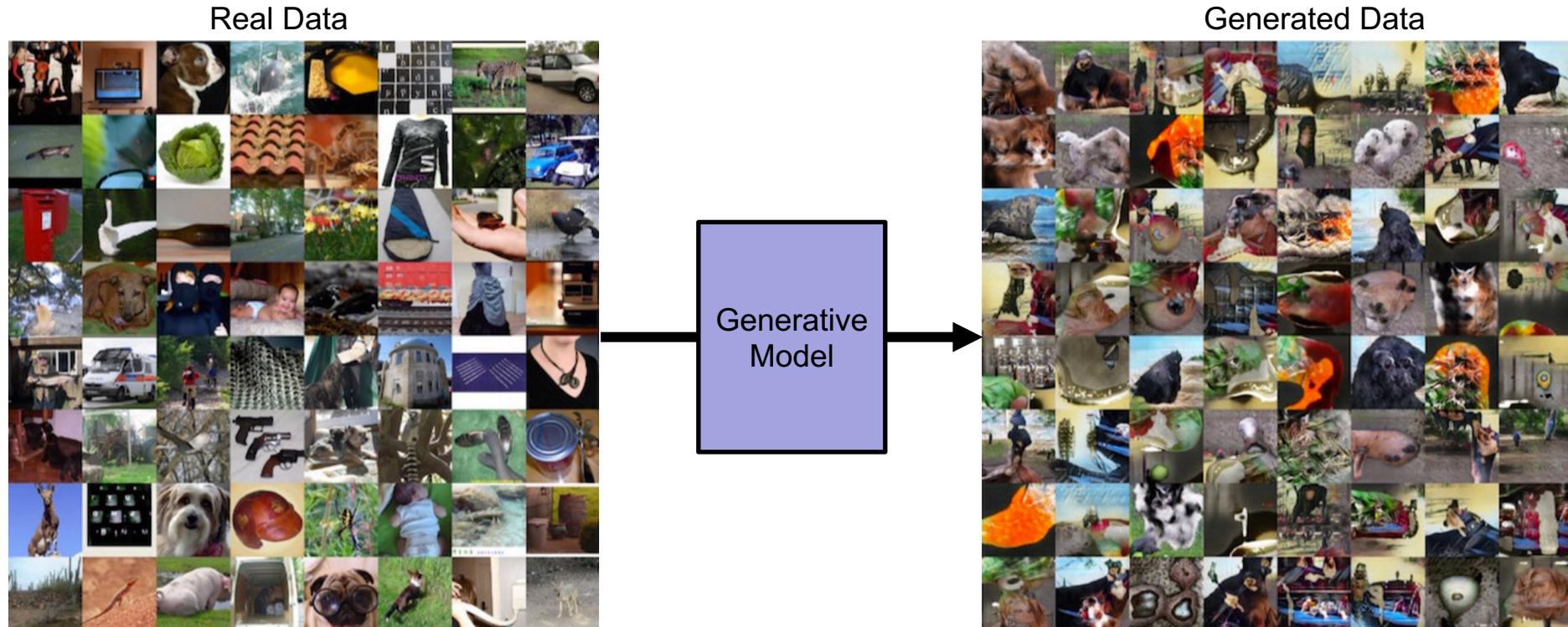


# Deep Generative Models: Generative Adversarial Networks

Slides adapted from Pieter Abbeel

This session will be audio  
recorded for educational use

# Generative Model



Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, 2016

# Generative Adversarial Networks

---

## Generative Adversarial Nets

---

Ian J. Goodfellow,<sup>\*</sup> Jean Pouget-Abadie,<sup>†</sup> Mehdi Mirza, Bing Xu, David Warde-Farley,  
Sherjil Ozair,<sup>‡</sup> Aaron Courville, Yoshua Bengio<sup>§</sup>

Département d'informatique et de recherche opérationnelle  
Université de Montréal  
Montréal, QC H3C 3J7

### Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model  $G$  that captures the data distribution, and a discriminative model  $D$  that estimates the probability that a sample came from the training data rather than  $G$ . The training procedure for  $G$  is to maximize the probability of  $D$  making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions  $G$  and  $D$ , a unique solution exists, with  $G$  recovering the training data distribution and  $D$  equal to  $\frac{1}{2}$  everywhere. In the case where  $G$  and  $D$  are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

# Generative Adversarial Networks



$x$  sampled from  
data

Real image

Figure from NeurIPS 2016  
GAN Tutorial (Goodfellow)

# Generative Adversarial Networks



$x$  sampled from  
data

Real image

Input noise  $z$

Figure from NeurIPS 2016  
GAN Tutorial (Goodfellow)

# Generative Adversarial Networks

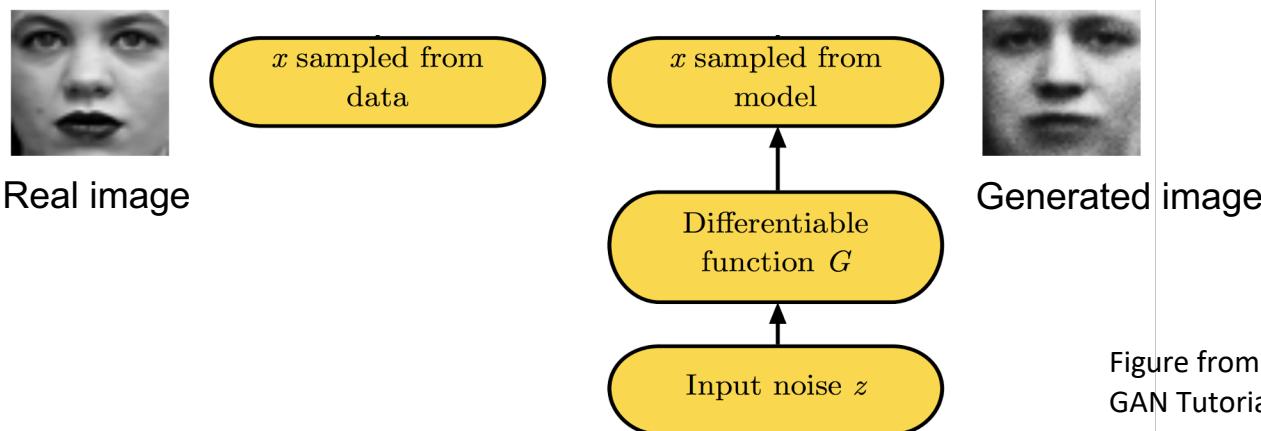


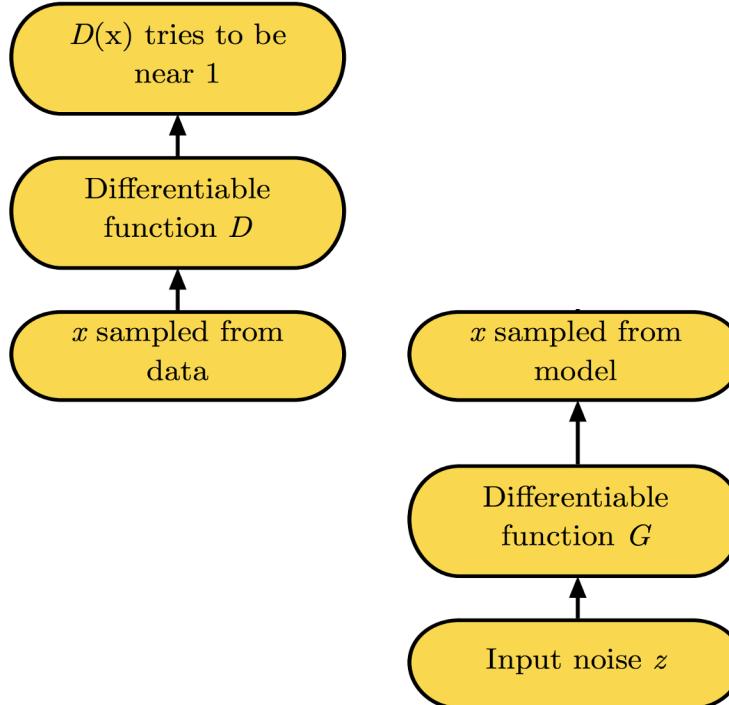
Figure from NeurIPS 2016  
GAN Tutorial (Goodfellow)

# Generative Adversarial Networks

Discriminator tries to classify all real data with label “1”



Real image



Generated image

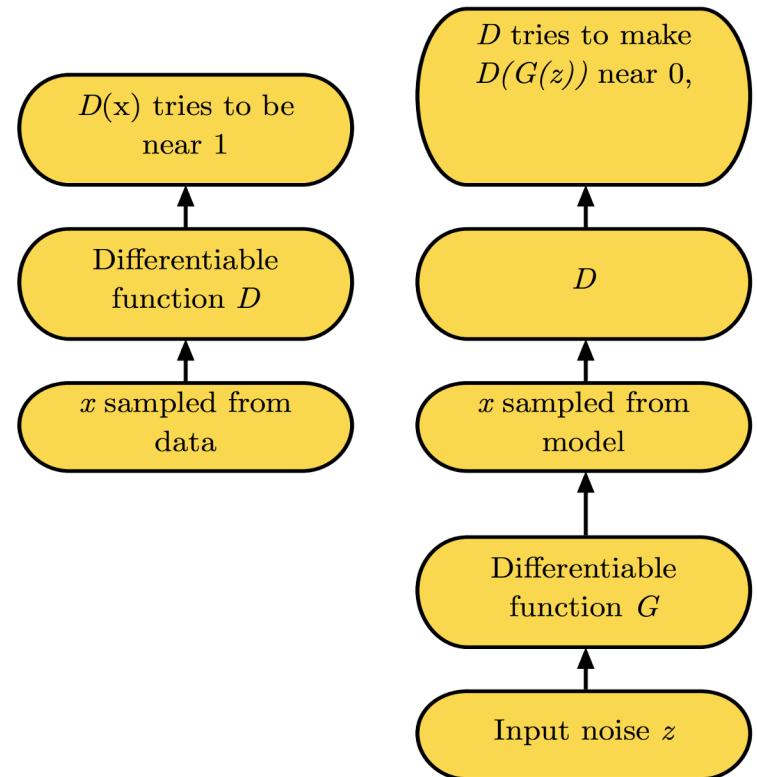
Figure from NeurIPS 2016  
GAN Tutorial (Goodfellow)

# Generative Adversarial Networks

Discriminator tries to classify all real data with label “1”



Real image



Discriminator tries to classify all generated data with label “0”



Generated image

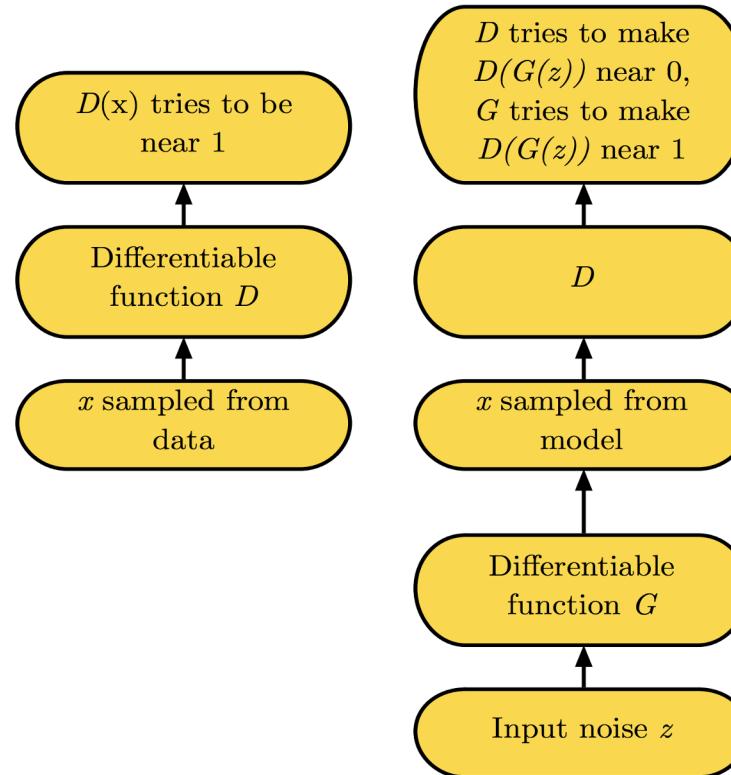
Figure from NeurIPS 2016  
GAN Tutorial (Goodfellow)

# Generative Adversarial Networks

Discriminator tries to classify all real data with label “1”



Real image



Discriminator tries to classify all generated data with label “0”

**Generator tries to “fool” the discriminator**



Generated image

Figure from NeurIPS 2016  
GAN Tutorial (Goodfellow)

# Generative Adversarial Networks

GAN objective:

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - \underline{D(G(z))})]$$

- Loss of generator is:

$$\mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

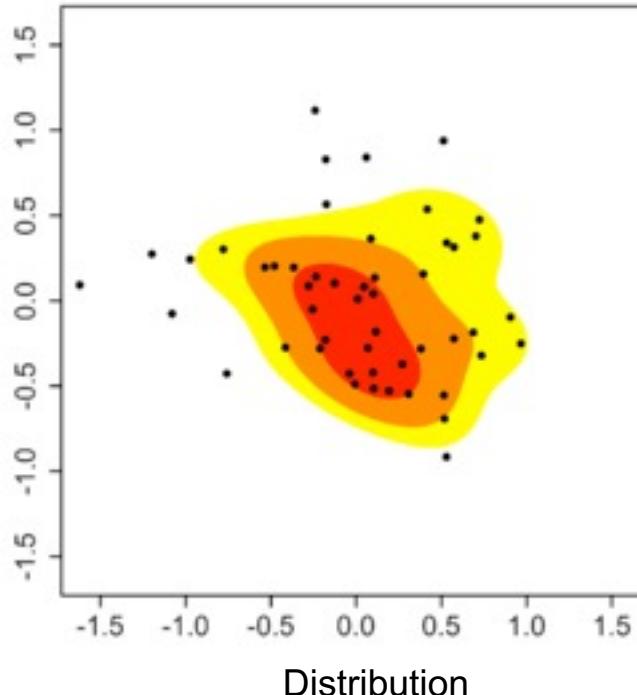
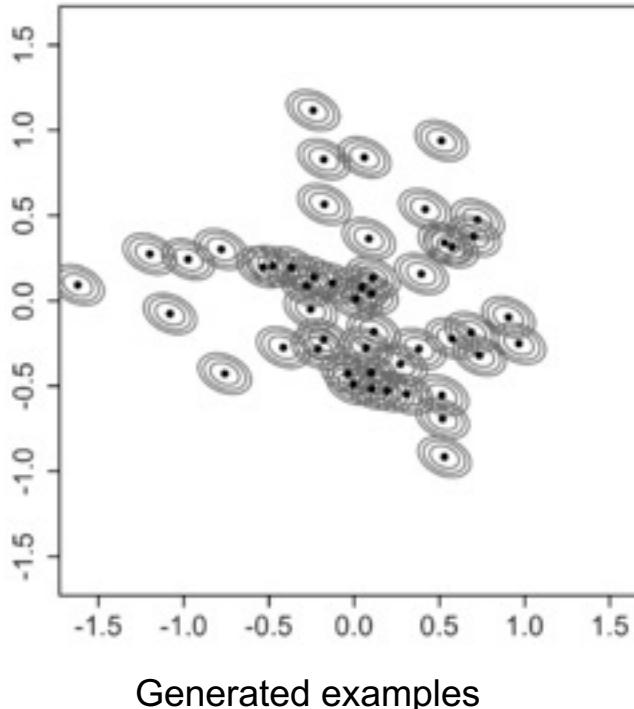
- The output of the generator passes through the discriminator to compute the loss
- Thus the discriminator needs to be differentiable so that we can backpropagate the loss to the generator
- We alternate between training the generator (k1 steps of SGD) and training the discriminator (k2 steps of SGD)

# How to evaluate if your GAN is working?

- Just look at the pictures?
- Quantitative evaluation for GANs (or any generative model) is still an open problem
- The following discussion applies to evaluating any generative model (not just GANs)

# Gaussian density estimator

- We treat a set of generated samples as a mixture of Gaussians
- We can then compute the probability of some test data under this mixture distribution



# Gaussian density estimator

- Decent metric for low-dimensional data
- But:
  - Sensitive to hyperparameters (number of generated samples, Kernel size)
  - For high-dimensional data we need a very large number of samples to cover the space
  - Usually too computationally intensive for high-dimensional data

# Inception Score

- Can we side-step high-dimensional density estimation?
- One idea: good generators generate samples that are semantically diverse
- Semantics predictor: trained image classifier (e.g. Inception Network v3)
  - $p(y|x)$ ,  $y$  is one of the [1000 ImageNet classes](#)
- What do we want from a good generator?
  - each image  $x$  should have distinctly recognizable object ->  $p(y|x)$  should have low entropy
  - **Is this sufficient? How could a generator “cheat” this metric?**
  - there should be as many classes generated as possible ->  $p(y)$  should have high entropy

# Inception Score

- Inception classifier:  $p(y|x)$ 
  - **What is the optimal distribution for  $p(y|x)$  for a generated example  $y$ , if the classifier and the generator are both optimal?**
  - We want a peaky  $p(y|x)$  to indicate that the generated example  $x$  is clearly identifiable as some class  $y$
- Marginal label distribution = probability of classifying an example as class  $y$ , independent of the input:  $p(y) = \sum p(y | x)p(x) = E_{x \sim G}[p(y | x)]$ 
  - **What is the optimal distribution of  $p(y)$ ?**
  - We want a uniform  $p(y)$  because each generated example  $x$  should be a different class  $y$

# Inception Score

- We want a peaky  $p(y|x)$  to indicate that the generated example  $x$  is clearly identifiable as some class  $y$
- We want a uniform  $p(y)$  because each generated example  $x$  should be a different class  $y$
- Inception Score:  $\text{IS}(x) = \exp(\mathbb{E}_{x \sim p_g} [D_{\text{KL}} [p(y|x) \parallel p(y)]] )$
- A good generator will have a high KL
  - $p(y)$  will be close to uniform and  $p(y|x)$  will be very peaky

# Inception Score

Samples				
Model	Real data	Our methods	-VBN+BN	-L+HA
Score $\pm$ std.	$11.24 \pm .12$	$8.09 \pm .07$	$7.54 \pm .07$	$6.86 \pm .06$

# Inception Score

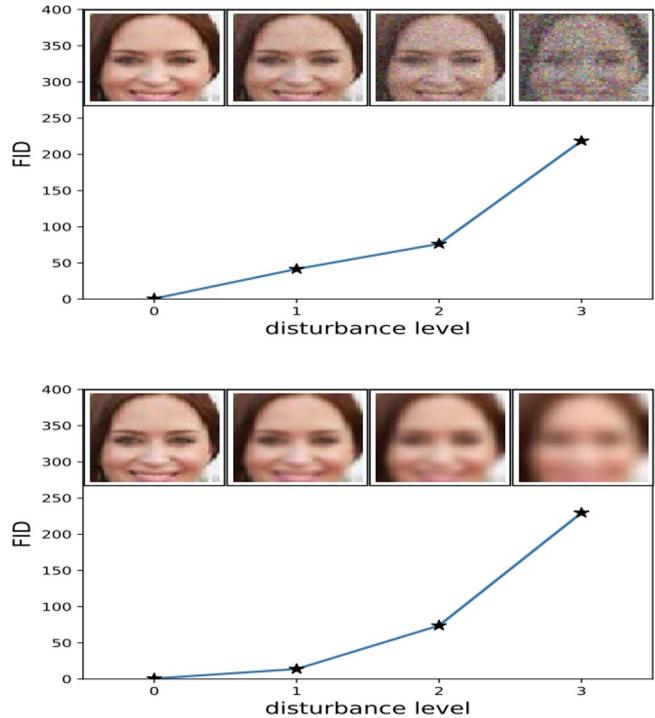
- We want a peaky  $p(y|x)$  to indicate that the generated example  $x$  is clearly identifiable as some class  $y$
- We want a uniform  $p(y)$  because each generated example  $x$  should be a different class  $y$
- **How could a generator “cheat” the inception score without doing what we want?**

# Fréchet Inception Distance

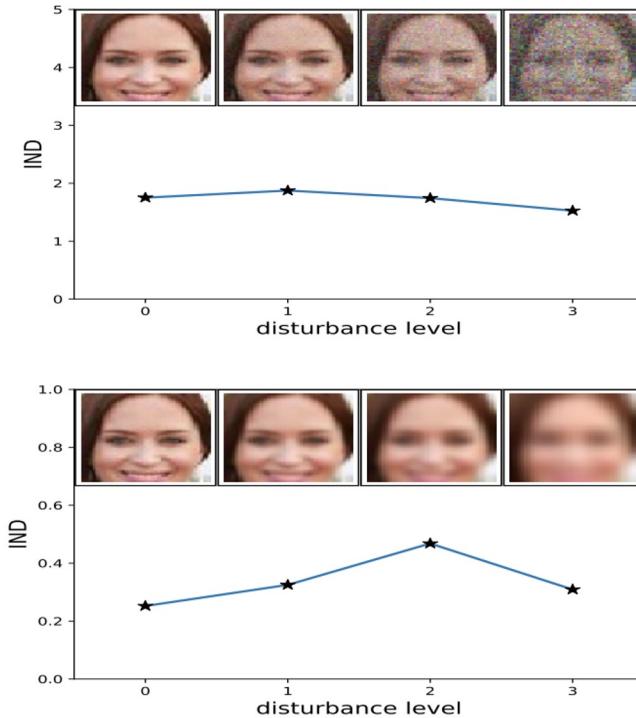
- A list of 1000 images (one of each class) can obtain perfect Inception Score
- Inception Score doesn't sufficiently measure diversity
  - It only captures class-level diversity but not within-class diversity
- FID was proposed to capture more details
  - We want the **features** to be realistic and diverse, not just the **labels**!
- Embed image  $x$  into some feature space (2048-dimensional activations of the Inception-v3 pool3 layer), then compare mean ( $m$ ) & covariance ( $C$ ) of those random features

$$d^2((\mathbf{m}, \mathbf{C}), (\mathbf{m}_w, \mathbf{C}_w)) = \|\mathbf{m} - \mathbf{m}_w\|_2^2 + \text{Tr}(\mathbf{C} + \mathbf{C}_w - 2(\mathbf{C}\mathbf{C}_w)^{1/2})$$

## Fréchet Inception Distance

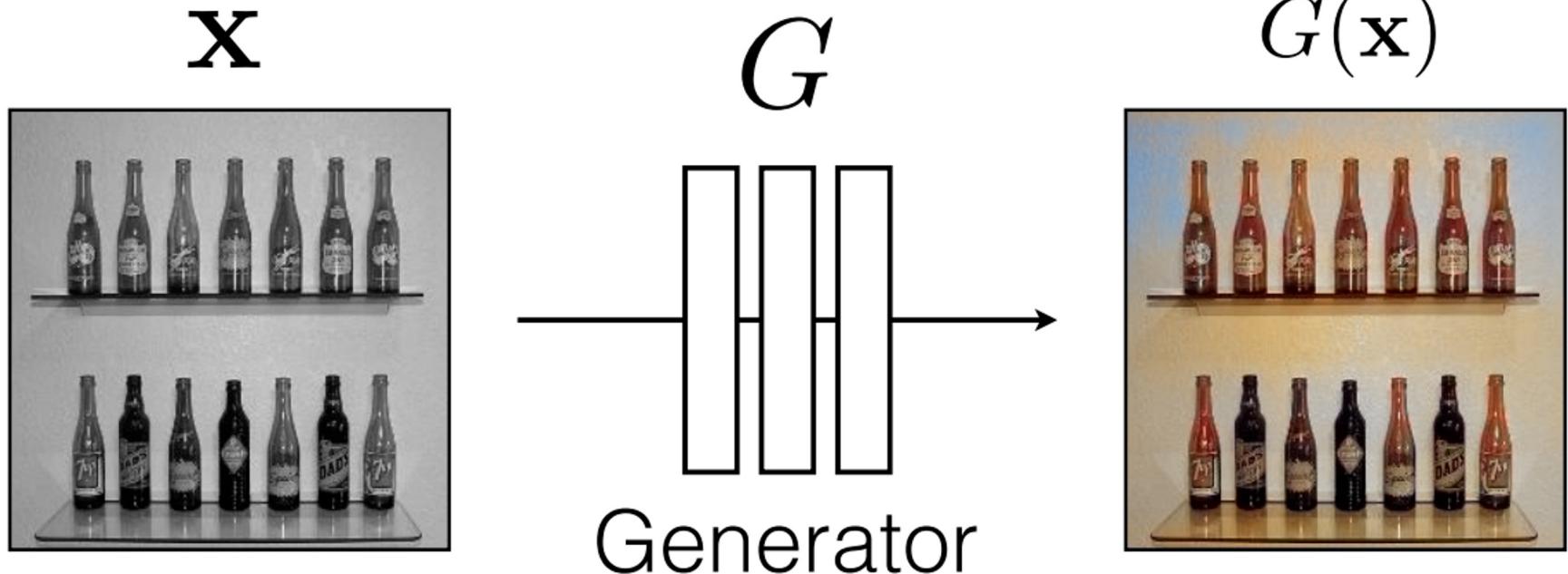


## Inception Score



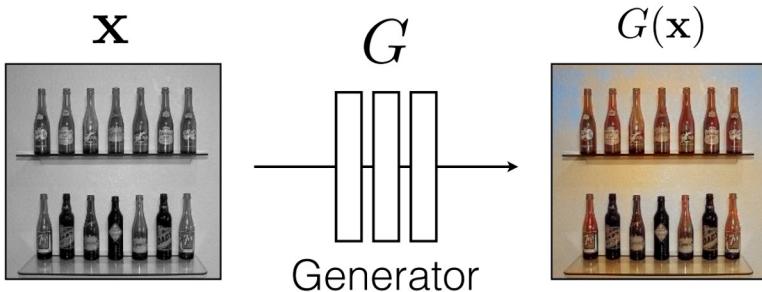
[Heusel et al, 2017]

# Conditional GANs / pix2pix



Slide: Phillip Isola

# Conditional GANs / pix2pix



Where do we get training data for this?

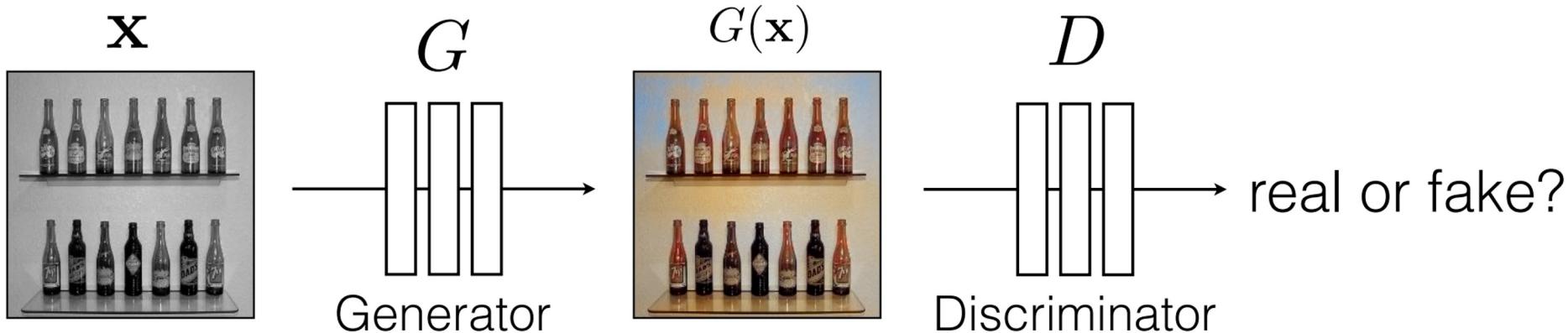
Why not just train this with supervised learning?

Why do we need a GAN?

Convert color images to black and white

Given a black and white image, there  
are multiple plausible colorizations

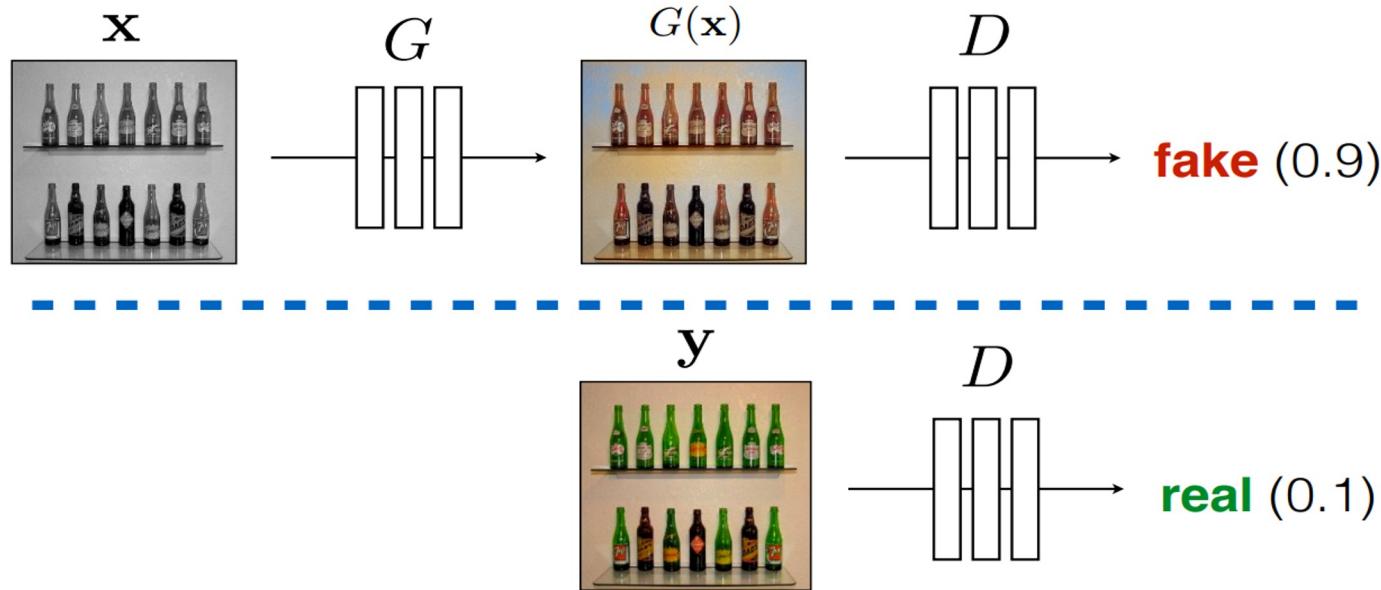
# Conditional GANs / pix2pix



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

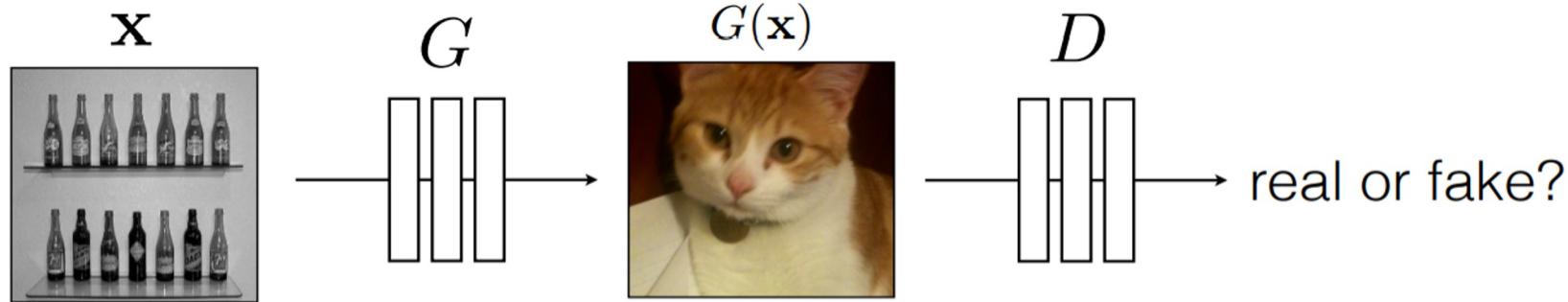
**D** tries to identify the fakes

# Conditional GANs / pix2pix

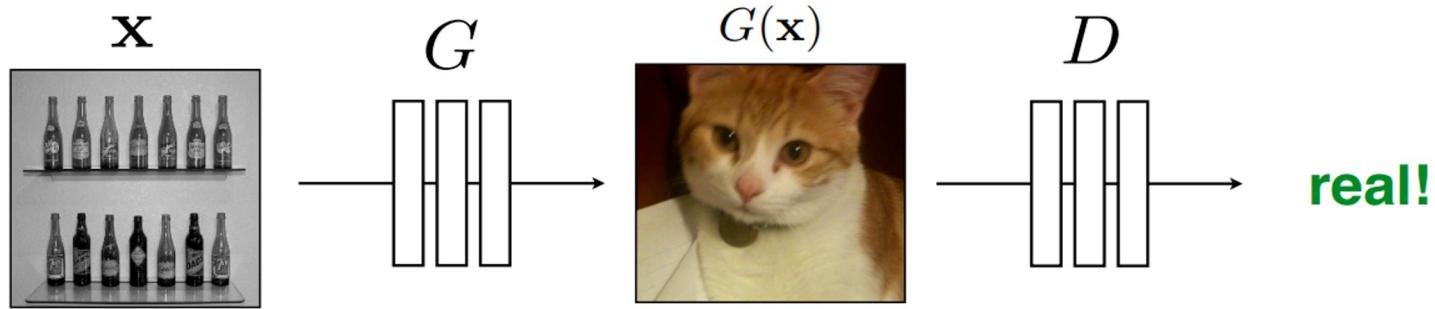


What is the problem with this approach?

# Conditional GANs / pix2pix

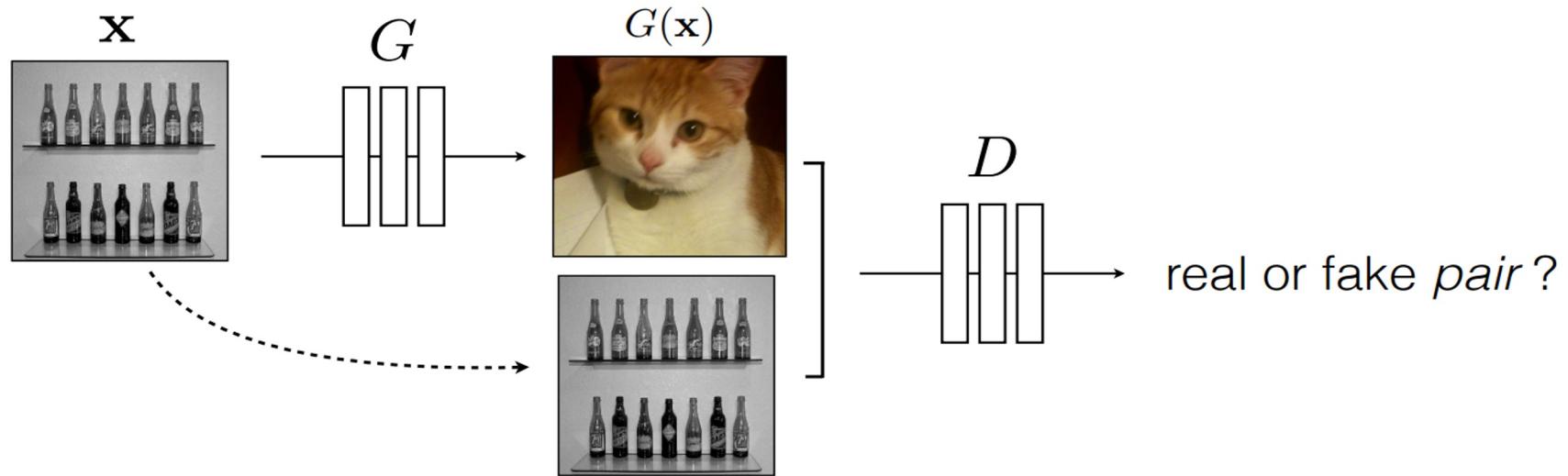


# Conditional GANs / pix2pix



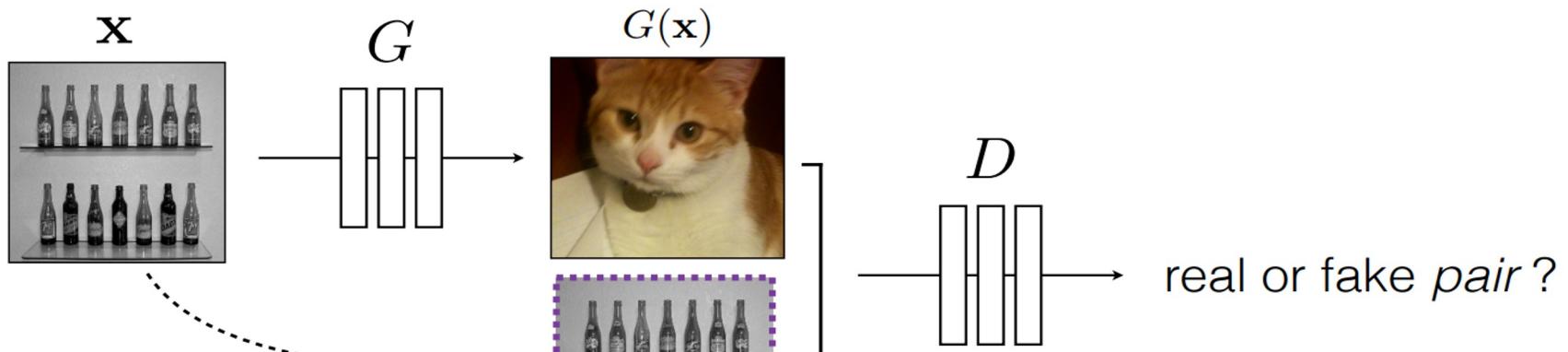
How can we fix this?

# Conditional GANs / pix2pix



Old objective:  $\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [ \log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y})) ]$  sola

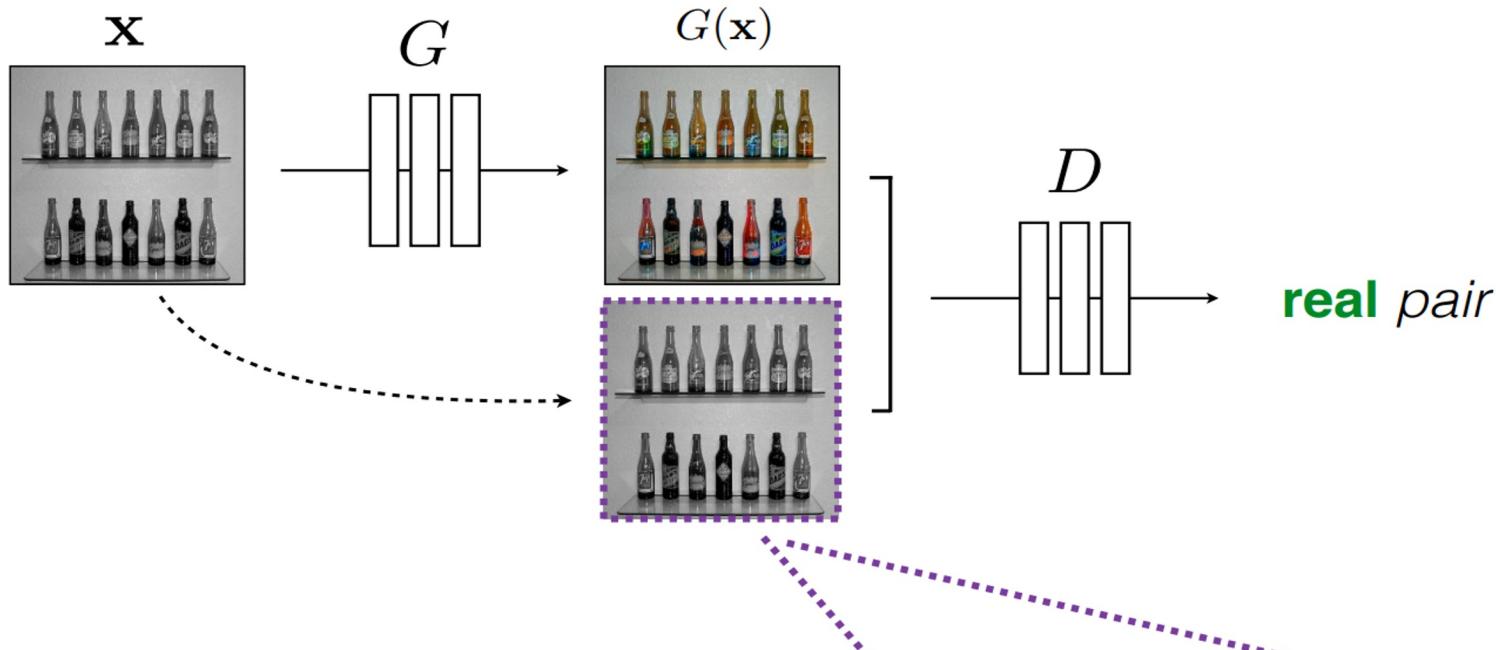
# Conditional GANs / pix2pix



New objective:  $\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y})) ]$

Old objective:  $\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [ \log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y})) ]$

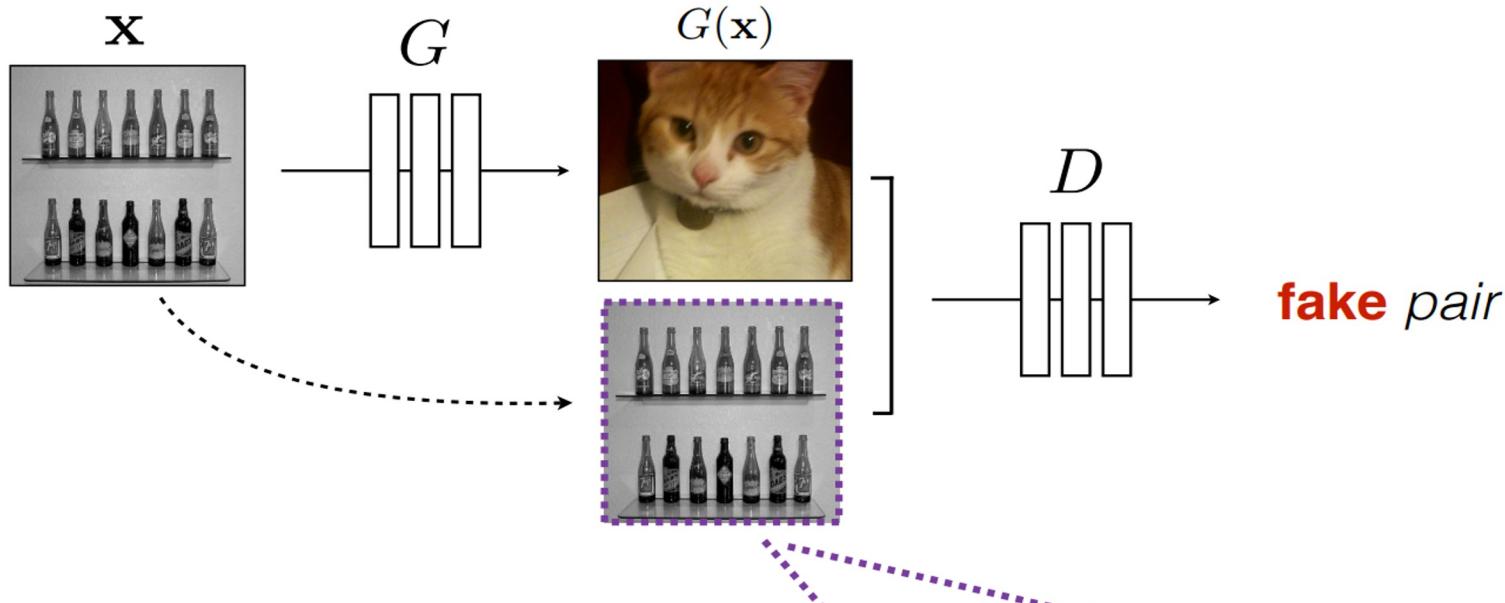
# Conditional GANs / pix2pix



New objective:  $\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y})) ]$

Old objective:  $\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [ \log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y})) ]$

# Conditional GANs / pix2pix

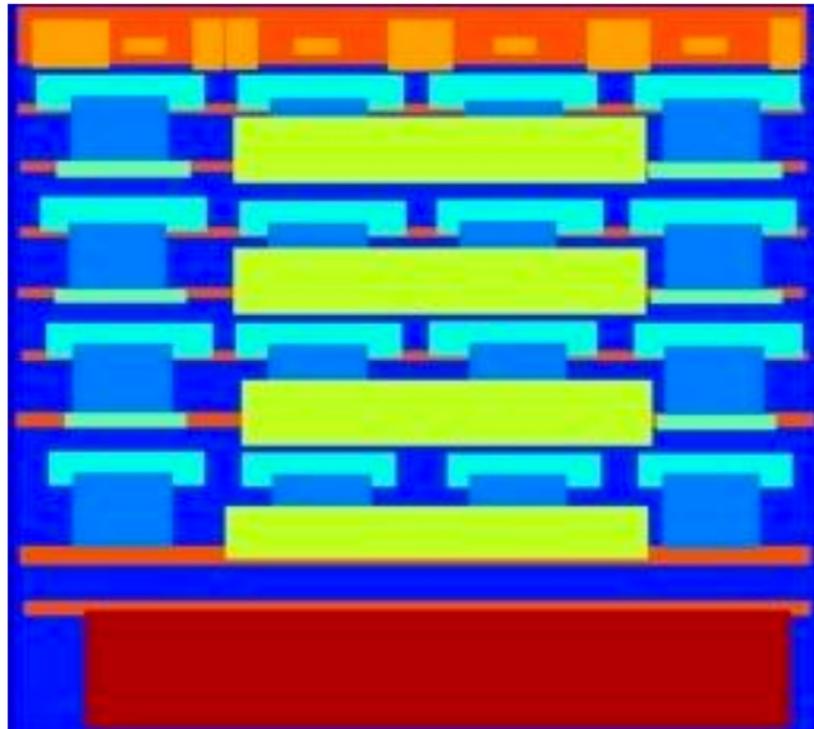


New objective:  $\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y})) ]$

Old objective:  $\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [ \log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y})) ]$

# Conditional GANs / pix2pix

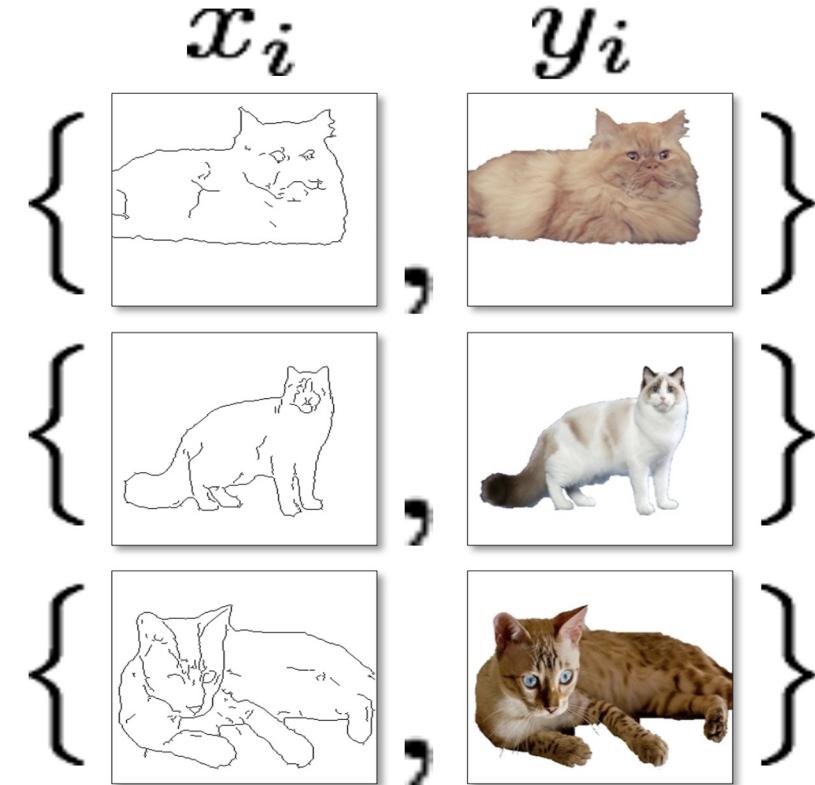
Input



16x16 Discriminator



# Conditional GANs / pix2pix



# Image-to-Image Translation with pix2pix

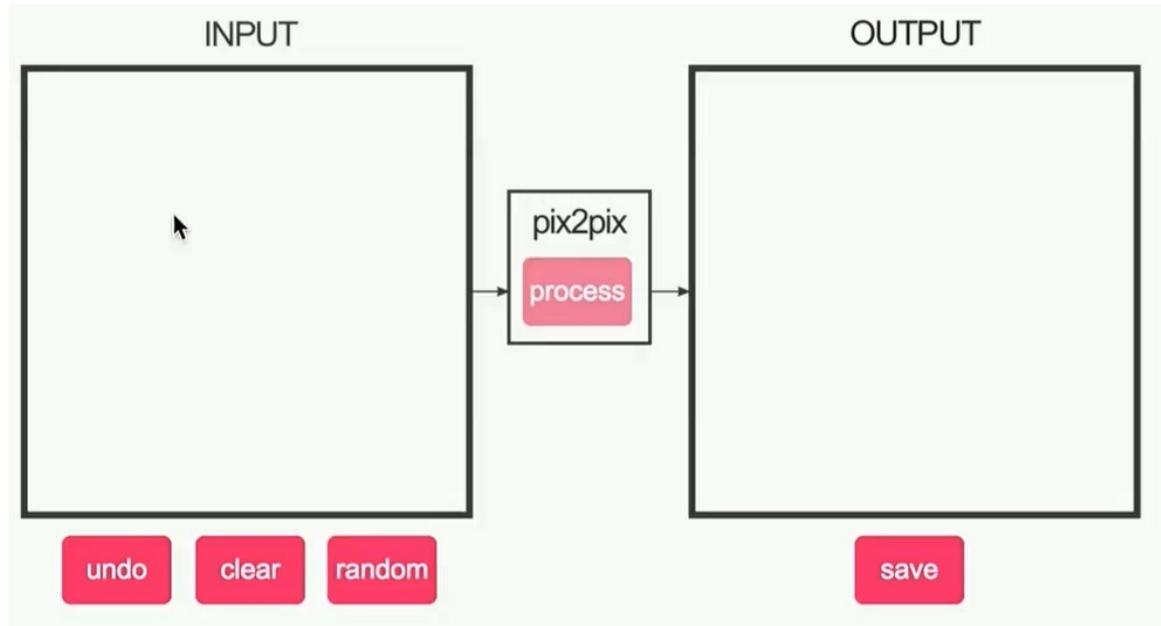
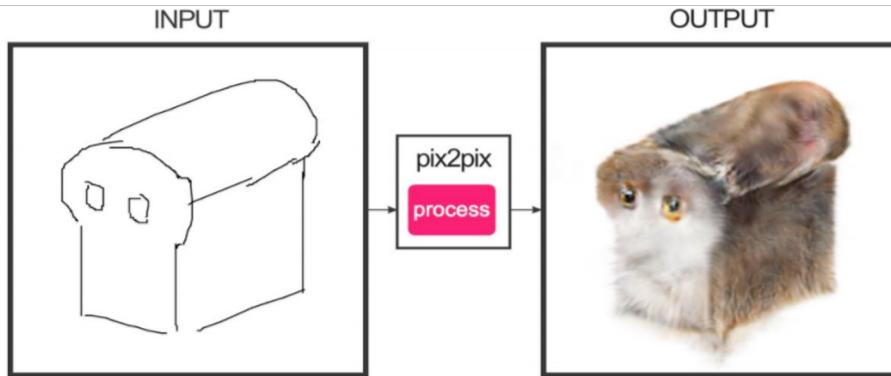


Image-to-image Translation with Conditional Adversarial Nets  
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. CVPR 2017

# Conditional GANs / pix2pix



Ivy Tasi @ivymyt



Vitaly Vidmirov @vvid

Slide: Phillip Isola

# Image-to-Image Translation with pix2pix

Labels to Street Scene

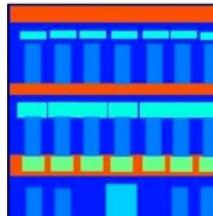


input



output

Labels to Facade



input



output  
Day to Night

BW to Color



input

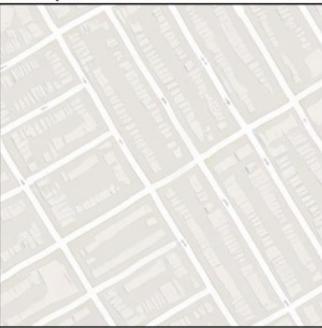


output

Aerial to Map



input



output

input  
output

Edges to Photo



input



output



input



output

# Conditional GANs / pix2pix

BW → Color

Input



Output



Input



Output



Input



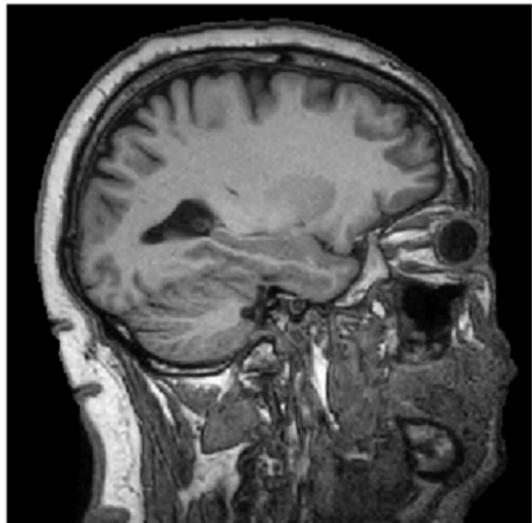
Output



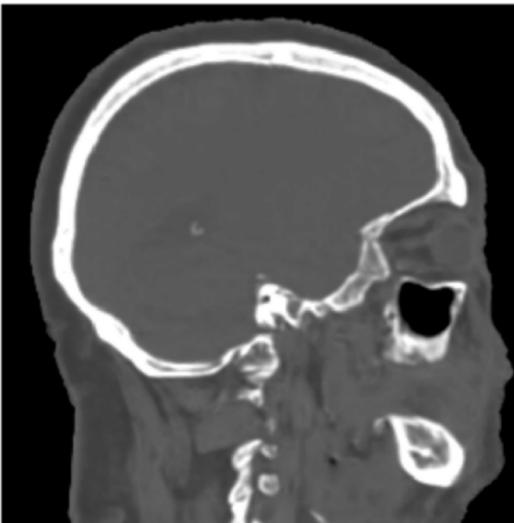
Slide: Phillip Isola

# pix2pix for medical imaging

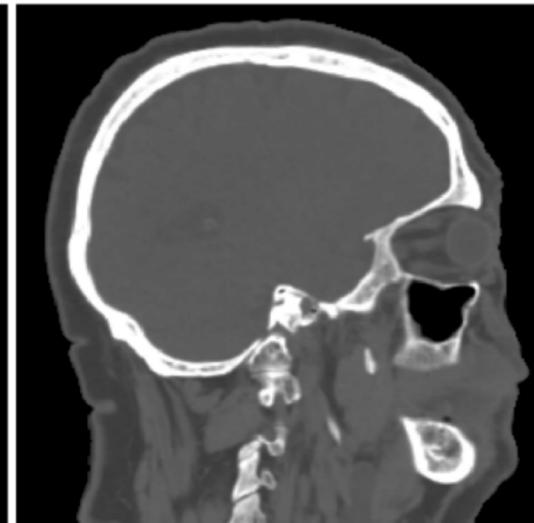
**MR → CT** [Wolterink et al] arxiv: 1708.01155



Input MR



Generated CT



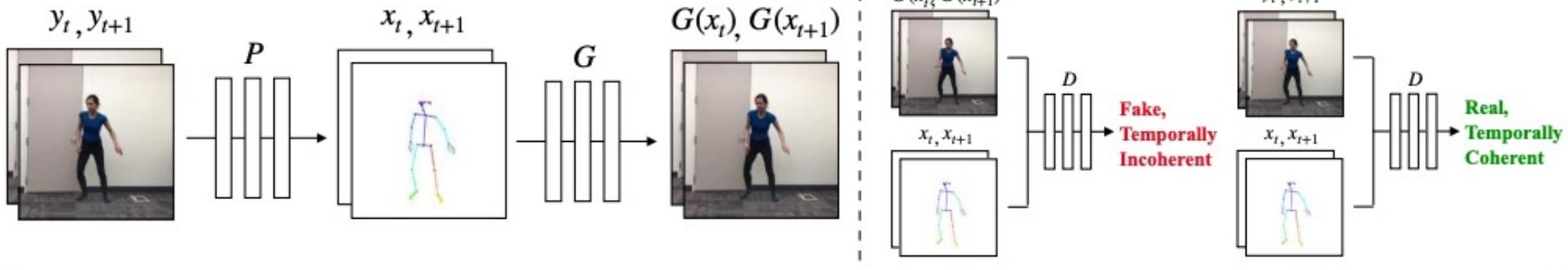
Ground truth CT

- MRI reconstruction [Quan et al.] arxiv:1709.00753
- Cardiac MR images from CT [Chartsias et al. 2017]

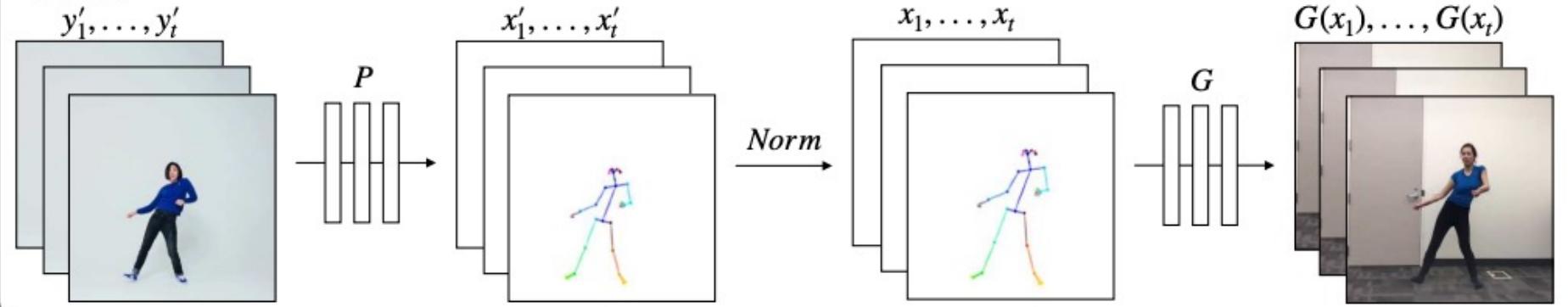


Chan, Caroline, et al. "Everybody dance now." /ICCV 2019.

## Training



## Transfer

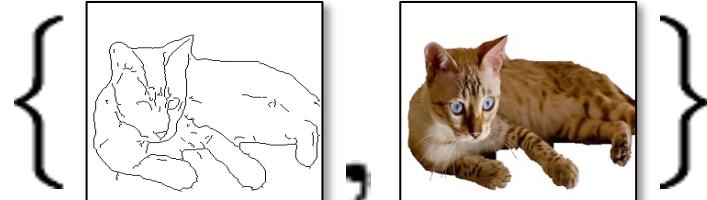
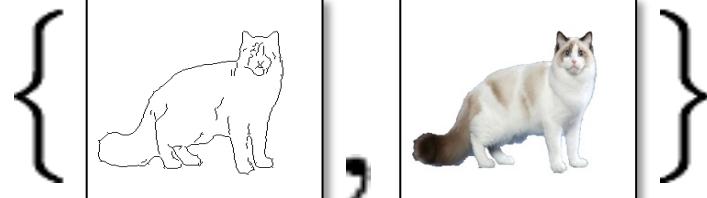
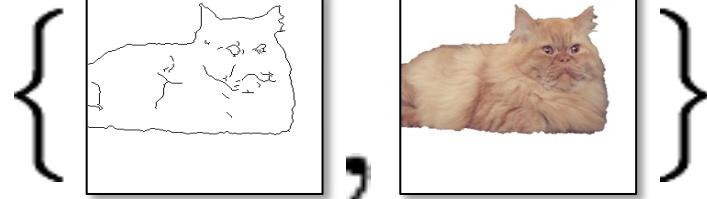


Chan, Caroline, et al. "Everybody dance now." ICCV 2019.

# Paired

$x_i$

$y_i$

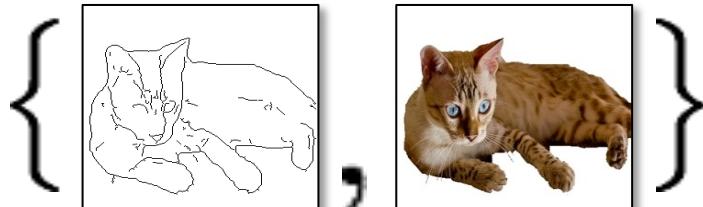
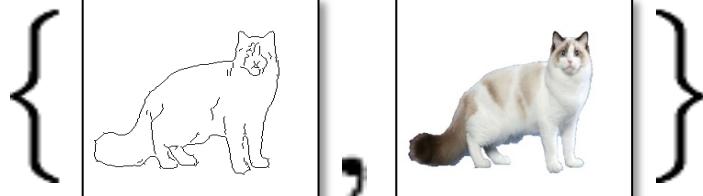
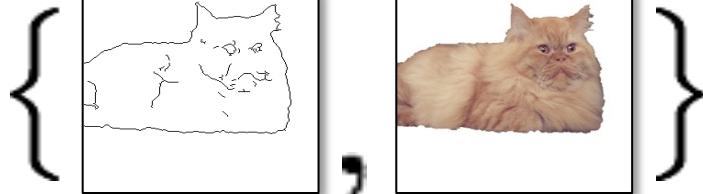


⋮

# Paired

$x_i$

$y_i$



⋮

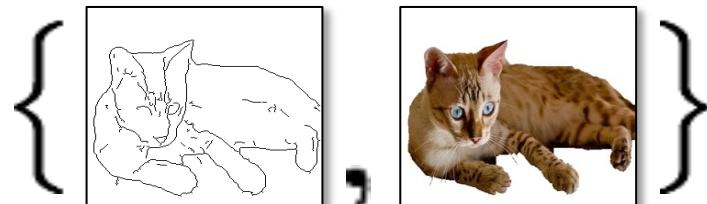
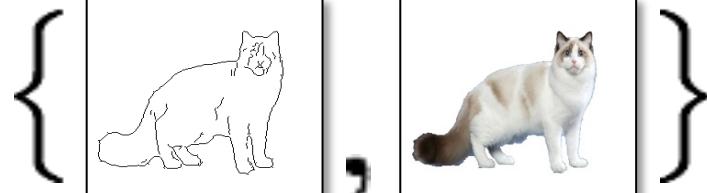
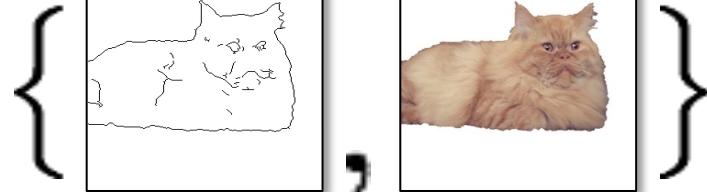


Horse  $\leftrightarrow$  zebra:  
how to get paired images of zebras?

# Paired

$x_i$

$y_i$



⋮

# Unpaired

$X$

$Y$



⋮

⋮

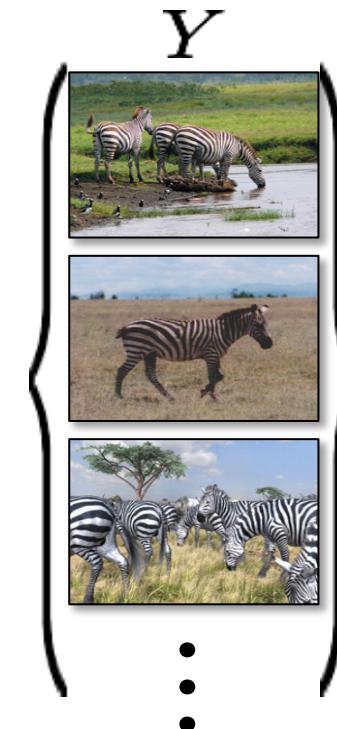
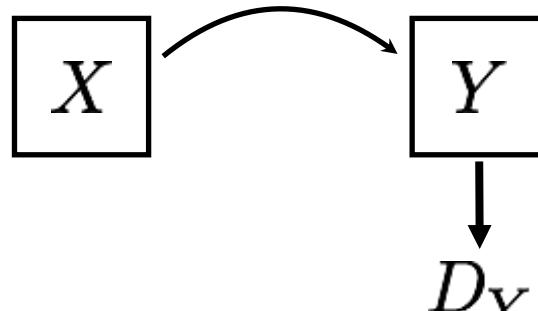
# Cycle-Consistent Adversarial Networks

How could a network “cheat” this?

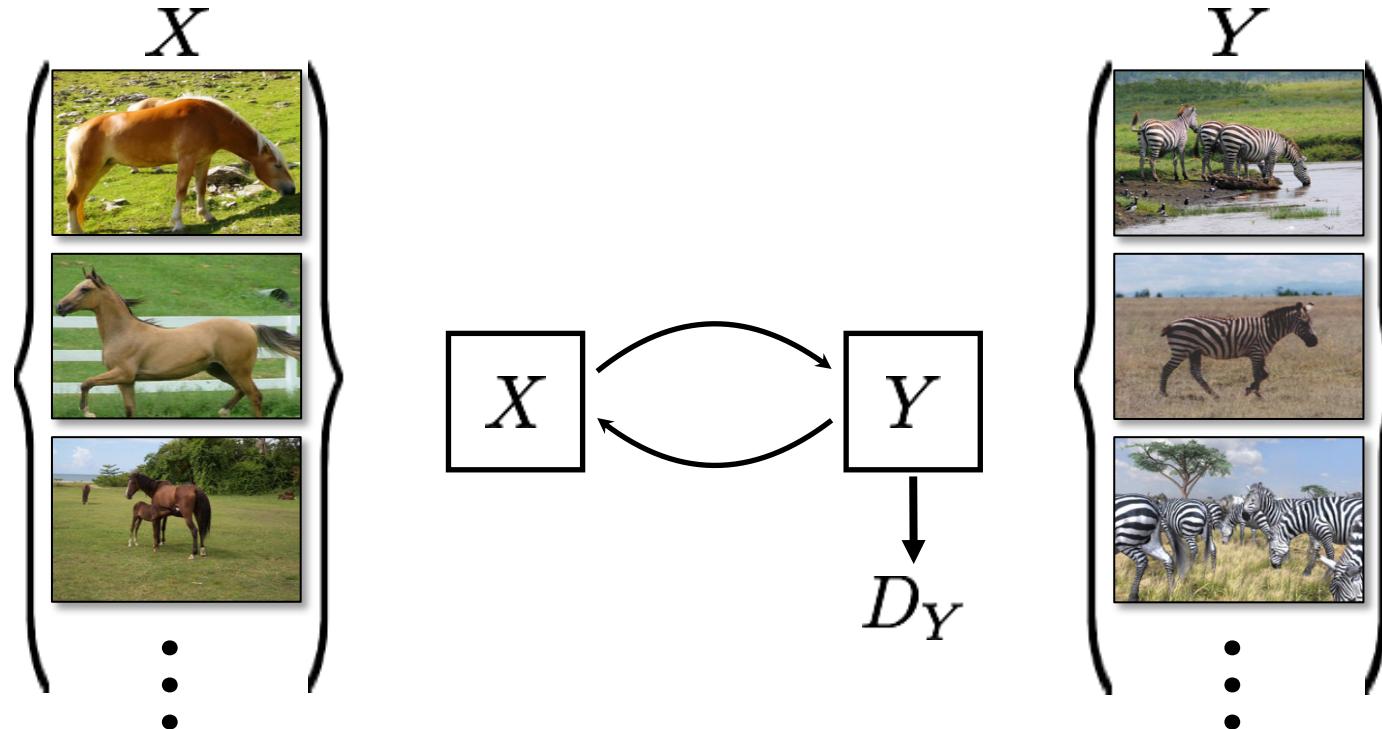


It could generate a zebra picture in a pose that is completely unrelated to the horse

How can we fix this if we do not have a dataset of horse-zebra pairs?

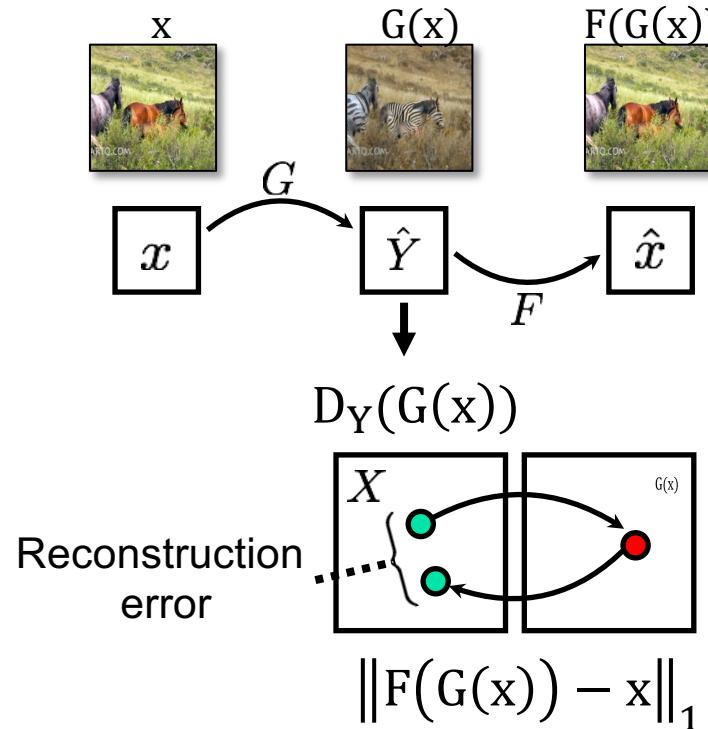


# Cycle-Consistent Adversarial Networks



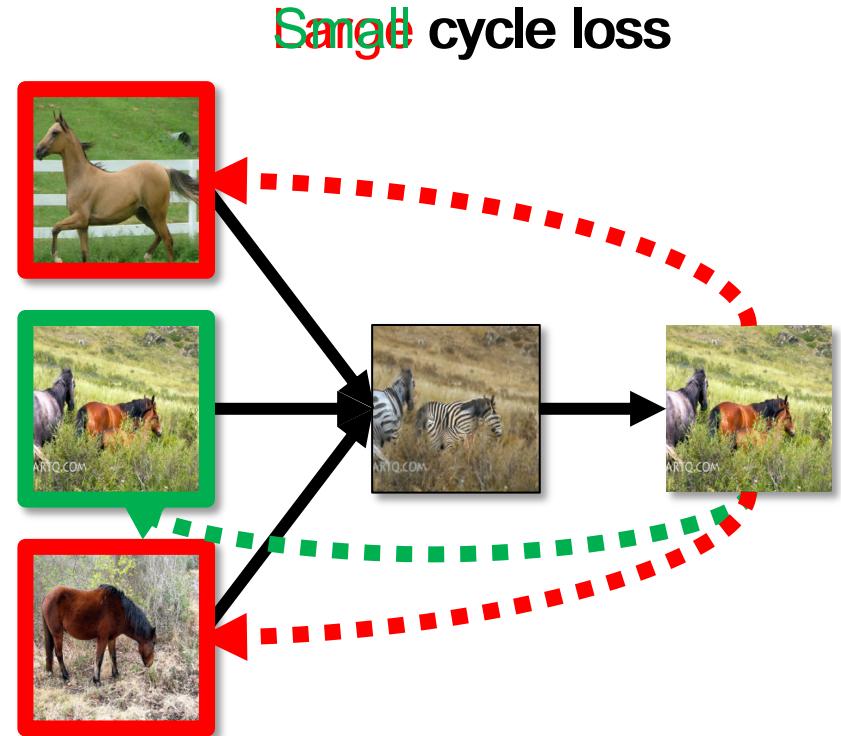
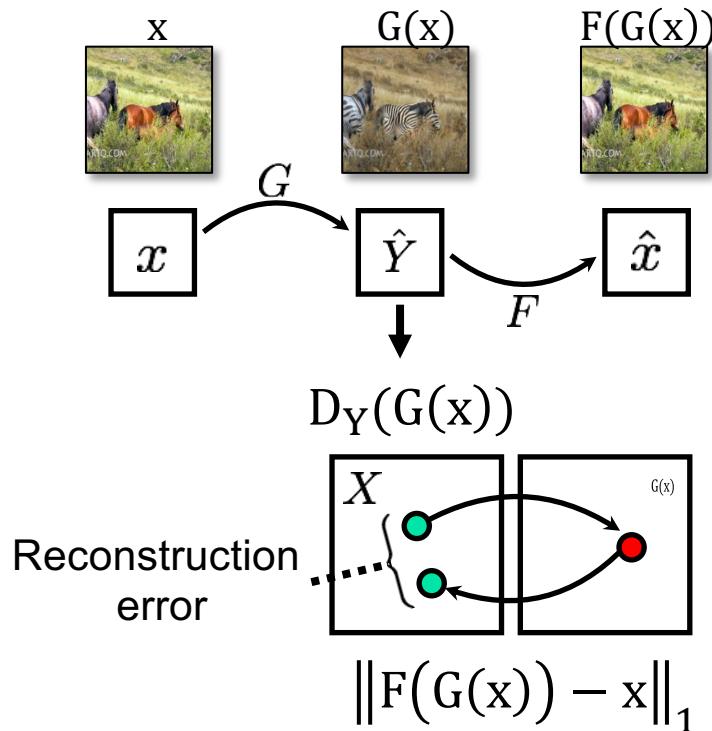
[Zhu\*, Park\*, Isola, and Efros, ICCV 2017]

# Cycle-Consistent Adversarial Networks



[Zhu\*, Park\*, Isola, and Efros, ICCV 2017]

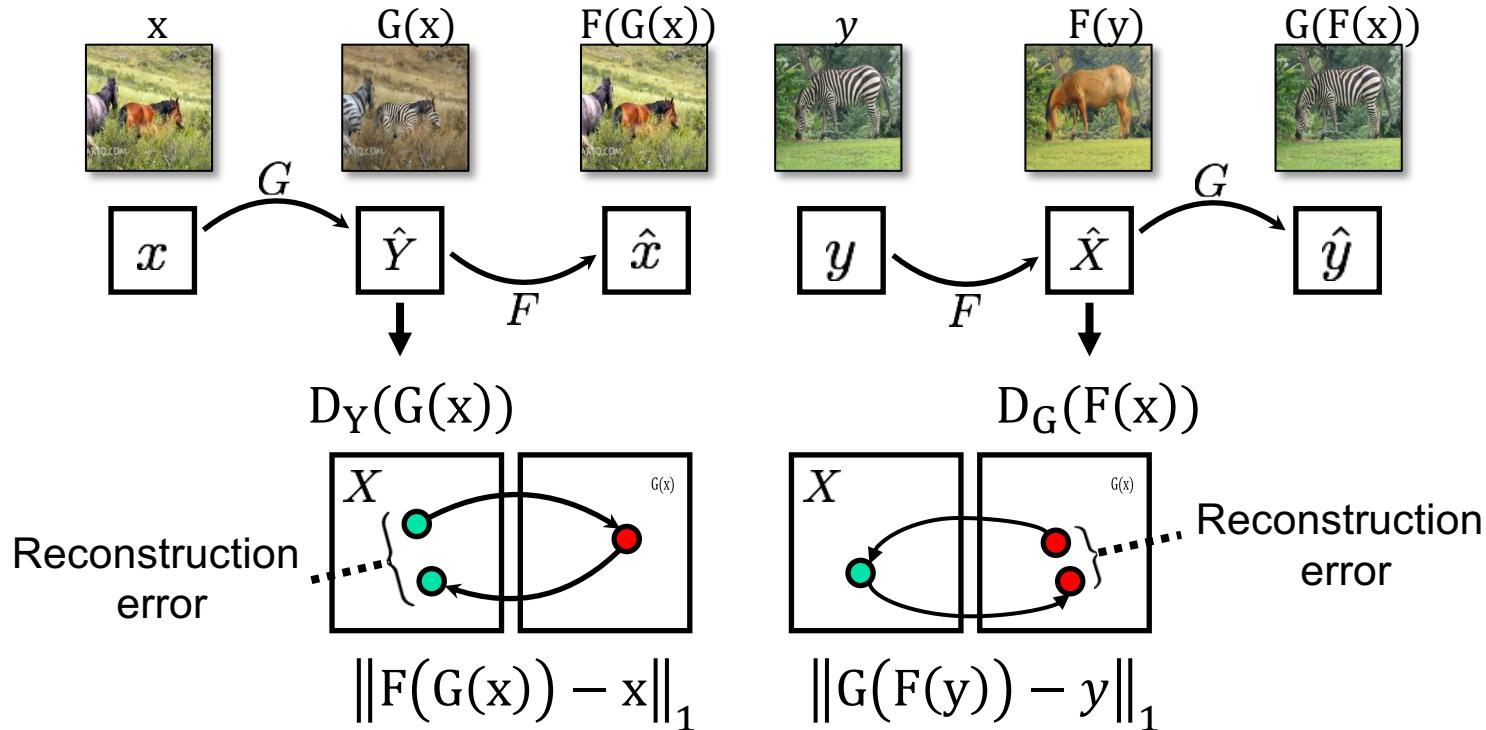
# Cycle Consistency Loss



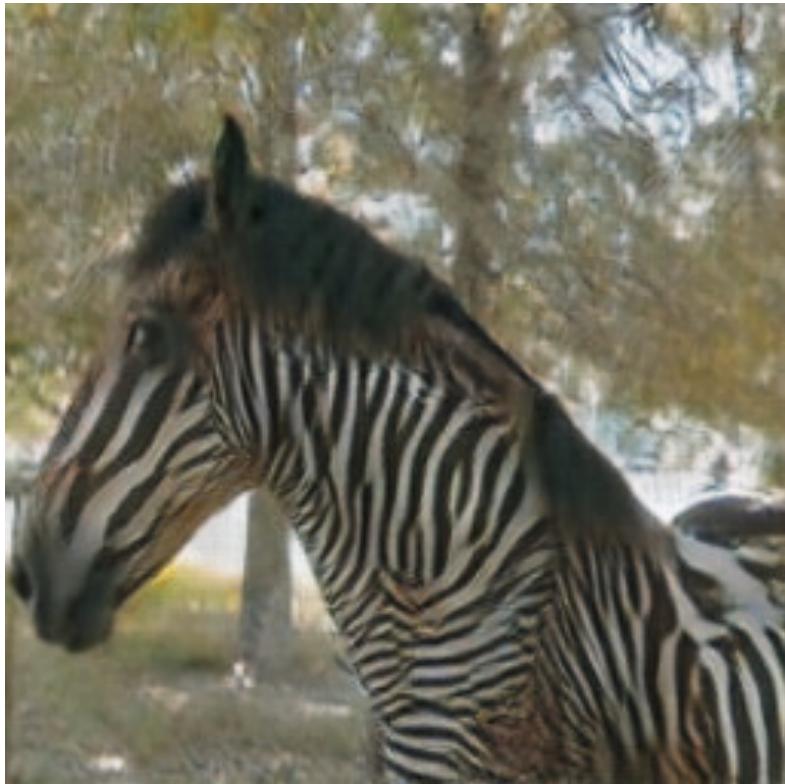
[Zhu\*, Park\*, Isola, and Efros, ICCV 2017]

# Cycle Consistency Loss

Could a network still cheat this? How?



See similar formulations [Yi et al. 2017], [Kim et al. [Zhu\*, Park\*, Isola, and Efros, ICCV 2017]





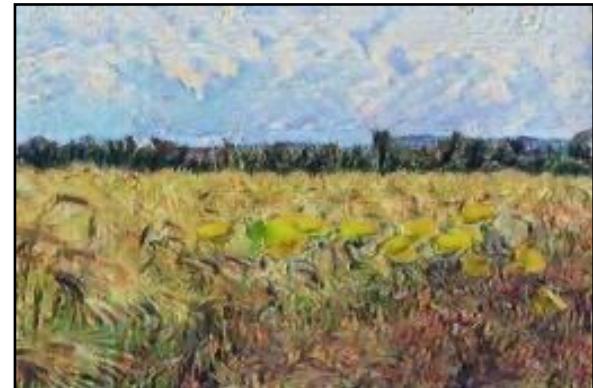
# Collection Style Transfer (no pairs exist)



Photograph  
@ Alexei Efros



Monet



Van Gogh



Cezanne



Ukiyo-e

Input



Monet



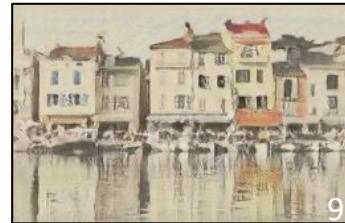
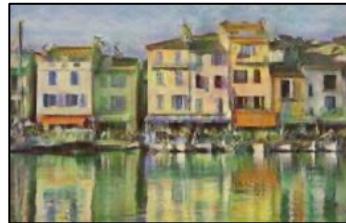
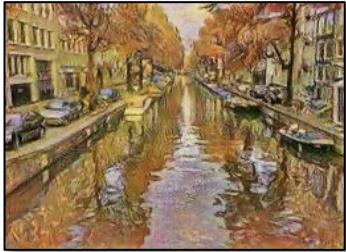
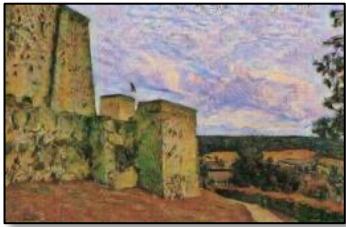
Van Gogh



Cezanne



Ukiyo-e



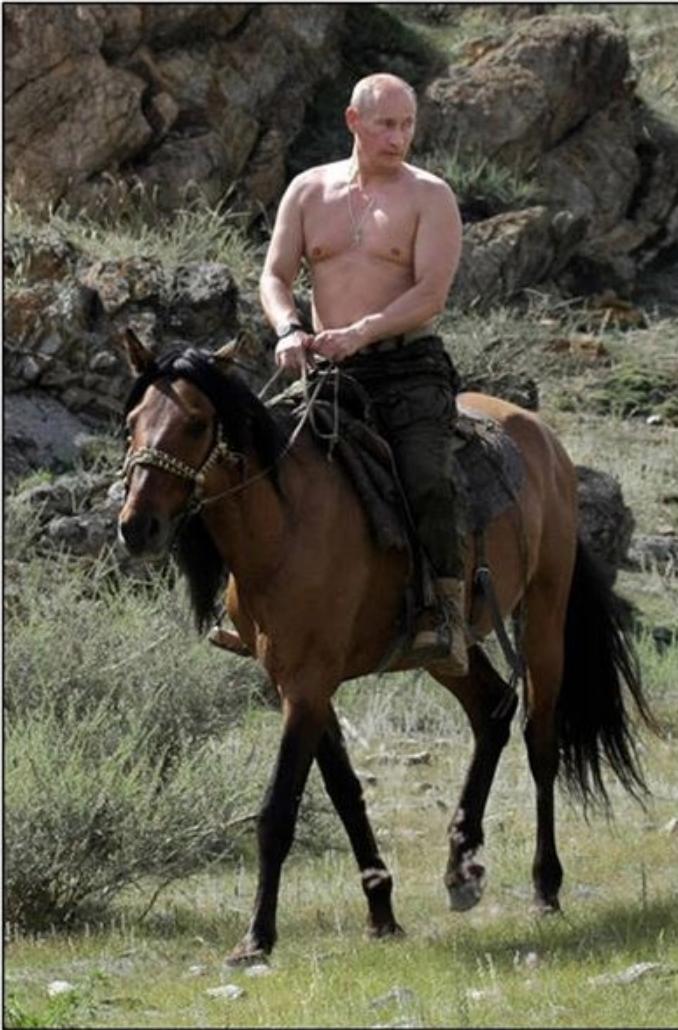
# Monet's paintings → photos



# Monet's paintings → photos









# CG2Real: GTA5 → real streetview



GTA5 CG Input



Output  
Inspired by [Johnson et al. 2011]

# Real2CG: real streetview → GTA



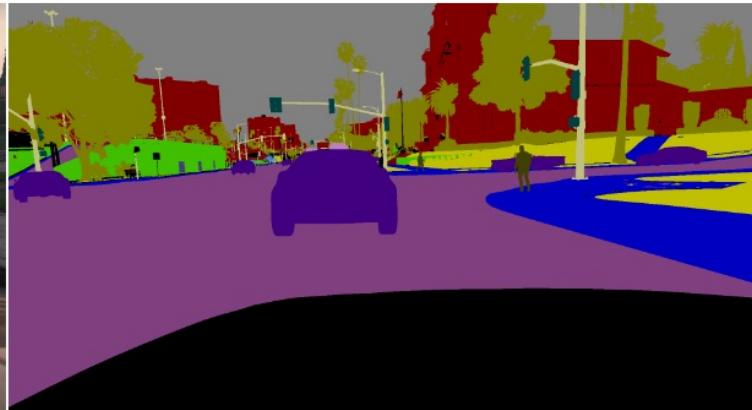
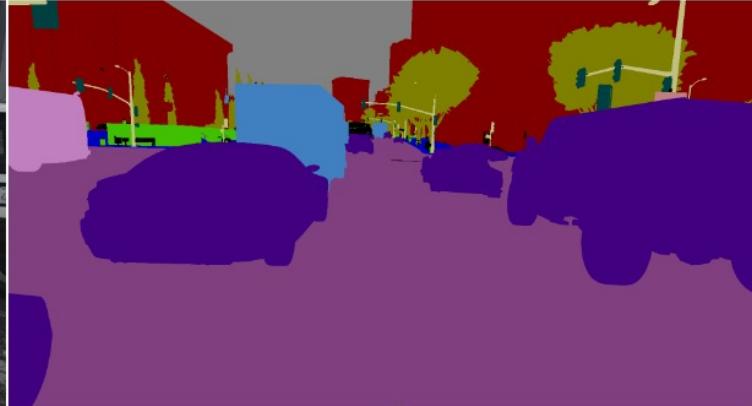
Cityscape Input

Output

# Synthetic Data as Supervision



GTA5 images



Segmentation labels

# Domain Adaptation with CycleGAN



Train on GTA5 data



Test on real images

	meanIOU	Per-pixel accuracy
Train on CG, test on Real	17.9	54.0

# Domain Adaptation with CycleGAN



Train on CycleGAN data



Test on real images

	meanIOU	Per-pixel accuracy
Train on CG, test on Real	17.9	54.0
Train on CycleGAN, test on Real	<b>34.8</b>	<b>82.8</b>

See Judy Hoffman's talk at 14:30 "Adversarial Domain Adaptation"

# Domain Adaptation with CycleGAN



Train on CycleGAN data

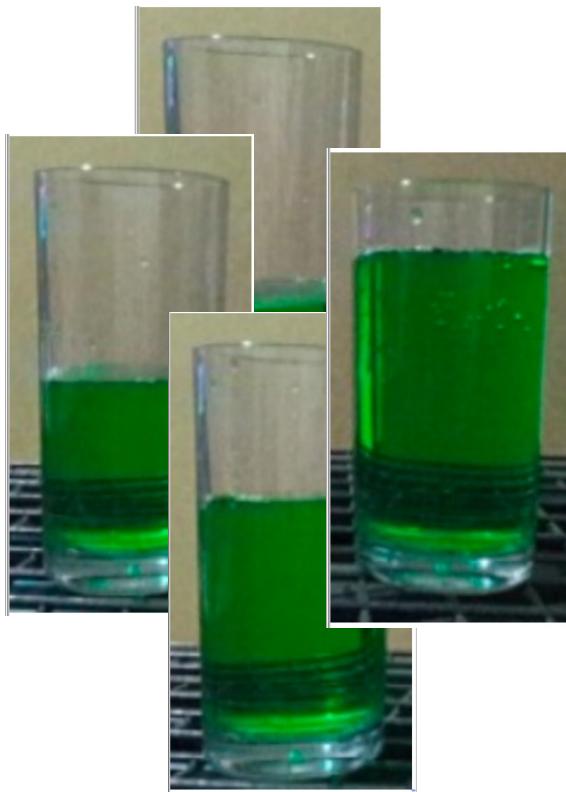


Test on real images

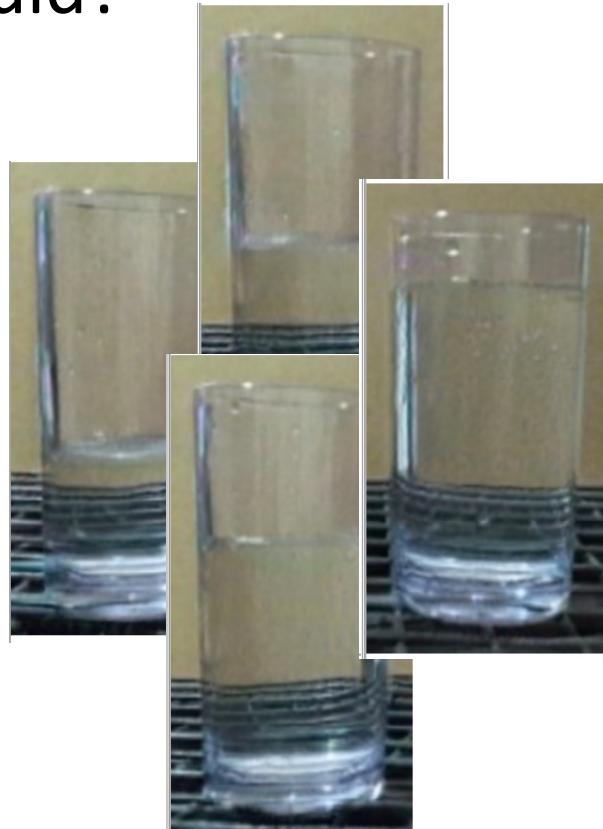
	meanIOU	Per-pixel accuracy
Oracle (Train and test on Real)	60.3	93.1
Train on CG, test on Real	17.9	54.0
Train on CycleGAN, test on Real	<b>34.8</b>	<b>82.8</b>

See Judy Hoffman's talk at 14:30 "Adversarial Domain Adaptation"

# Can we use Colored Liquid to Learn about Transparent Liquid?

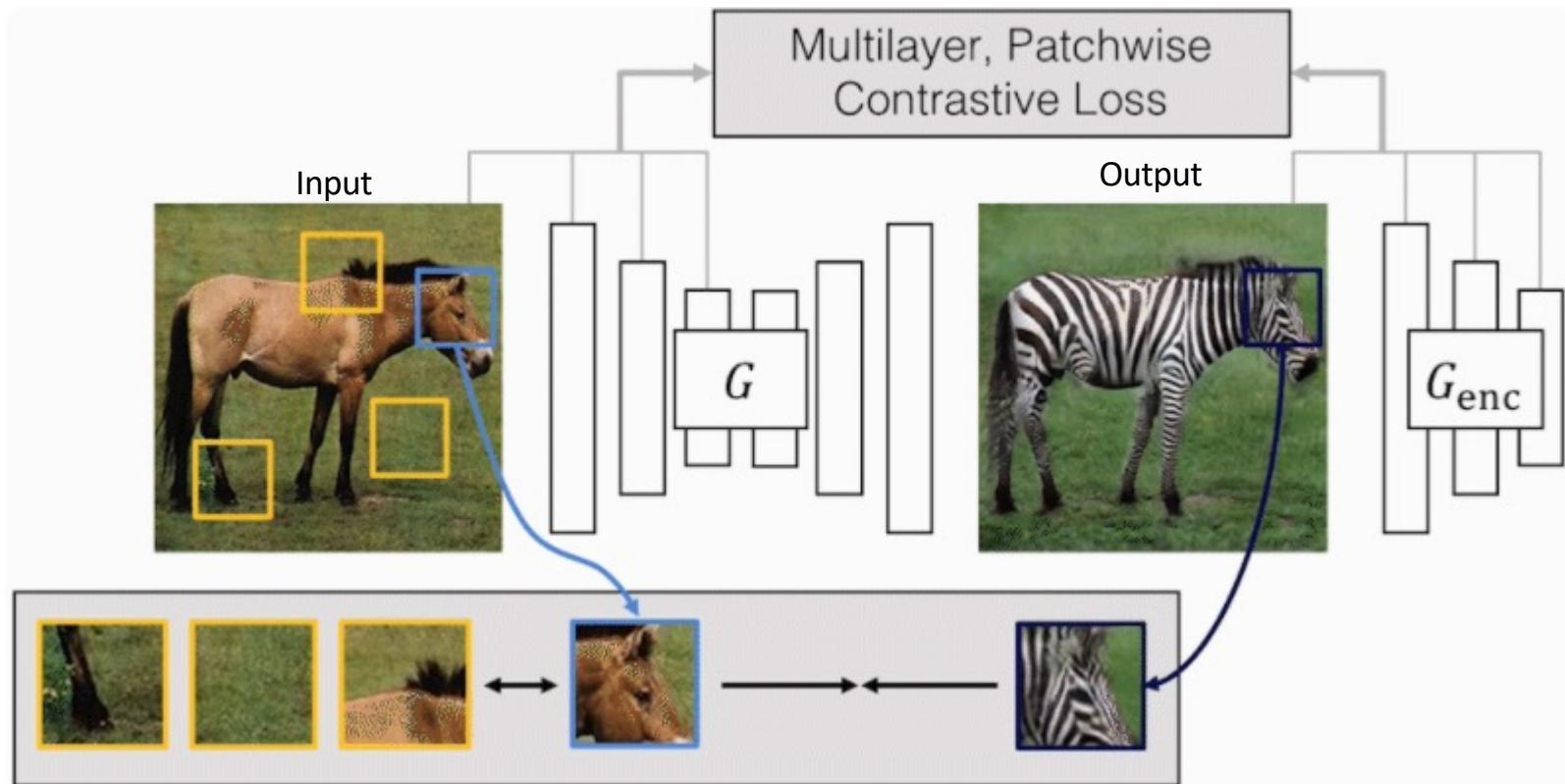


Dataset of Colored Liquids



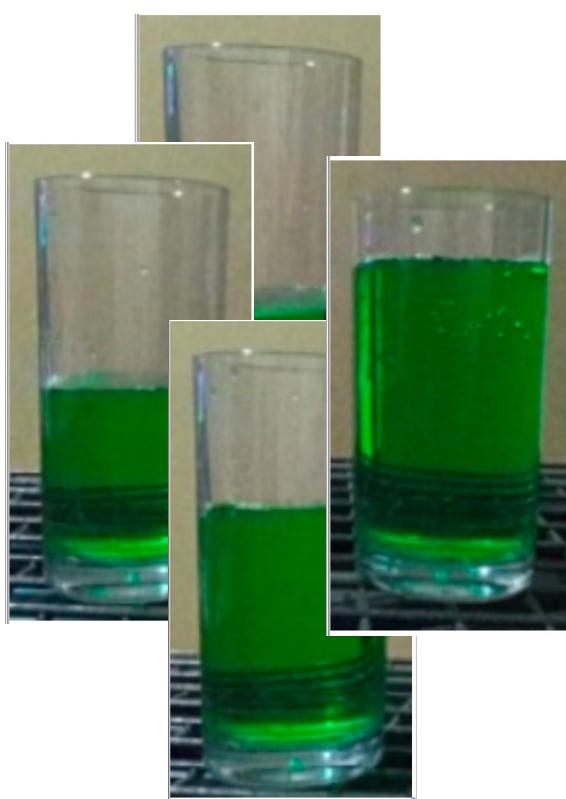
Dataset of Transparent Liquids

# Convert a horse into a zebra

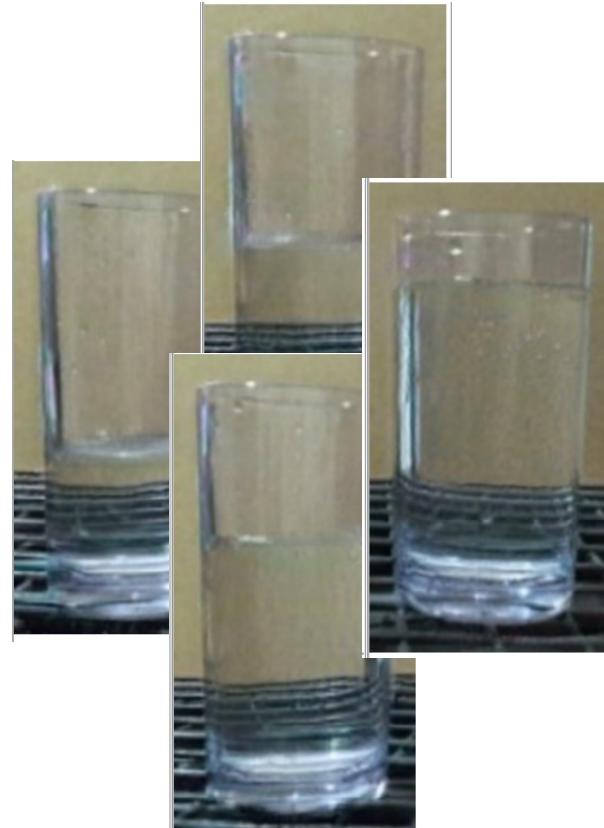


Park, Taesung, et al. "Contrastive learning for unpaired image-to-image translation." ECCV 2020.

# Convert Colored Liquid to Transparent Liquid

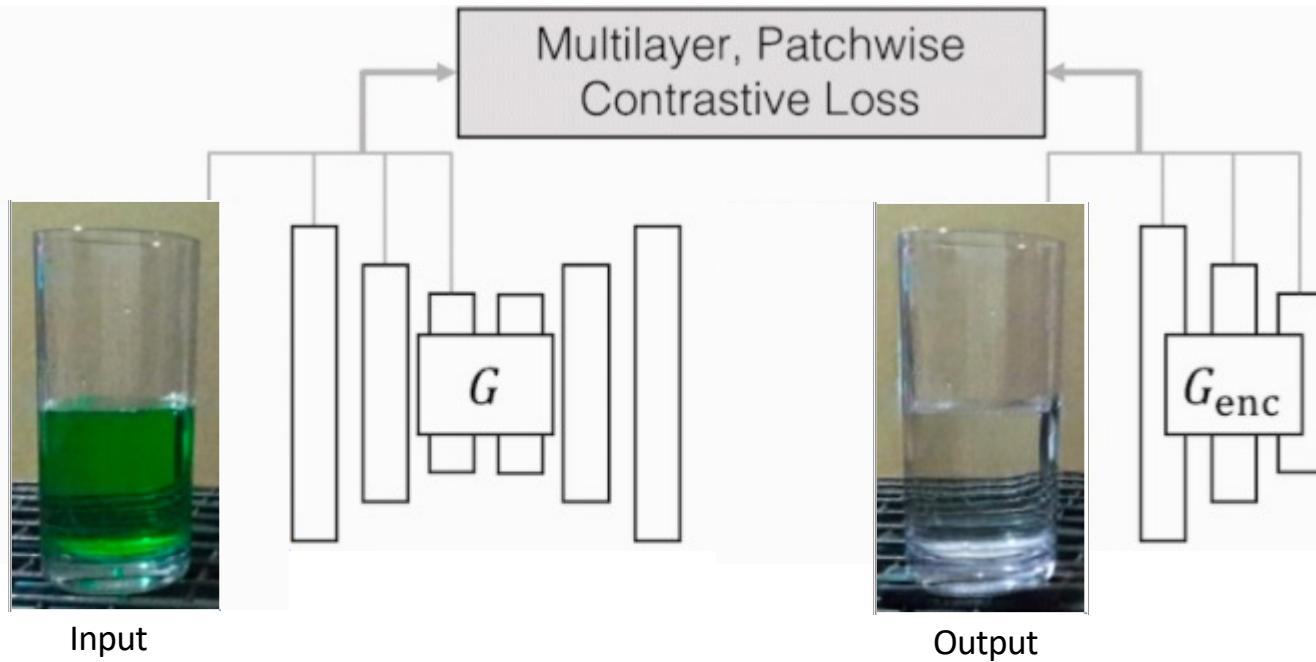


Dataset of Colored Liquids

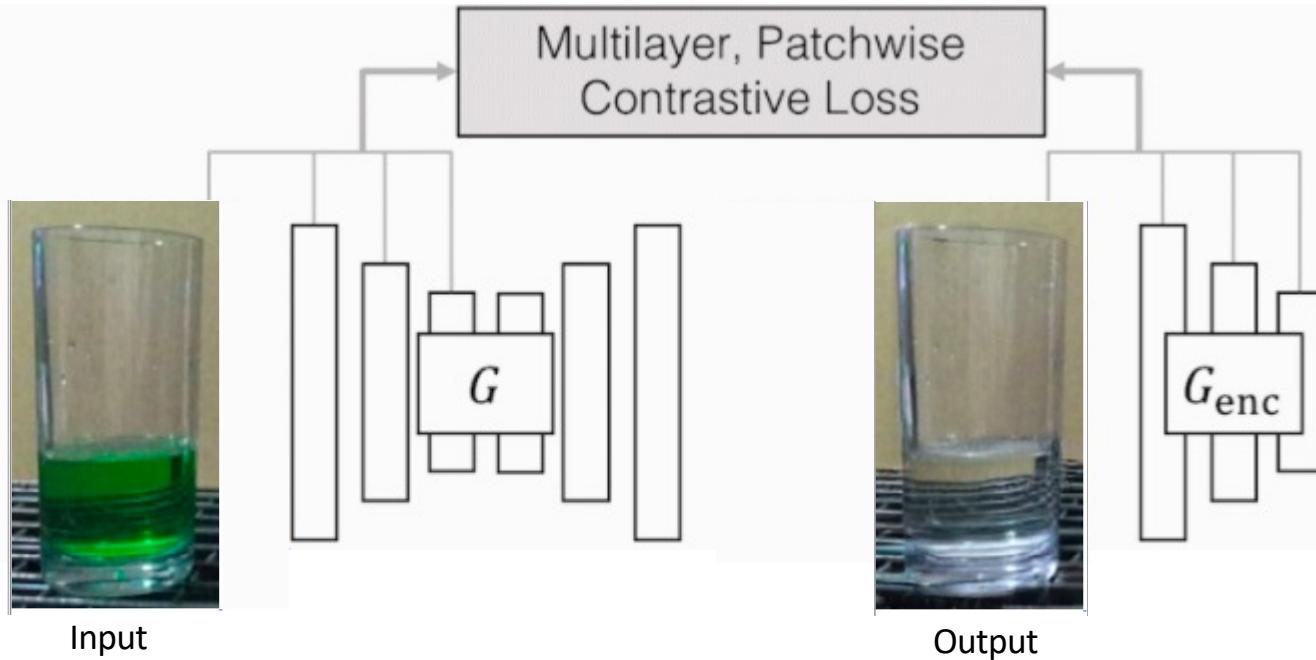


Dataset of Transparent Liquids

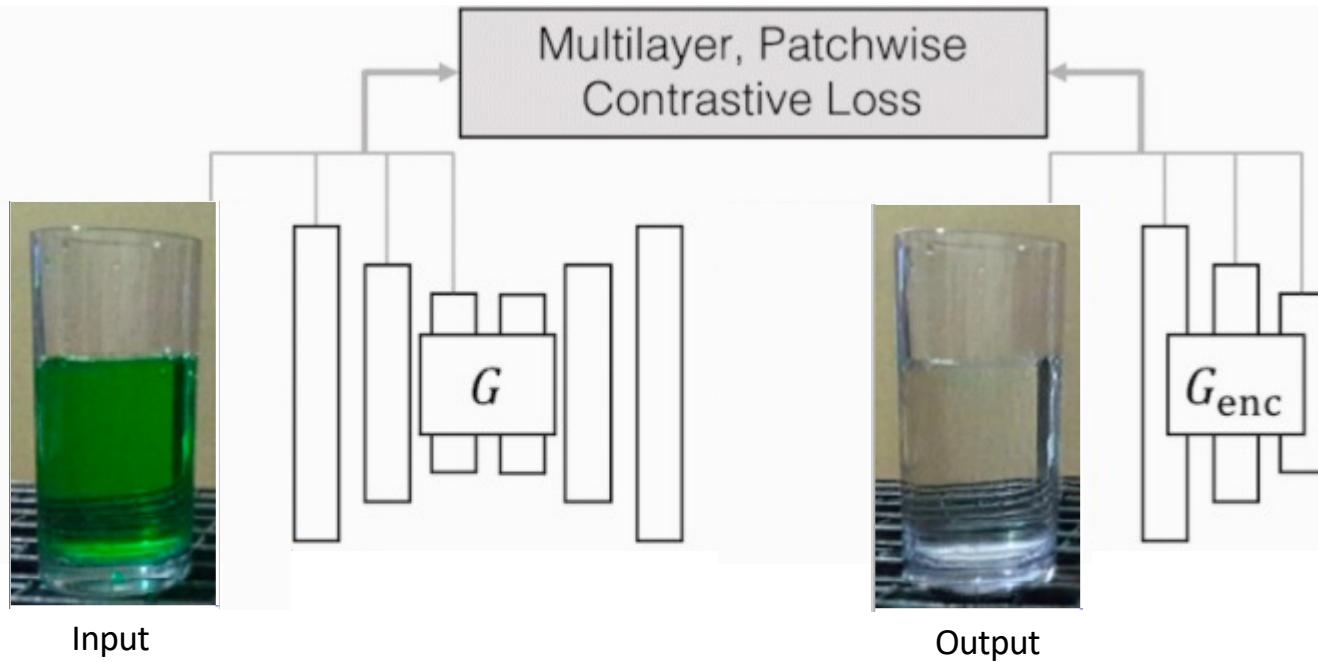
# Convert Colored Liquid to Transparent Liquid



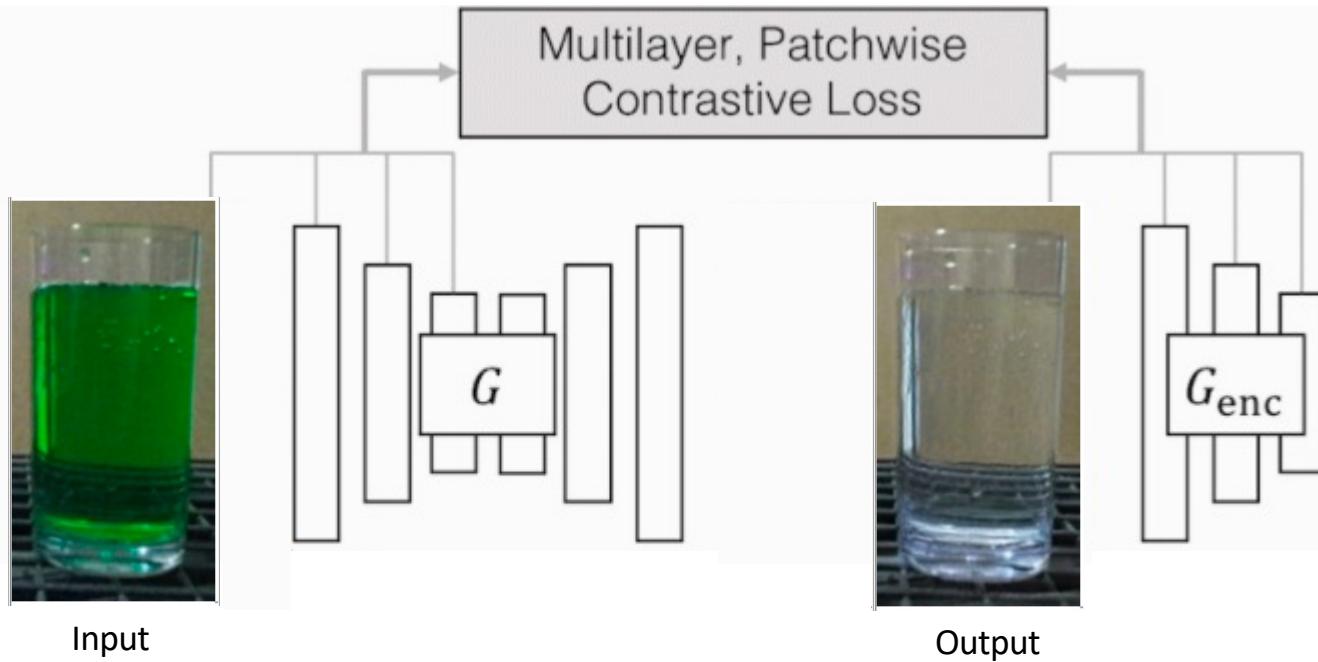
# Convert Colored Liquid to Transparent Liquid



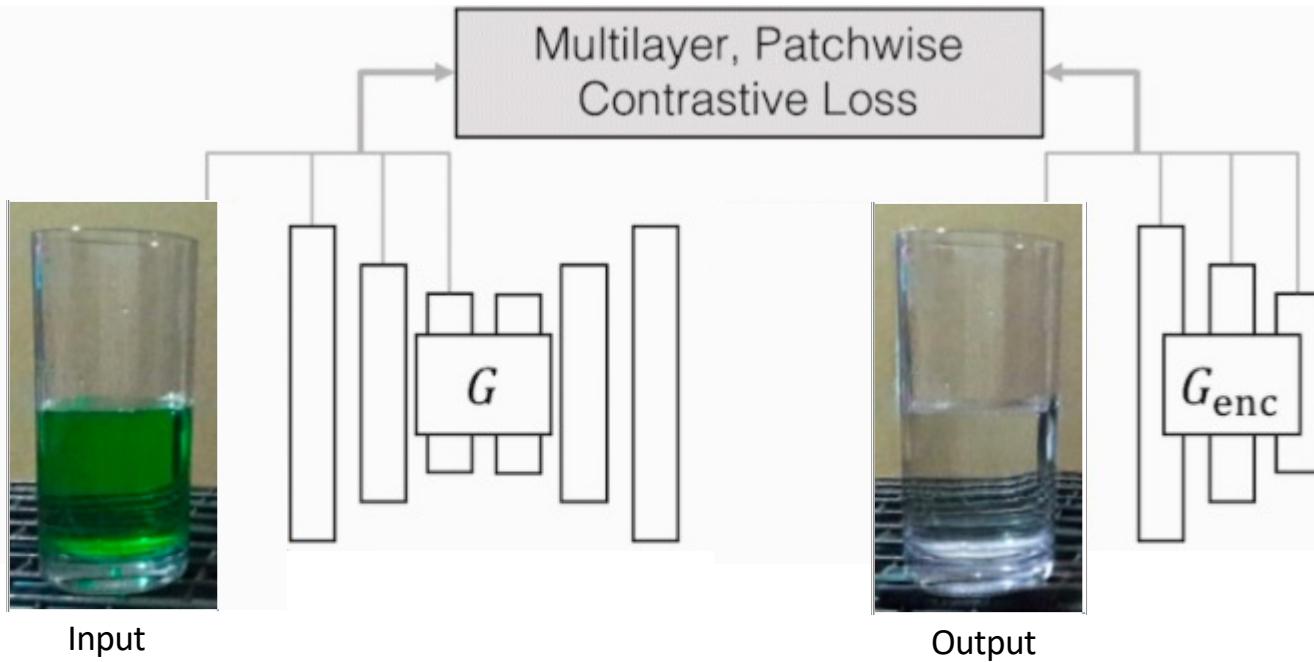
# Convert Colored Liquid to Transparent Liquid



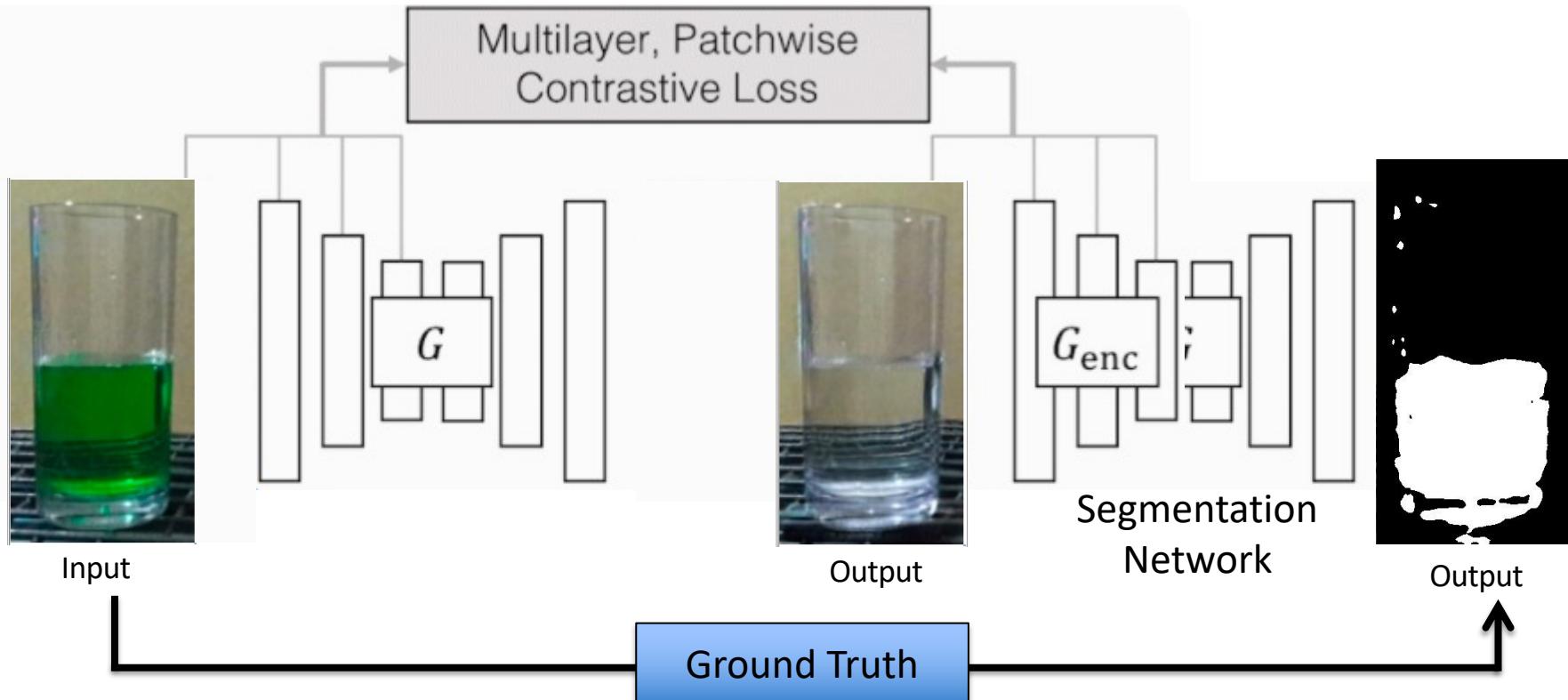
# Convert Colored Liquid to Transparent Liquid



# Convert Colored Liquid to Transparent Liquid



# Use colored liquid to get a segmentation mask for transparent liquid

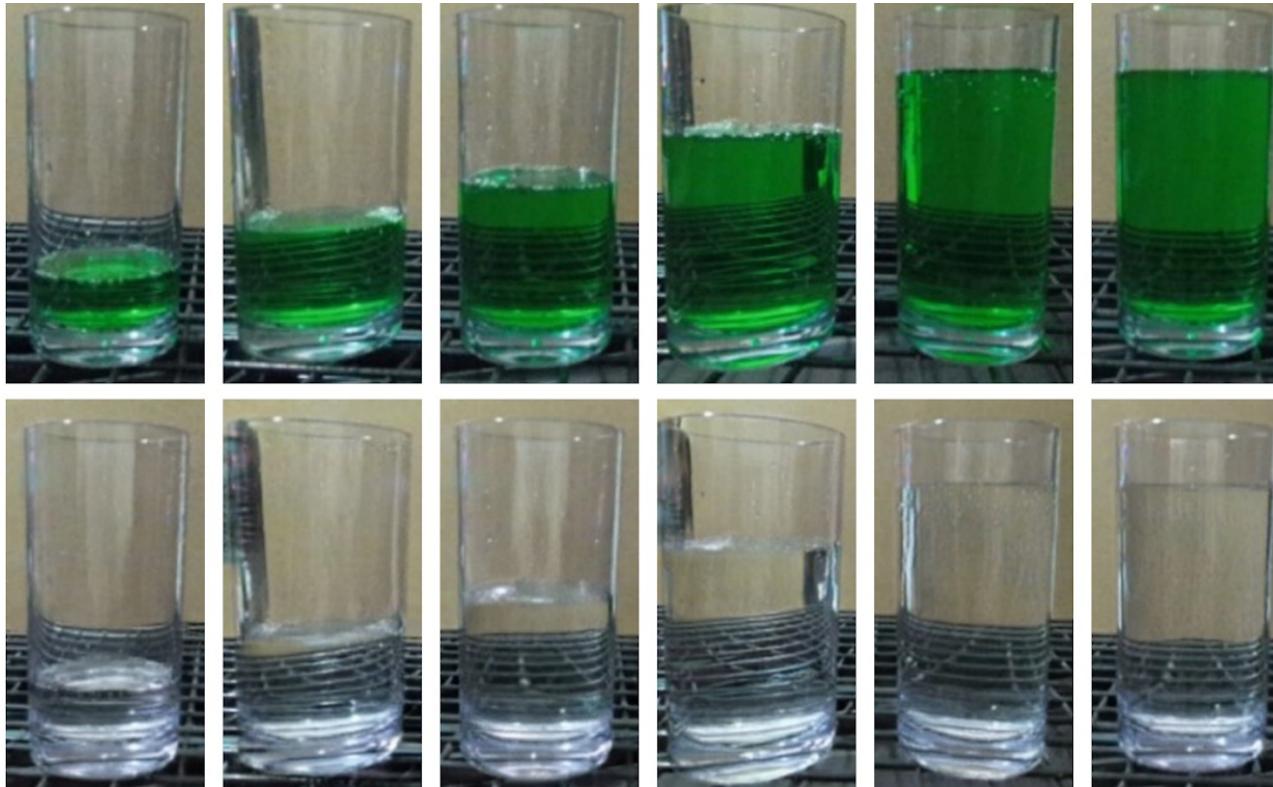


# Cup Segmentation: Translab

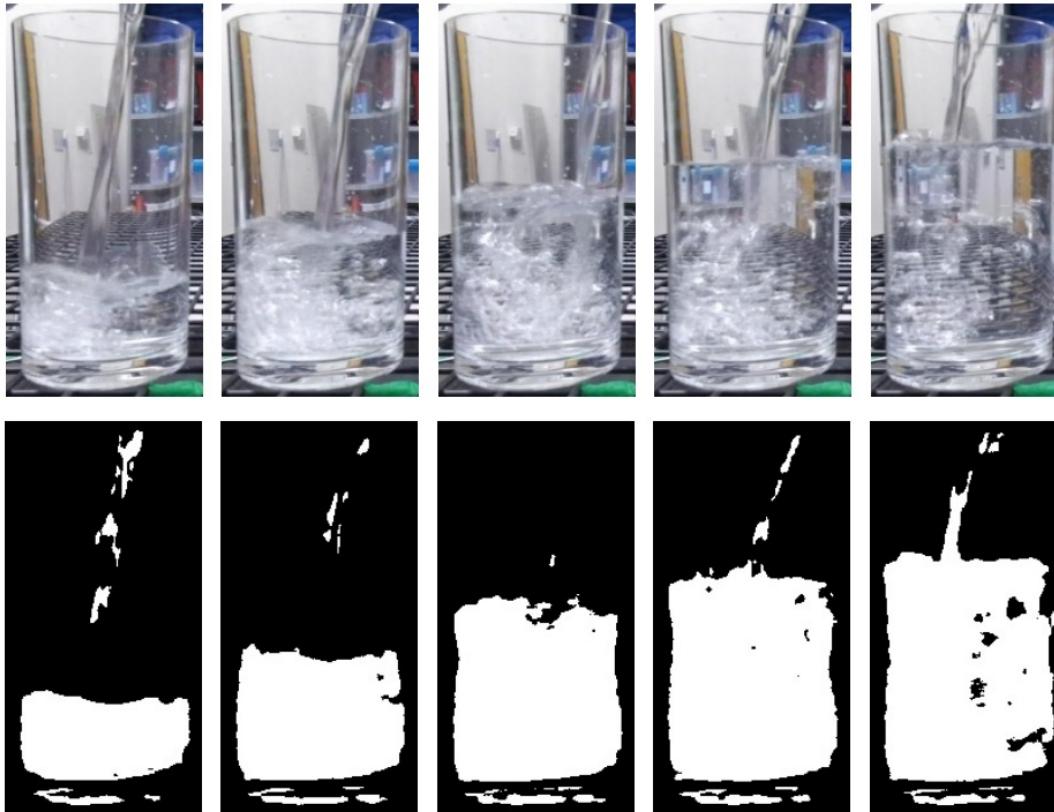


Xie, Enze, et al. "Segmenting transparent objects in the wild." *arXiv preprint arXiv:2003.13948* (2020).

# Results: Image Translation



# Results: Liquid Segmentation





**Short Liquid**



**Short Liquid Segmentation**



**Medium Liquid**



**Medium Liquid Segmentation**

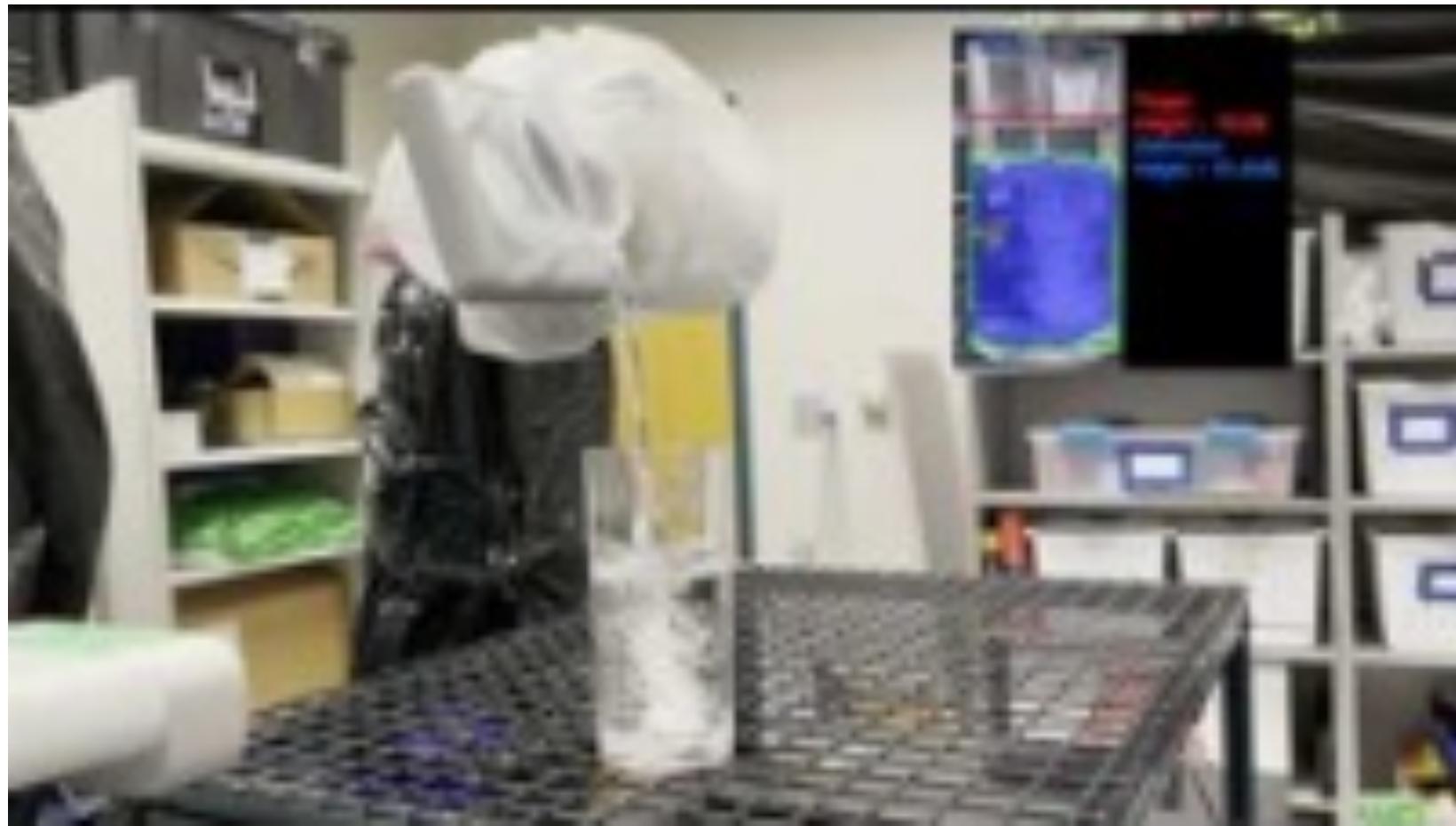


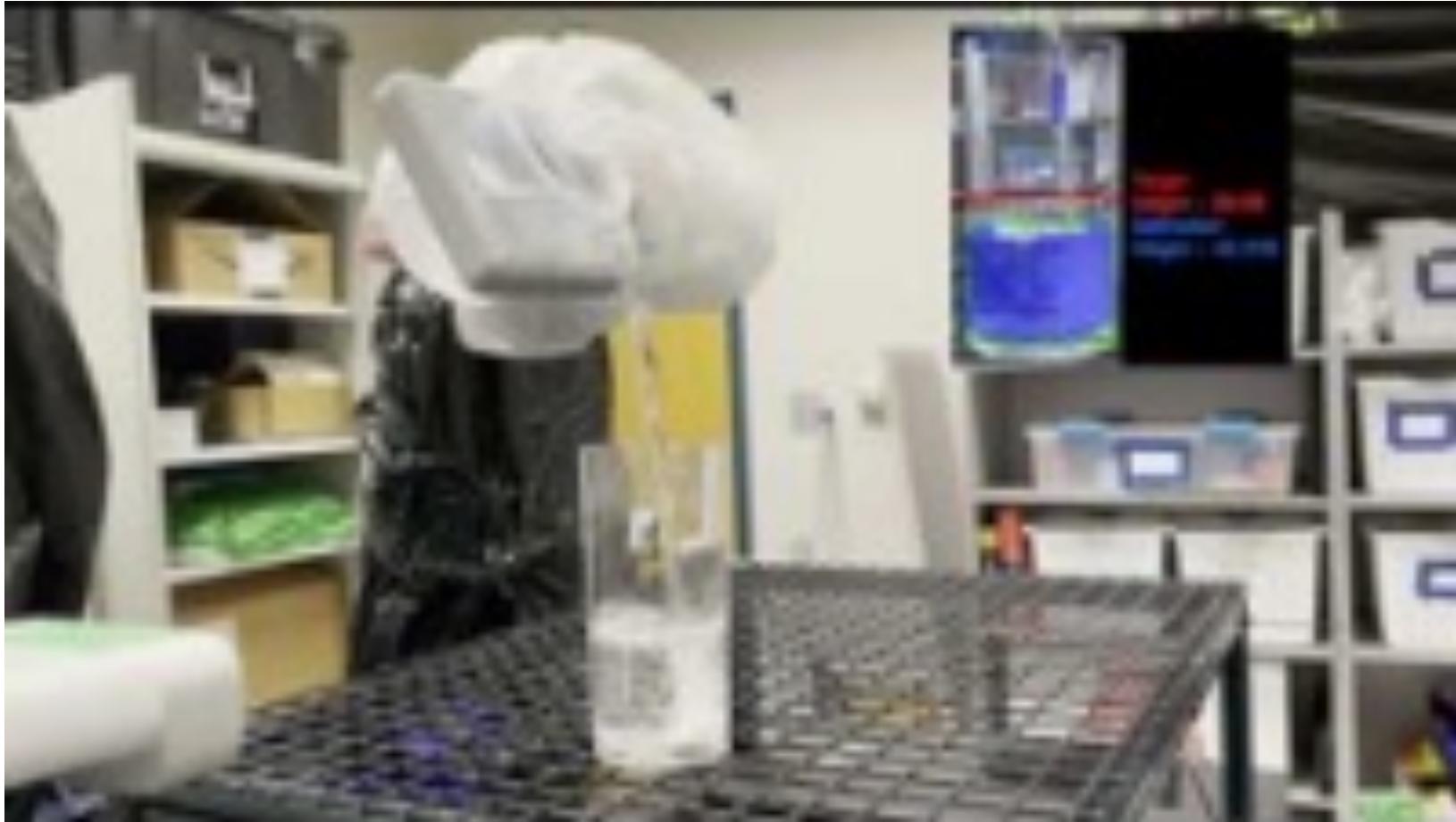
**Tall Liquid**



**Tall Liquid Segmentation**











# Progressive growing of GANs

---

## PROGRESSIVE GROWING OF GANs FOR IMPROVED QUALITY, STABILITY, AND VARIATION

**Tero Karras**

NVIDIA

**Timo Aila**

NVIDIA

**Samuli Laine**

NVIDIA

**Jaakko Lehtinen**

NVIDIA and Aalto University

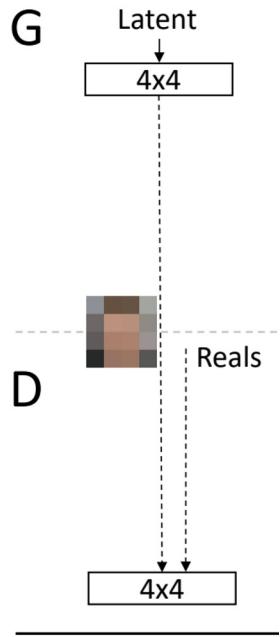
{tkarras, taila, slaine, jlehtinen}@nvidia.com

### ABSTRACT

We describe a new training methodology for generative adversarial networks. The key idea is to grow both the generator and discriminator progressively: starting from a low resolution, we add new layers that model increasingly fine details as training progresses. This both speeds the training up and greatly stabilizes it, allowing us to produce images of unprecedented quality, e.g., CELEBA images at  $1024^2$ . We also propose a simple way to increase the variation in generated images, and achieve a record inception score of 8.80 in unsupervised CIFAR10. Additionally, we describe several implementation details that are important for discouraging unhealthy competition between the generator and discriminator. Finally, we suggest a new metric for evaluating GAN results, both in terms of image quality and variation. As an additional contribution, we construct a higher-quality version of the CELEBA dataset.

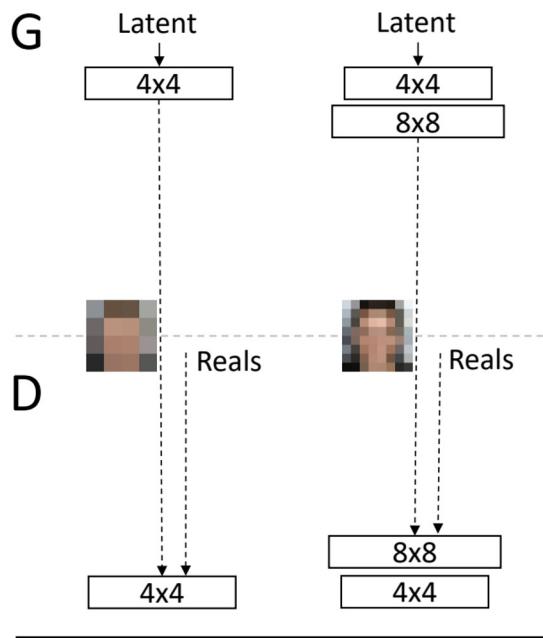
Karras et al 2017

# Progressive growing of GANs



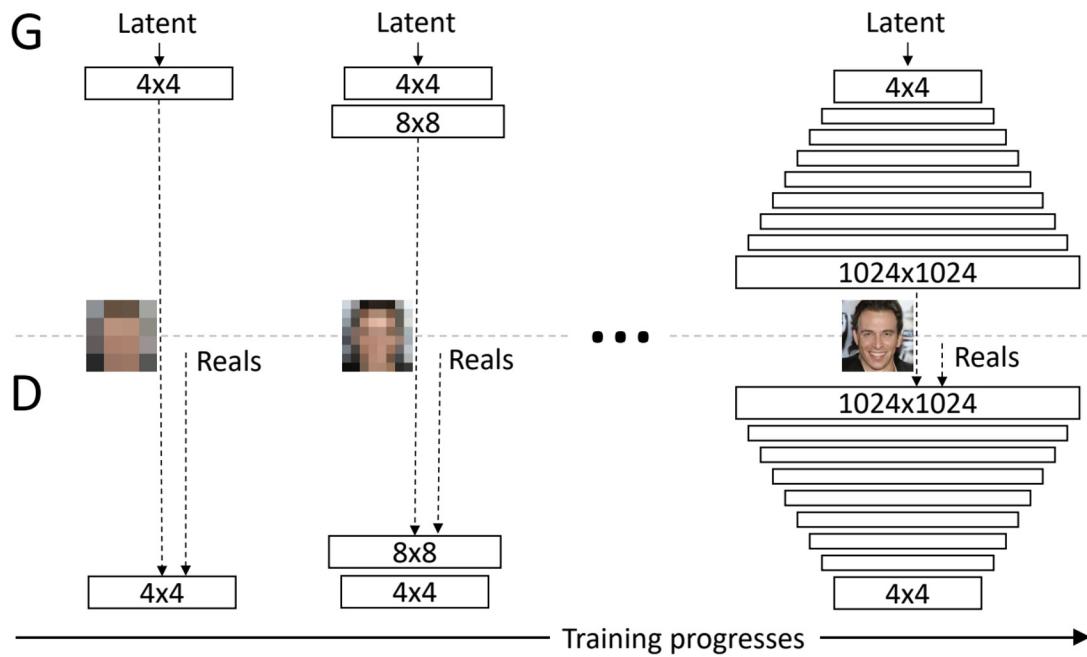
Karras et al 2017

# Progressive growing of GANs



Karras et al 2017

# Progressive growing of GANs



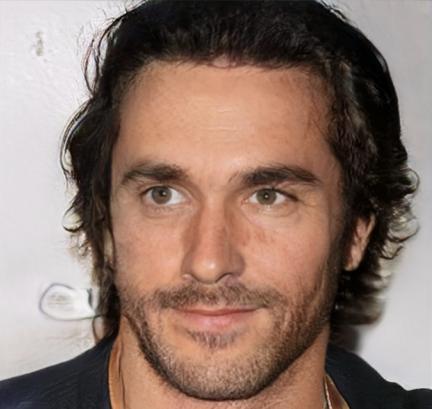
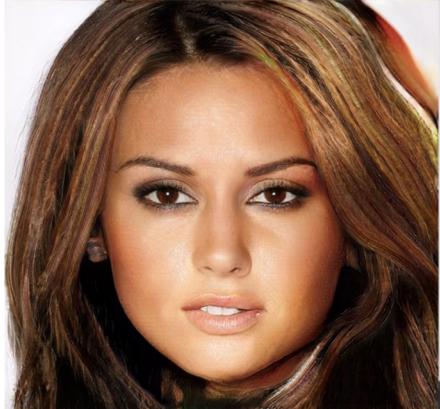
Karras et al 2017

# Progressive growing of GANs



Karras et al 2017

# Progressive growing of GANs



# StyleGAN

Latent  $\mathbf{z} \in \mathcal{Z}$

Normalize

Fully-connected

PixelNorm

Conv 3×3

PixelNorm

4×4

Upsample

Conv 3×3

PixelNorm

Conv 3×3

PixelNorm

8×8

(a) Traditional

Latent  $\mathbf{z} \in \mathcal{Z}$

Normalize

Mapping network  $f$

FC

FC

FC

FC

FC

FC

$\mathbf{w} \in \mathcal{W}$

Synthesis network  $g$

Noise

Const 4×4×512

style

AdaIN

Conv 3×3

style

AdaIN

4×4

Upsample

Conv 3×3

style

AdaIN

Conv 3×3

style

AdaIN

8×8

(b) Style-based generator

# StyleGAN - Adaptive Instance Norm

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

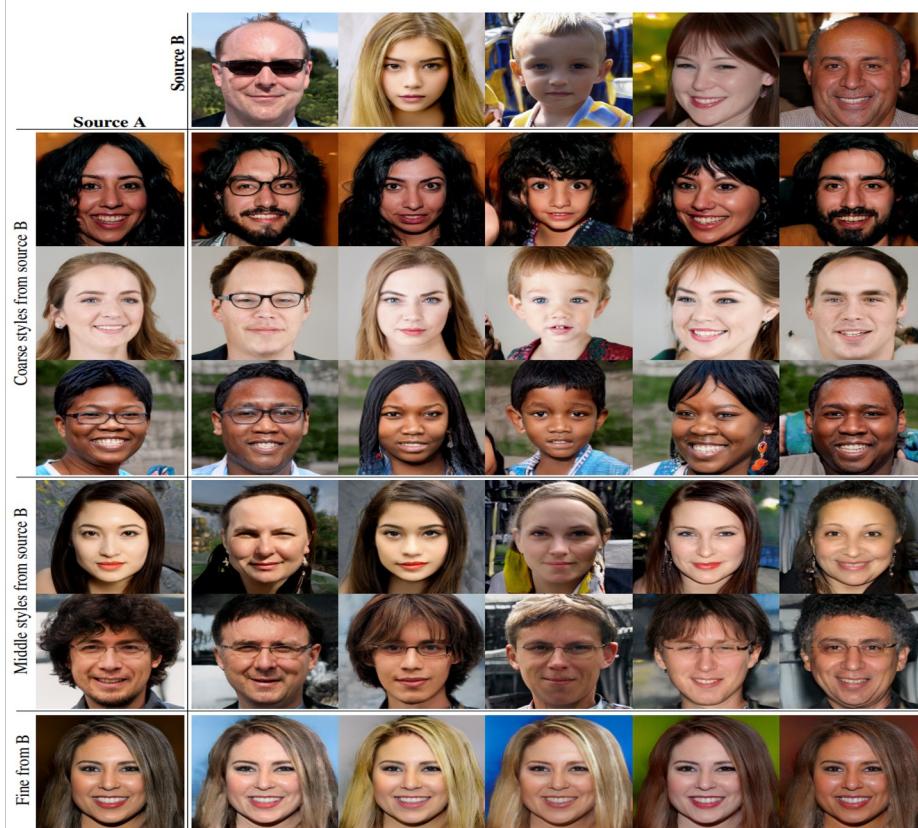
The diagram illustrates the AdaIN equation with several annotations:

- A green bracket labeled "features" spans the entire term  $\mathbf{x}_i - \mu(\mathbf{x}_i)$ .
- A green bracket labeled "feature mean" spans the term  $\mu(\mathbf{x}_i)$ .
- A green bracket labeled "feature variance" spans the term  $\sigma(\mathbf{x}_i)$ .
- A green bracket labeled "normalized features" spans the term  $\frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)}$ .
- A green bracket labeled "Style variables" spans the terms  $\mathbf{y}_{s,i}$  and  $\mathbf{y}_{b,i}$ .

# StyleGAN results



# StyleGAN - Style Transfer



# Summary

- Basic GAN
- Evaluating GANs
- Conditional GAN
- CycleGAN
- Latent vector arithmetic
- Latent vector interpolation
- Progressive Growing of GANs
- StyleGAN