Deep Generative Models:

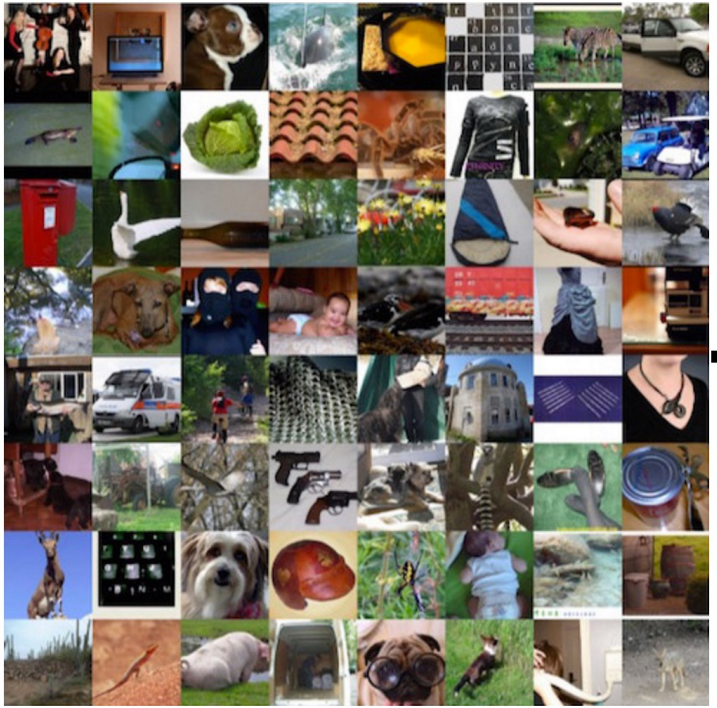
Generative Adversarial Networks

Slides adapted from Pieter Abbeel

This session will be audio recorded for educational

Generative Model

Real Data



Generative Model

Generated Data

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

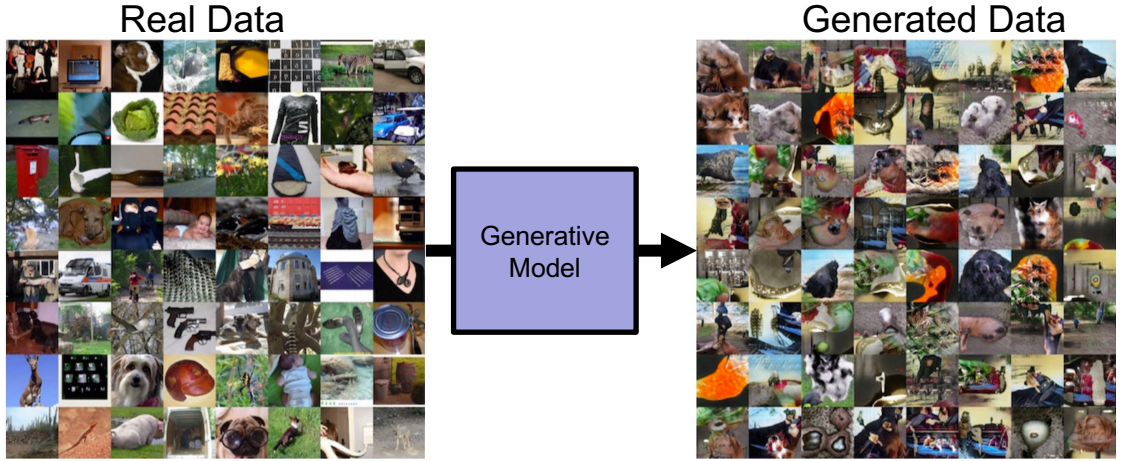
Building a sampler

- How about this generative model?

```
1) Load training data
2) Sample a training image
3) Return training image
```

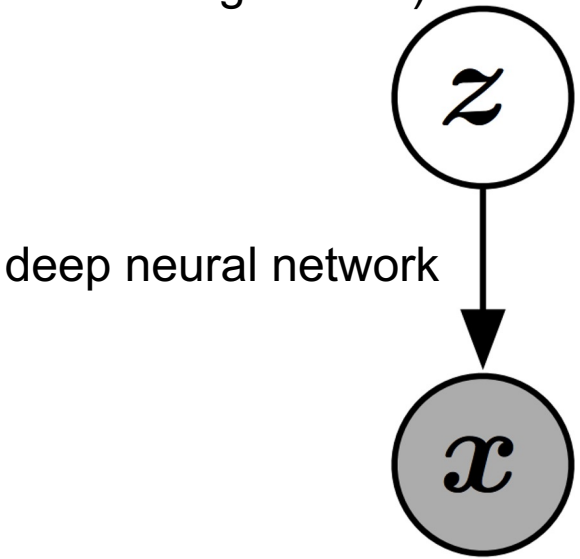
Building a sampler

- You don’t just want to sample the exact data points you have.
- You want to build a generative model that can understand the underlying distribution of data points and
  - output samples representative of the underlying factors of variation in the training distribution.
  - Example: new objects with unseen poses, lighting, etc.



Implicit Models

Sample z from p(z) = fixed noise source distribution (uniform or gaussian).



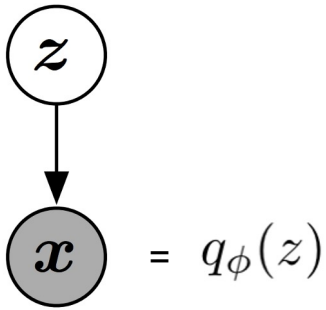
z represents the latent factors of variation: what objects to generate, where to place them in the image, lighting, viewpoint, etc

sample x (e.g. an image)

- In “Implicit” models we cannot compute p(x) for an arbitrary x
- We can only sample from an implicit model

Implicit Models

- We sample the latent variables $z \sim p(z)$
- We train a generative model $q_\phi(z)$ to output samples x
- We want these samples to be similar to the training data: $p_{\text{data}} : x_1, x_2, \cdots, x_n$
- Let us call the distribution of generated samples p_{model}
- What loss should we put train the generator?
- We want a loss that minimizes the "distance" between p_{model} and p_{data}
- **GAN trick: learn a second neural network to produce a loss!**



Generative Adversarial Networks

Generative Adversarial Nets

Ian J. Goodfellow[*], **Jean Pouget-Abadie**[†], **Mehdi Mirza**, **Bing Xu**, **David Warde-Farley**, **Sherjil Ozair**[‡], **Aaron Courville**, **Yoshua Bengio**[§]
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.

NeurIPS 2014

Generative Adversarial Networks

Real image

x sampled from data

Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

Generative Adversarial Networks

Real image

x sampled from data

Input noise z

Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

Generative Adversarial Networks

Real image

x sampled from data

x sampled from model

Differentiable function G

Input noise z

Generated image

Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

Generative Adversarial Networks

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

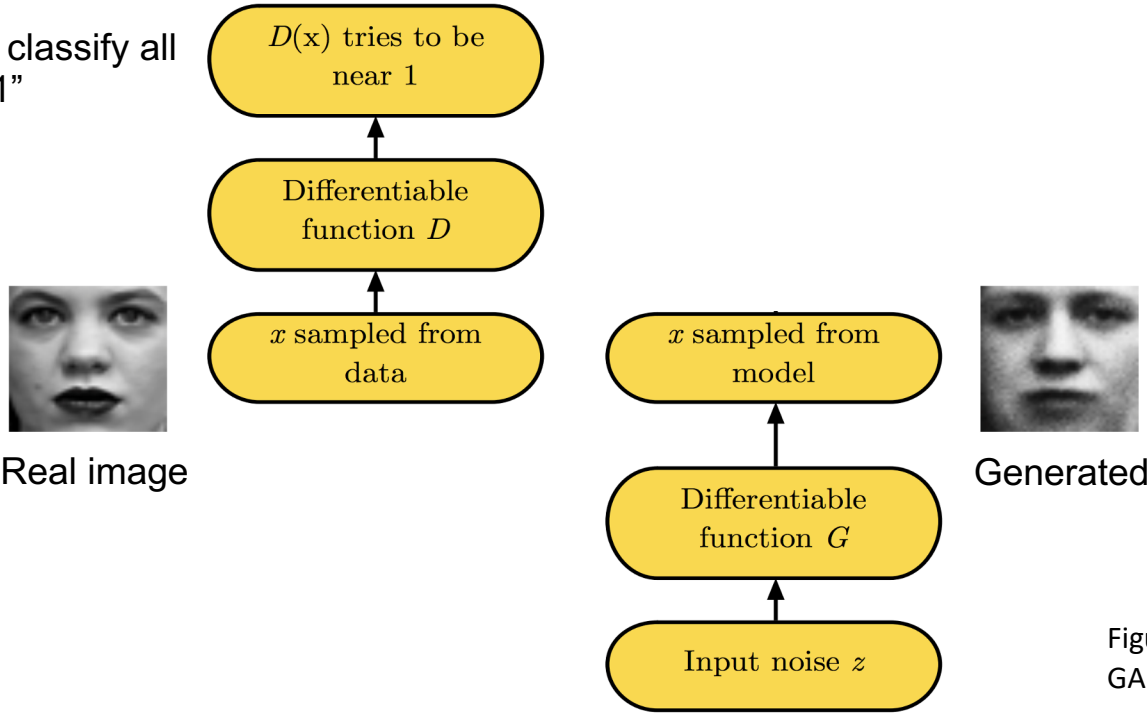
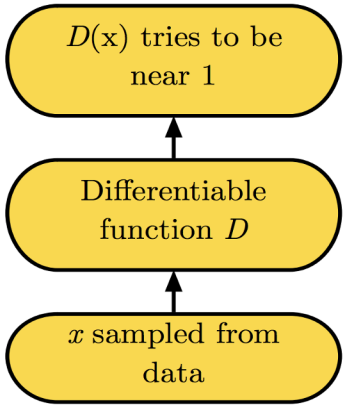


Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

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”

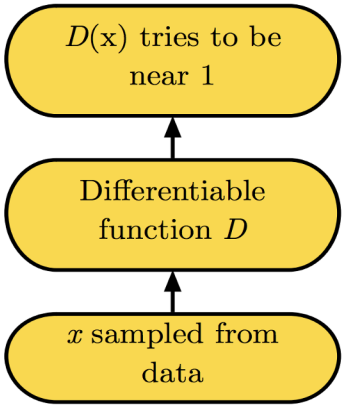
Generated image

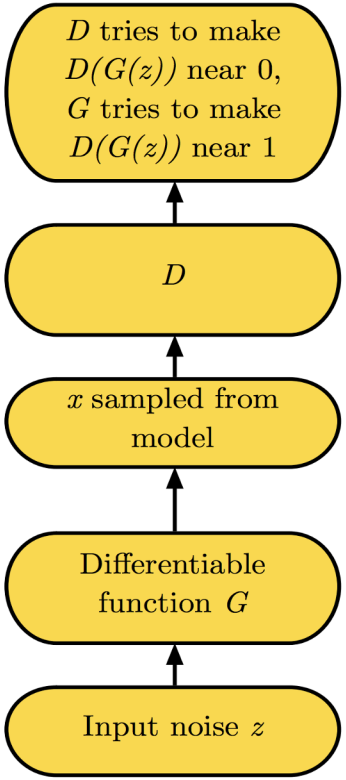
Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

Generative Adversarial Networks

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

Real image





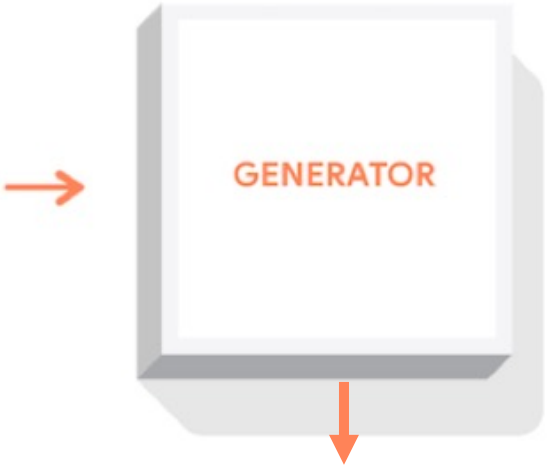
Discriminator tries to classify all generated data with label “0”

Generator tries to “fool” the discriminator

Generated image

Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

GENERATOR
"The Artist"
A neural network trying to create pictures of cats that look real.



DISCRIMINATOR
"The Art Critic"
A neural network examining cat pictures to determine if they're real or fake.

Thousands of real-world images labeled "CAT"

Review: Binary Cross-Entropy Loss

Classifier: p = f(x), where p = probability of x belonging to the positive class

$$Loss = \begin{cases} -\log p & \text{for positive examples} \\ -\log(1-p) & \text{for negative examples} \end{cases}$$

Want to maximize log p or minimize –log p

Want to minimize p or maximize 1-p or maximize log (1-p) or minimize -log (1-p)

$$Loss = -(y \log p + (1-y) \log(1-p))$$

Positive examples: y = 1 **Negative examples: y = 0**

$$Loss = -(y \log f(x) + (1-y) \log(1-f(x)))$$

Generative Adversarial Networks

$$Loss = -(y \log f(x) + (1 - y) \log(1 - f(x)))$$

Positive examples: y = 1 **Negative examples: y = 0**

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

real data — generated data

GAN objective

- Two player minimax game between generator (G) and discriminator (D)
- (D) tries to maximize the log-likelihood for the binary classification problem

  - data: real (1)

  - generated: fake (0)
- (G) tries to minimize the log-probability of its samples being classified as “fake” by the discriminator (D)

Generative Adversarial Networks

GAN objective:

real data generated data

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

- 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)

Generative Adversarial Networks

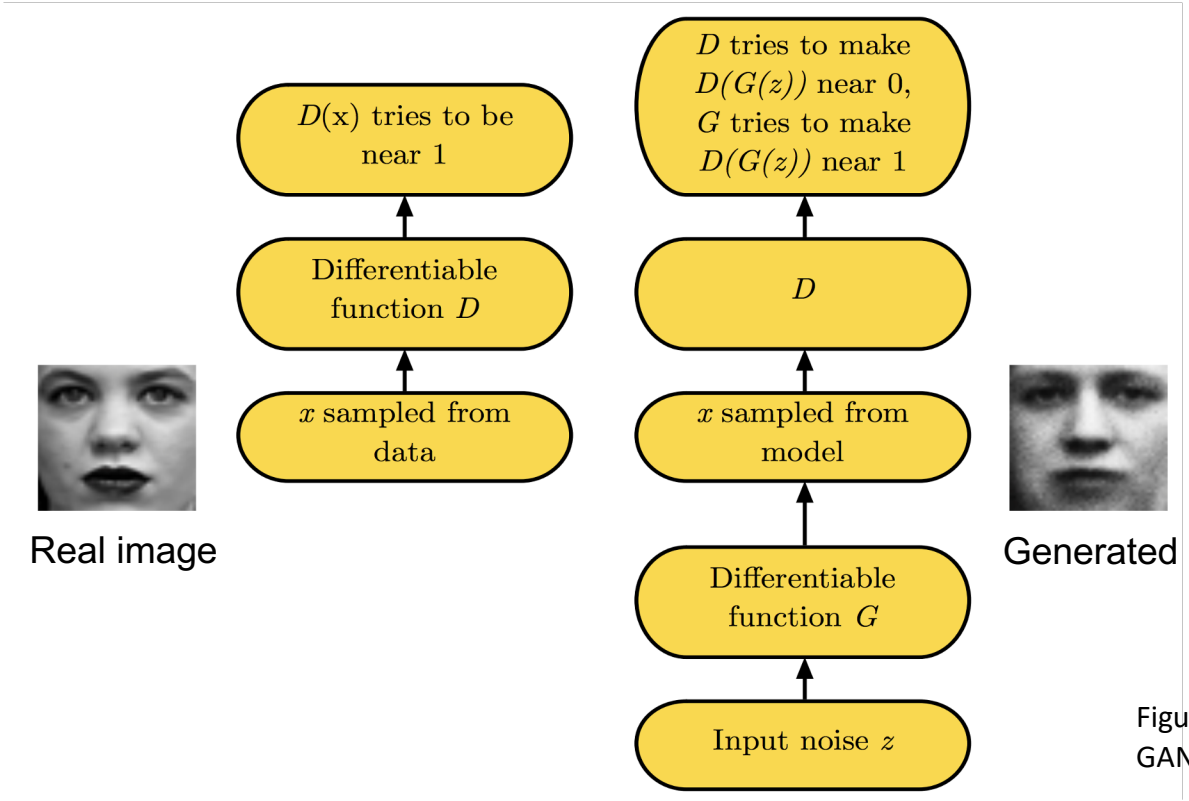


Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

For the optimal generator, what is the output of the optimal discriminator?

GAN: Bayes-Optimal Discriminator

- For a fixed **suboptimal** generator, what's the optimal discriminator?

$$\implies D^*(x) = \frac{p_{\text{data}}(x)}{(p_{\text{data}}(x) + p_g(x))}$$

- Example: Training data $p_{data}(x)$ contains 50% cats and 50% dogs
- Suboptimal generator produces only perfectly photorealistic dogs (100%)
- **The discriminator sees an image of a cat. What is the probability that it is a real image?**
  - **100%**
- **The discriminator sees an image of a dog. What is the probability that it is a real image?**
  - **33%**
- **What should the generator do to maximally fool the discriminator?**
  - **Generate more cats**

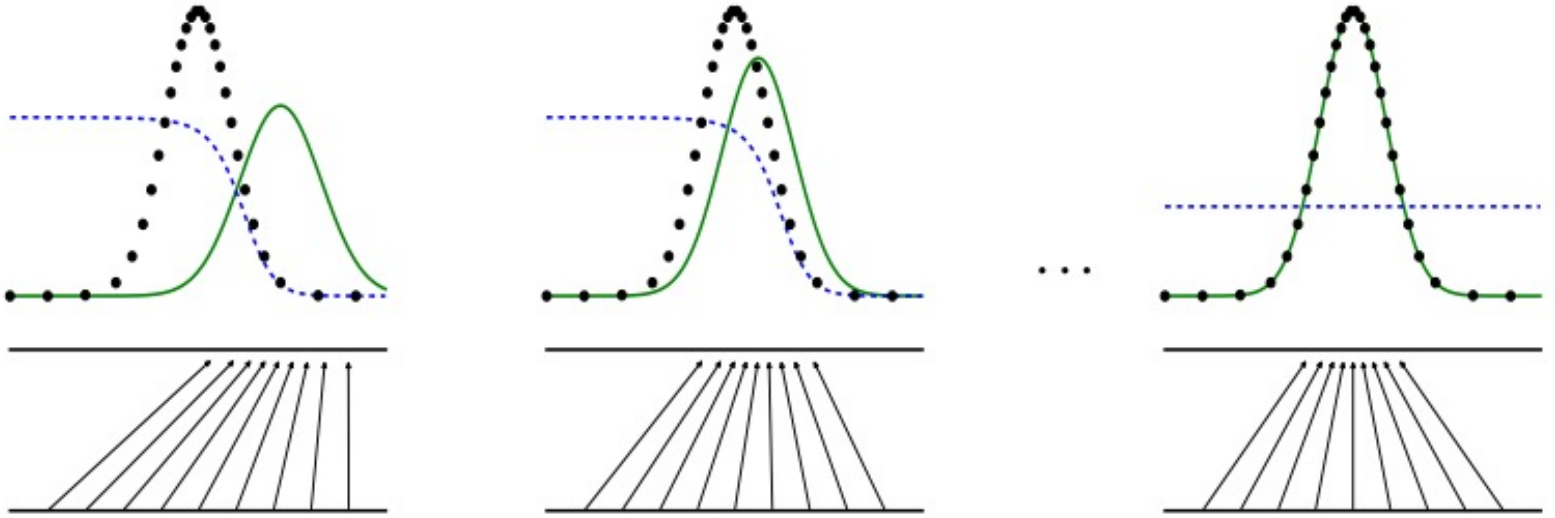
GAN: Bayes-Optimal Discriminator

- For a fixed suboptimal generator, the optimal discriminator is:

$$\implies D^*(x) = \frac{p_{\text{data}}(x)}{(p_{\text{data}}(x) + p_g(x))}$$

- For the **optimal** generator, $p_{data}(x) = p_g(x) \ \ \forall x$
- Example: Training data $p_{data}(x)$ contains 50% cats and 50% dogs
- Optimal generator produces 50% perfectly photorealistic dogs and 50% perfectly photorealistic cats

- Then optimal discriminator for the optimal discriminator is: $D^*(x) = \frac{1}{2}$



Near-optimal generator

Optimal generator and discriminator

Blue: Discriminator probability
Black: Data distribution
Green: Generator distribution

[Figure Source: Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.]

GAN

See it in action: https://poloclub.github.io/ganlab/

GAN Progress

Picture credit: Ian Goodfellow

StyleGAN (2020)

GAN Art

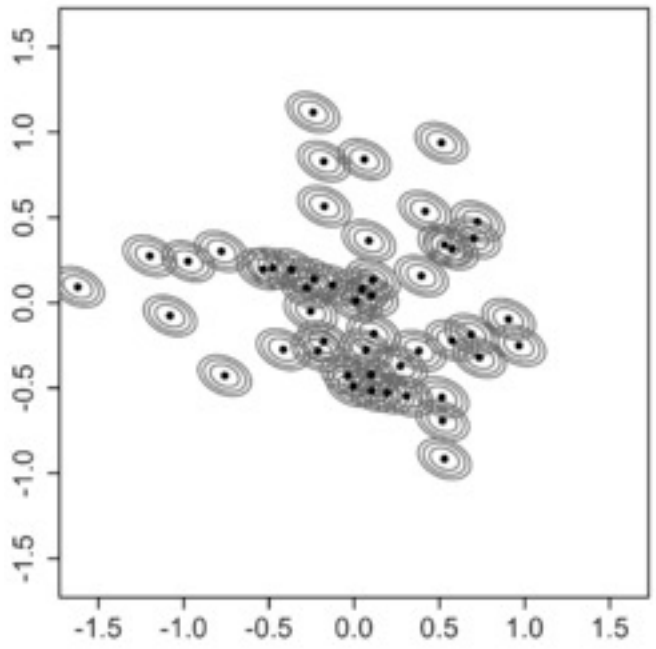
Sold at a British art auction house for $432,500

How to evaluate if your GAN is working?

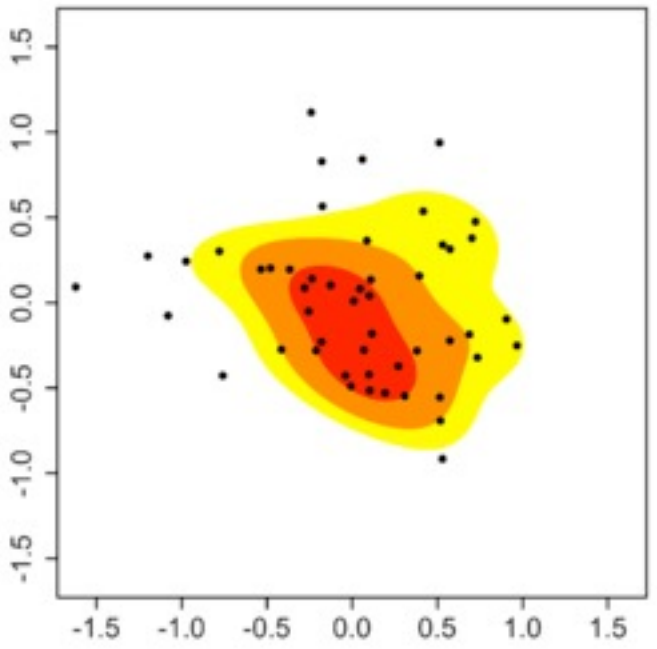
- Just look at the pictures?
- Quantitative evaluation for GANs (or any generative model) is still an open problem
- The following discussion applies to evaluating any generative model (not just GANs)

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



Generated examples



Distribution

One approach: 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
- Known as Kernel Density Estimator (KDE) or Parzen window density estimator:

real data point — generated sample

$$\hat{p_h}(x) = \frac{1}{nh} \sum_i K\left(\frac{x - x_i}{h}\right)$$

An estimator with kernel K and bandwidth h

- In generative model evaluation, K is usually the density function of a standard Normal distribution
- **What do you think this is the limitation of this approach?**

Bishop 2006

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

Bishop 2006

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: $p(y) = \sum p(y \mid x)p(x) = E_{x\sim G}[p(y \mid x)]$
  - **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

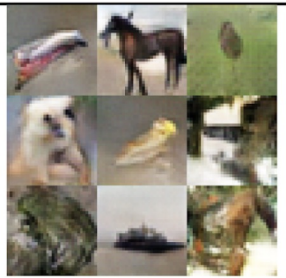
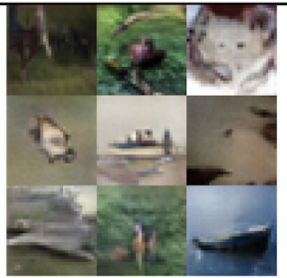
[Improved GAN: Salimans et al 2016]

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: $\mathrm{IS}(x) = \exp(\mathbb{E}_{x \sim p_g}\left[D_{\mathrm{KL}}\left[p(y|x) \parallel p(y)\right]\right])$
- A good generator will have a high KL
  - p(y) will be close to uniform and p(y|x) will be very peaky

[Improved GAN: Salimans et al 2016]

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?**

[Improved GAN: Salimans et al 2016]

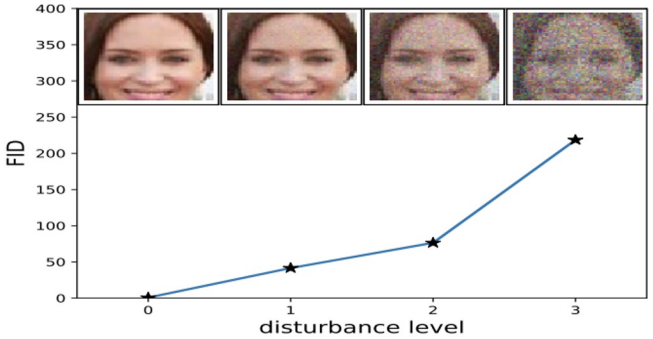
Fréchet Inception Distance

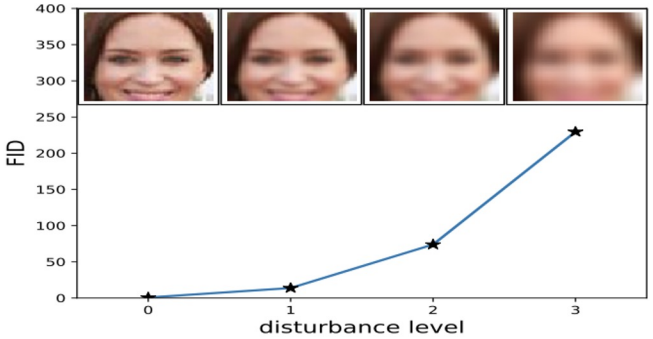
- A list of 1000 images (one of each class) can obtain perfect Inception Score
- Inception Score doesn't sufficiently measure diversity
  - It only captures class-level diversity but not within-class diversity
- FID was proposed to capture more details
  - We want the **features** to be realistic and 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

$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \mathrm{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2\left(\boldsymbol{C}\boldsymbol{C}_w\right)^{1/2}\right)$$

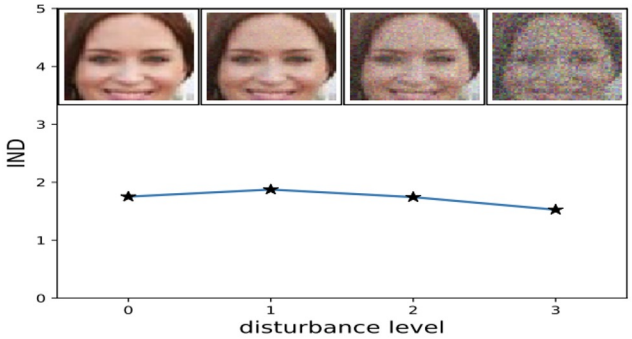
[Heusel et al, 2017]

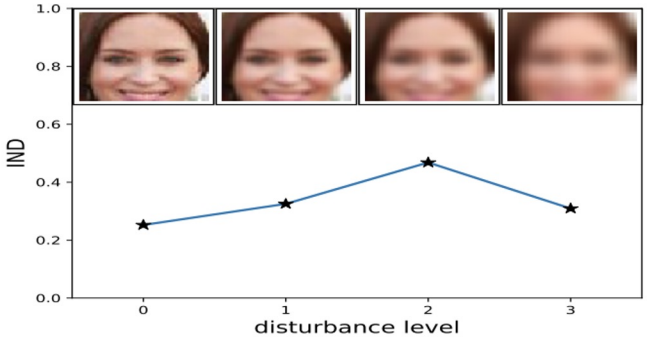
Fréchet Inception Distance





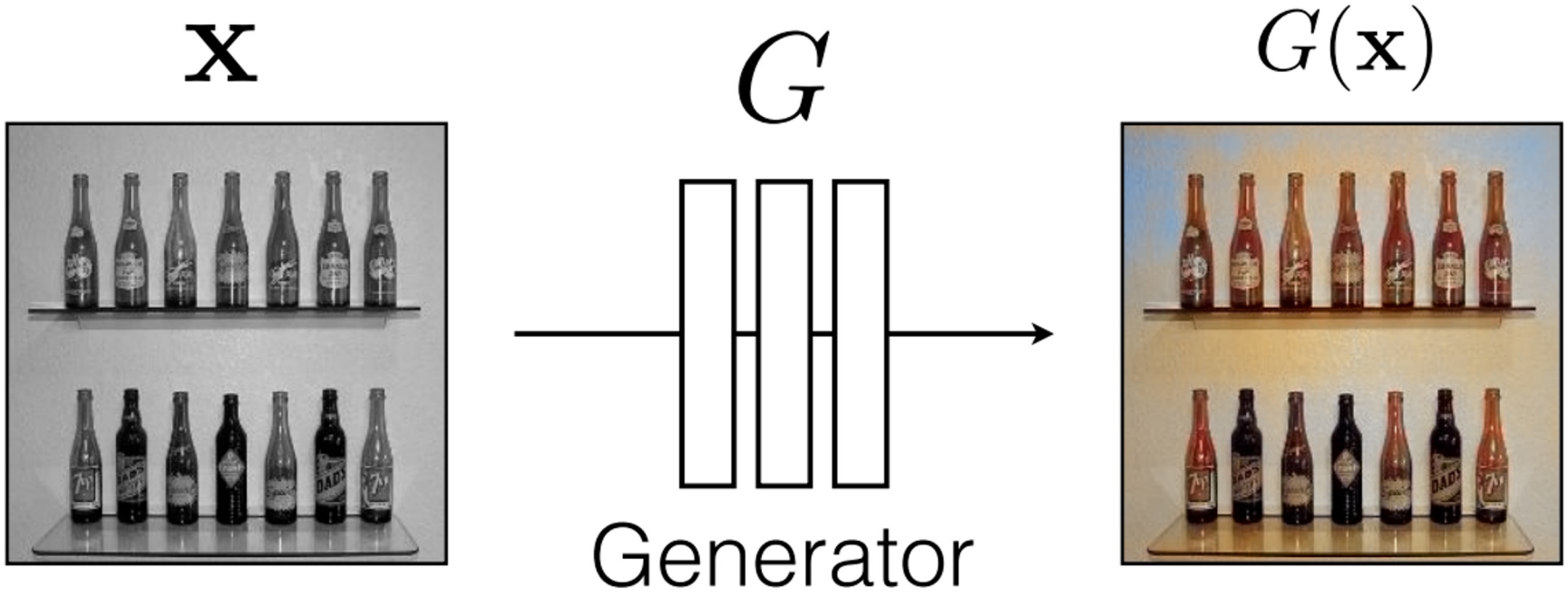
Inception Score





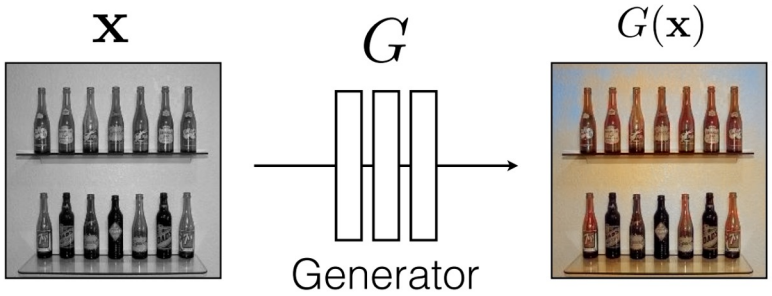
[Heusel et al, 2017]

Conditional GANs / pix2pix



Slide: Phillip Isola

Conditional GANs / pix2pix



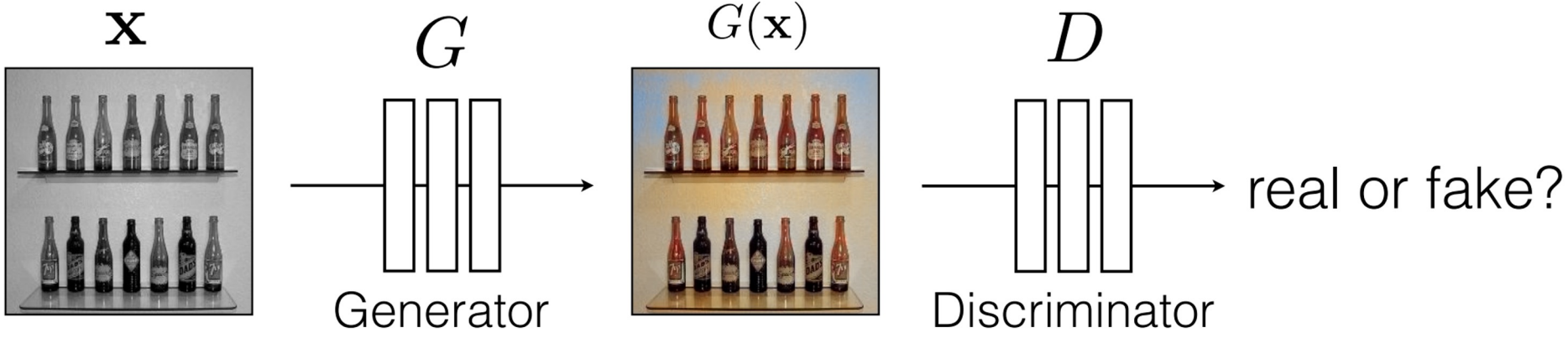
Where do we get training data for this?

Convert color images to black and white

Why not just train this with supervised learning?
Why do we need a GAN?

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

Conditional GANs / pix2pix

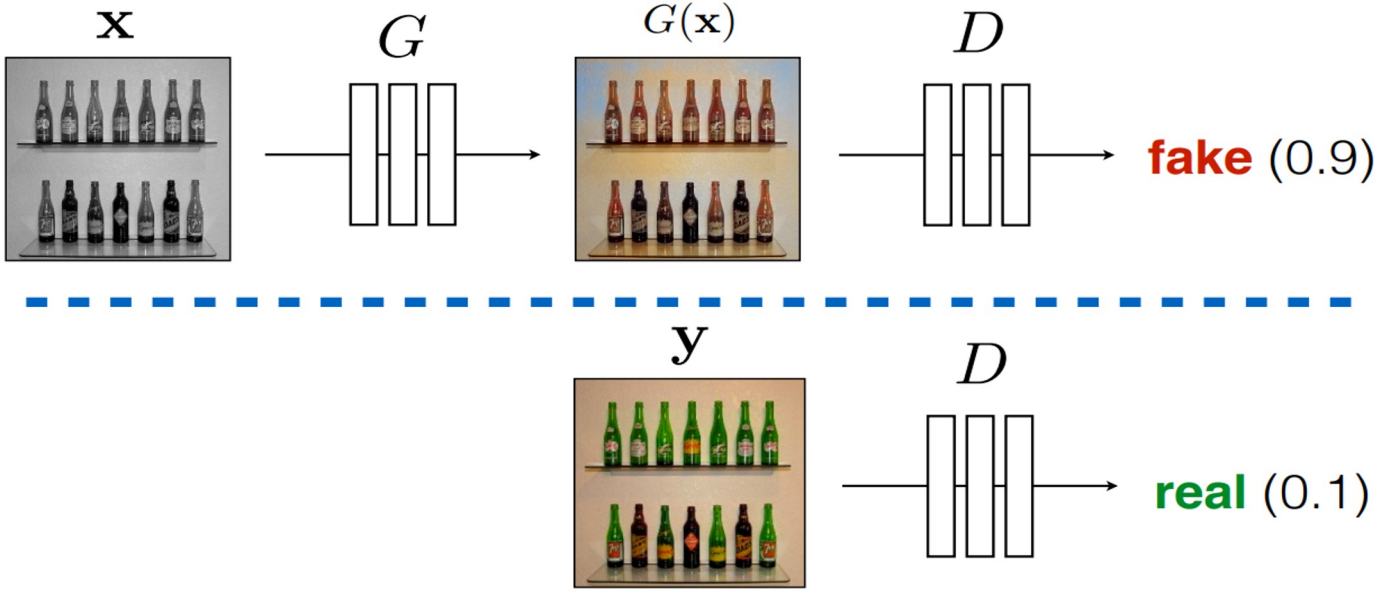


G tries to synthesize fake images that fool **D**

D tries to identify the fakes

Slide: Phillip Isola

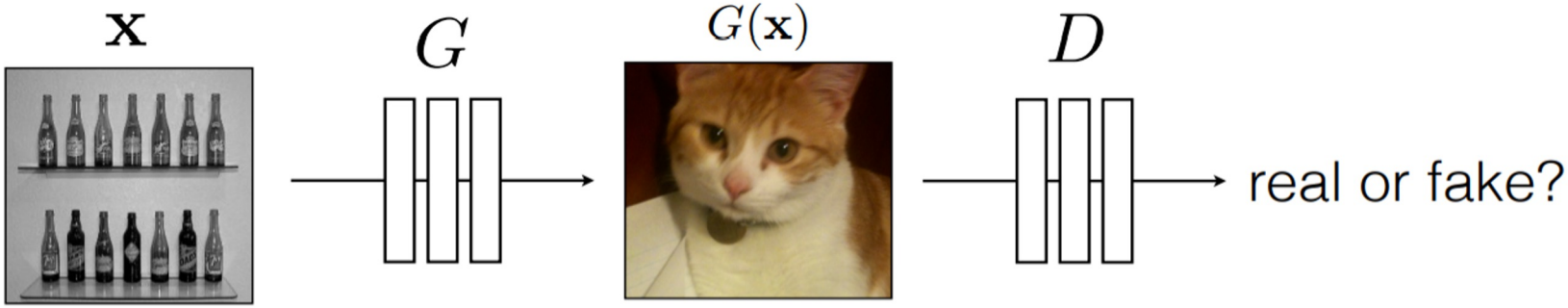
Conditional GANs / pix2pix



What is the problem with this approach?

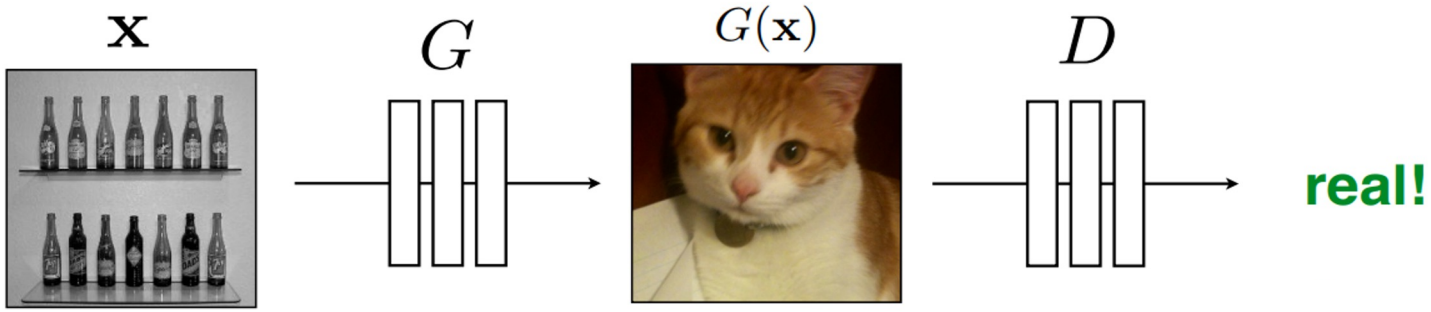
Slide: Phillip Isola

Conditional GANs / pix2pix



Slide: Phillip Isola

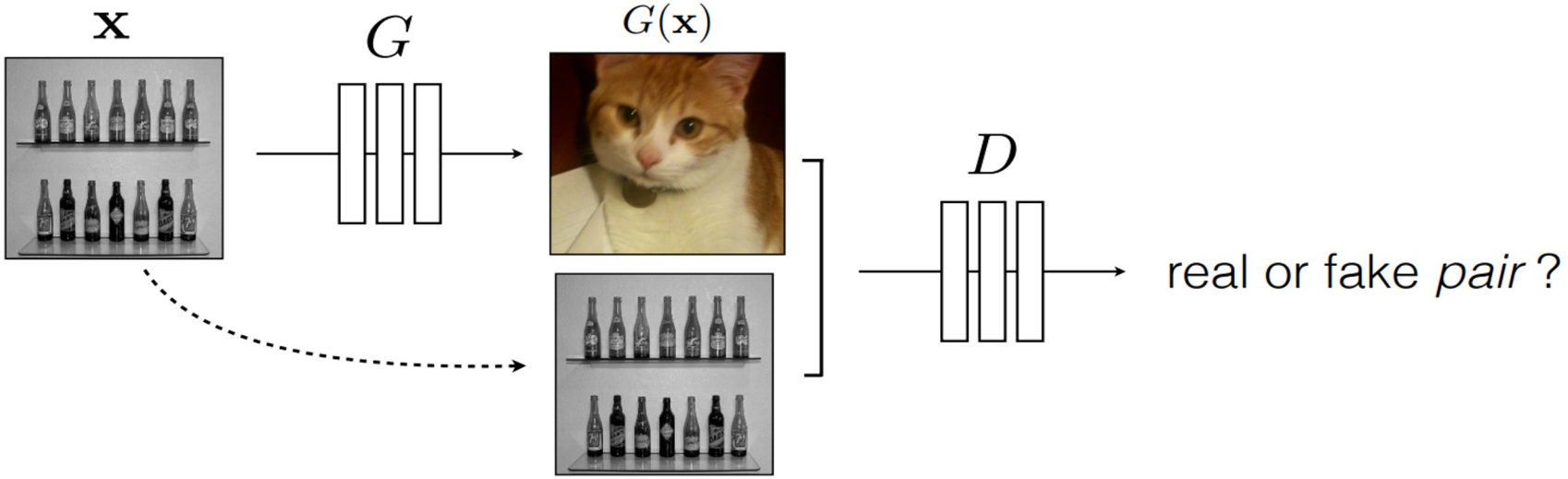
Conditional GANs / pix2pix



How can we fix this?

Slide: Phillip Isola

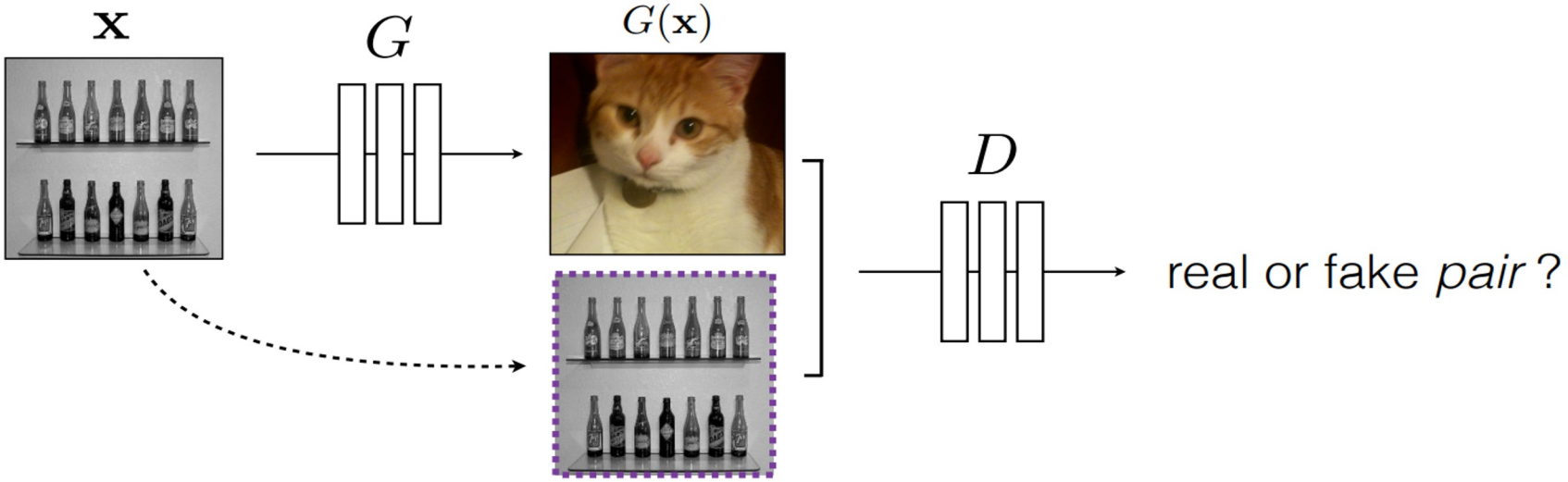
Conditional GANs / pix2pix



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

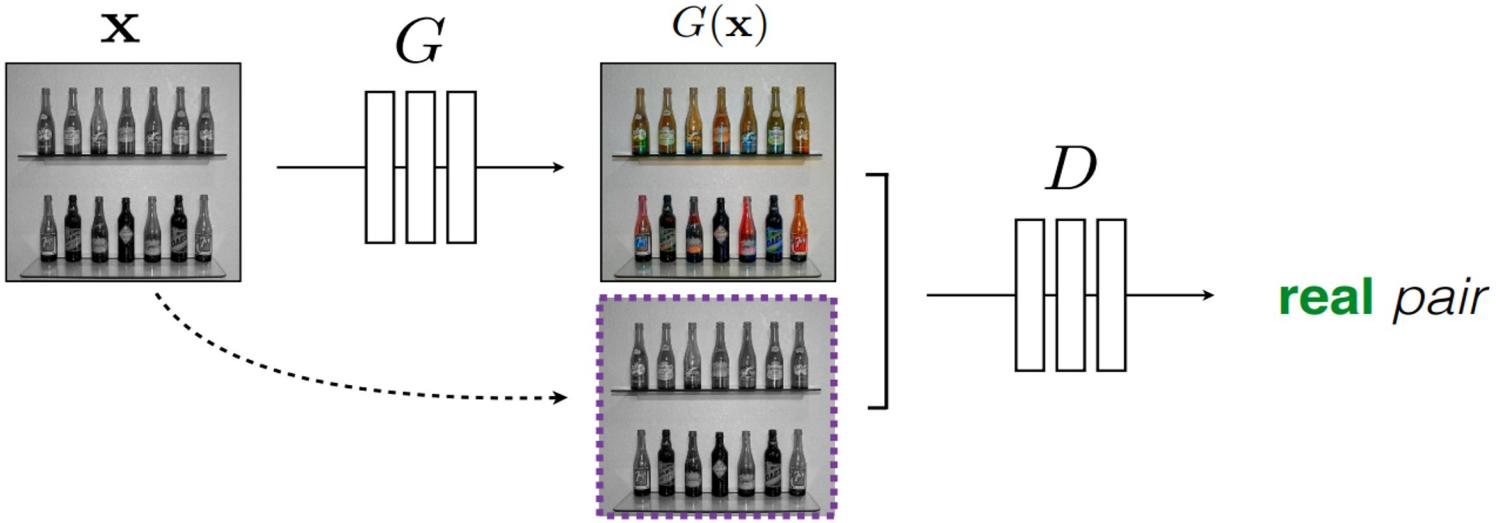


New objective: $\arg\min_G \max_D \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$

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

Slide: Phillip Isola

Conditional GANs / pix2pix

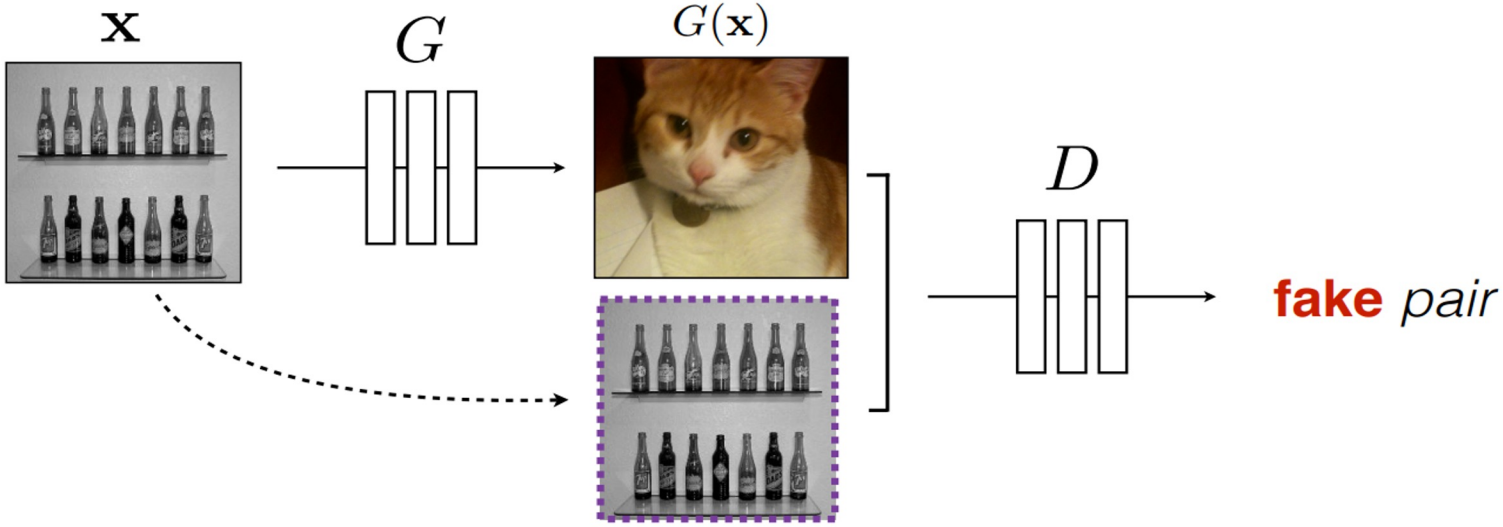


New objective: $\arg\min_G \max_D \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$

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

Slide: Phillip Isola

Conditional GANs / pix2pix



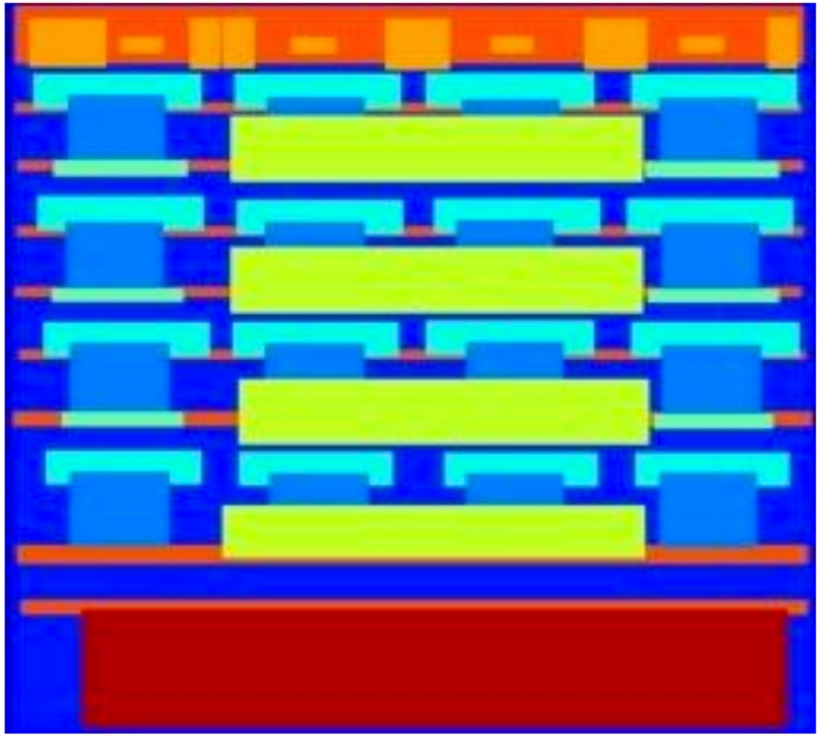
New objective: $\arg\min_G \max_D \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$

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

Slide: Phillip Isola

Conditional GANs / pix2pix

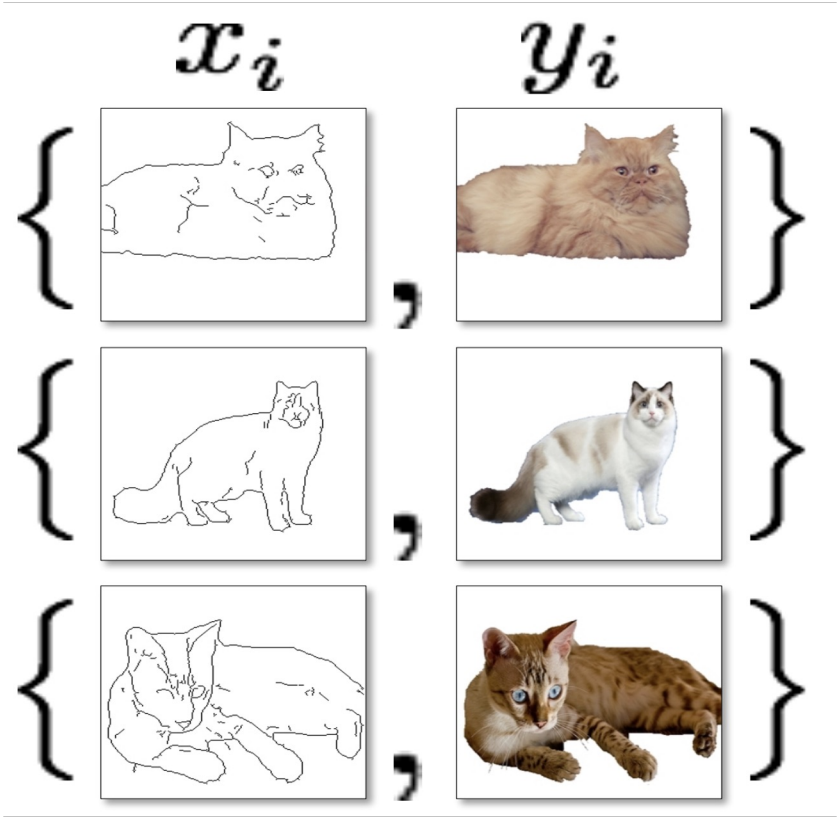
Input



16x16 Discriminator

Slide: Phillip Isola

Conditional GANs / pix2pix



Slide: Phillip Isola

Image-to-Image Translation with **pix2pix**

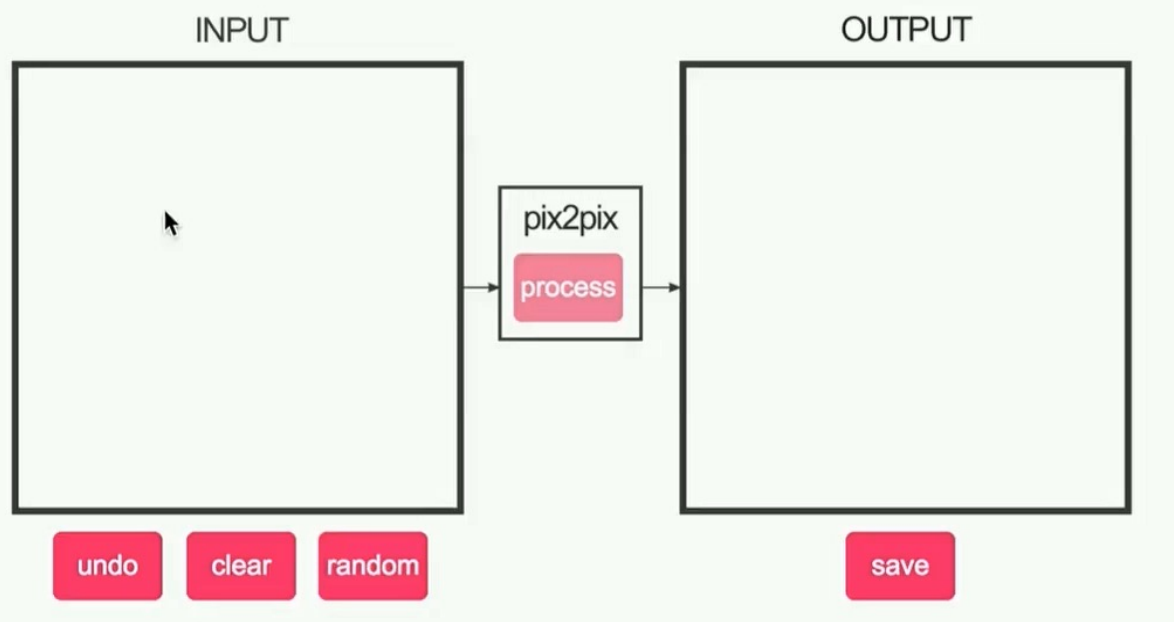
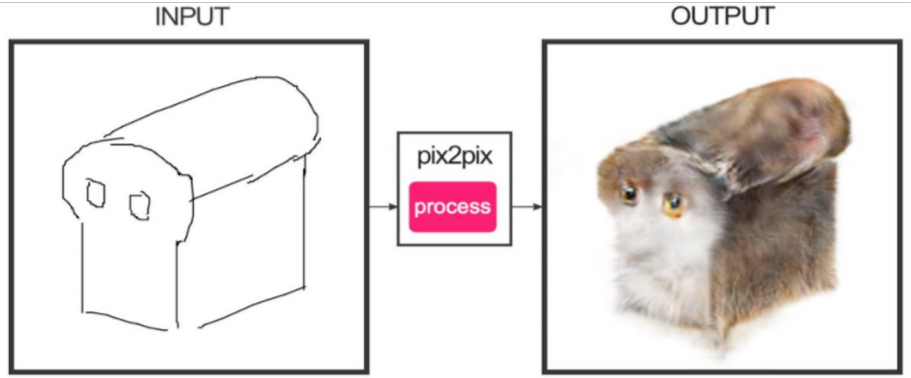


Image-to-image Translation with Conditional Adversarial Nets
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. CVPR 2017

Conditional GANs / pix2pix

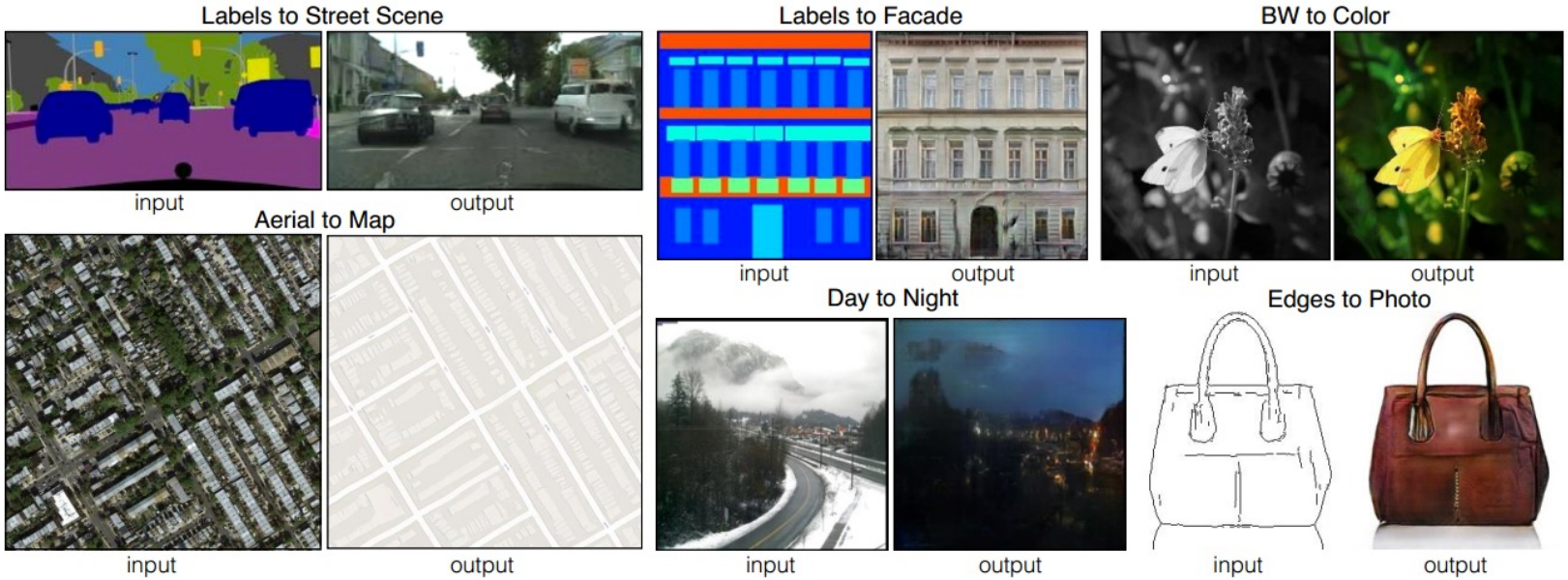


Ivy Tasi @ivymyt

Vitaly Vidmirov @vvid

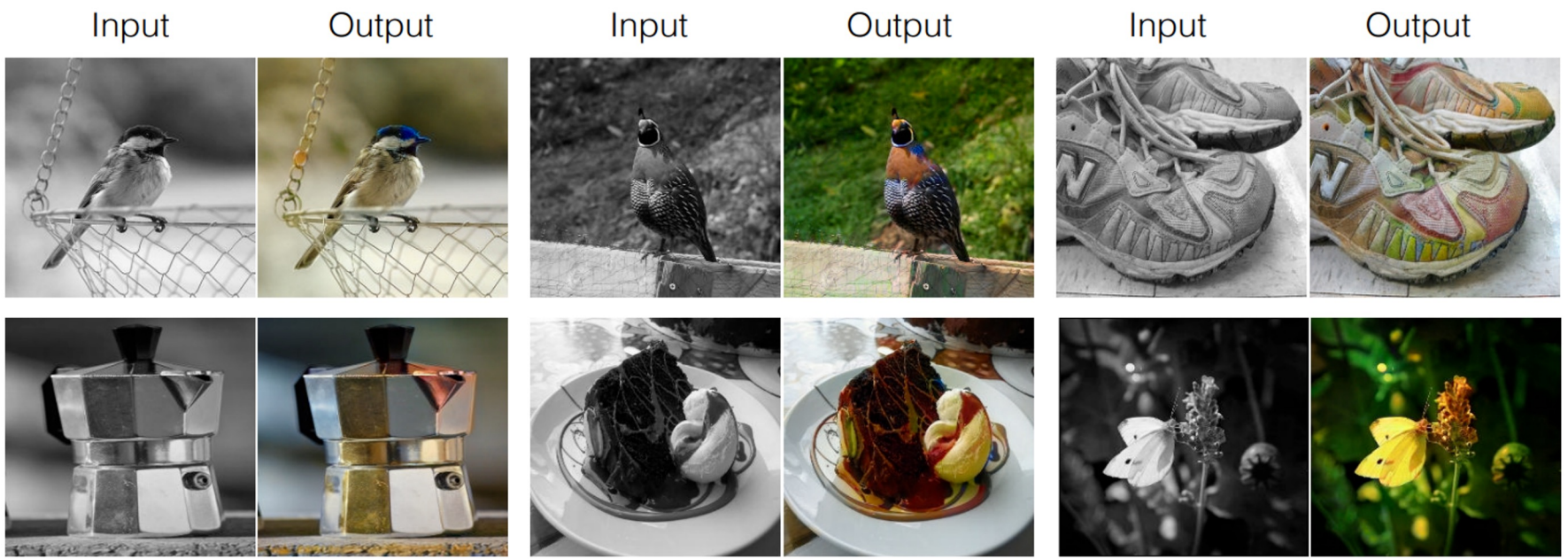
Slide: Phillip Isola

Image-to-Image Translation with **pix2pix**



Conditional GANs / pix2pix

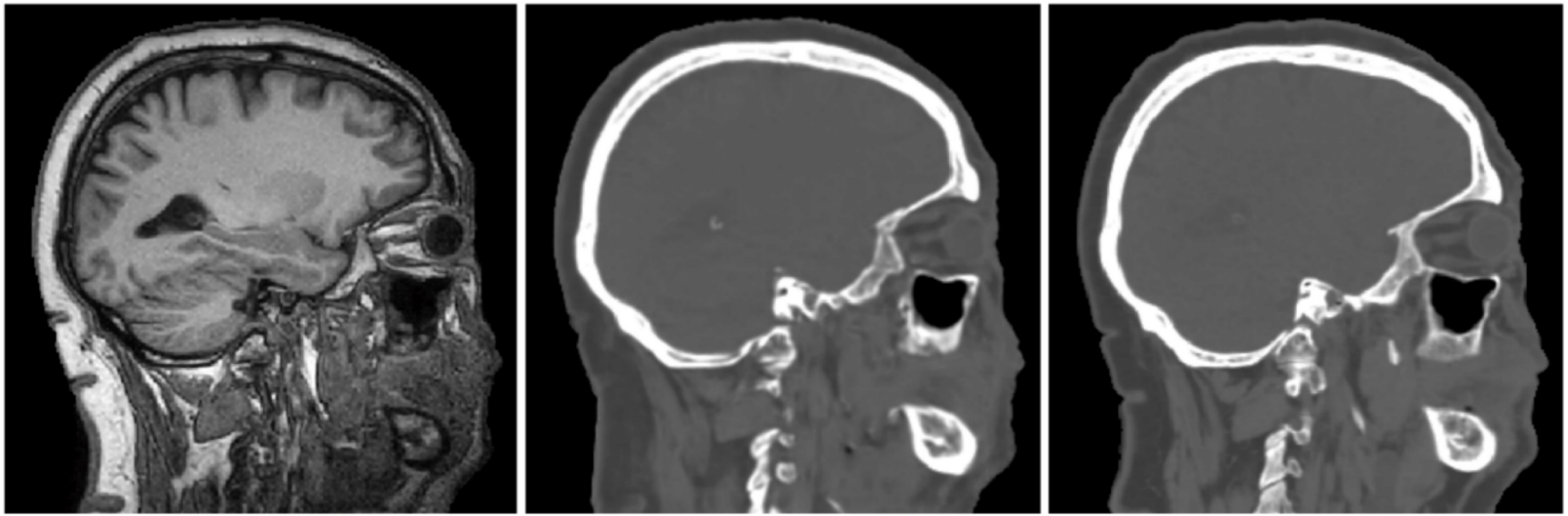
BW → Color



Slide: Phillip Isola

pix2pix for medical imaging

MR → CT [Wolterink et al] arxiv: 1708.01155

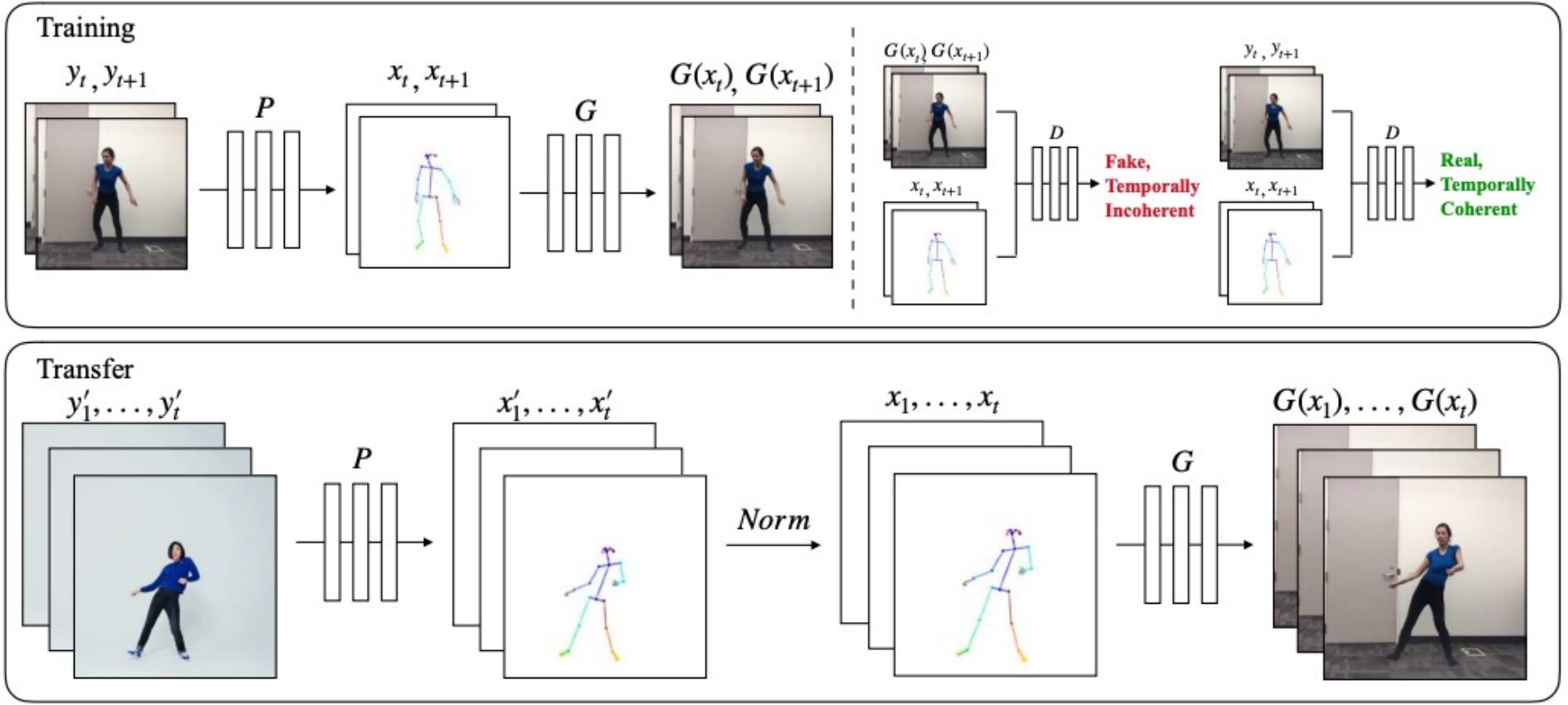


Input MR | Generated CT | Ground truth CT

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

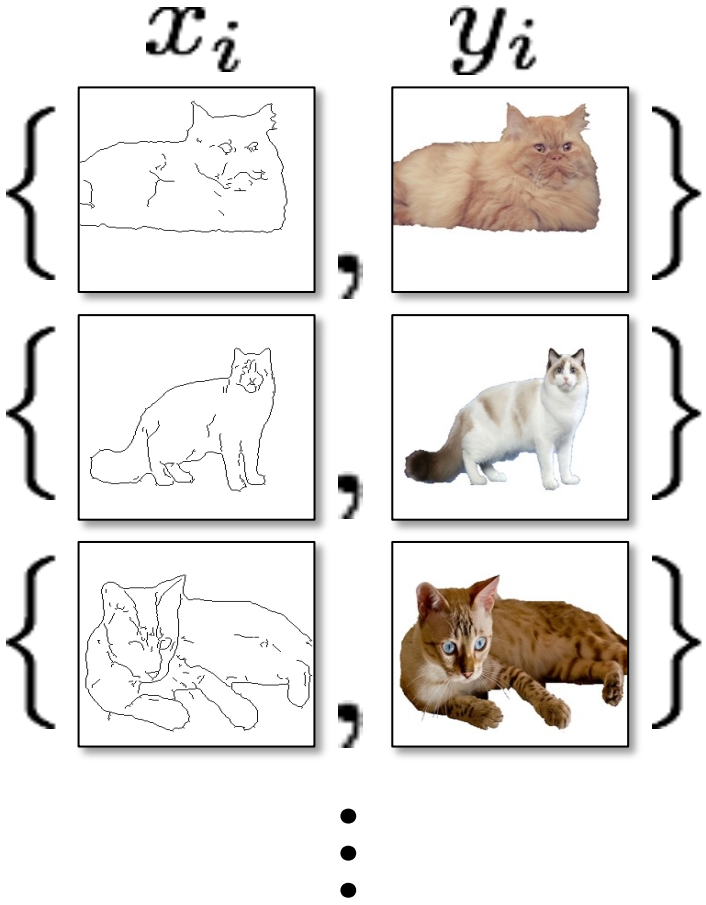
Slide: Phillip Isola

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

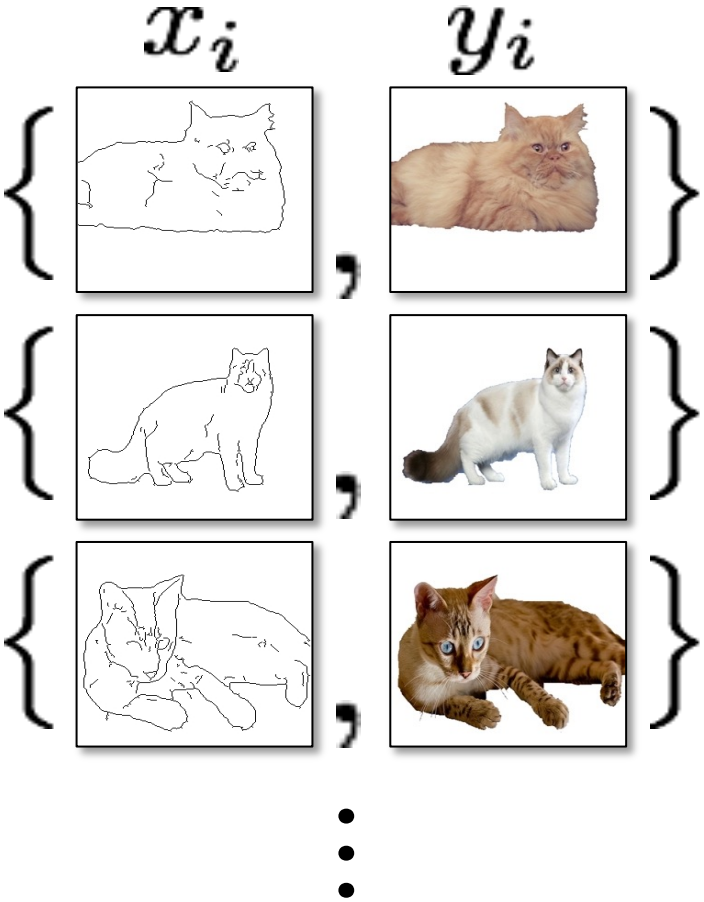


Chan, Caroline, et al. "Everybody dance now." *ICCV* 2019.

Paired



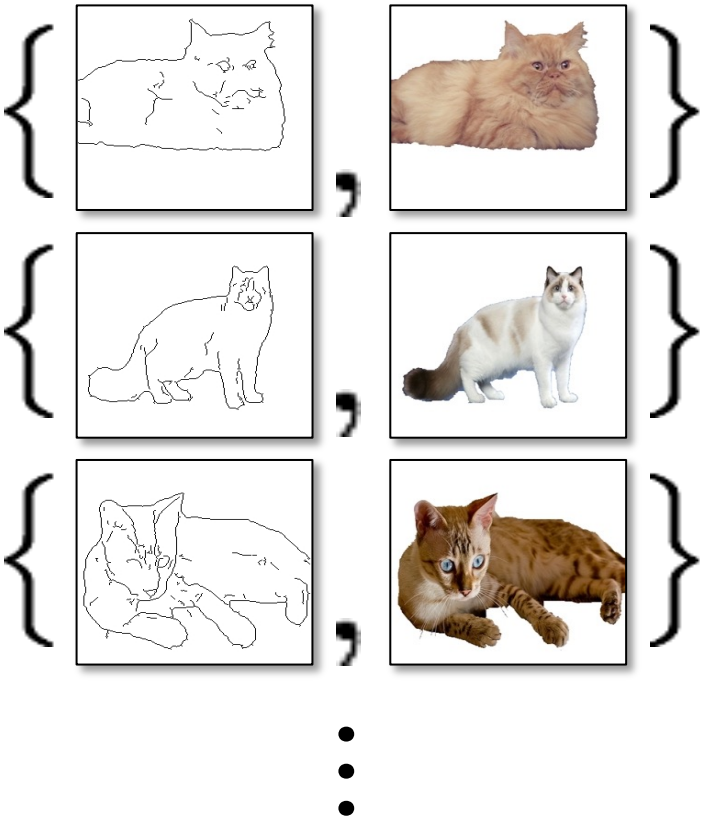
Paired



Horse ↔ zebra:
how to get paired images of zebras?

Paired

x_i y_i



Unpaired

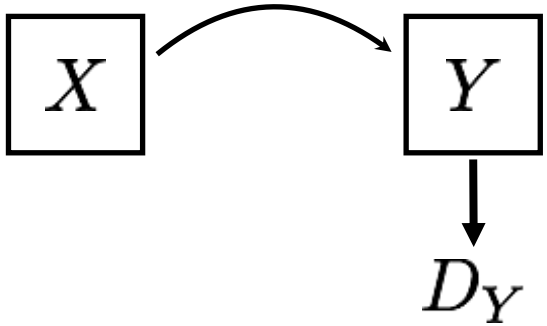
X Y

Cycle-Consistent Adversarial Networks

How could a network "cheat" this?

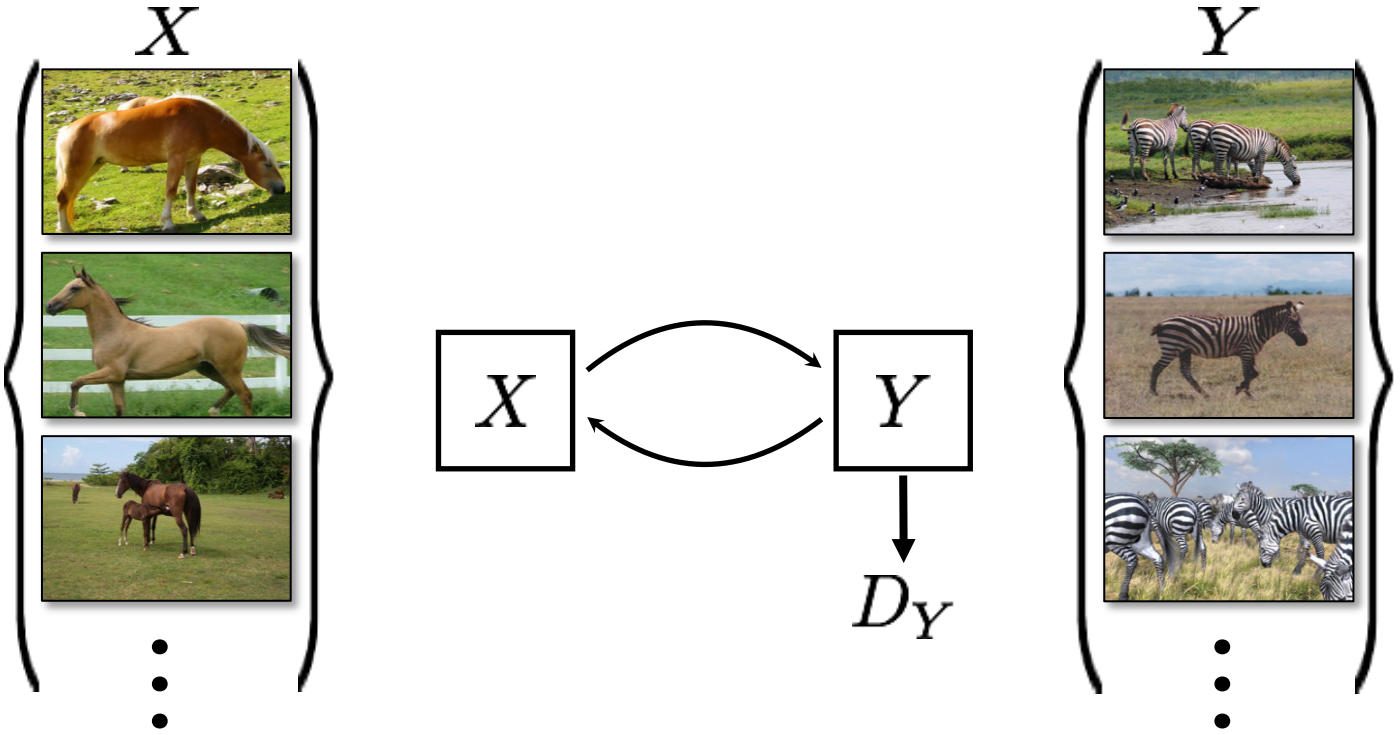
It could generate a zebra picture in a pose that is completely unrelated to the horse

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



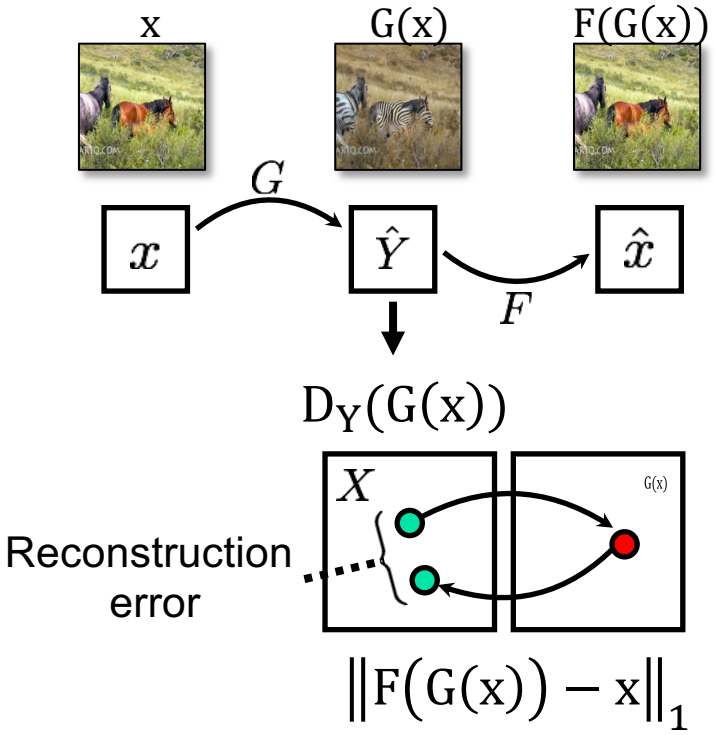
[Zhu*, Park*, Isola, and Efros, ICCV 2017]

Cycle-Consistent Adversarial Networks



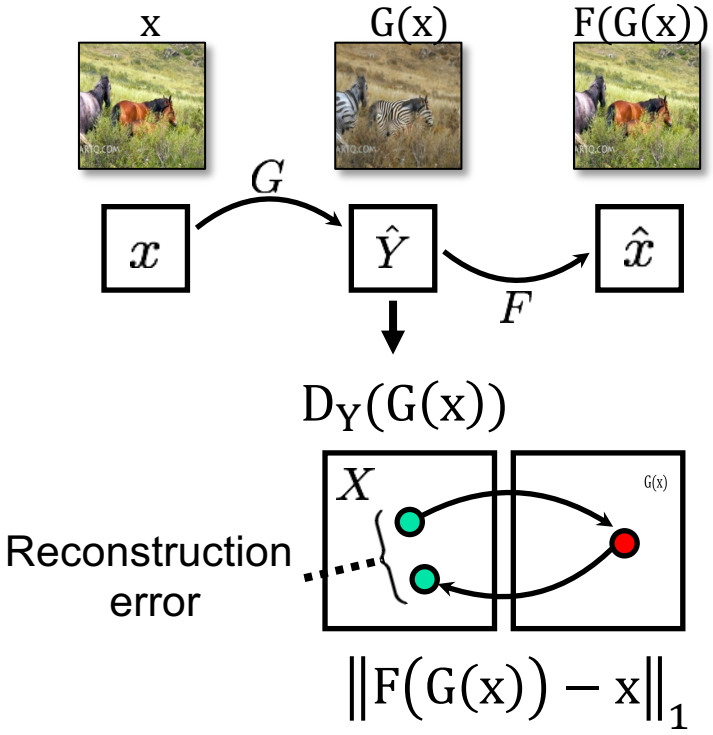
[Zhu*, Park*, Isola, and Efros, ICCV 2017]

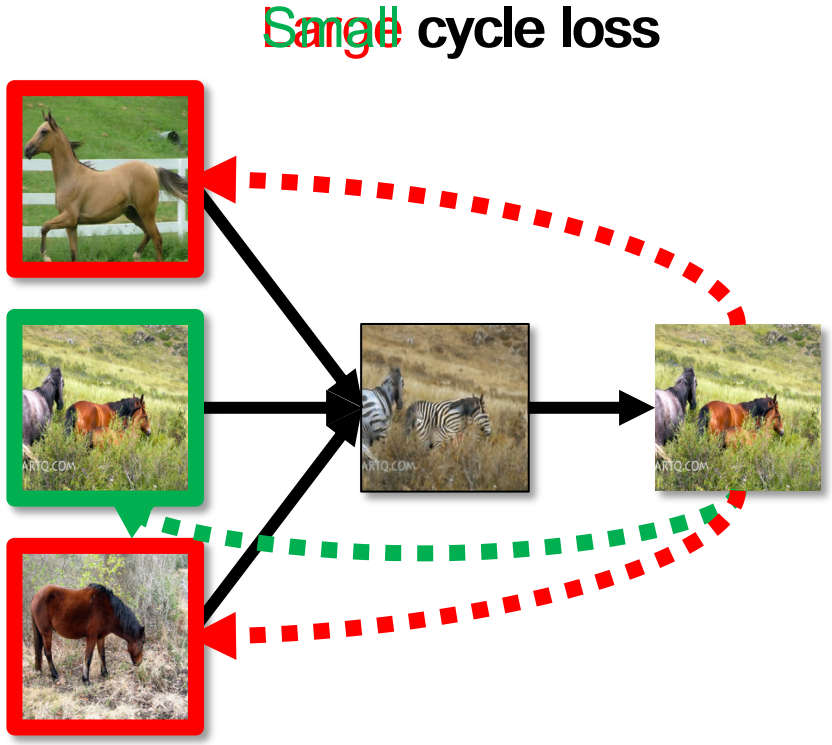
Cycle-Consistent Adversarial Networks



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

Cycle Consistency Loss

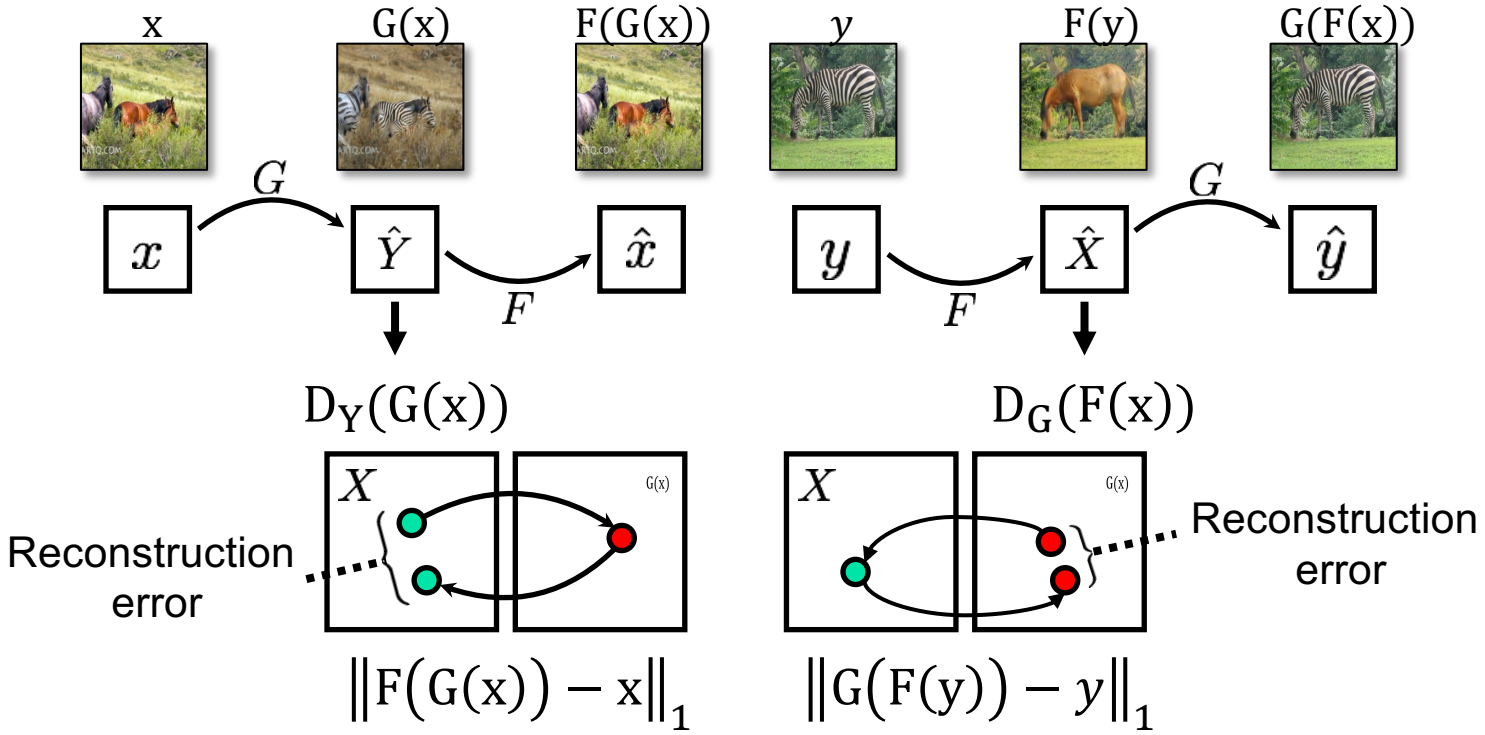




[Zhu*, Park*, Isola, and Efros, ICCV 2017]

Cycle Consistency Loss

Could a network still cheat this? How?



See similar formulations [Yi et al. 2017], [Kim et al. 2017]

Collection Style Transfer (no pairs exist)

Photograph
@ Alexei Efros

Monet

Van Gogh

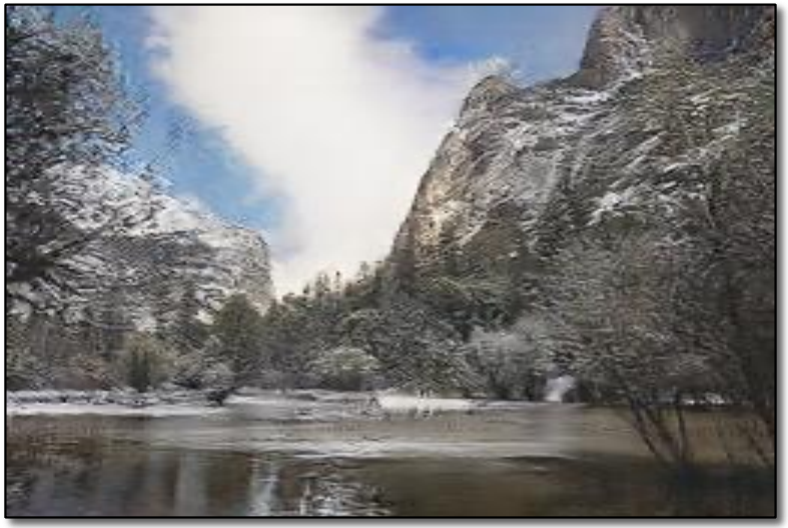
Cezanne

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

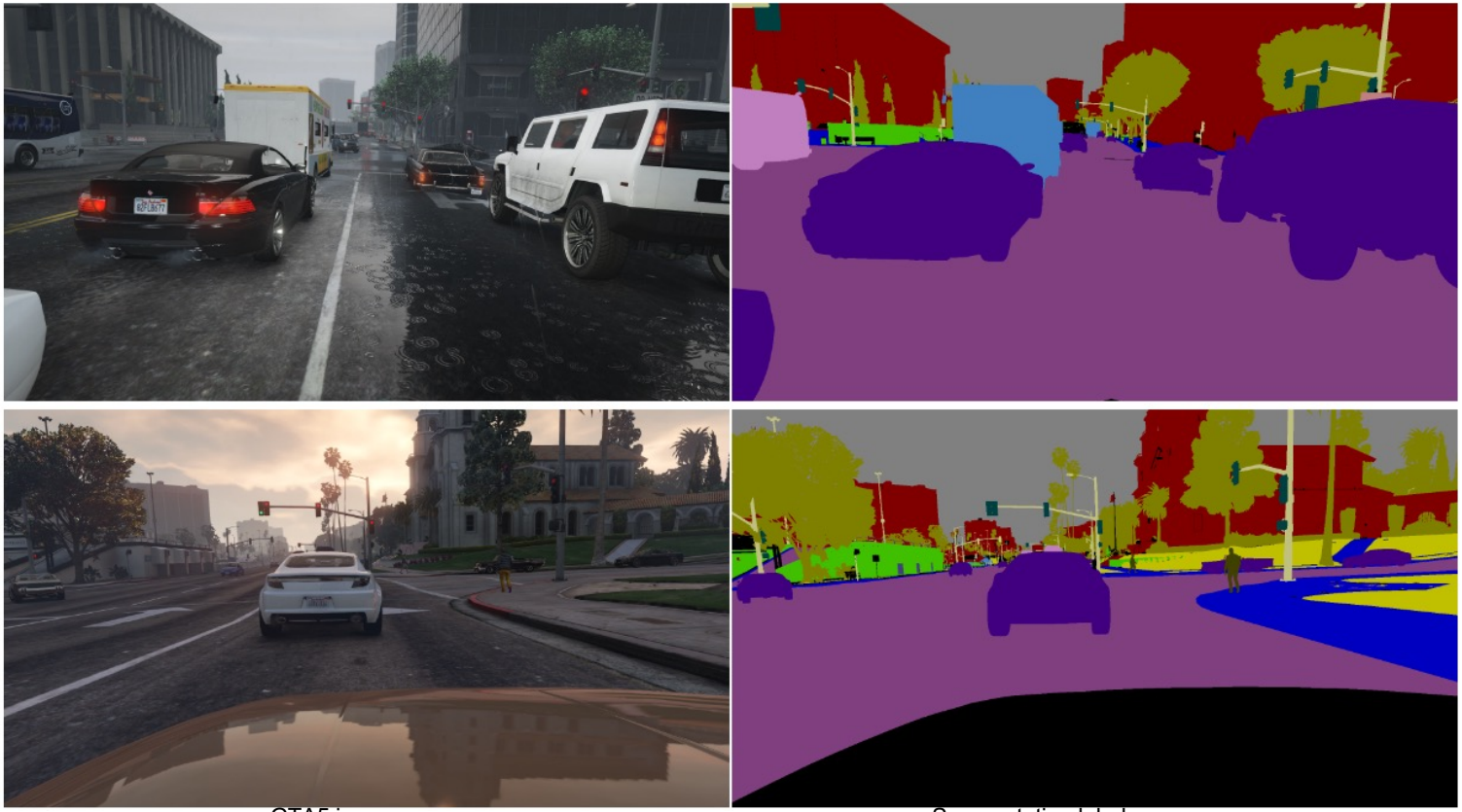
Inspired by [Johnson et al. 2011]

Real2CG: real streetview → GTA

Cityscape Input

Output

Synthetic Data as Supervision

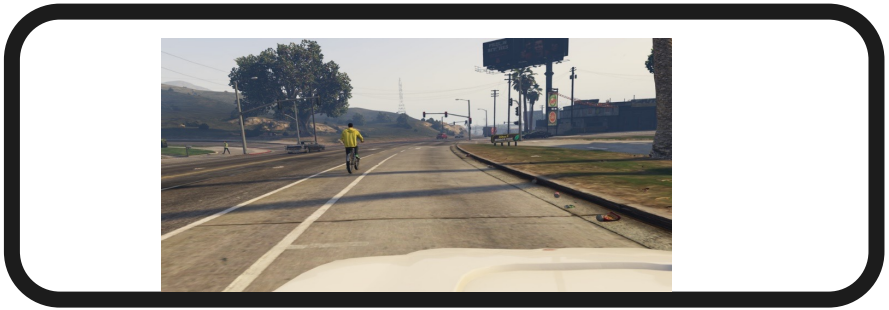


GTA5 images

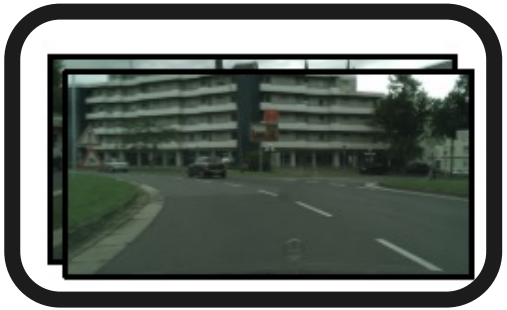
Segmentation labels

[Richter*, Vineet* et al. 2016] [Krähenbühl et al. 2018]

Domain Adaptation with CycleGAN



Train on GTA5 data



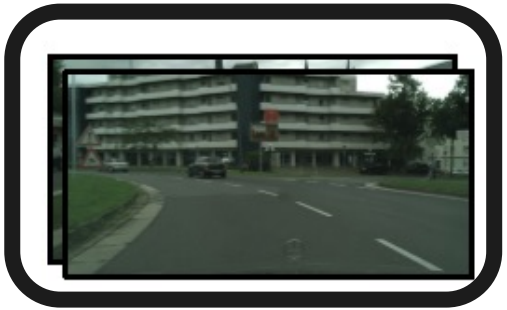
Test on real images

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

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

Domain Adaptation with CycleGAN

Train on CycleGAN data



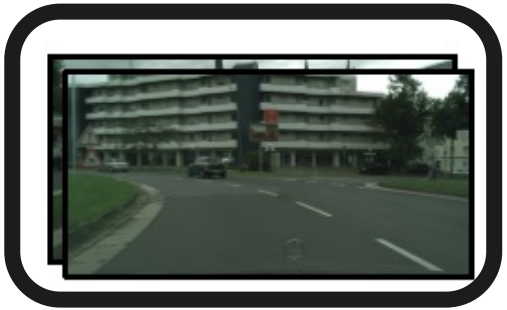
Test on real images

	meanIOU	Per-pixel accuracy
Train on CG, test on Real	17.9	54.0
Train on CycleGAN, test on Real	**34.8**	**82.8**

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

Domain Adaptation with CycleGAN

Train on CycleGAN data

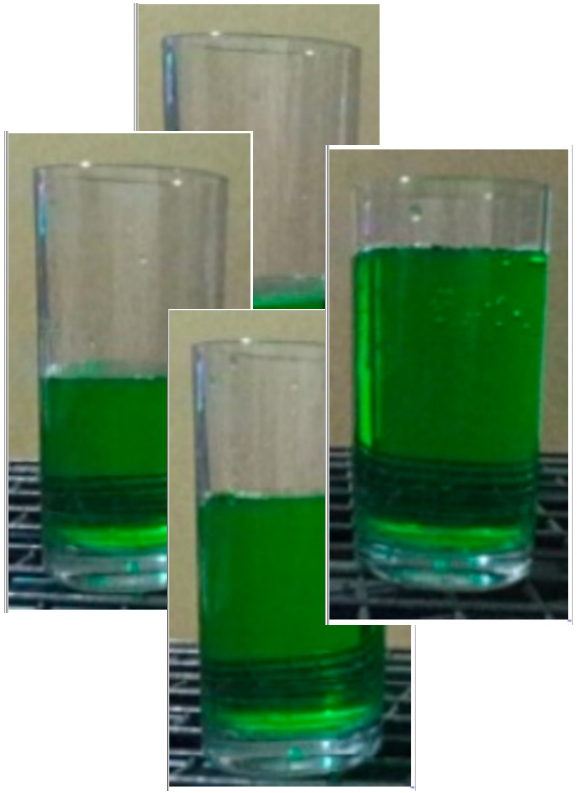


Test on real images

	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	**34.8**	**82.8**

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

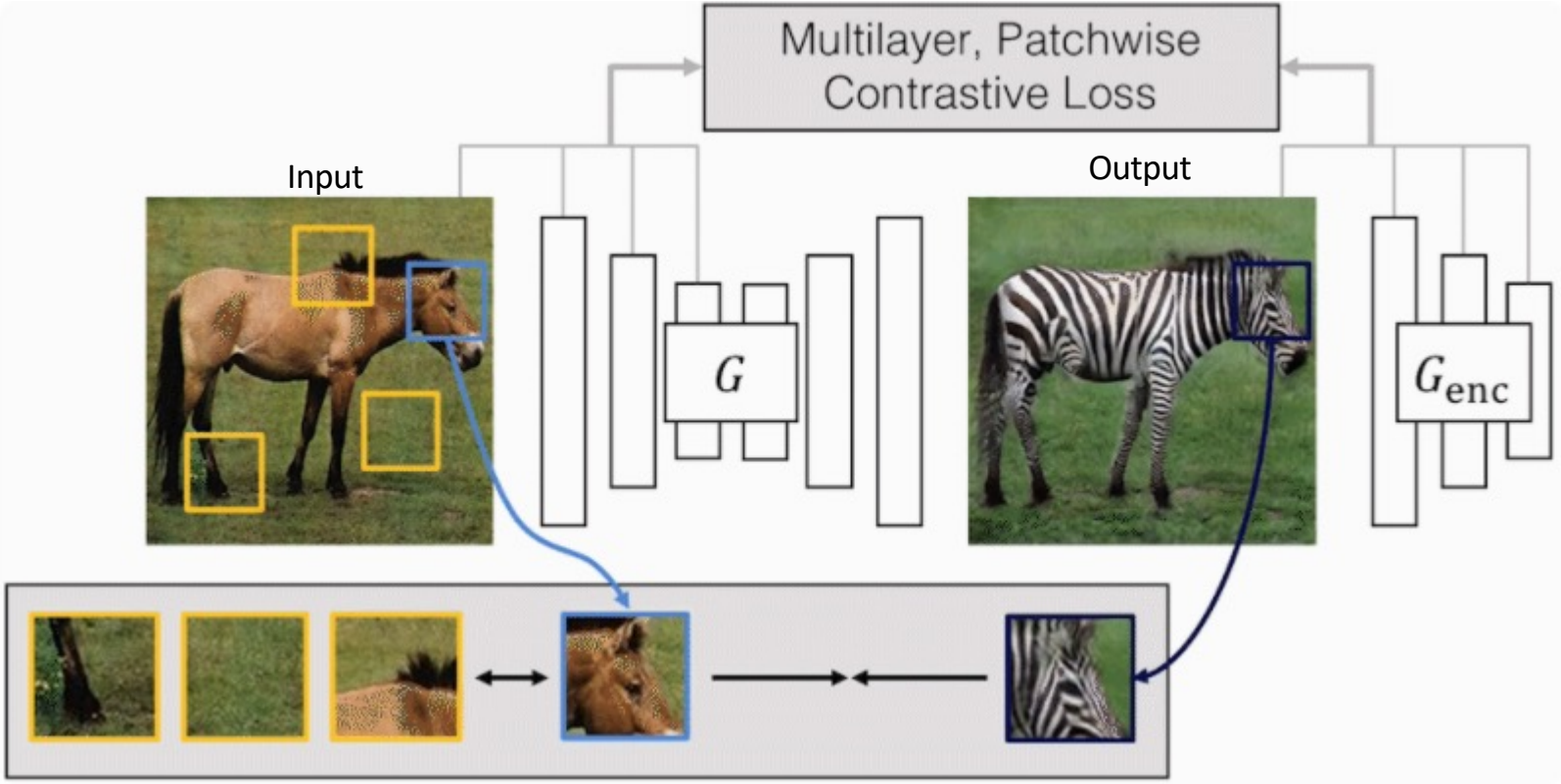
Can we use Colored Liquid to Learn about Transparent Liquid?



Dataset of Colored Liquids

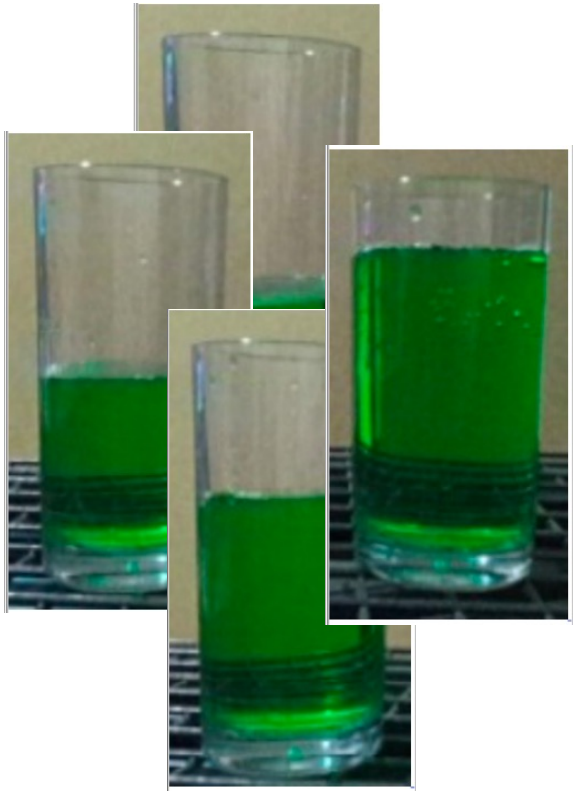
Dataset of Transparent Liquids

Convert a horse into a zebra

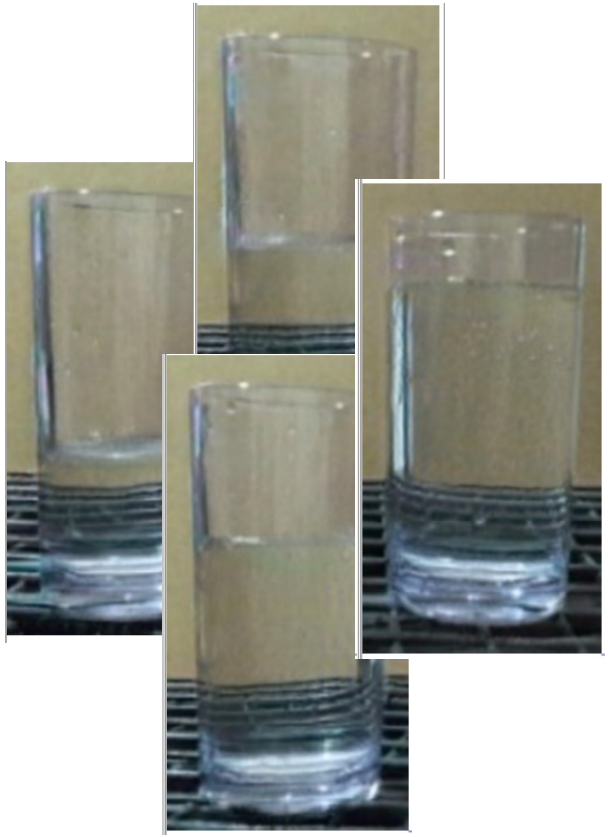


Park, Taesung, et al. "Contrastive learning for unpaired image-to-image translation." ECCV 2020.

Convert Colored Liquid to Transparent Liquid

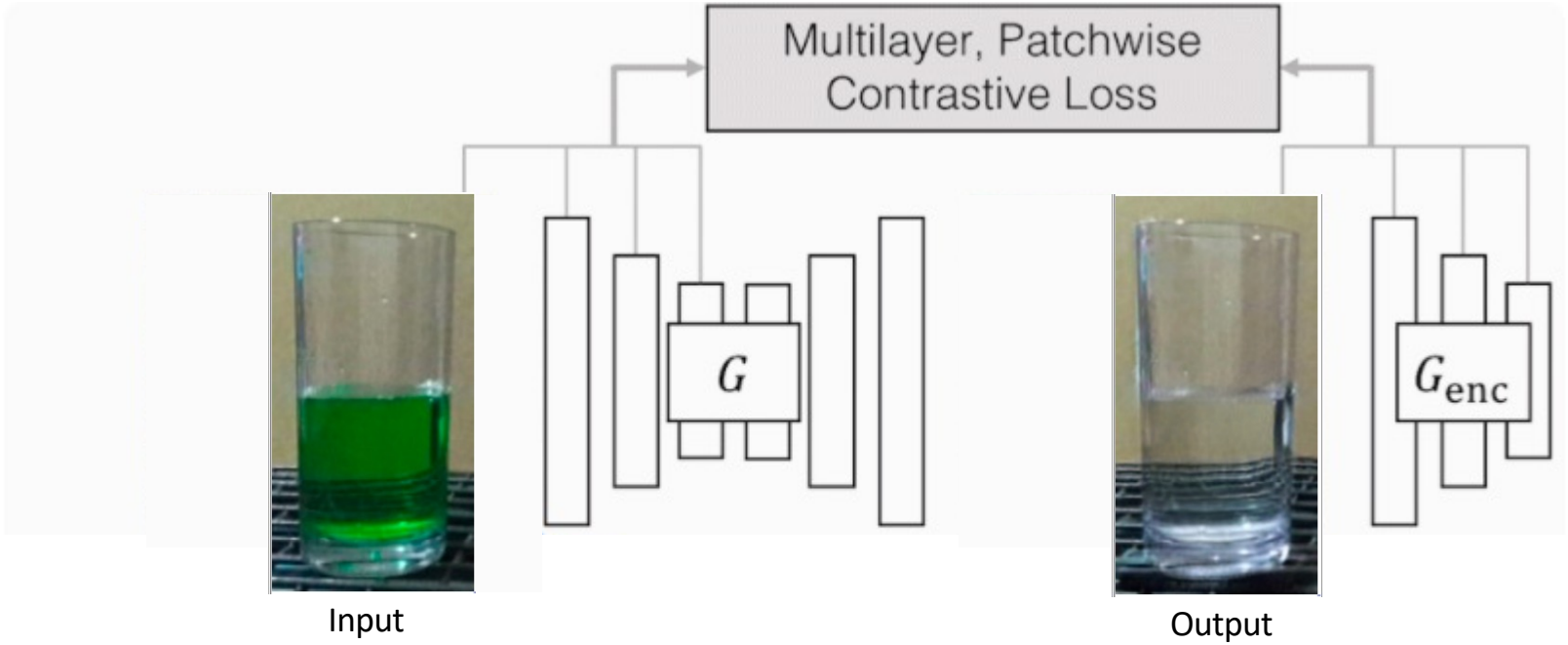


Dataset of Colored Liquids

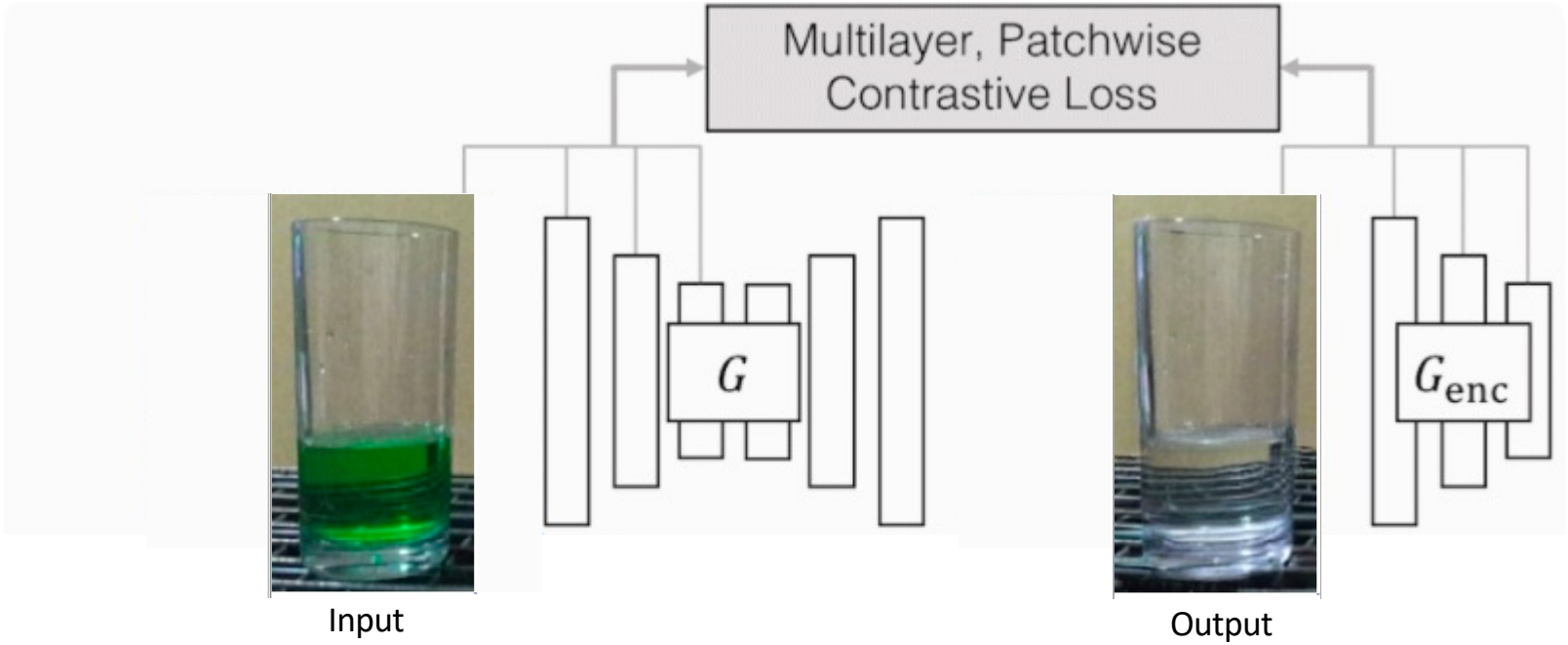


Dataset of Transparent Liquids

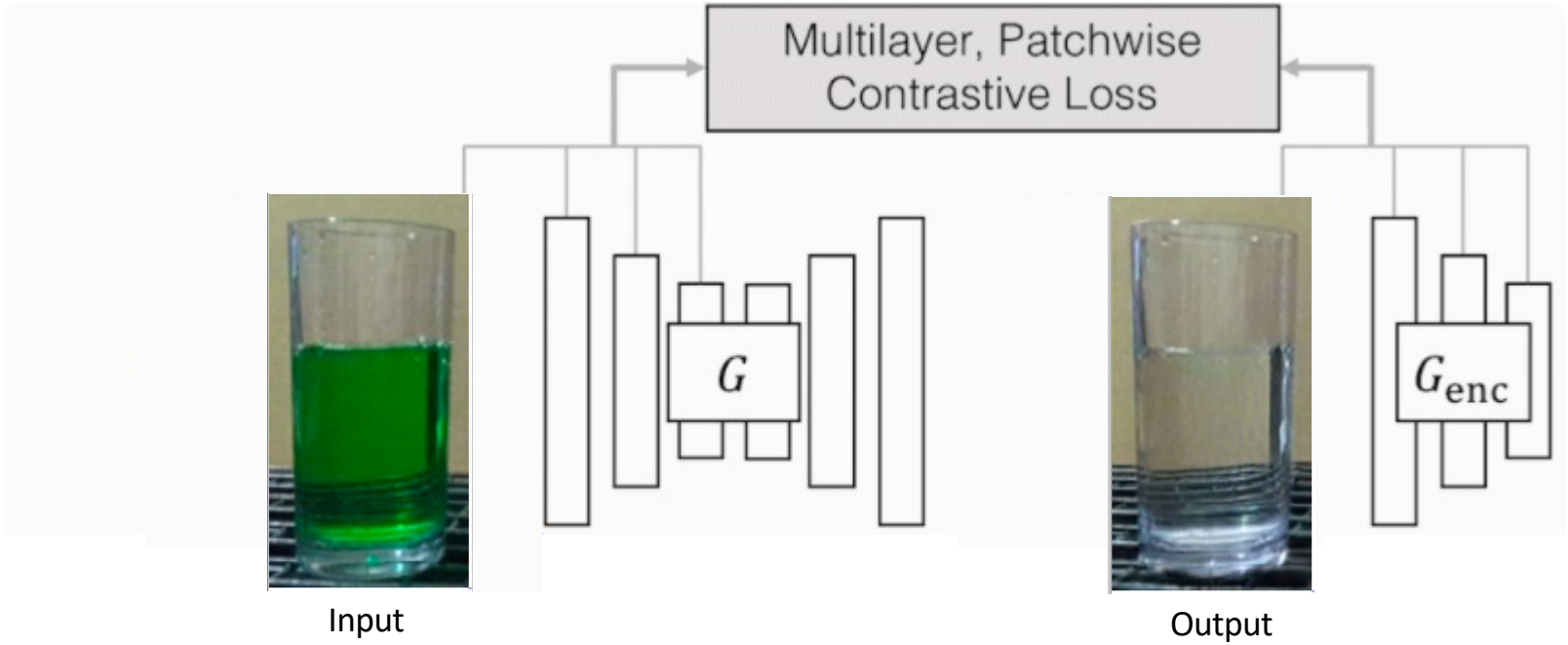
Convert Colored Liquid to Transparent Liquid



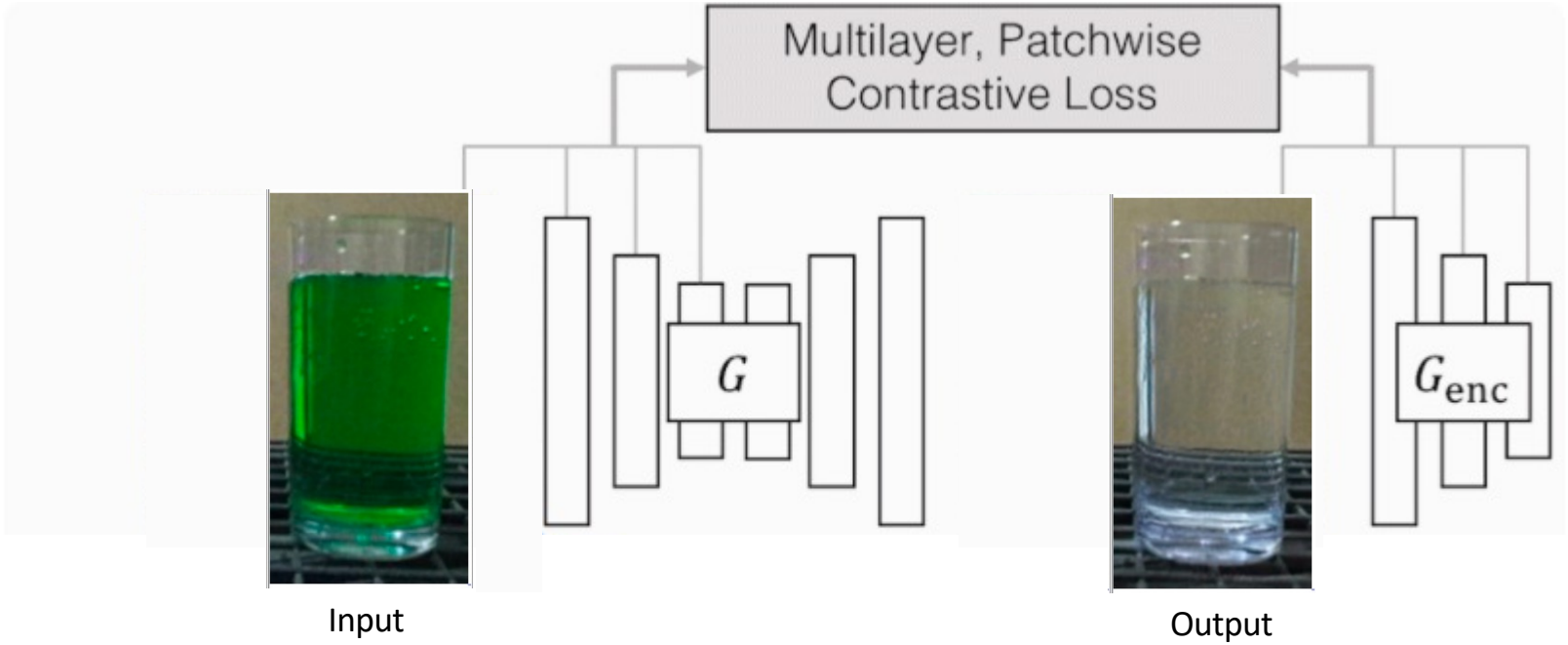
Convert Colored Liquid to Transparent Liquid



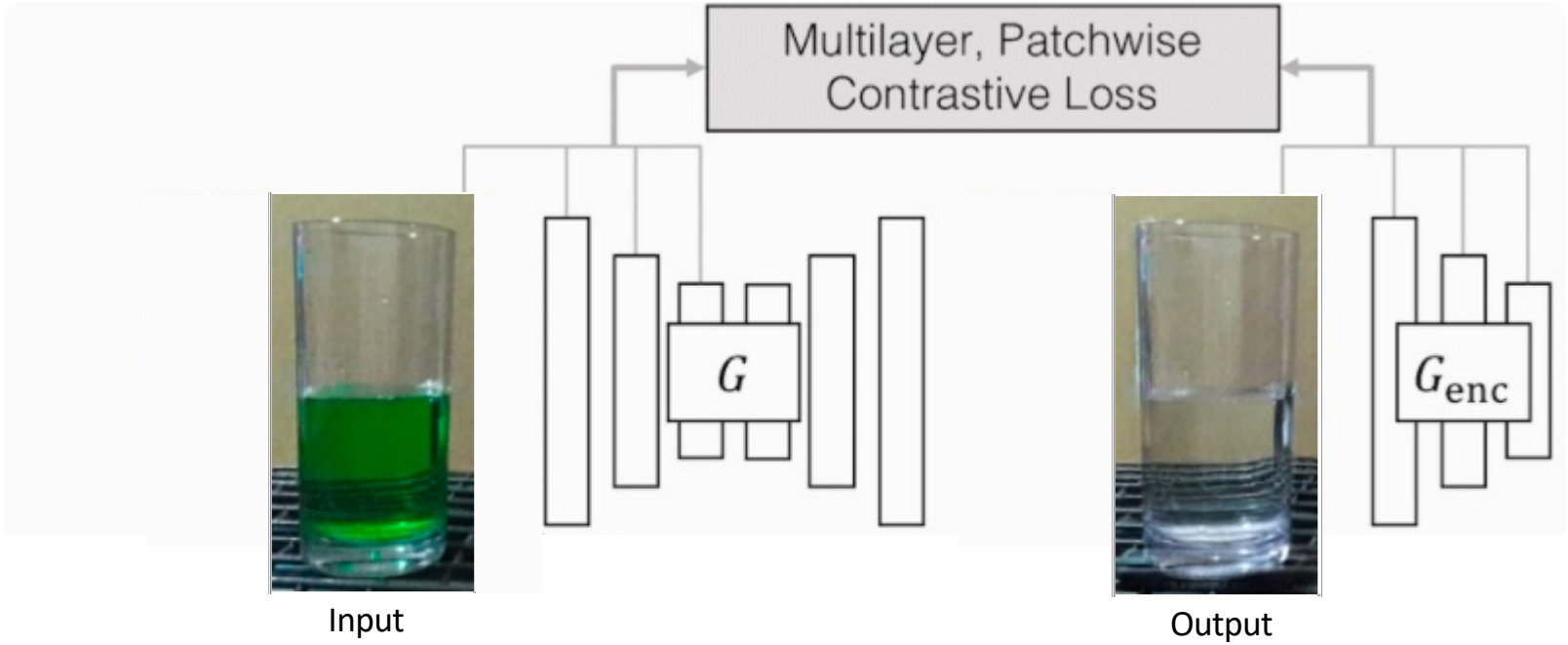
Convert Colored Liquid to Transparent Liquid



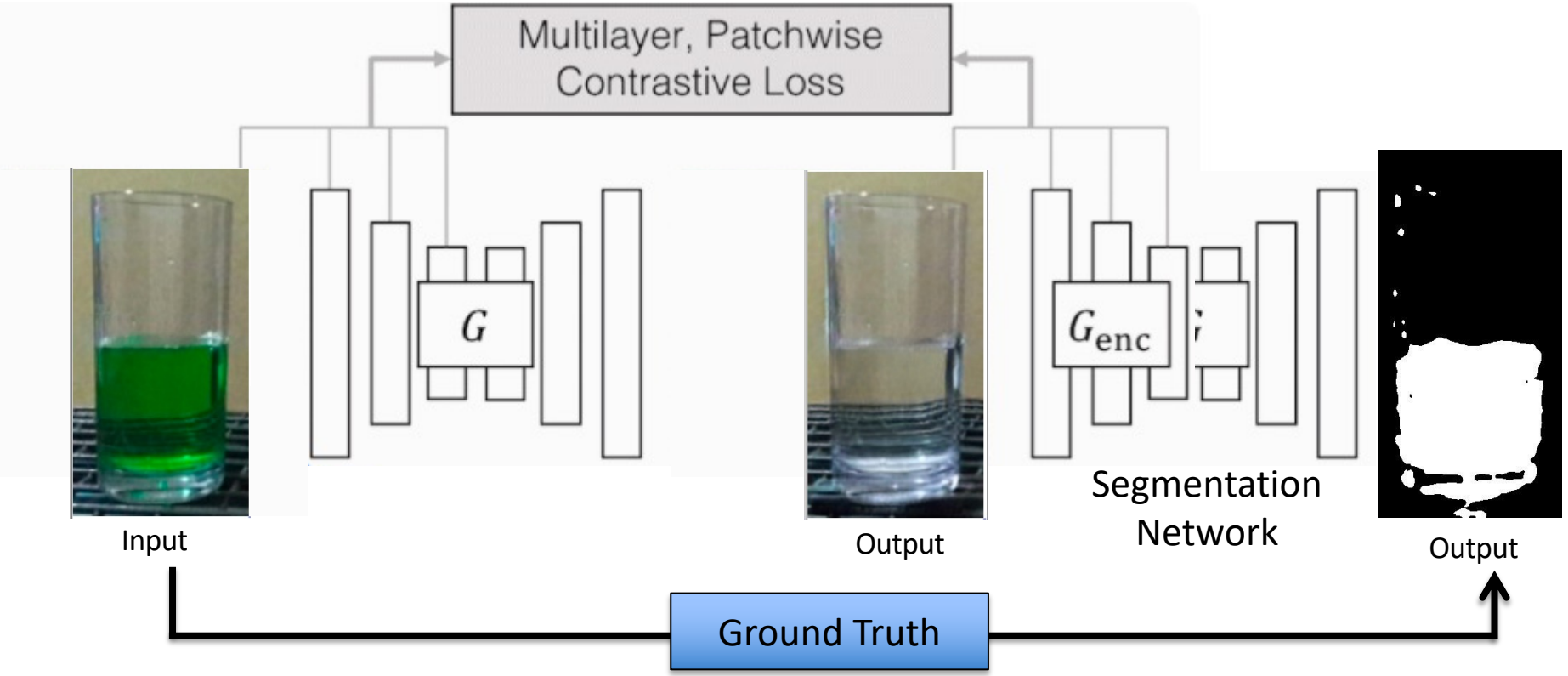
Convert Colored Liquid to Transparent Liquid



Convert Colored Liquid to Transparent Liquid



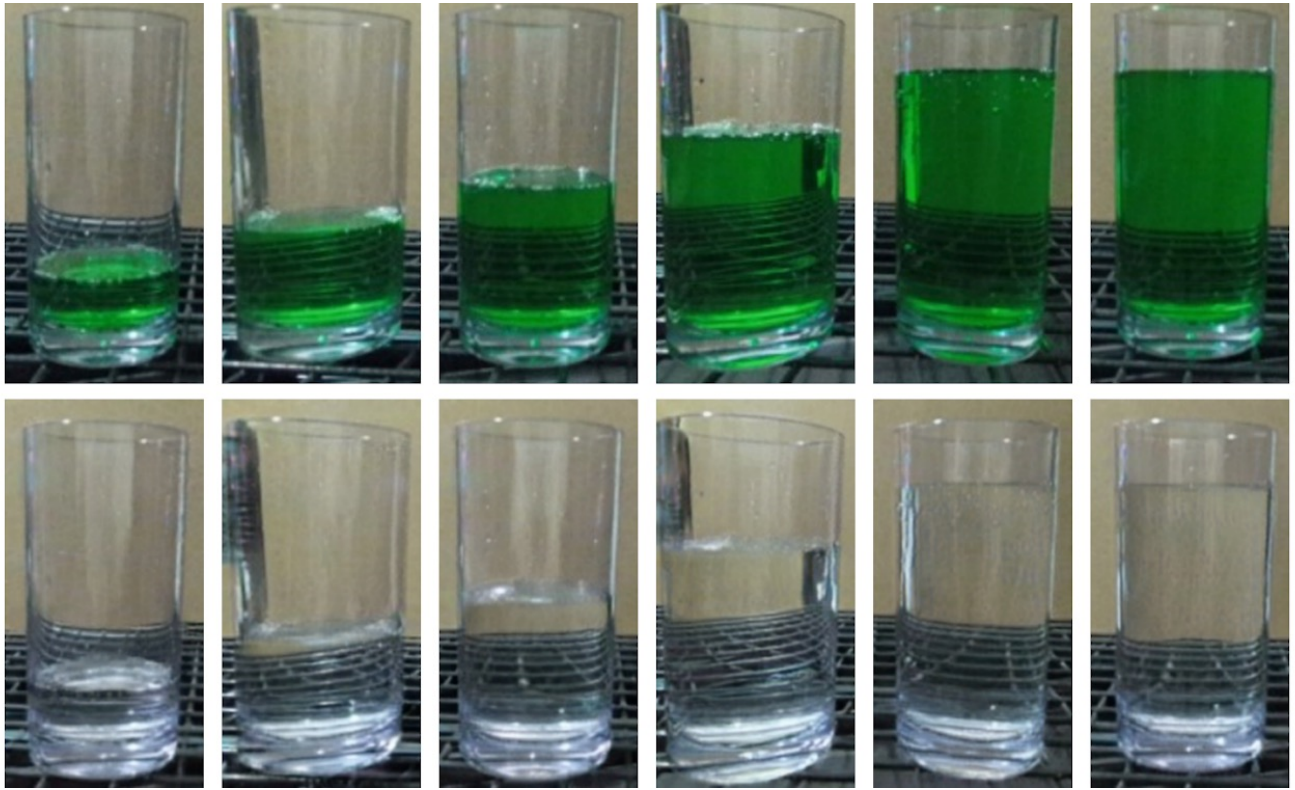
Use colored liquid to get a segmentation mask for transparent liquid



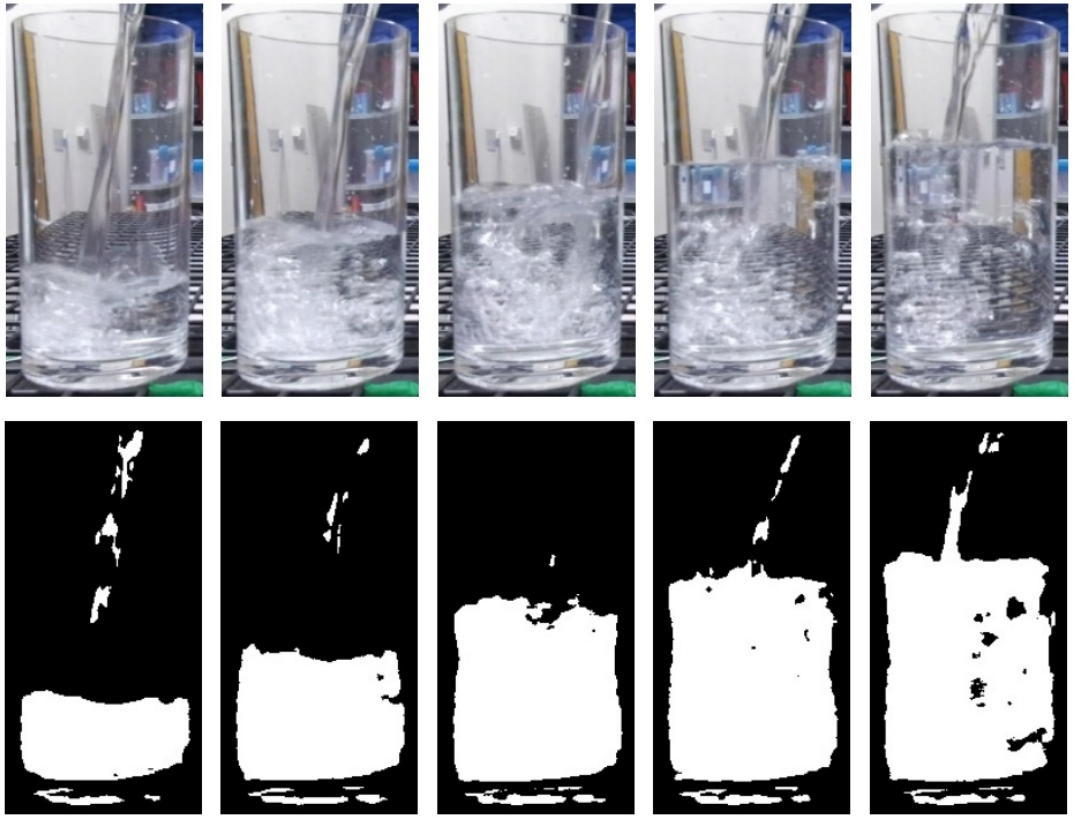
Cup Segmentation: Translab

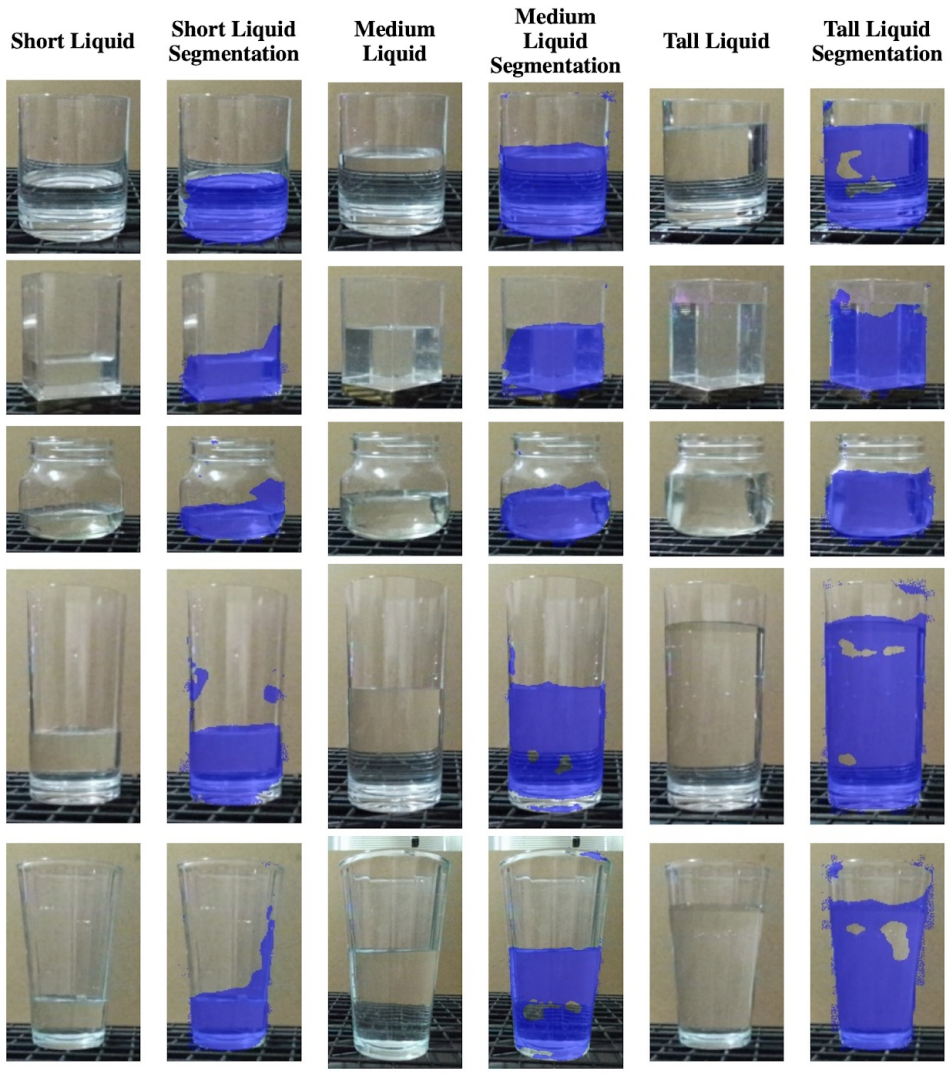
Xie, Enze, et al. "Segmenting transparent objects in the wild." *arXiv preprint arXiv:2003.13948* (2020).

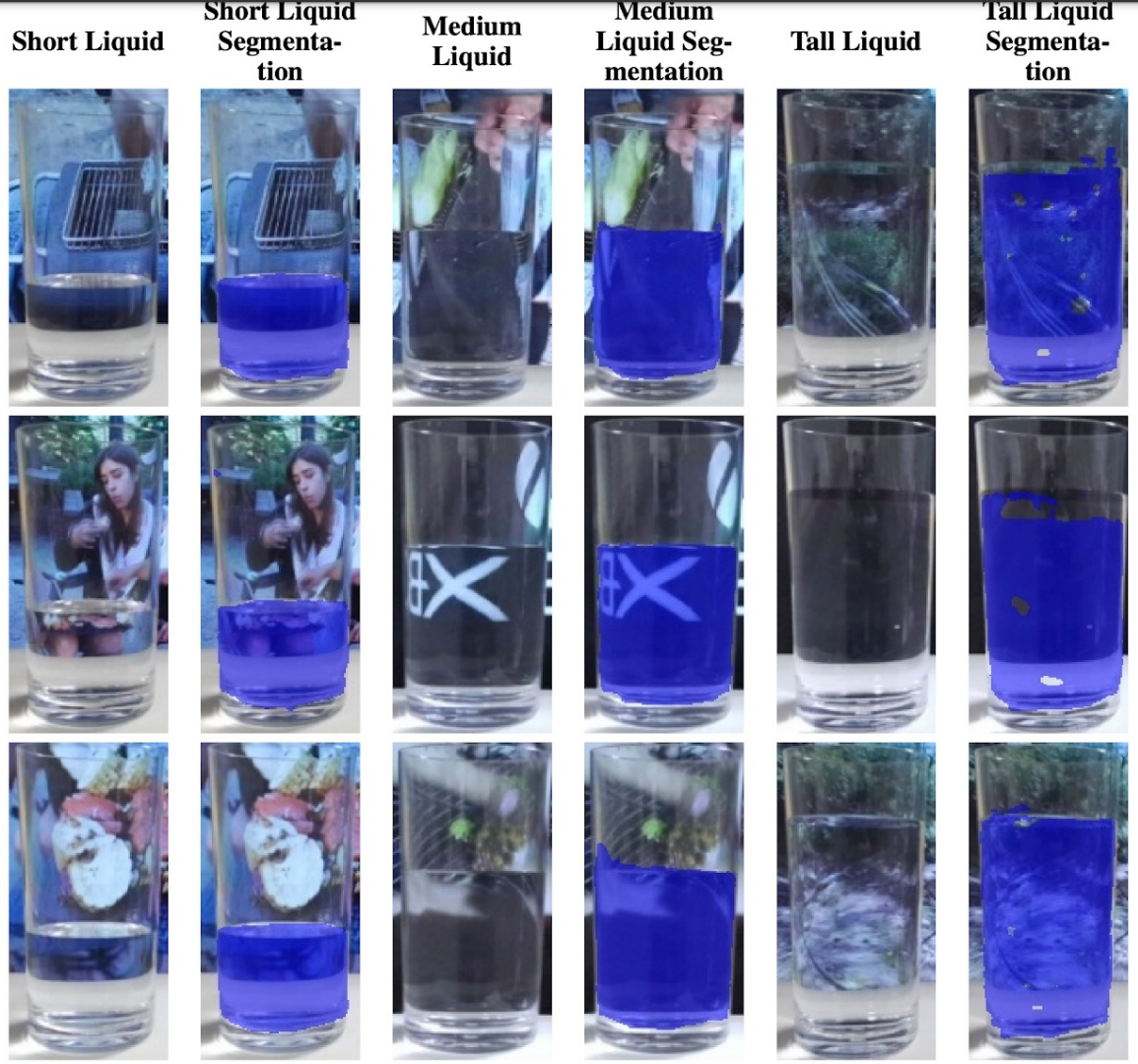
Results: Image Translation

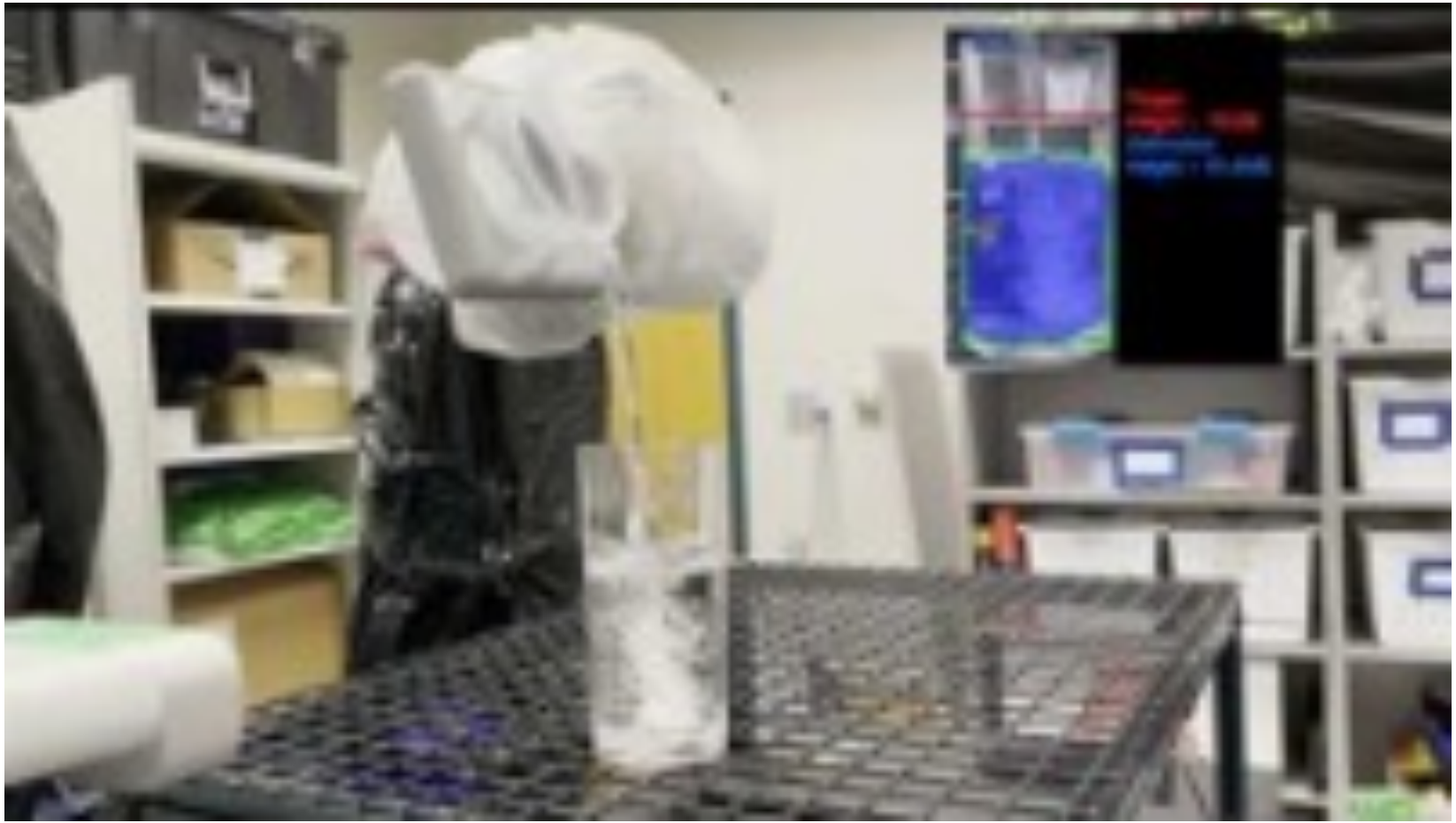


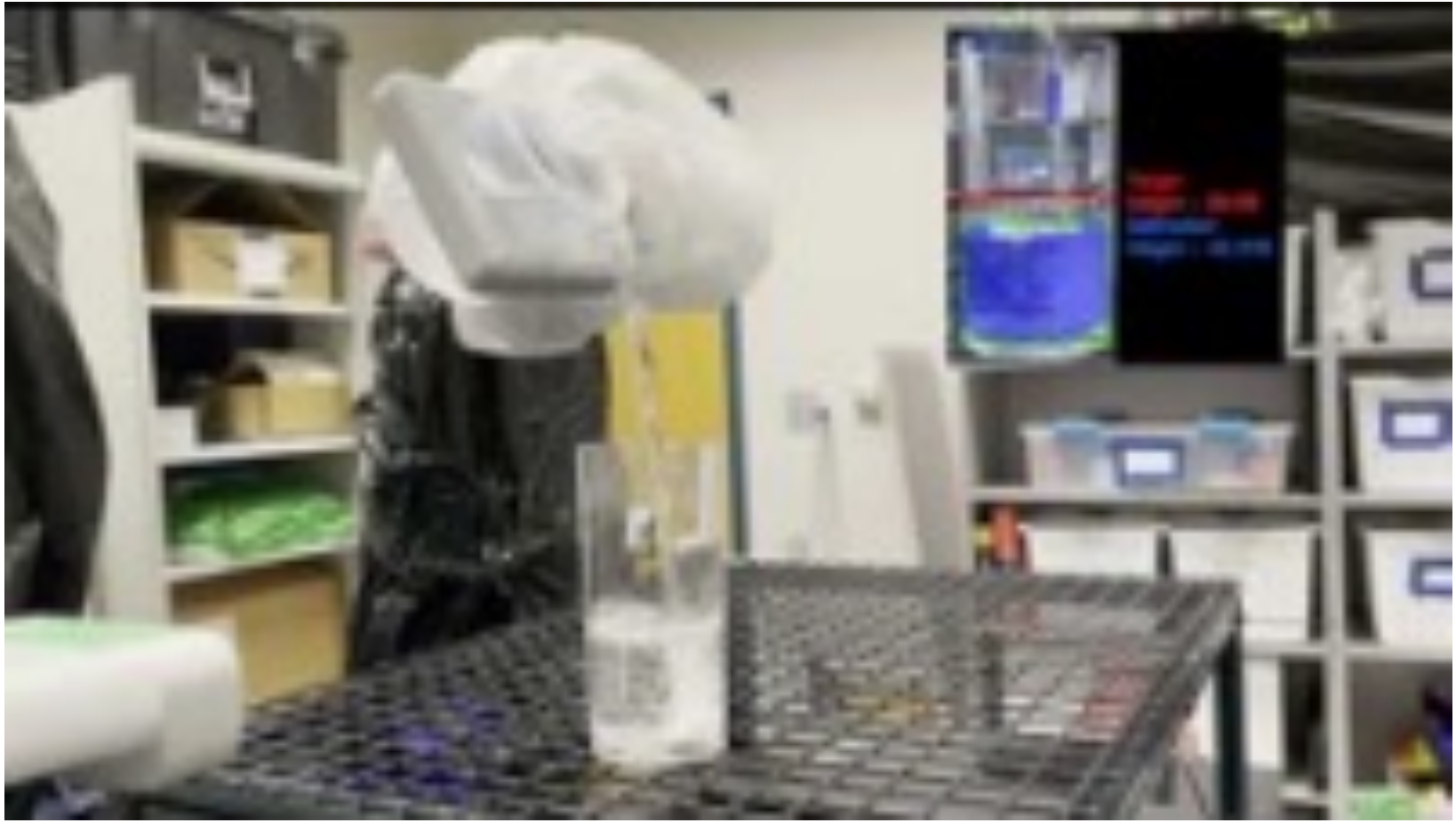
Results: Liquid Segmentation

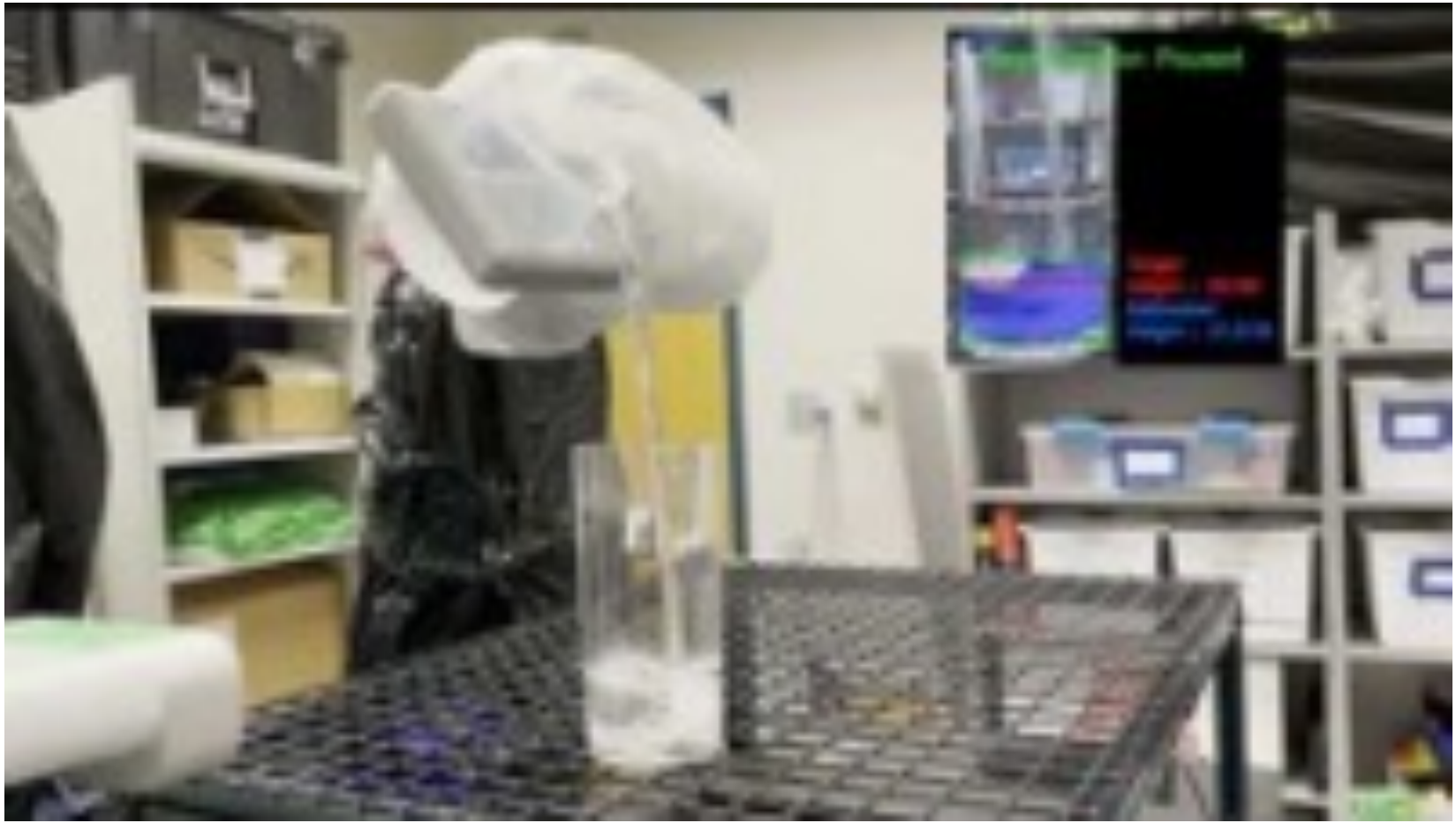


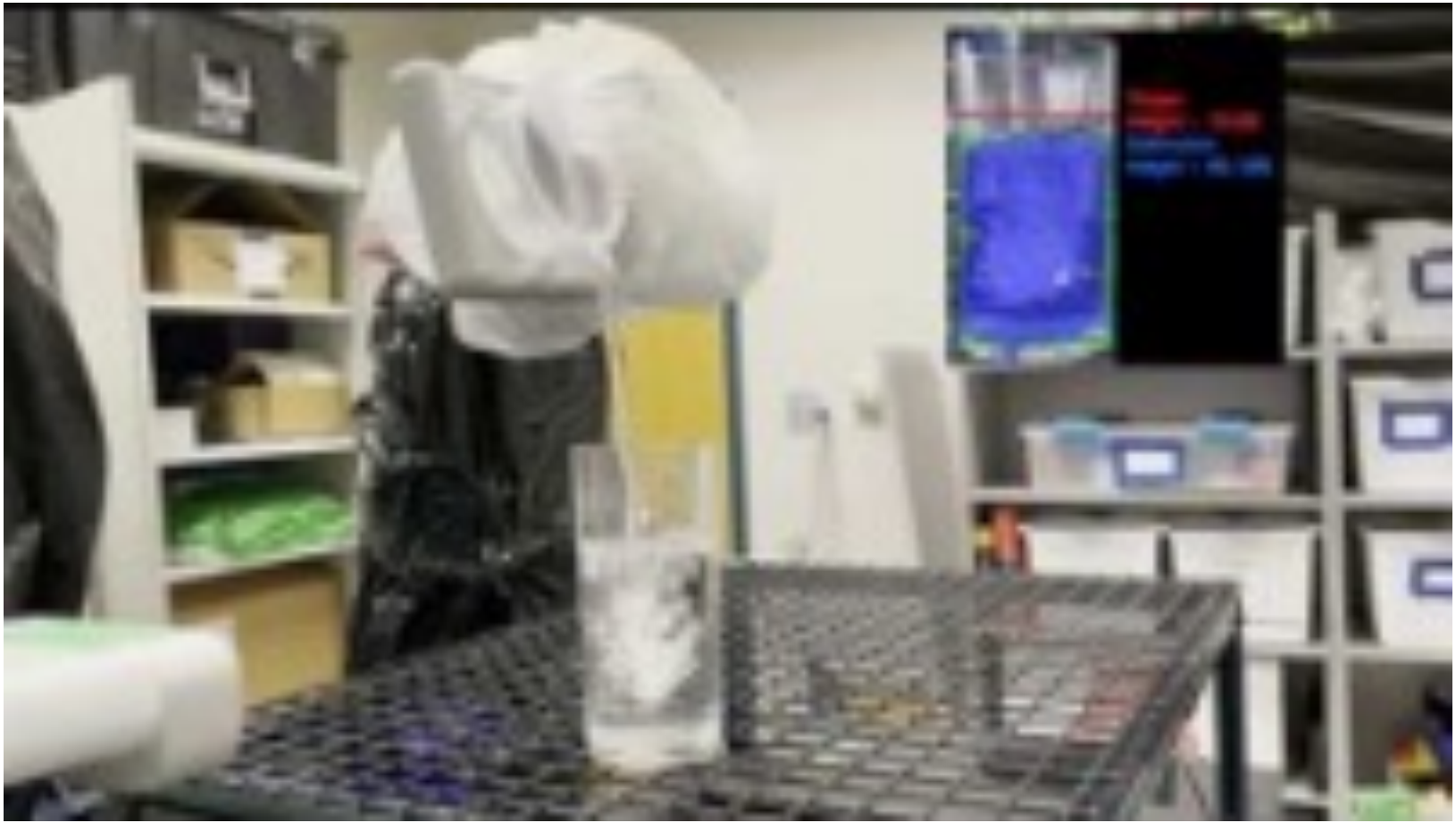

Short Liquid
Short Liquid Segmentation
Medium Liquid
Medium Liquid Segmentation
Tall Liquid
Tall Liquid Segmentation


Short Liquid
Short Liquid Segmenta-tion
Medium Liquid
Medium Liquid Seg-mentation
Tall Liquid
Tall Liquid Segmenta-tion

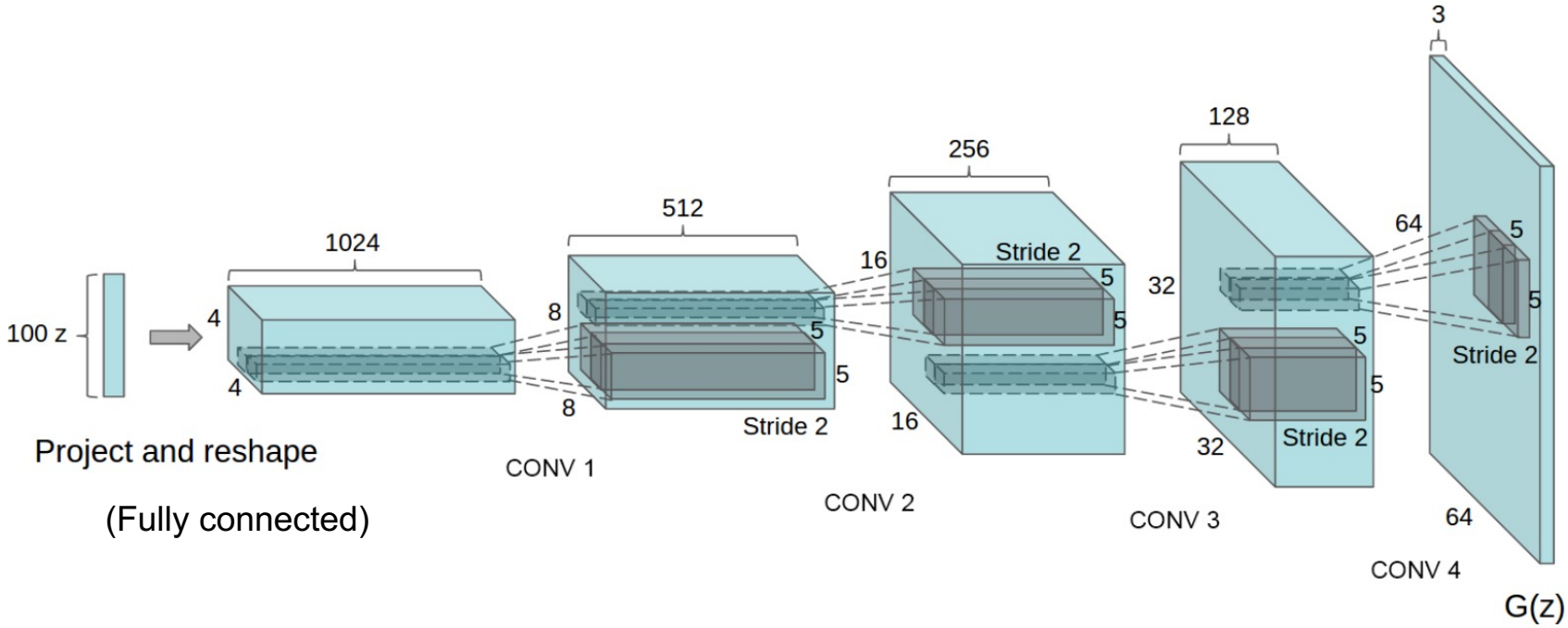




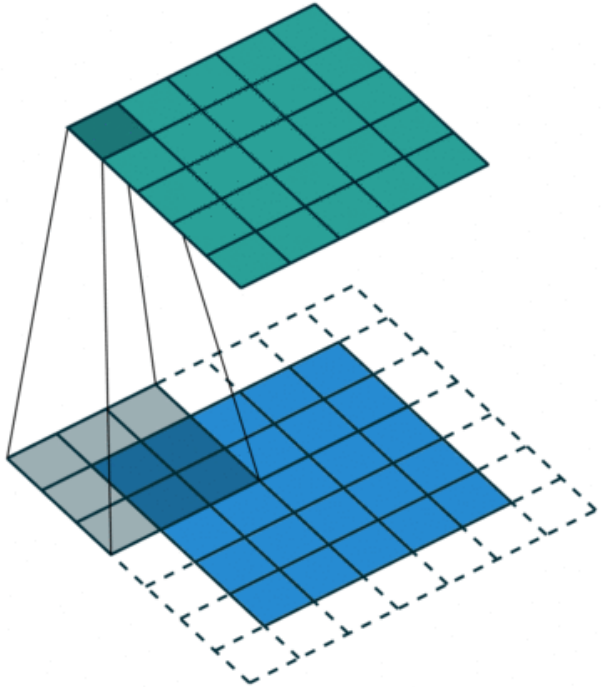




DCGAN Architecture

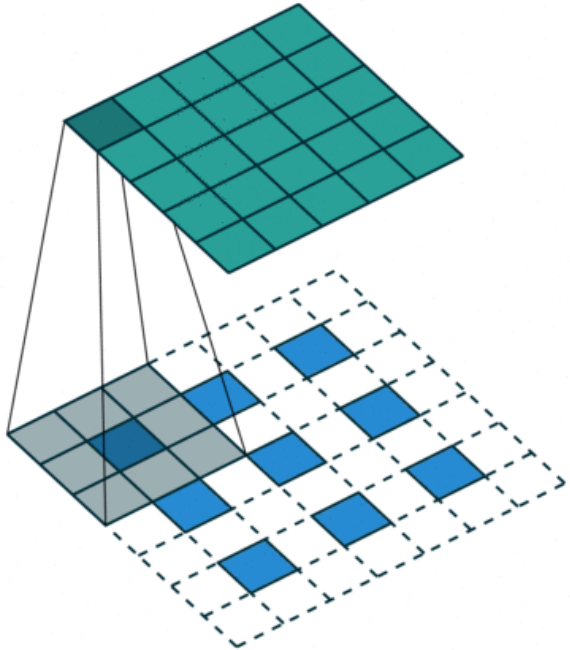


Convolutions



Regular convolution

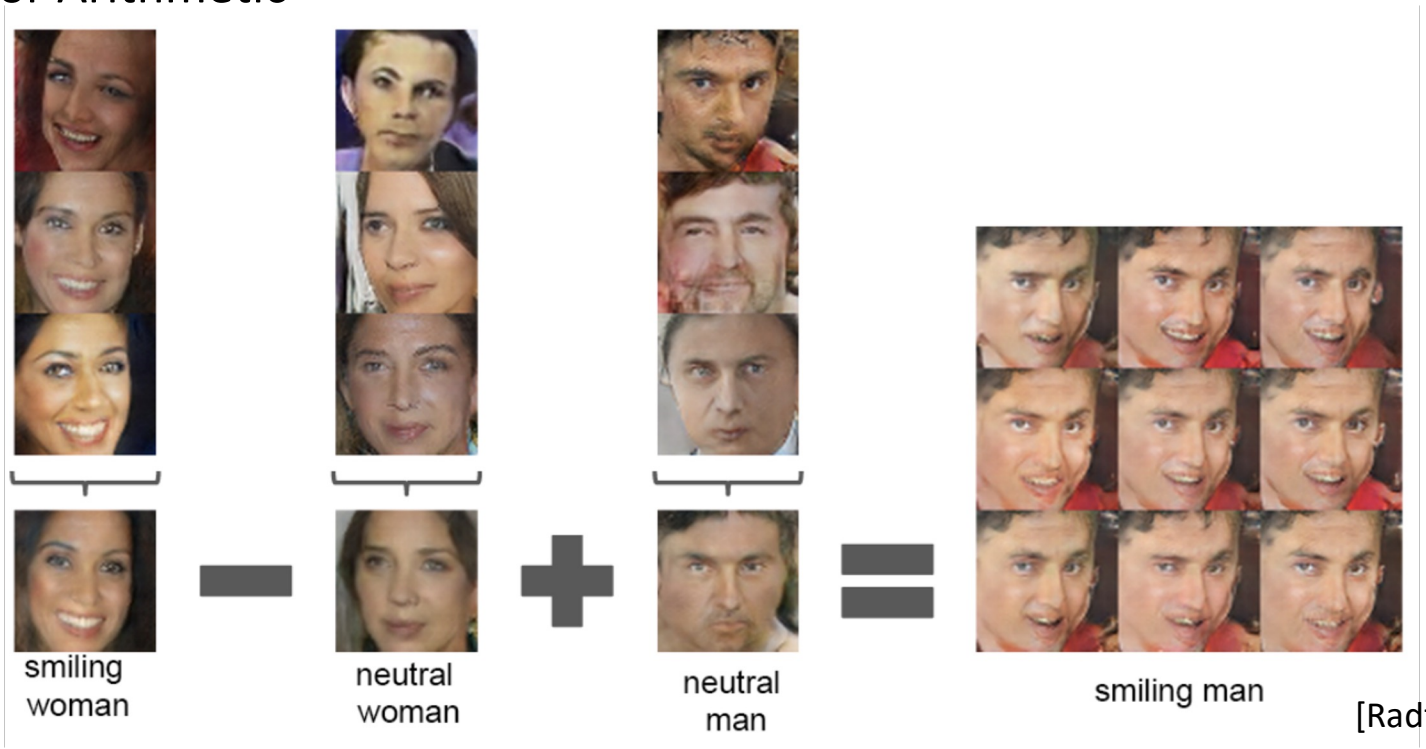
Blue is input
Green is output



Fractionally strided convolutions
Transposed convolution
Deconvolution

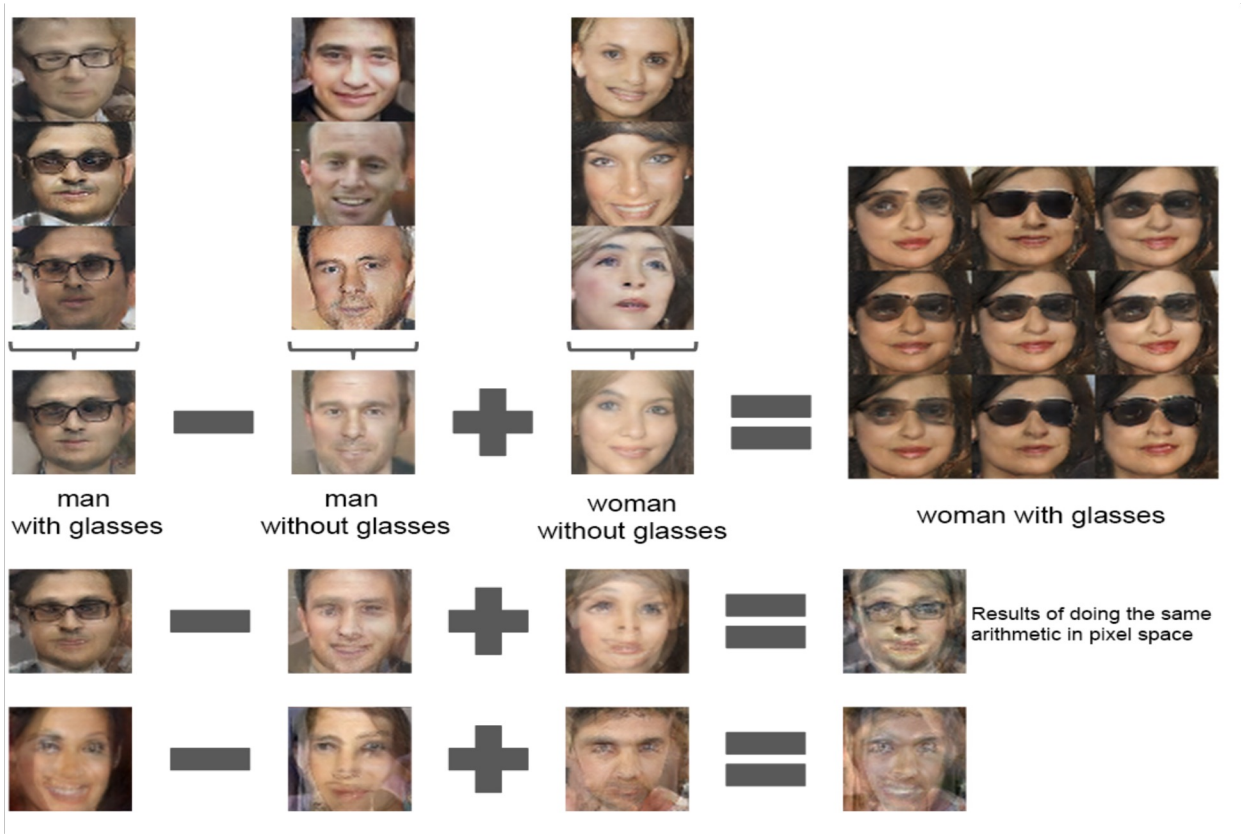
DCGAN - Key Results

- Vector Arithmetic



[Radford et al 2016]

DCGAN - Key Results



[Radford et al 2016]

DCGAN - Key Results

[Radford et al 2016]

Progressive growing of GANs

PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION

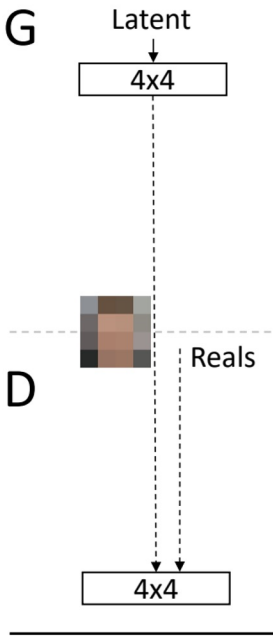
Tero Karras NVIDIA
Timo Aila NVIDIA
Samuli Laine NVIDIA
Jaakko Lehtinen NVIDIA and Aalto University
{tkarras,taila,slaine,jlehtinen}@nvidia.com

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.

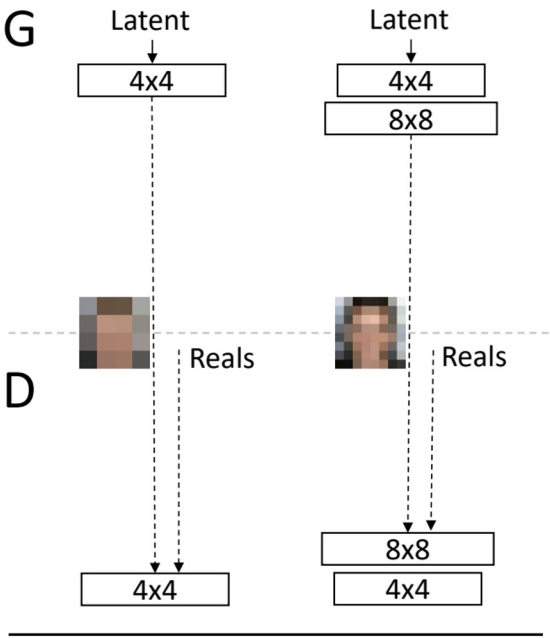
Karras et al 2017

Progressive growing of GANs



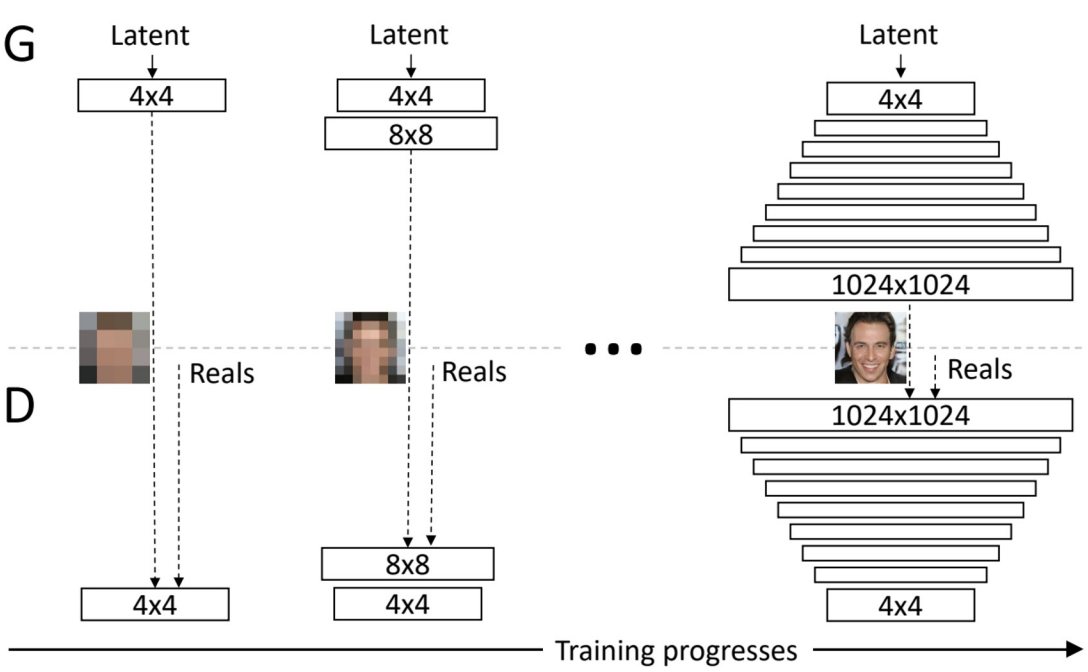
Karras et al 2017

Progressive growing of GANs



Karras et al 2017

Progressive growing of GANs



Karras et al 2017

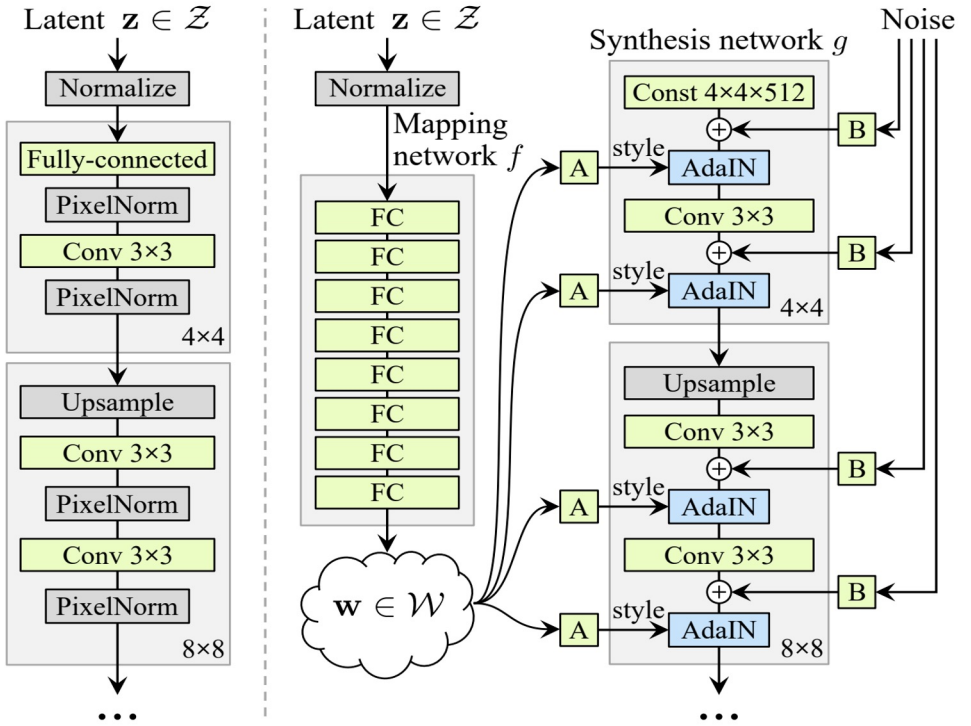
Progressive growing of GANs

Karras et al 2017

Progressive growing of GANs

Karras et al 2017

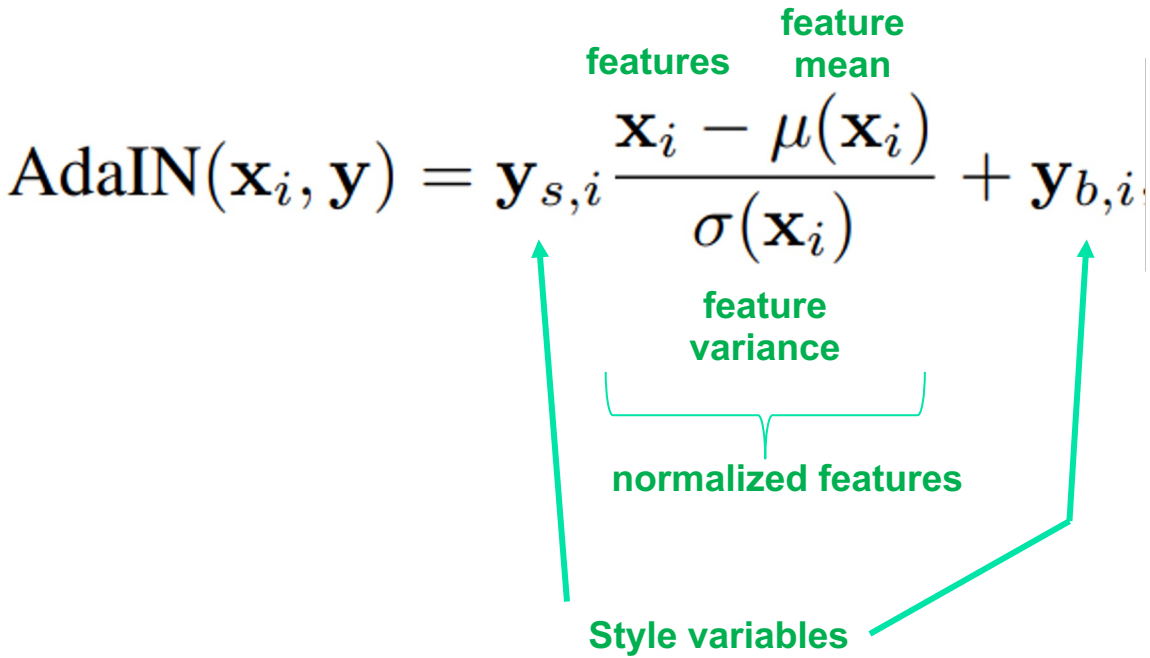
StyleGAN



(a) Traditional

(b) Style-based generator

StyleGAN - Adaptive Instance Norm



StyleGAN results

StyleGAN - Style Transfer

Summary

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