Image Generation

Slides adapted from S. Lazebnik

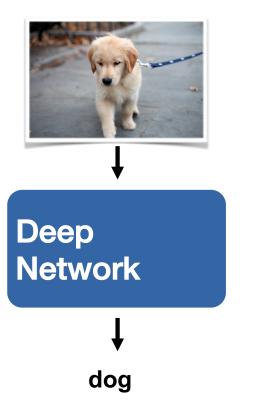
Image generation

Input: Image

- High dimensional
- Structured

Output: Label

- Low dimensional
- Easy



© 2019 Philipp Krähenbühl and Chao-Yuan Wu

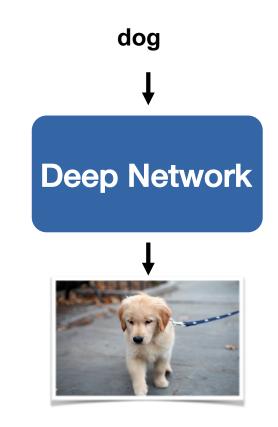
Input: Label / nothing

Low dimensional

Output: Image

- High dimensional
- Many possibilities

Very hard



Autoencoder

Image to image

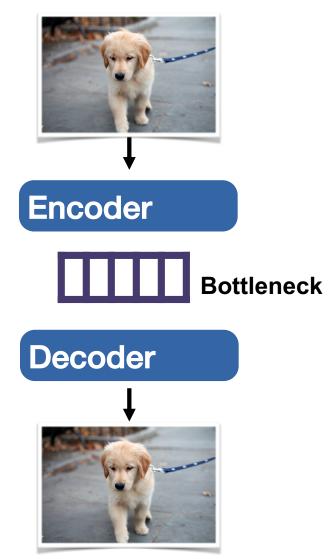
- with bottleneck
- Le
- Compression

Invertible mapping

- Does it learn to understand the image?
 - Only in the limit / best compression

Reducing the Dimensionality of Data with Neural Networks, Hinton and Salakhutdinov, Science, 2006

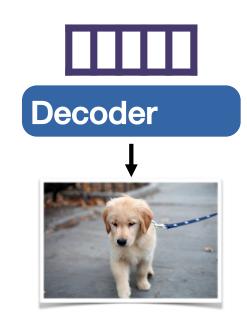
© 2019 Philipp Krähenbühl and Chao-Yuan Wu



Alternative that works

Deep image prior

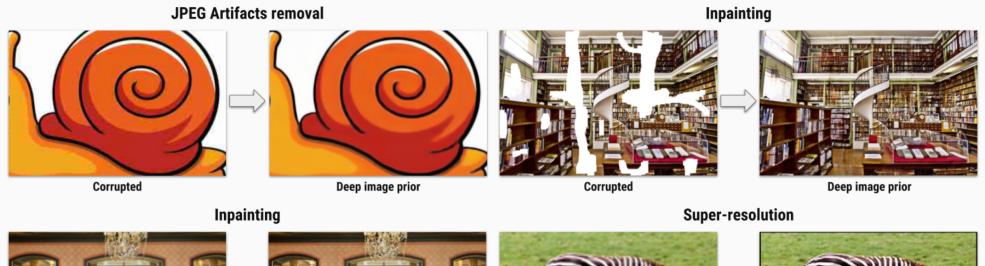
- Learn decoder of autoencoder
- Fixed random input
- Learns to denoise
- Learn how to generate an image from a single random code
- Search in the space on NN parameters to generate the image



Deep image prior, Ulyanov et al., CVPR 2018

© 2019 Philipp Krähenbühl and Chao-Yuan Wu

Deep image prior





Corrupted



Deep image prior



Corrupted



Deep image prior

Inpainting







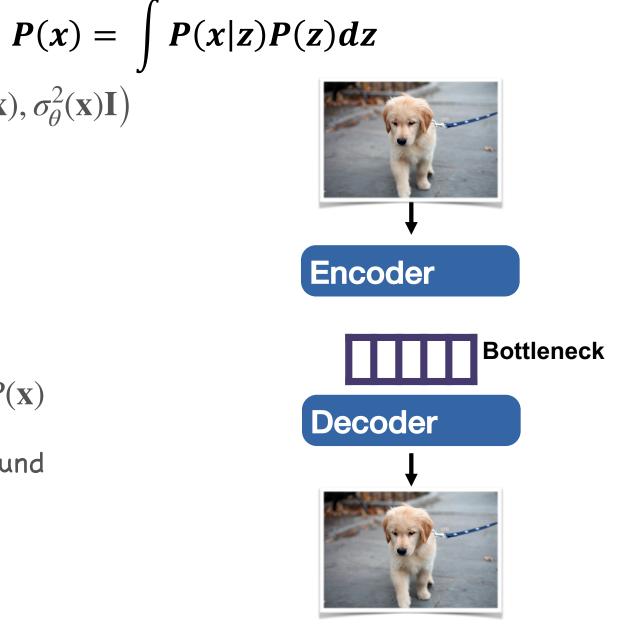


Variational Autoencoder

• Encoder

•
$$q(\mathbf{z} | \mathbf{x}) = \mathcal{N}\left(\mathbf{z}; \mu_{\theta}(\mathbf{x}), \sigma_{\theta}^{2}(\mathbf{x})\mathbf{I}\right)$$

- Sampling $\mathbf{f} \sim q(\mathbf{z} \,|\, \mathbf{x})$
- Decoder
 - $P(\mathbf{x} \mid \mathbf{f})$
- Approximately learns $P(\mathbf{x})$
 - Variational lower bound



VAE faces

Alec Radford



Generative adversarial networks

EBGAN (2017)



BigGAN (2018)

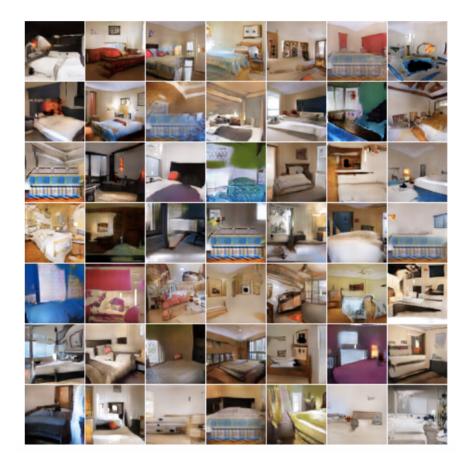


Outline

- Generative tasks
- Formulations
 - Original GAN
 - DCGAN
 - WGAN, improved WGAN
 - LSGAN
- Issues
 - Stability
 - Mode collapse
 - Evaluation methodology

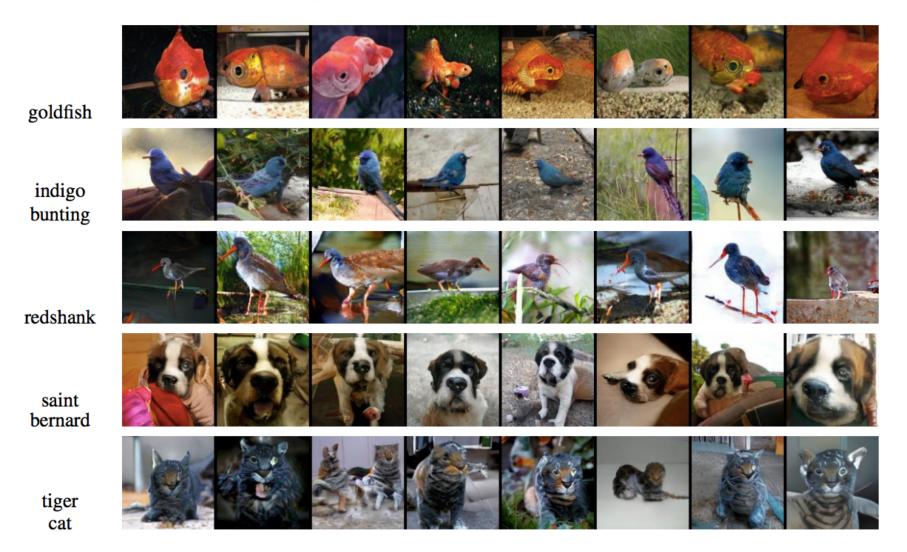
Generative tasks

- Generation (from scratch): learn to sample from the distribution represented by the training set
 - Unsupervised learning task



Generative tasks

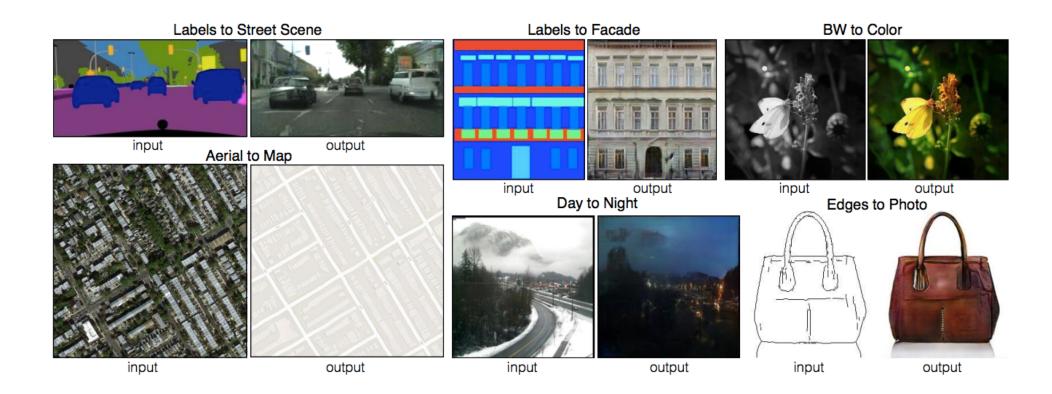
Conditional generation





Generative tasks

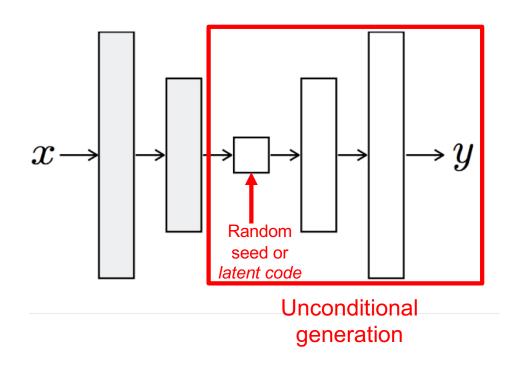
Image-to-image translation



P. Isola, J.-Y. Zhu, T. Zhou, A. Efros, <u>Image-to-Image Translation with Conditional Adversarial</u> <u>Networks</u>, CVPR 2017

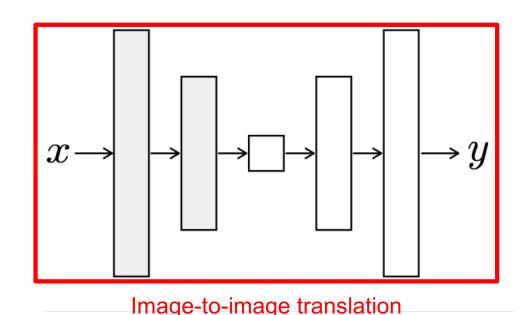
Designing a network for generative tasks

- 1. We need an architecture that can generate an image
 - Recall upsampling architectures for dense prediction



Designing a network for generative tasks

- 1. We need an architecture that can generate an image
 - Recall upsampling architectures for dense prediction

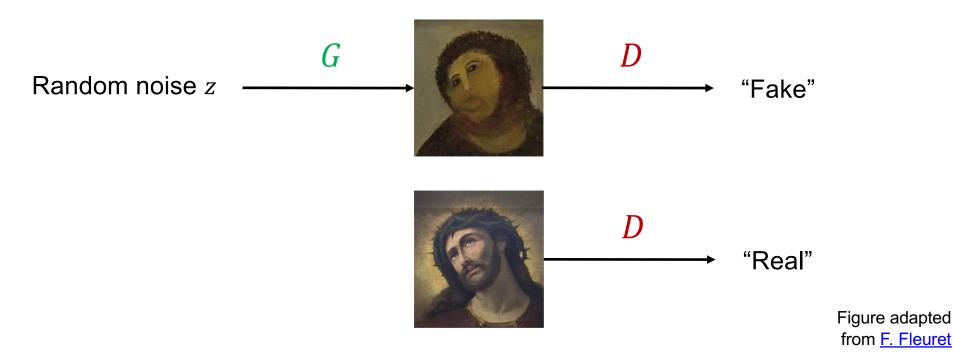


Designing a network for generative tasks

- 1. We need an architecture that can generate an image
 - Recall upsampling architectures for dense prediction
- 2. We need to design the right loss function

Generative adversarial networks

- Train two networks with opposing objectives:
 - Generator: learns to generate samples
 - Discriminator: learns to distinguish between generated and real samples



I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, <u>Generative adversarial nets</u>, NIPS 2014

GAN objective

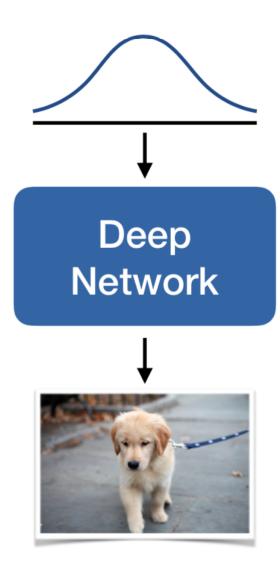
- The discriminator D(x) should output the probability that the sample x is real
 - That is, we want D(x) to be close to 1 for real data and close to 0 for fake
- Expected conditional log likelihood for real and generated data:

 $\mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{x \sim p_{\text{gen}}} \log (1 - D(x))$

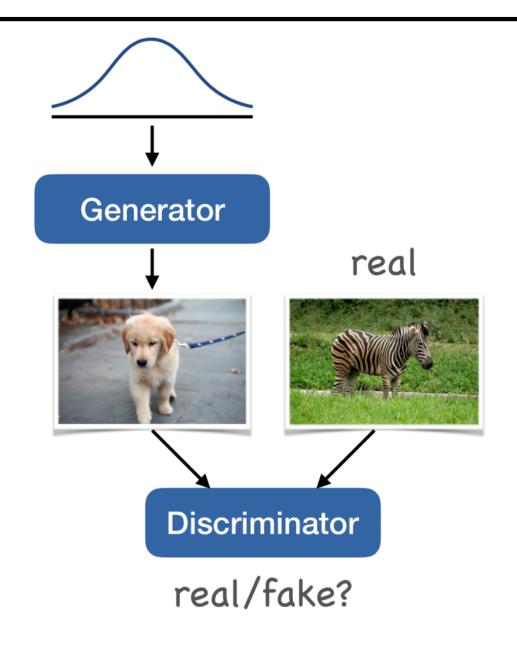
 $= \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$

We seed the generator with noise *z* drawn from a simple distribution *p* (Gaussian or uniform)

GAN



GAN



 $V(G,D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$

• The discriminator wants to correctly distinguish real and fake samples:

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

• The generator wants to fool the discriminator:

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

• Train the generator and discriminator jointly in a *minimax game*

 $V(G,D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$

- Assuming unlimited capacity for generator and discriminator and unlimited training data:
 - The objective $\min_{G} \max_{D} V(G, D)$ is equivalent to Jensen-Shannon divergence between p_{data} and p_{gen} and global optimum (Nash equilibrium) is given by $p_{data} = p_{gen}$
 - If at each step, *D* is allowed to reach its optimum given *G*, and *G* is updated to decrease V(G, D), then p_{gen} with eventually converge to p_{data}

 $V(G,D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$

- Alternate between
 - *Gradient ascent* on discriminator:

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

• *Gradient descent* on generator (minimize log-probability of discriminator being right):

 $G^* = \arg \min_G V(G, D)$ = $\arg \min_G \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$

 In practice, do gradient ascent on generator (maximize log-probability of discriminator being wrong):

 $G^* = \arg \max_G \mathbb{E}_{z \sim p} \log(D(G(z)))$

GAN training algorithm

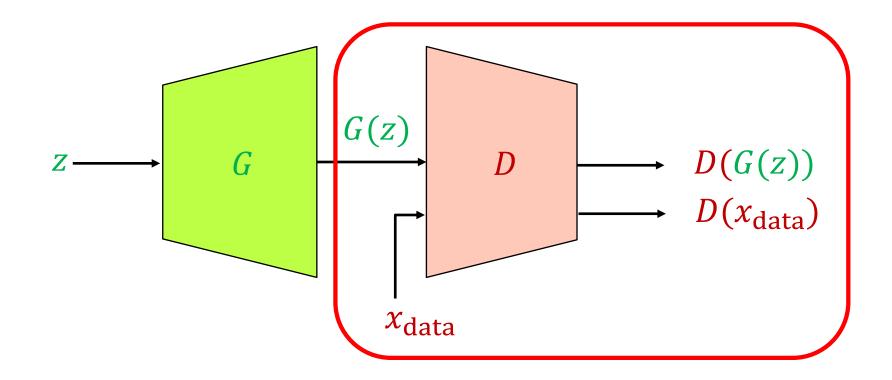
- Update discriminator
 - Repeat for *k* steps:
 - Sample mini-batch of noise samples z_1, \dots, z_m and mini-batch of real samples x_1, \dots, x_m
 - Update parameters of *D* by stochastic gradient ascent on $\frac{1}{m} \sum_{m} [\log D(x_m) + \log(1 - D(G(z_m)))]$
- Update generator
 - Sample mini-batch of noise samples z_1, \dots, z_m
 - Update parameters of *G* by stochastic gradient ascent on

$$\frac{1}{m}\sum_{m}\log D(G(z_m))$$

• Repeat until happy with results

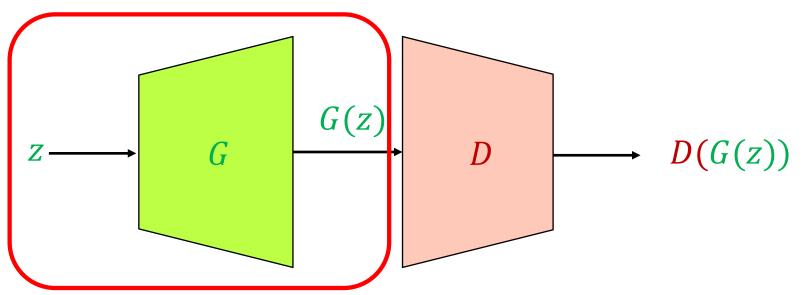
GAN: Conceptual picture

- Update discriminator: push $D(x_{data})$ close to 1 and D(G(z)) close to 0
 - The generator is a "black box" to the discriminator



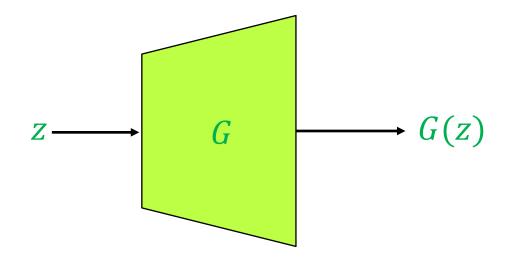
GAN: Conceptual picture

- Update generator: increase D(G(z))
 - Requires back-propagating through the composed generator-discriminator network (i.e., the discriminator cannot be a black box)
 - The generator is exposed to real data only via the output of the discriminator (and its gradients)



GAN: Conceptual picture

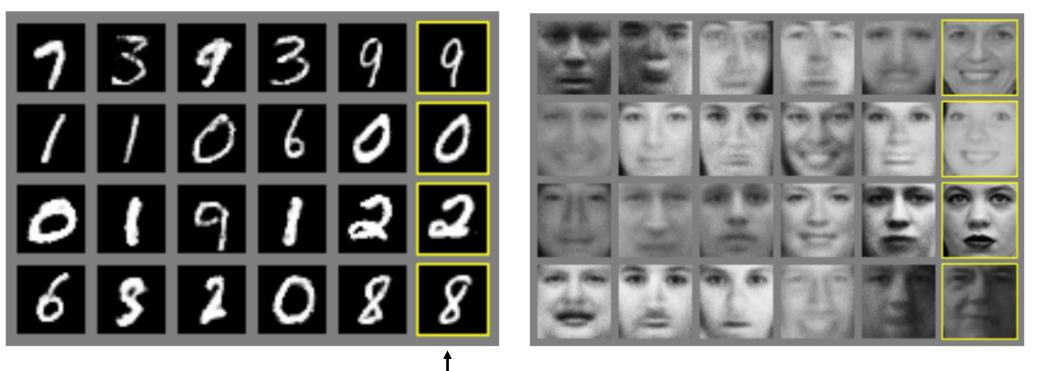
• Test time



Original GAN results

MNIST digits

Toronto Face Dataset



Nearest real image for sample to the left

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, <u>Generative adversarial nets</u>, NIPS 2014

Original GAN results

CIFAR-10 (FC networks)

CIFAR-10 (conv networks)



I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, <u>Generative adversarial nets</u>, NIPS 2014

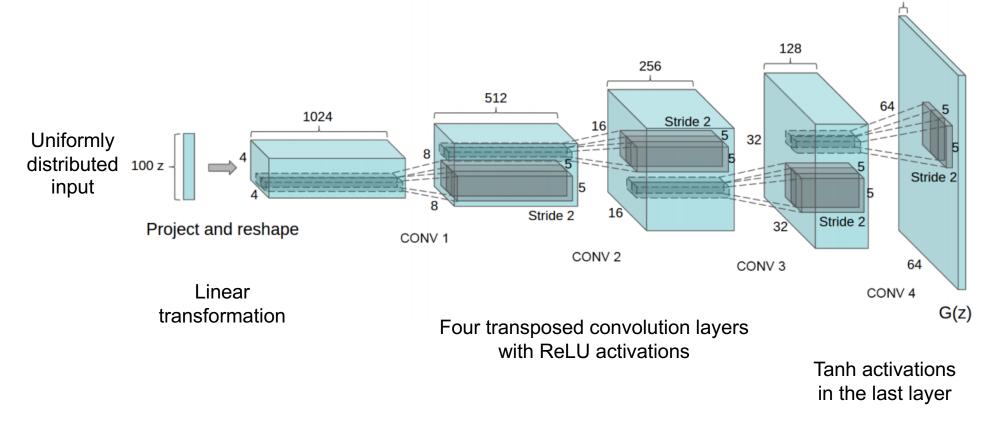
DCGAN

- Propose principles for designing convolutional architectures for GANs
- Generator architecture

A. Radford, L. Metz, S. Chintala, <u>Unsupervised representation learning with deep</u> <u>convolutional generative adversarial networks</u>, ICLR 2016

DCGAN

- Propose principles for designing convolutional architectures for GANs
- Generator architecture



A. Radford, L. Metz, S. Chintala, <u>Unsupervised representation learning with deep</u> <u>convolutional generative adversarial networks</u>, ICLR 2016

DCGAN

- Propose principles for designing convolutional architectures for GANs
- Discriminator architecture
 - Don't use pooling, only strided convolutions
 - Use Leaky ReLU activations (sparse gradients cause problems for training)
 - Use only one FC layer before the softmax output
 - Use batch normalization after most layers (in the generator also)

A. Radford, L. Metz, S. Chintala, <u>Unsupervised representation learning with deep</u> <u>convolutional generative adversarial networks</u>, ICLR 2016

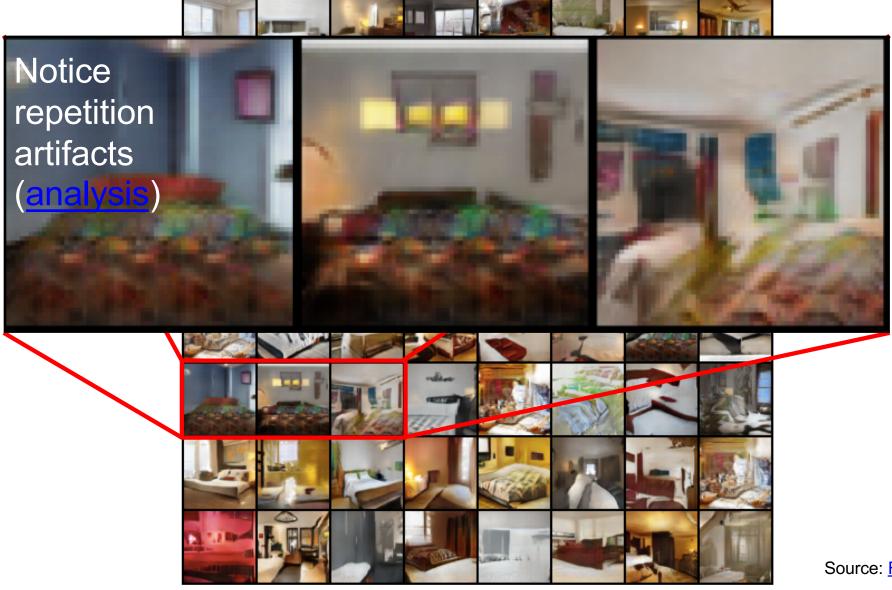
Generated bedrooms after one epoch



Generated bedrooms after five epochs

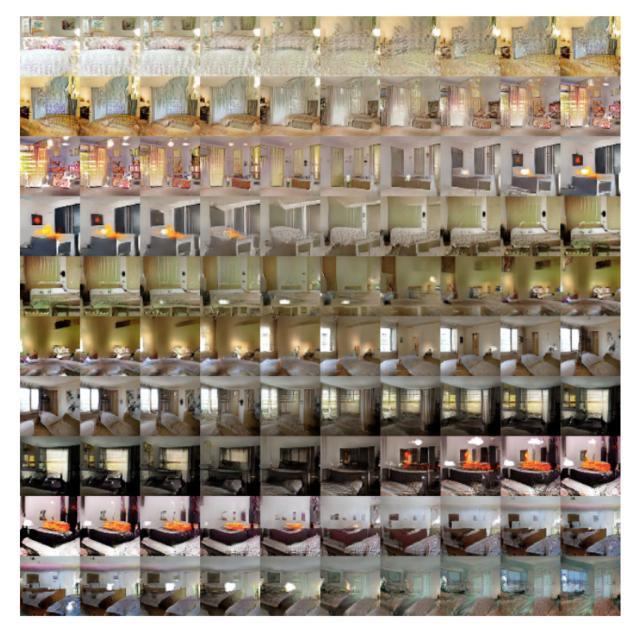


Generated bedrooms from reference implementation

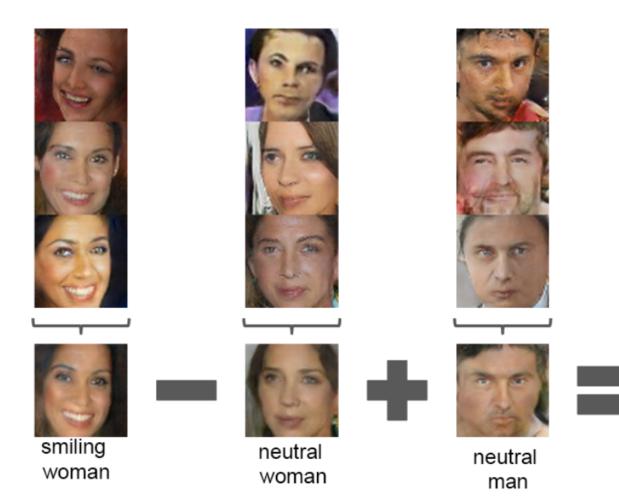


Source: F. Fleuret

Interpolation between different points in the z space

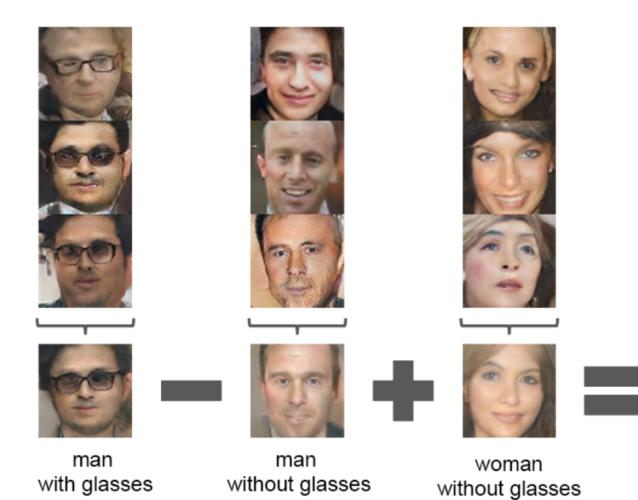


• Vector arithmetic in the z space



DCGAN results

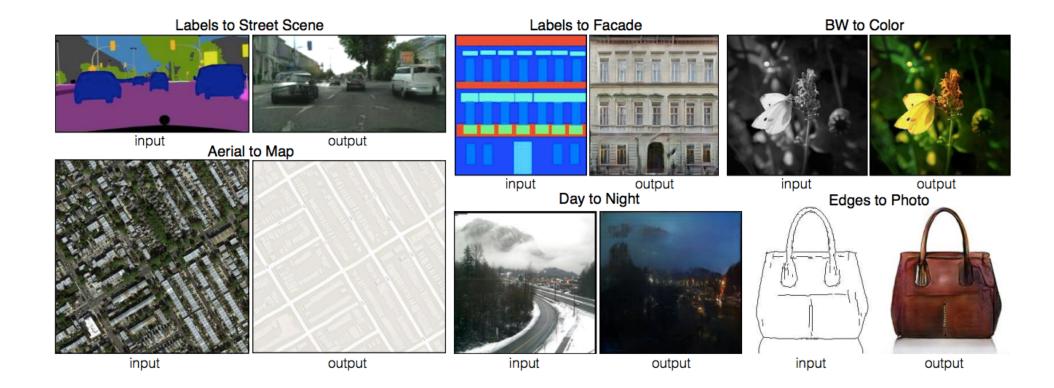
• Vector arithmetic in the z space



DCGAN results

• Pose transformation by adding a "turn" vector





P. Isola, J.-Y. Zhu, T. Zhou, A. Efros, <u>Image-to-Image Translation with Conditional Adversarial</u> <u>Networks</u>, CVPR 2017

Pix2Pix - Image2Image translation

real

"Autoencoder"

 Different input and output

GAN loss

• High fidelity reconstruction

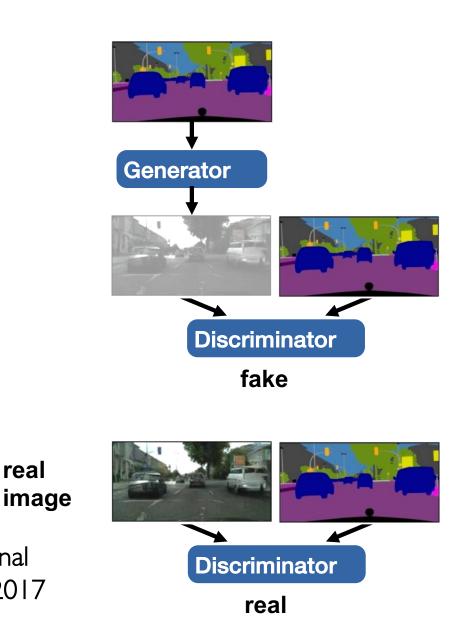
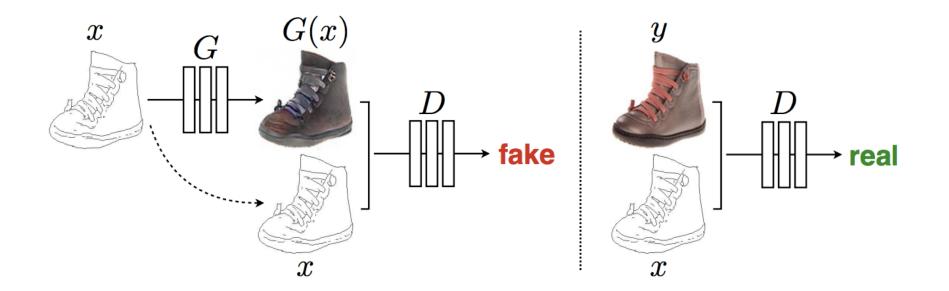
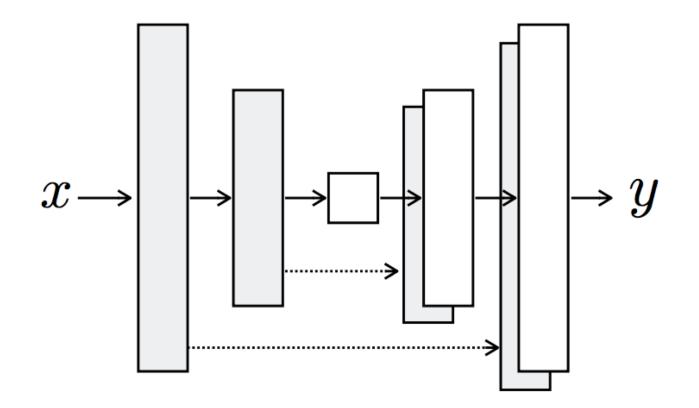


Image-to-Image Translation with Conditional Adversarial Networks, Isola et al., CVPR 2017

- Produce modified image y conditioned on input image x (note change of notation)
 - Generator receives *x* as input
 - Discriminator receives an x, y pair and has to decide whether it is real or fake

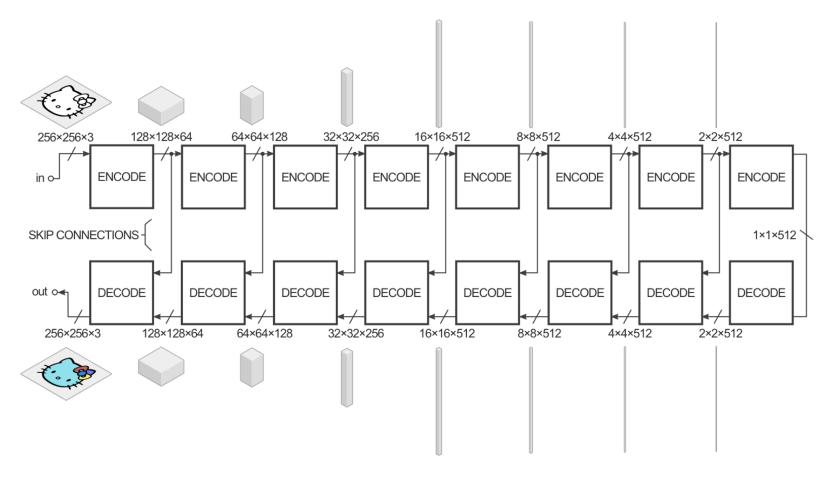


• Generator architecture: U-Net



• Note: no z used as input, transformation is basically deterministic

• Generator architecture: U-Net



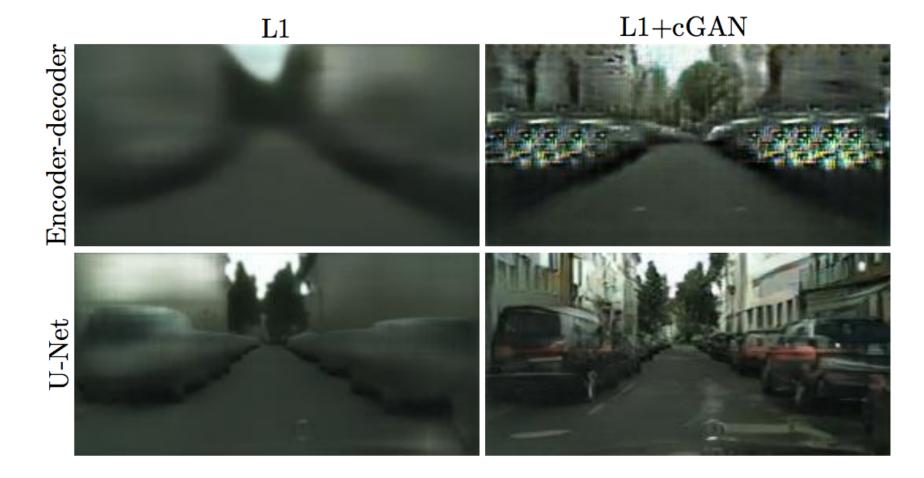
Encode: convolution \rightarrow BatchNorm \rightarrow ReLU

Decode: transposed convolution \rightarrow BatchNorm \rightarrow ReLU



• Generator architecture: U-Net

Effect of adding skip connections to the generator



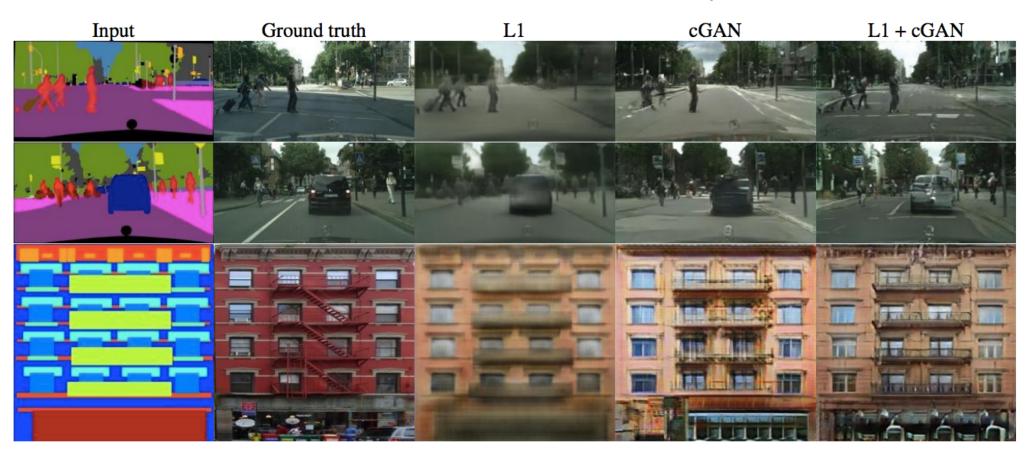
 Generator loss: GAN loss plus L1 reconstruction penalty

 $G^* = \arg\min_G \max_D \mathcal{L}_{GAN}(G, D) + \lambda \sum_i ||y_i - G(x_i)||_1$

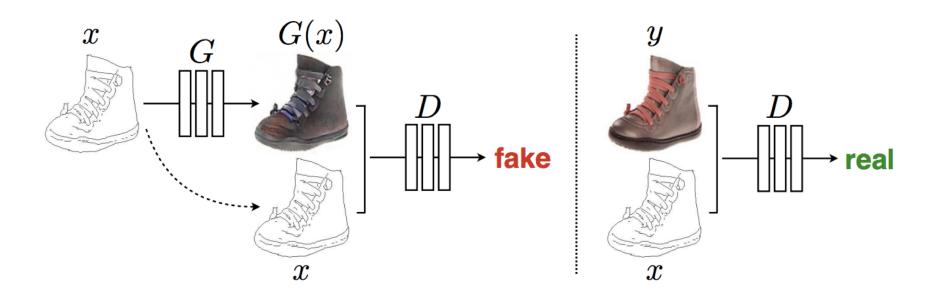
Generated output $G(x_i)$ should be close to ground truth target y_i

 Generator loss: GAN loss plus L1 reconstruction penalty

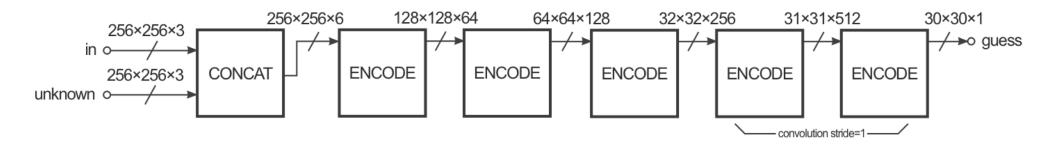
 $G^* = \arg\min_G \max_D \mathcal{L}_{GAN}(G, D) + \lambda \sum_i ||y_i - G(x_i)||_1$



- Discriminator: PatchGAN
 - Given input image *x* and second image *y*, decide whether *y* is a ground truth target or produced by the generator



- Discriminator: PatchGAN
 - Given input image *x* and second image *y*, decide whether *y* is a ground truth target or produced by the generator
 - Output is a 30 x 30 map where each value (0 to 1) represents the quality of the corresponding section of the output image
 - Fully convolutional network, effective patch size can be increased by increasing the depth



- Discriminator: PatchGAN
 - Given input image *x* and second image *y*, decide whether *y* is a ground truth target or produced by the generator
 - Output is a 30 x 30 map where each value (0 to 1) represents the quality of the corresponding section of the output image
 - Fully convolutional network, effective patch size can be increased by increasing the depth

Effect of discriminator patch size on generator output



• Translating between maps and aerial photos

Map to aerial photo

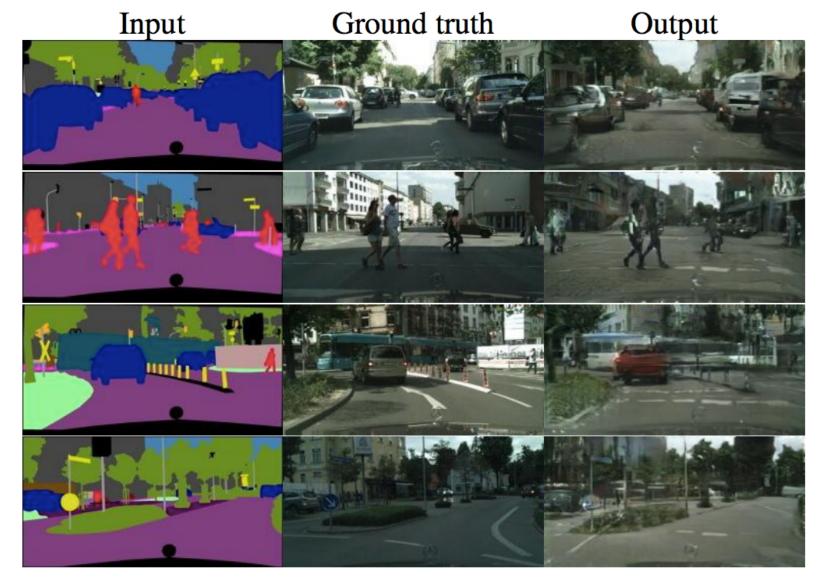
Aerial photo to map



- Translating between maps and aerial photos
- Human study:

	Photo \rightarrow Map	$\mathbf{Map} \rightarrow \mathbf{Photo}$
Loss	% Turkers labeled real	% Turkers labeled real
L1	$2.8\%\pm1.0\%$	$0.8\%\pm0.3\%$
L1+cGAN	$6.1\% \pm 1.3\%$	$\textbf{18.9\%} \pm \textbf{2.5\%}$

• Semantic labels to scenes



- Semantic labels to scenes
- Evaluation: FCN score
 - The higher the quality of the output, the better the FCN should do at recovering the original semantic labels

Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	0.42	0.15	0.11
GAN	0.22	0.05	0.01
cGAN	0.57	0.22	0.16
L1+GAN	0.64	0.20	0.15
L1+cGAN	0.66	0.23	0.17
Ground truth	0.80	0.26	0.21

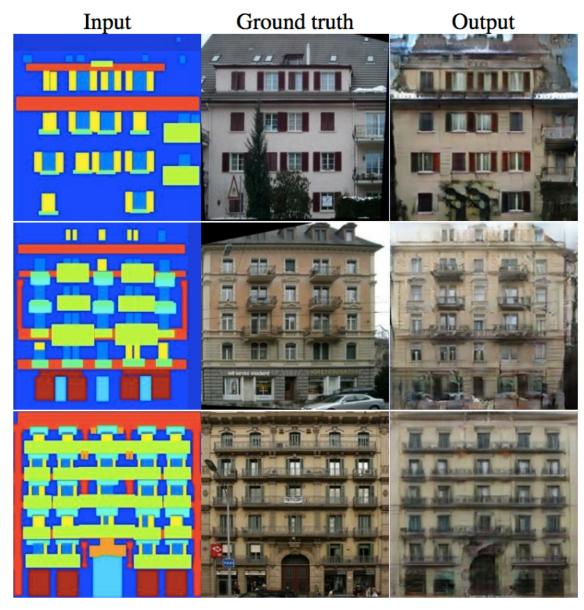
• Scenes to semantic labels

cGAN Ground truth L1 Input

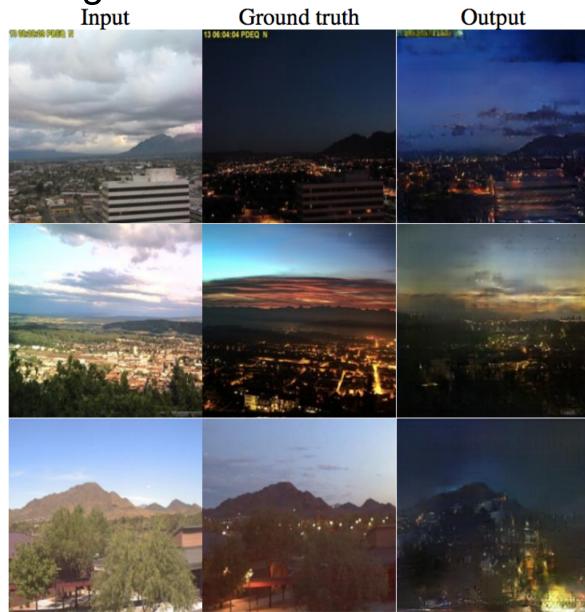
- Scenes to semantic labels
- Accuracy is worse than that of regular FCNs or generator with L1 loss

Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	0.86	0.42	0.35
cGAN	0.74	0.28	0.22
L1+cGAN	0.83	0.36	0.29

• Semantic labels to facades



• Day to night



• Edges to photos



• pix2pix demo

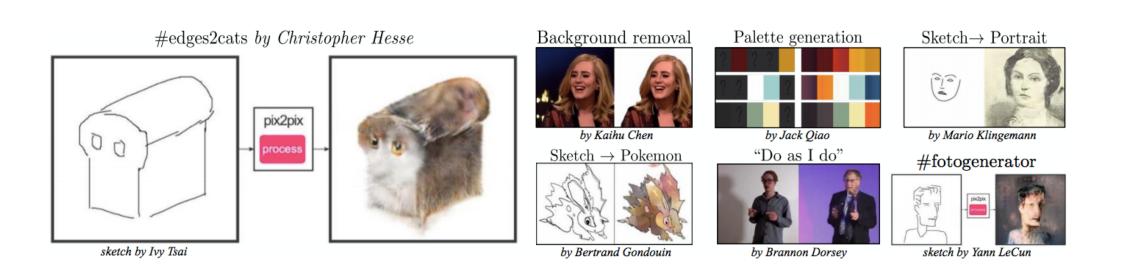
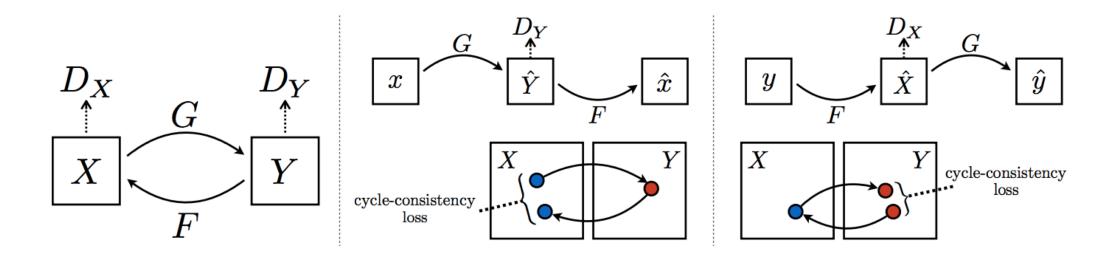


Image-to-image translation: Limitations

- Visual quality could be improved
- Requires *x*, *y* pairs for training
- Does not model conditional distribution P(y|x), returns a single mode instead

CycleGAN

- Given: domains X and Y
- Train two generators F and G and two discriminators D_X and D_Y
 - *G* translates from *X* to *Y*, *F* translates from *Y* to *X*
 - D_X recognizes images from X, D_Y from Y
 - We want $F(G(x)) \approx x$ and $G(F(y)) \approx y$



CycleGAN: Architecture

• Generators:

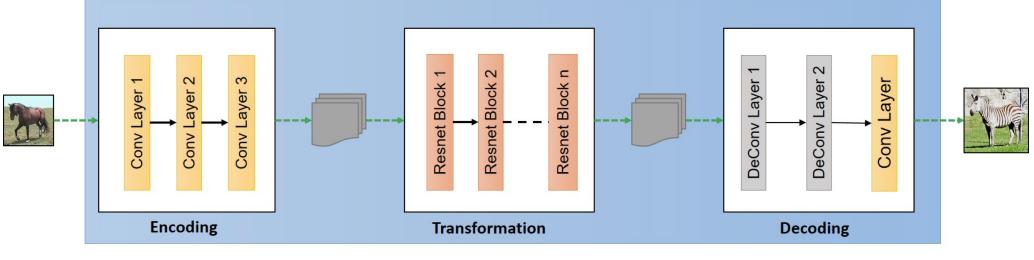


Figure source

• Discriminators: PatchGAN on 70 x 70 patches

CycleGAN: Loss

- Requirements:
 - G translates from X to Y, F translates from Y to X
 - D_X recognizes images from X, D_Y from Y
 - We want $F(G(x)) \approx x$ and $G(F(y)) \approx y$
- CycleGAN discriminator loss: LSGAN

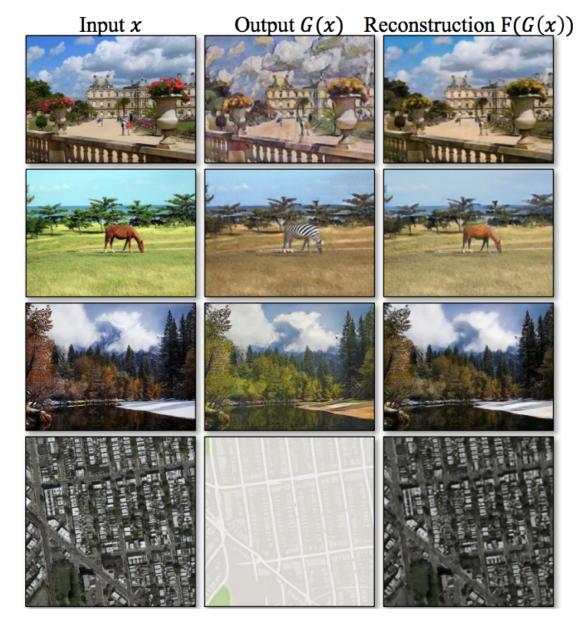
$$\mathcal{L}_{\text{GAN}}(D_Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)} \left[(D_Y(y) - 1)^2 \right] + \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[D_Y(G(x))^2 \right]$$
$$\mathcal{L}_{\text{GAN}}(D_X) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[(D_X(x) - 1)^2 \right] + \mathbb{E}_{y \sim p_{\text{data}}(y)} \left[D_X(F(y))^2 \right]$$

CycleGAN generator loss:

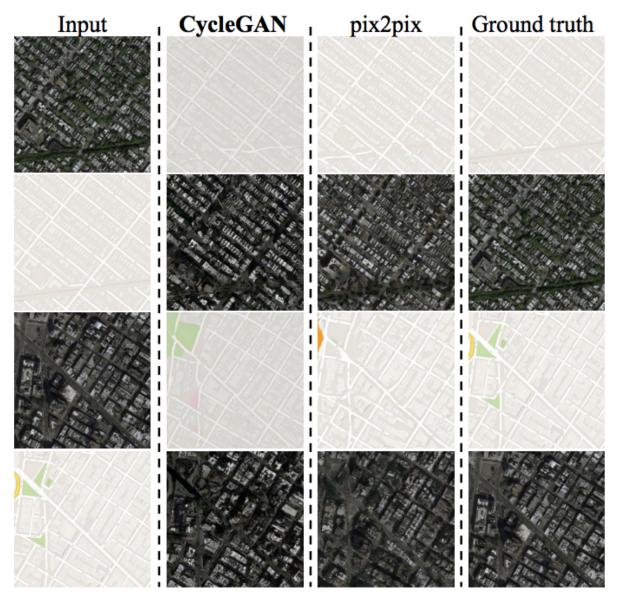
 $\mathcal{L}_{\text{cyc}}(G,F) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [D_Y(G(x) - 1)^2] + \mathbb{E}_{y \sim p_{\text{data}}(y)} [D_X(F(y) - 1)^2]$ $+ \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\left\| F(G(x)) - x \right\|_1 \right] + \mathbb{E}_{y \sim p_{\text{data}}(y)} \left[\left\| G(F(y)) - y \right\|_1 \right]$

CycleGAN

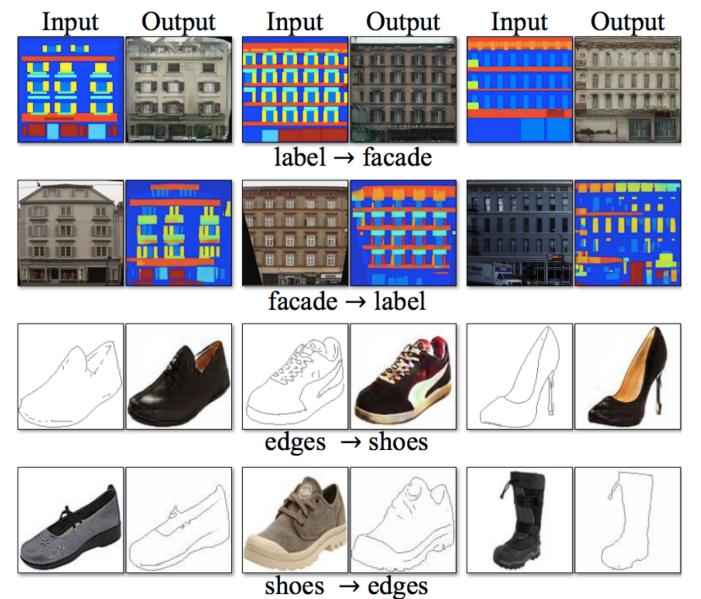
• Illustration of cycle consistency:



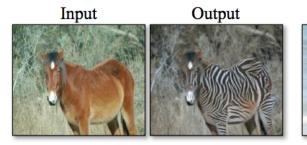
• Translation between maps and aerial photos

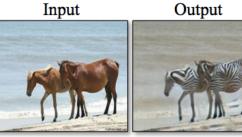


• Other pix2pix tasks



• Tasks for which paired data is unavailable





horse \rightarrow zebra



 $zebra \rightarrow horse$



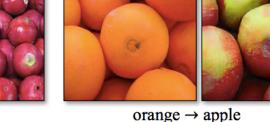
apple \rightarrow orange



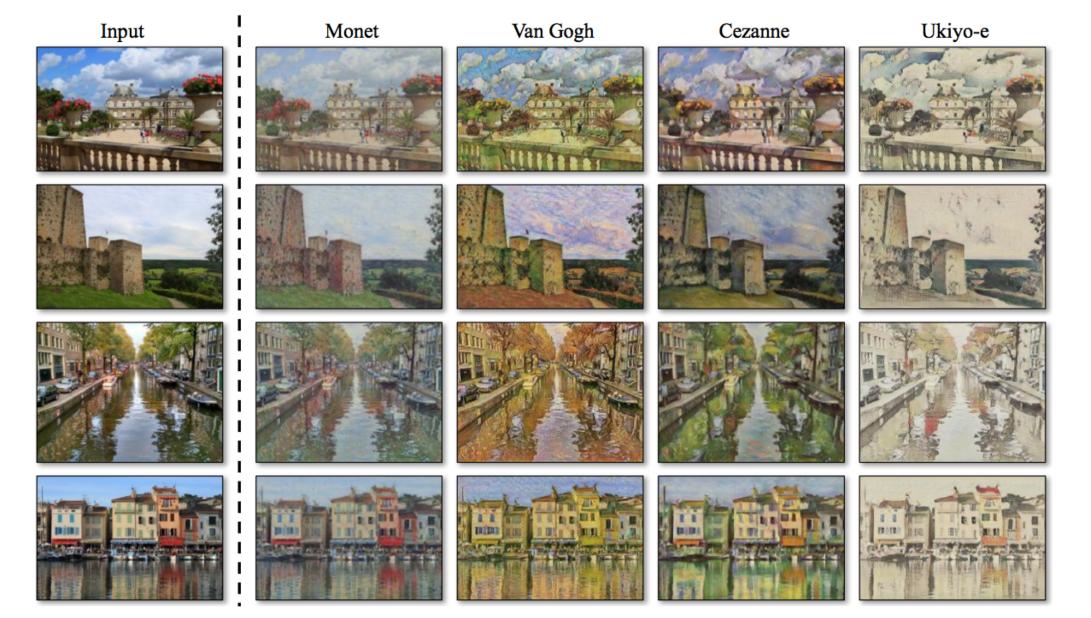
Output

Input





• Style transfer



CycleGAN: Failure cases

Failure cases

