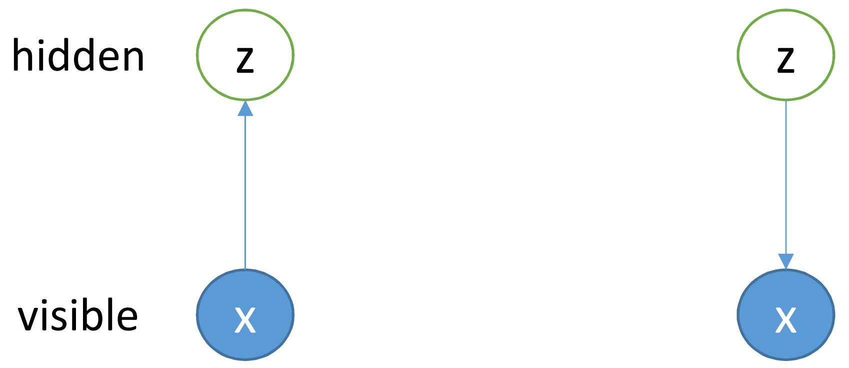
Ch09. Generative Adversarial Networks (GANs)

Hwanjo Yu

POSTECH

http://hwanjoyu.org

Why Deep Generative Models?



Discrinimative model

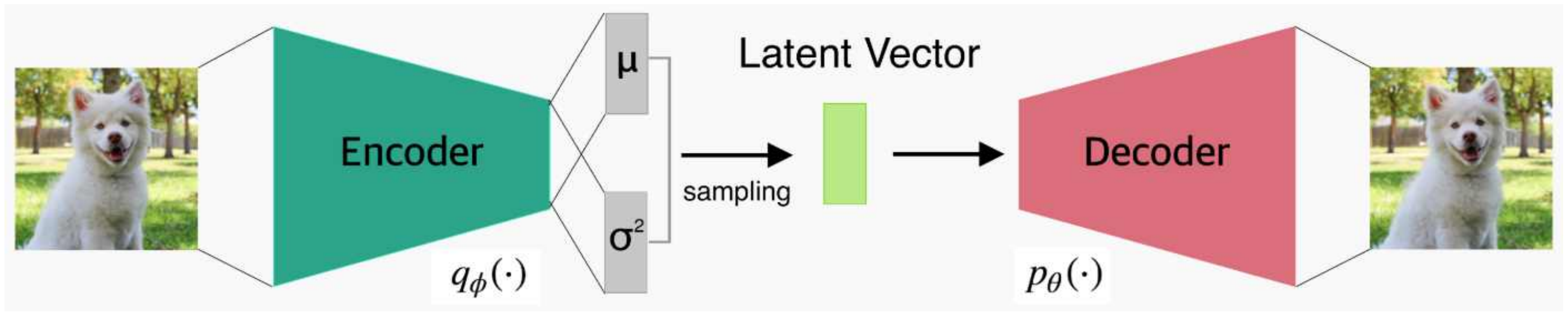
(supervised)

Generative model

(unsupervised)

Variational Autoencoders (VAEs)

[D P Kingma and M Welling (2014), ”Auto-encoding variational Bayes,” NIPS.]

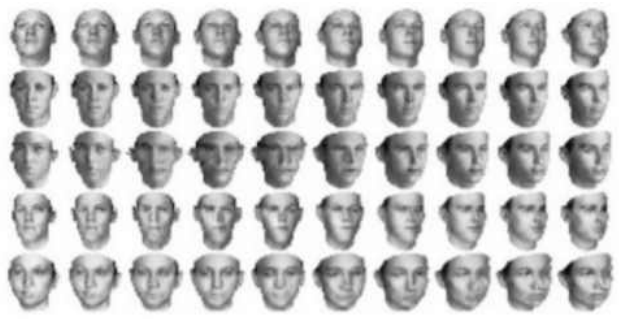


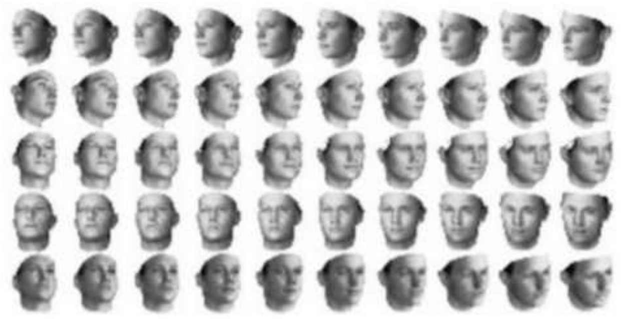
- Kind of probabilistic autoencoder: Compared to vanilla autoencoder, VAE could generate diverse data by sampling from a distribution

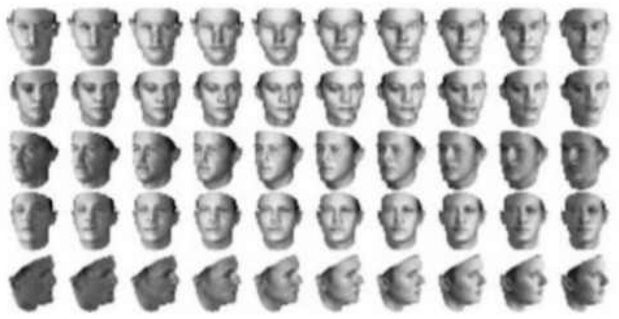
GAN example

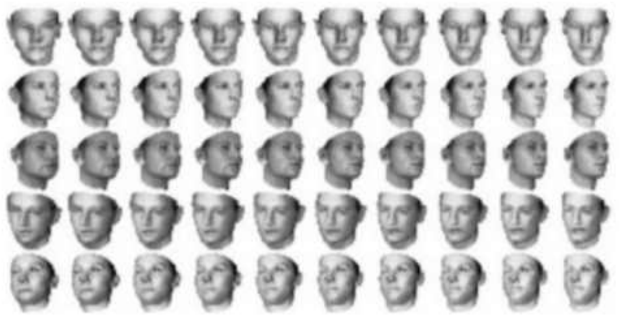
[X. Chen et al. (2016), ”InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets,” NIPS.]

<u>Learning disentangled & interpretable representations!</u>


(a) Azimuth (pose)


(b) Elevation

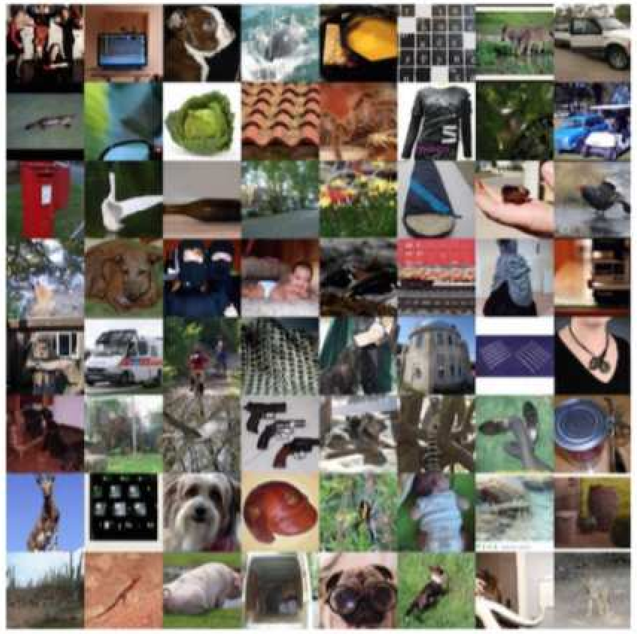

(c) Lighting


(d) Wide or Narrow

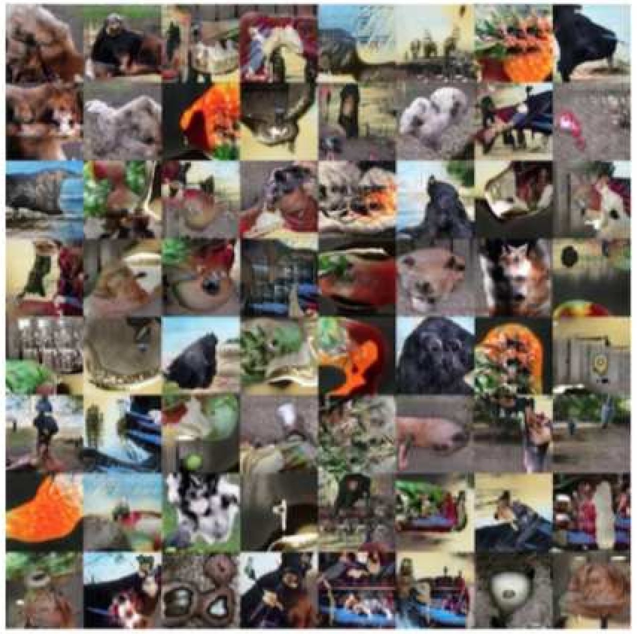
GAN example

[T. Salimans et al. (2016), ”Improved Techniques for Training GANs,” NIPS.]

- Learning to generate new data, whose distribution follows the distribution over training examples



Real images (ImageNet)



Generated images

- https://www.youtube.com/watch?v=XOxxPcy5Gr4

Adversarial Training

Dueling neural networks: 10 breakthrough technologies in MIT Technology Review 2018

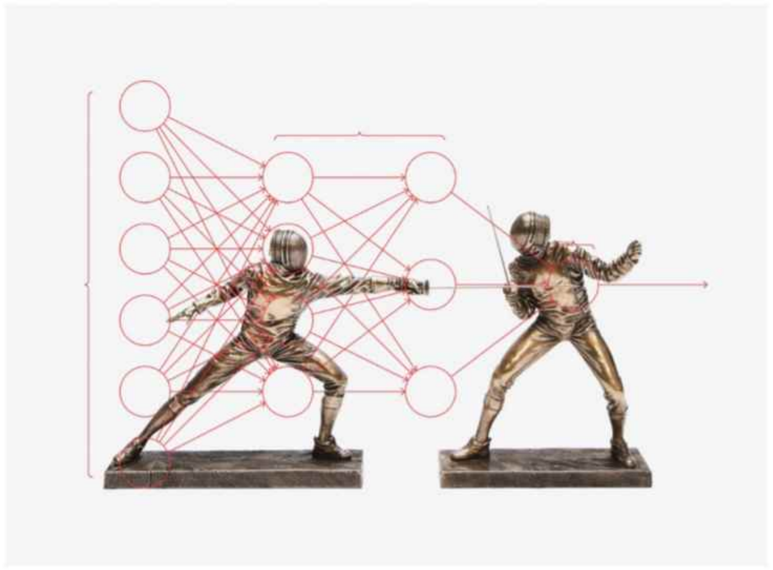


ILLUSTRATION BY DEREK BRAHNEY | DIAGRAM COURTESY OF MICHAEL NIELSEN, "NEURAL NETWORKS AND DEEP LEARNING", DETERMINATION PRESS, 2015

- A set of machines learn together by pursuing competing goals
- A fascinating new training method
- Bypasses the need of loss functions in learning
- A new way of regularizing learning machines

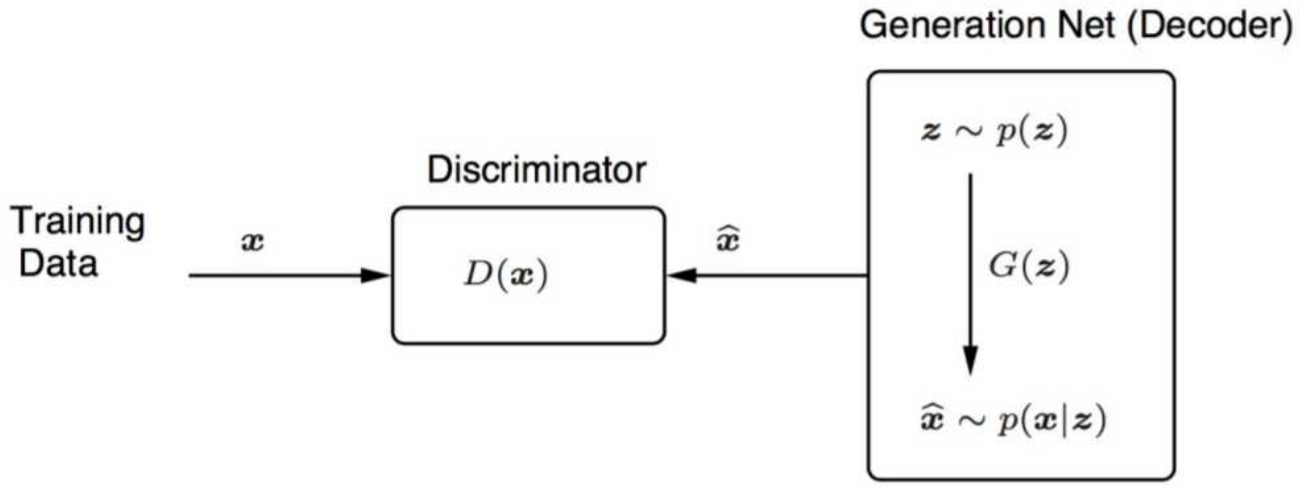
Generative Adversarial Network [Goodfellow et al., 2014]

Generator, $G(\mathbf{z};\theta): \mathbb{R}^K \to \mathbb{R}^D$

- Captures the data distribution
- <u>Counterfeiters:</u> Tries to fake discriminator

Discriminator, $D(\mathbf{x};\phi): \mathbb{R}^D \to \{0,1\}$

- Scoring function: $D(\mathbf{x};\phi) = \mathbb{P}[y = 1|\mathbf{x}]$ ($y = 1$: training data)
- Learns features with rich semantics
- <u>Police:</u> Tries to detect counterfeit images



How to train the generative model?

What we have is:

- Samples drawn from the model $p_\theta(x)$ (not known), via the generator.
- Samples from the data distribution $p_d(x)$.

What we want:

- Learn a generator $G(\cdot)$ such that the distribution of its generated samples well match the data distribution.

Training GAN

- Denote by $p_d(\mathbf{x})$ the true data distribution and by $p_g(\mathbf{x})$ our model distribution (generator's distribution over $\mathbf{x}$).

Training D

- Train the discriminator $D(\mathbf{x};\phi)$ to maximize the probability of assigning the correct label to training data as well as fake data generated by G (θ is fixed.)

$$\max_{\phi}\left[\mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[\log D(\mathbf{x};\phi)] + \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}\left[\log\bigl(1 - D(\mathbf{x};\phi)\bigr)\right]\right]$$
$$= \max_{\phi}\left[\mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[\log D(\mathbf{x};\phi)] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\bigl(1 - D(G(\mathbf{z};\theta);\phi)\bigr)\right]\right].$$

Training G

- Train the generator G to minimize the probability of the negative (generated-data) class, $\log\bigl(1 - D(G(\mathbf{z};\theta);\phi)\bigr)$ (ϕ is fixed.)

$$\min_{\theta}\left[\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\bigl(1 - D(G(\mathbf{z};\theta);\phi)\bigr)\right]\right].$$

- Two-player minimax game:

$$\min_{\theta} \max_{\phi} \mathcal{J}(\theta, \phi)$$

- where

$$\mathcal{J}(\theta, \phi) = \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})}[\log D(\mathbf{x}; \phi)] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}\big[\log\big(1 - D(G(\mathbf{z}; \theta); \phi)\big)\big]$$

- Both G and D are deep neural networks.
- Does not require any sophisticated inference methods (variational or sampling).
- Alternate between optimizing D (cross-entropy loss minimization) and optimizing G
  - $\max_{\phi} \mathbb{E}_{\mathbf{x} \sim q(\mathbf{x})}[\log D(\mathbf{x}; \phi)] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}\big[\log\big(1 - D(G(\mathbf{z}; \theta); \phi)\big)\big]$
  - $\min_{\theta} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}\big[\log\big(1 - D(G(\mathbf{z}; \theta); \phi)\big)\big]$
- In practice, train G: $\max_{\theta} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\log D(G(\mathbf{z}; \theta); \phi)]$. (stronger gradients early in learning)

Optimality in GAN

- The generator implicitly defines a probability distribution p_g as the distribution of samples $G(\mathbf{z};\theta)$ obtained when $\boldsymbol{z}\sim p(\mathbf{z})$.
- It was shown that the optimal discriminator has the shape:

$$D^*(\mathbf{x}) = \frac{p_d(\mathbf{x})}{p_d(\mathbf{x}) + p_g(\mathbf{x})}.$$

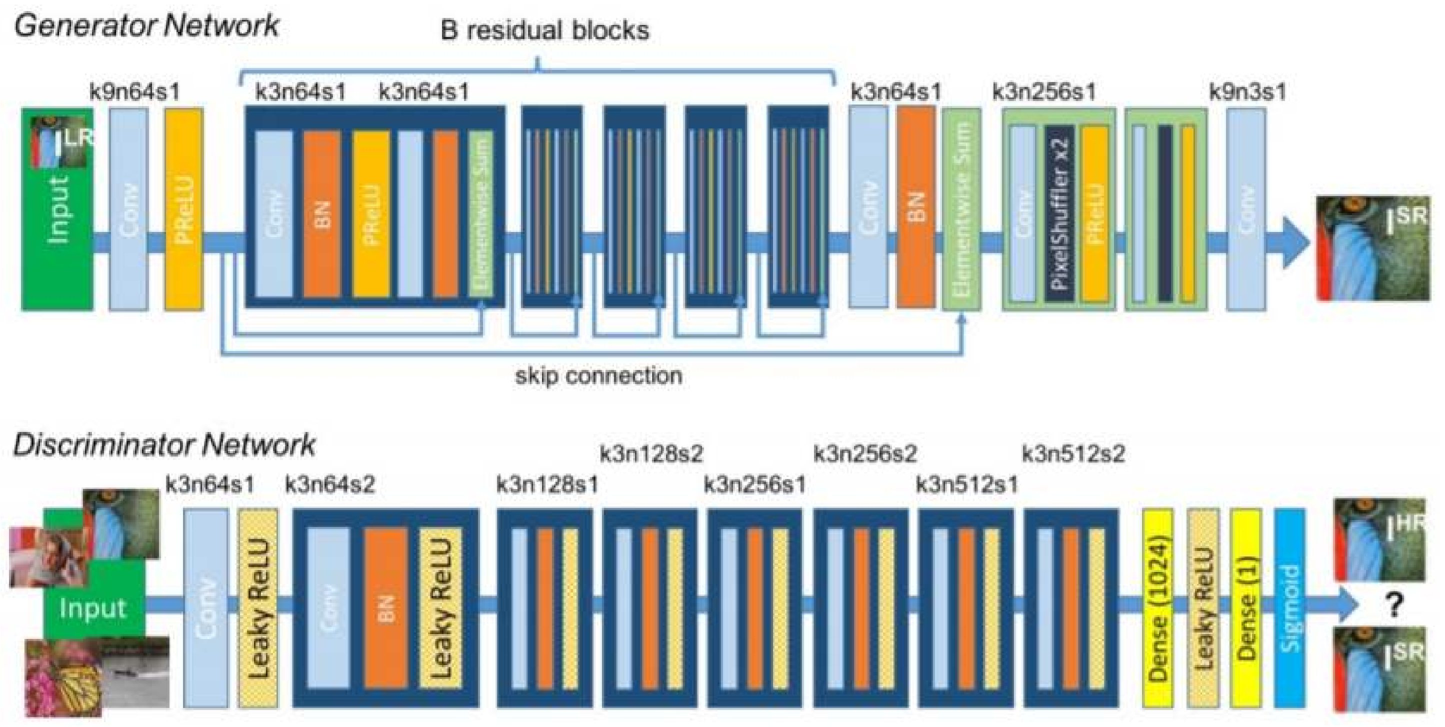
- When the discriminator is optimal, $\min_G \mathbb{E}_{\boldsymbol{z}\sim p(\mathbf{z})}\left[\log\left(1 - D\big(G(\mathbf{z};\theta)\big)\right)\right]$ is equivalent to minimizing the Jensen-Shannon divergence between p_d and p_g:

$$\mathrm{JSD}\big[p_d||p_g\big] = D_{KL}\left[p_d||\frac{p_d+p_g}{2}\right] + D_{KL}\left[p_g||\frac{p_d+p_g}{2}\right]$$

GAN example: Image Super-Resolution

Christian Ledig et al. (2016), ”Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network,” Preprint arXiv:1609.04802

- Goal: Estimate a high-resolution, superresolved image from a low-resolution input image.



GAN example: Image Super-Resolution

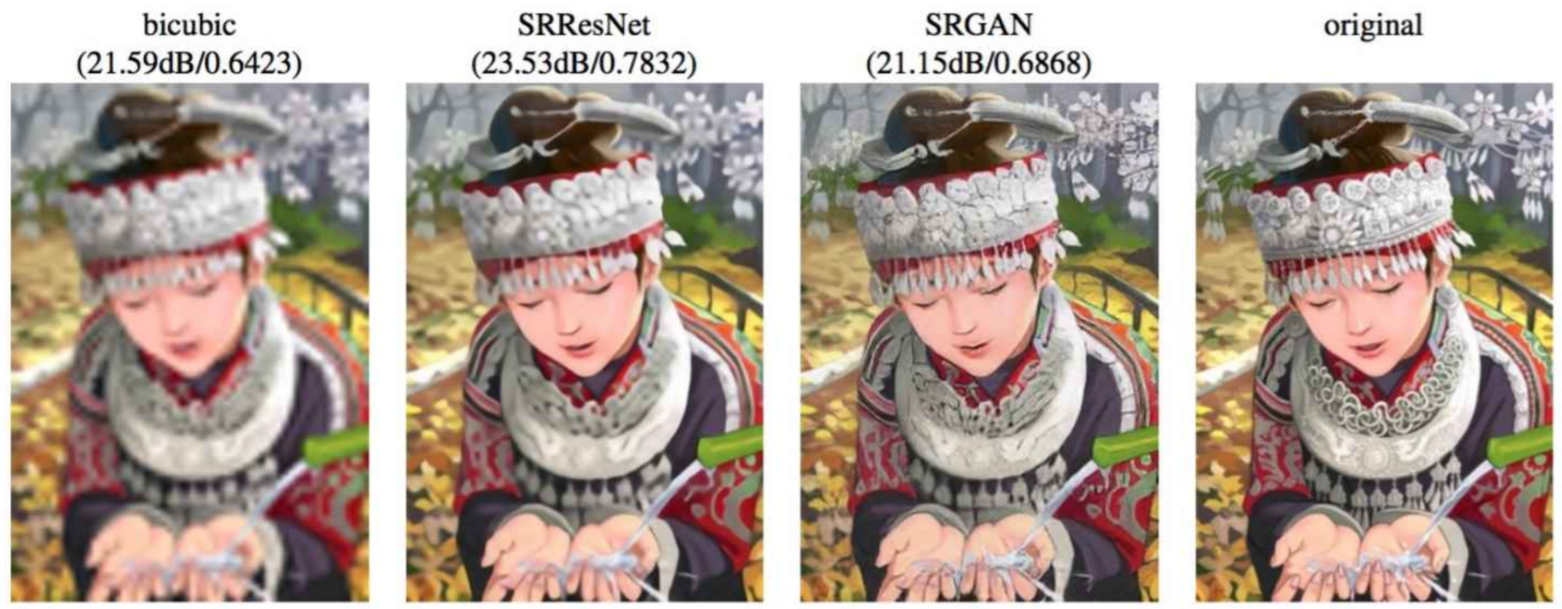
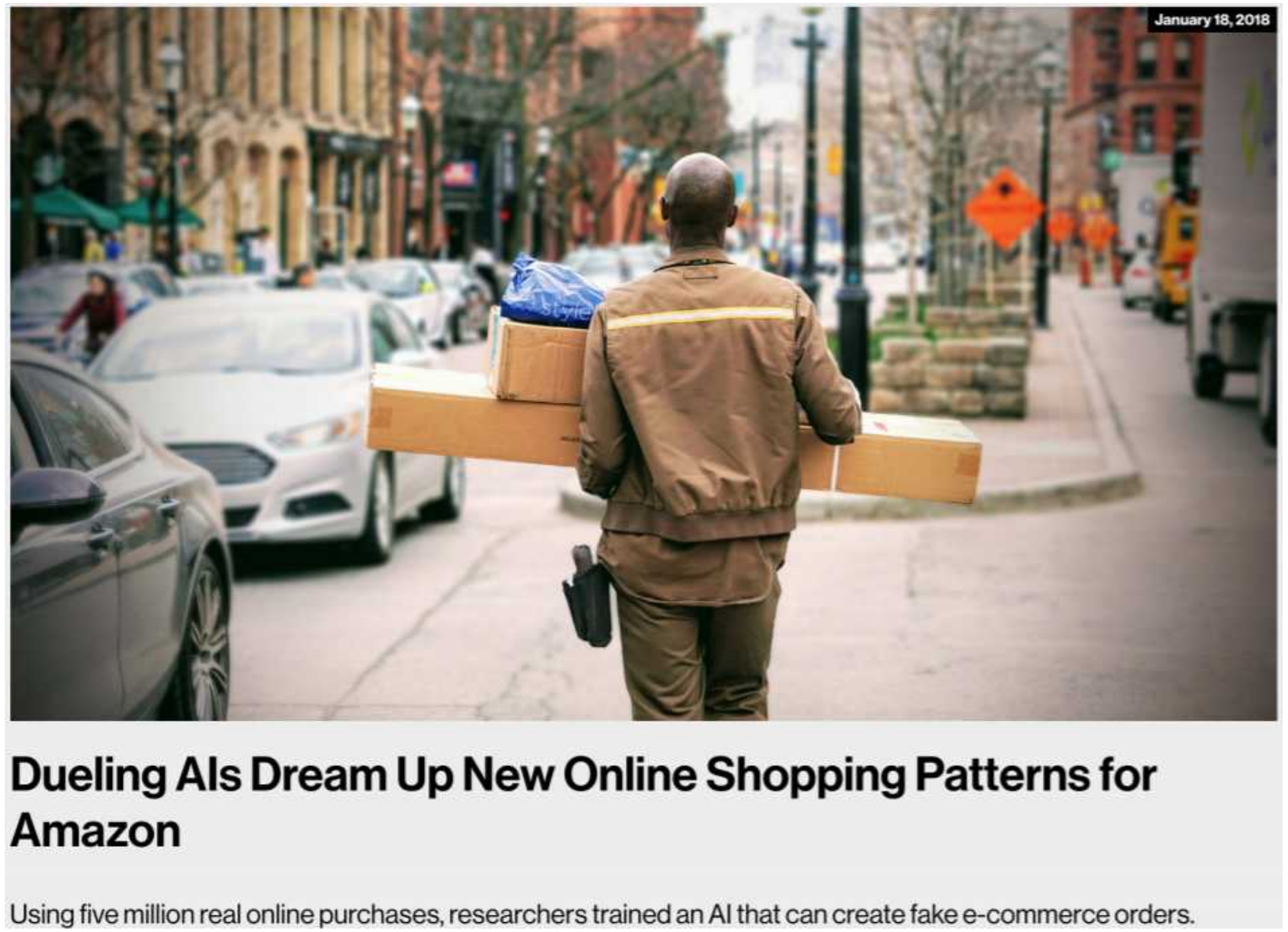


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

GAN example: eCommerceGAN

A. Kumar, A. Biswas, and S. Sanyal (2018), ”eCommerceGAN: A Generative Adversarial Network for E-commerce,” Preprint arXiv:1801.03244.



January 18, 2018

Dueling AIs Dream Up New Online Shopping Patterns for Amazon

Using five million real online purchases, researchers trained an AI that can create fake e-commerce orders.

Feature Matching to Train G

T. Salimans et al. (2016), ”Improved techniques for training GANs,” NIPS

- Training GANs = Finding a Nash equilibrium of a non-convex game with continuous and high-dimensional parameters.
- Note that the modification of parameters in D increase

$$\mathcal{J}(\phi) = \mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[\log D(\mathbf{x};\phi)] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(1 - D\big(G(\mathbf{z};\theta)\big)\right)\right].$$

- But G is modified to decrease

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(1 - D\big(G(\mathbf{z};\theta)\big)\right)\right].$$

- Thus, <u>gradient methods may fail to converge for many games.</u>
- Rather than directly optimizing the out of the discriminator

$$\min_{\theta} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(1 - D\big(G(\mathbf{z};\theta)\big)\right)\right],$$

- train the generator to match the expected value of the features $\Phi(\cdot)$ on an intermediate layer of the discriminator

$$\min_{\theta} \left|\left|\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\Phi(\mathbf{x}) - \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\Phi(G(\mathbf{z};\theta))\right|\right|_2^2.$$

GAN Trained with Denoising Feature Matching

D. Warde-Farley and Y. Bengio (2017), ”Improving adversarial generative networks with denoising feature matching,” ICLR.

Training G = Denoising autoencoder (in the space of discriminator features) + adversarial discriminator

- The discriminator $D = d \circ \Phi$ ($d(\cdot)$: $\mathbb{R}^D \to \{0,1\}$ is a classifier and is a feature extractor) is trained as in the standard GAN:

$$\max_{\phi} \mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[\log D(\mathbf{x};\phi)] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(1 - D(G(\mathbf{z};\theta);\phi)\right)\right].$$

- The generator G is trained:

$$\min_{\theta} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\lambda_{dae}\left|\left|\Phi\left(G(\mathbf{z};\theta)\right) - DAE\left(\Phi\left(G(\mathbf{z};\theta)\right)\right)\right|\right|^2 - \lambda_{gan}\log D\left(G(\mathbf{z};\theta)\right)\right],$$

- where $DAE(\cdot)$ is treated as constant w.r.t. gradient computations, which is trained by

$$\min_{DAE} \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left|\left|\Phi(\mathbf{x}) - DAE\left(\eta(\Phi(\mathbf{x}))\right)\right|\right|^2$$

- $\eta(\cdot)$ is the corruption function.

Unrolled GANs

L. Metz et al. (2017), "Unrolled generative adversarial networks," ICLR.

- Stabilizes training of GANs and solves mode collapsing problem.
- Increases the diversity and coverage of the data distribution by the generator
- Unrolled optimization for updating generator parameters.

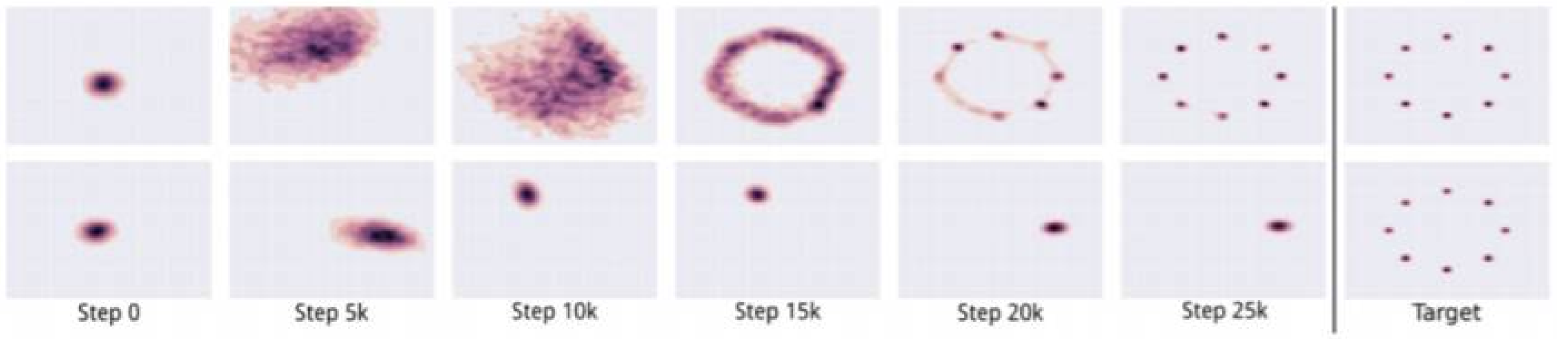


Figure 2: Unrolling the discriminator stabilizes GAN training on a toy 2D mixture of Gaussians dataset. Columns show a heatmap of the generator distribution after increasing numbers of training steps. The final column shows the data distribution. The top row shows training for a GAN with 10 unrolling steps. Its generator quickly spreads out and converges to the target distribution. The bottom row shows standard GAN training. The generator rotates through the modes of the data distribution. It never converges to a fixed distribution, and only ever assigns significant probability mass to a single data mode at once.

Training of GANs: Revisited

- GAN finds the optimal θ^* for a generator by solving the following two-player minimax game:

$$\theta^* = \operatorname*{argmin}_\theta \max_\phi \mathcal{J}(\theta, \phi) = \operatorname*{argmin}_\theta \mathcal{J}(\theta, \phi^*(\theta)),$$

- where

$$\phi^*(\theta) = \operatorname*{argmax}_\theta \mathcal{J}(\theta, \phi),$$

- and the objective function $\mathcal{J}$ is given by

$$\mathcal{J}(\theta, \phi) = \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})}[\log D(\mathbf{x}; \phi)] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}\left[\log\left(1 - D\big(G(\mathbf{z}; \theta)\big)\right)\right]$$

Unrolling GANs

- A local optimum of the discriminator parameter ϕ^* can be expressed as

$$\phi^0 = \phi,$$
$$\phi^{k+1} = \phi^k + \eta^k \frac{\partial \mathcal{J}(\theta, \phi^k)}{\partial \phi^k},$$
$$\phi^*(\theta) = \lim_{k\to\infty} \phi^k .$$

- Unrolling for K steps, a surrogate objective for the update of the generator is given by

$$\mathcal{J}_K(\theta, \phi) = \mathcal{J}\big(\theta, \phi^K(\theta, \phi)\big).$$

- Update generator and discriminator parameters using the surrogate loss:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{J}_K(\theta, \phi)}{\partial \theta},$$
$$\phi \leftarrow \phi + \eta \frac{\partial \mathcal{J}(\theta, \phi)}{\partial \phi}.$$

Semi-Supervised Learning with GANs

T. Salimans et al. (2016), ”Improved techniques for training GANs,” NIPS.

- Classifier for K classes: $p_{model}(y = k|\mathbf{x}) = \frac{\exp\{\ell_k\}}{\sum_{j=1}^{K} \exp\{\ell_j\}}$, where ℓ_k are logits.
- GAN: Label generated samples with $y = K + 1$, i.e., $p_{model}(y = K + 1|\mathbf{x})$ yields the probability that $\mathbf{x}$ is fake.
- Loss: $\mathcal{L} = \mathcal{L}_{supervised} + \mathcal{L}_{unsupervised}$,

$$\mathcal{L}_{supervised} = -\mathbb{E}_{(\mathbf{x},y)\sim p_{data}(\mathbf{x},y)} \log p_{model}(y|\mathbf{x}, y \le K),$$

$$\mathcal{L}_{unsupervised} = -\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})} \log\big(1 - p_{model}(y = K + 1|\mathbf{x})\big)$$

$$= -\mathbb{E}_{\mathbf{x}\sim G} \log p_{model}(y = K + 1|\mathbf{x}).$$

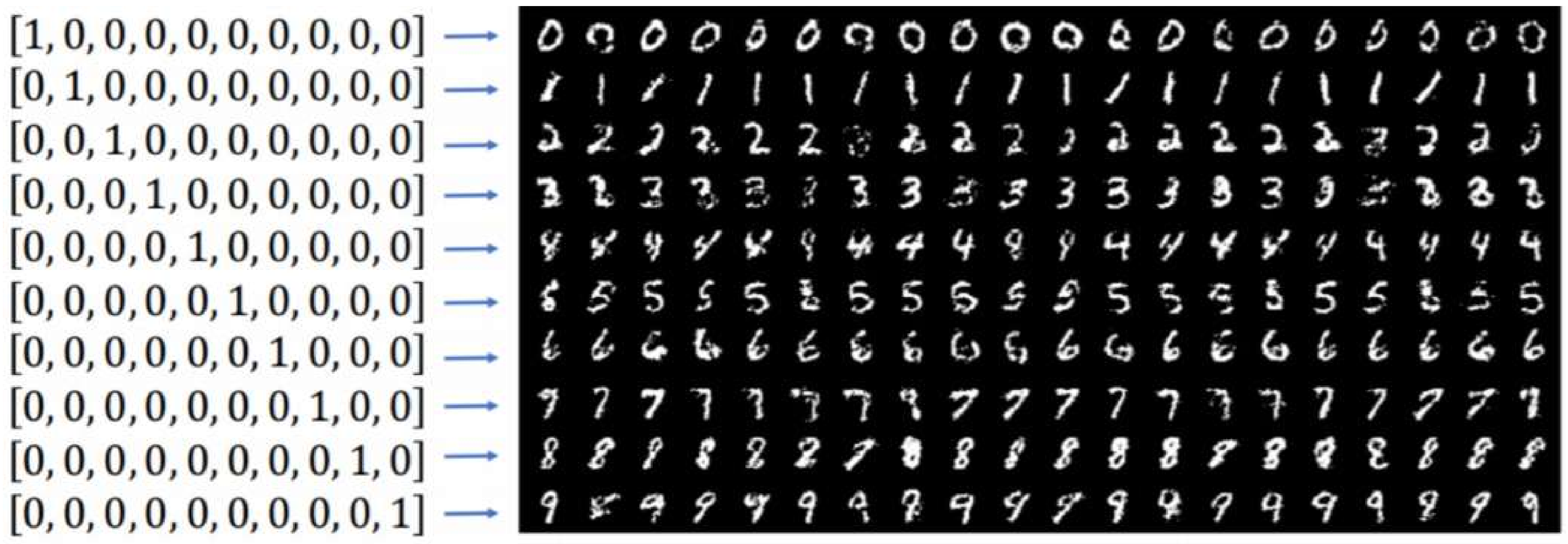
- This $\mathcal{L}_{unsupervised}$ is the case where in the standard GAN game value, we use $D(\mathbf{x}) = 1 - p_{model}(y = K + 1|\mathbf{x})$, yielding

$$\mathcal{L}_{unsupervised} = -\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})} \log D(\mathbf{x}) - \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \log\Big(1 - D\big(G(\mathbf{z})\big)\Big).$$

Conditional GANs

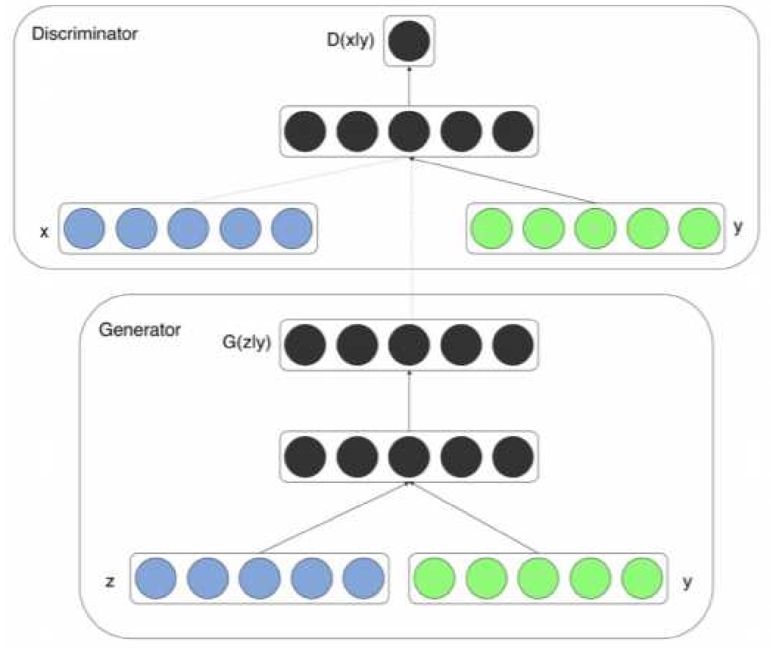
Mehdi Mirza and Simon Osindero (2014), ”Conditional generative adversarial nets,” Preprint arXiv:1411.1784.

- Generator is trained to **generate a fake sample x with a condition y** (e.g., class label or data from other modalities) provided as another input, in addition to noise **z**.



- Generator: Input noise **z** and **y** are combined in joint hidden representation.
- Discriminator: **x** and **y** are presented as inputs to the discriminator.
- Optimization:

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



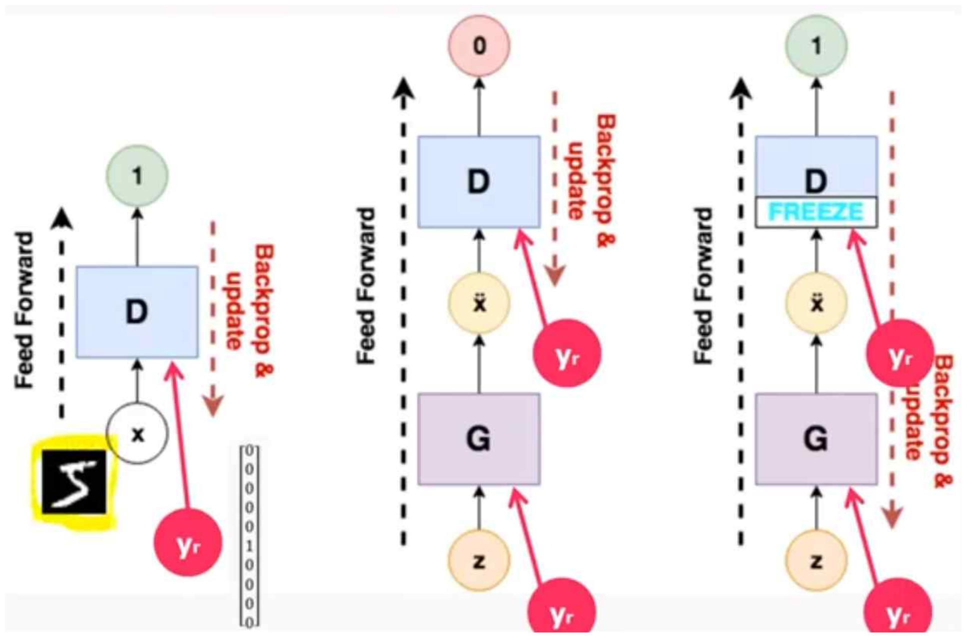
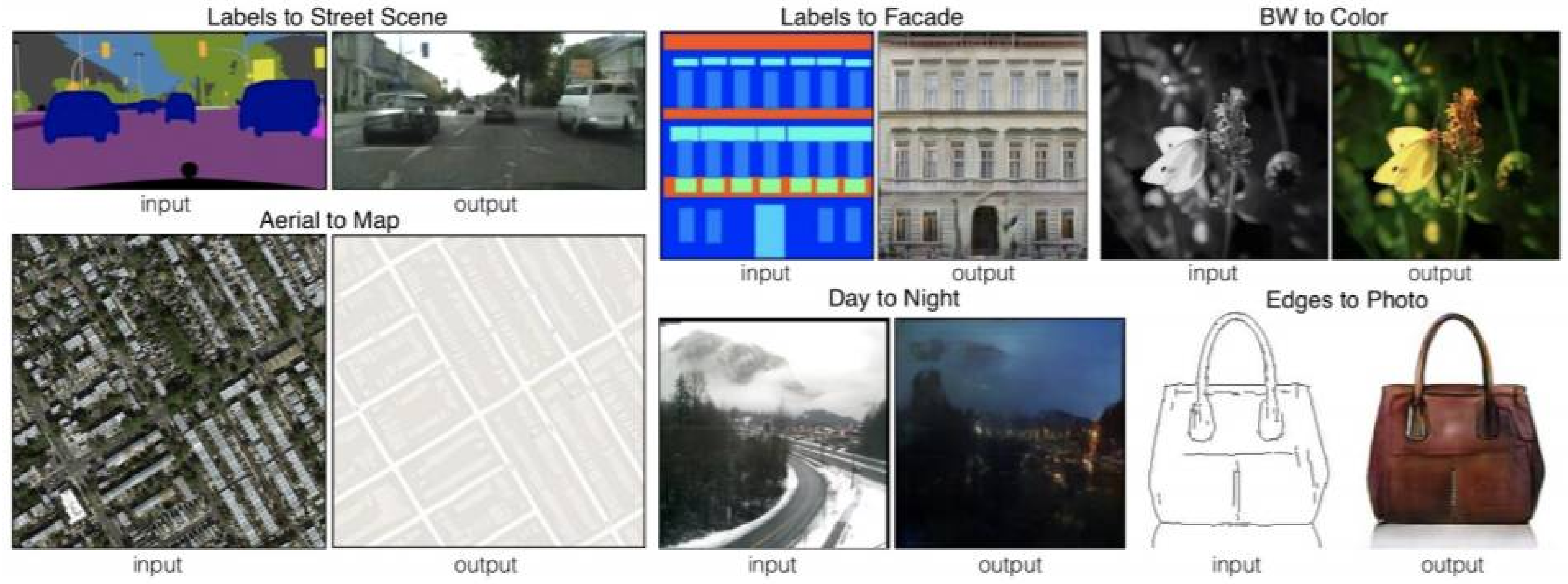


Image to Image Translation: Examples

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros (2017), ”Image-to-Image Translation with Conditional Adversarial Networks,” CVPR.



Map Edges to Photo via cGAN

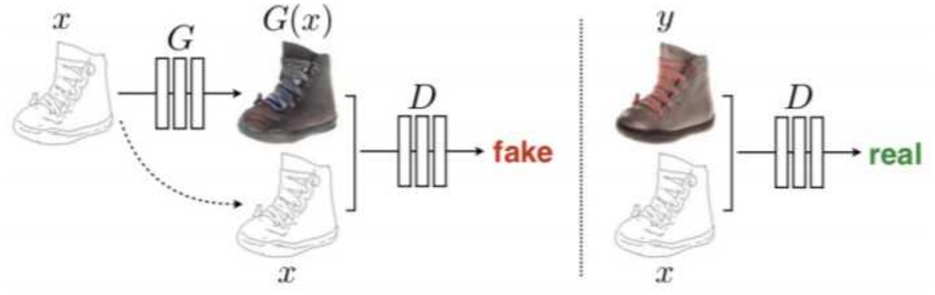


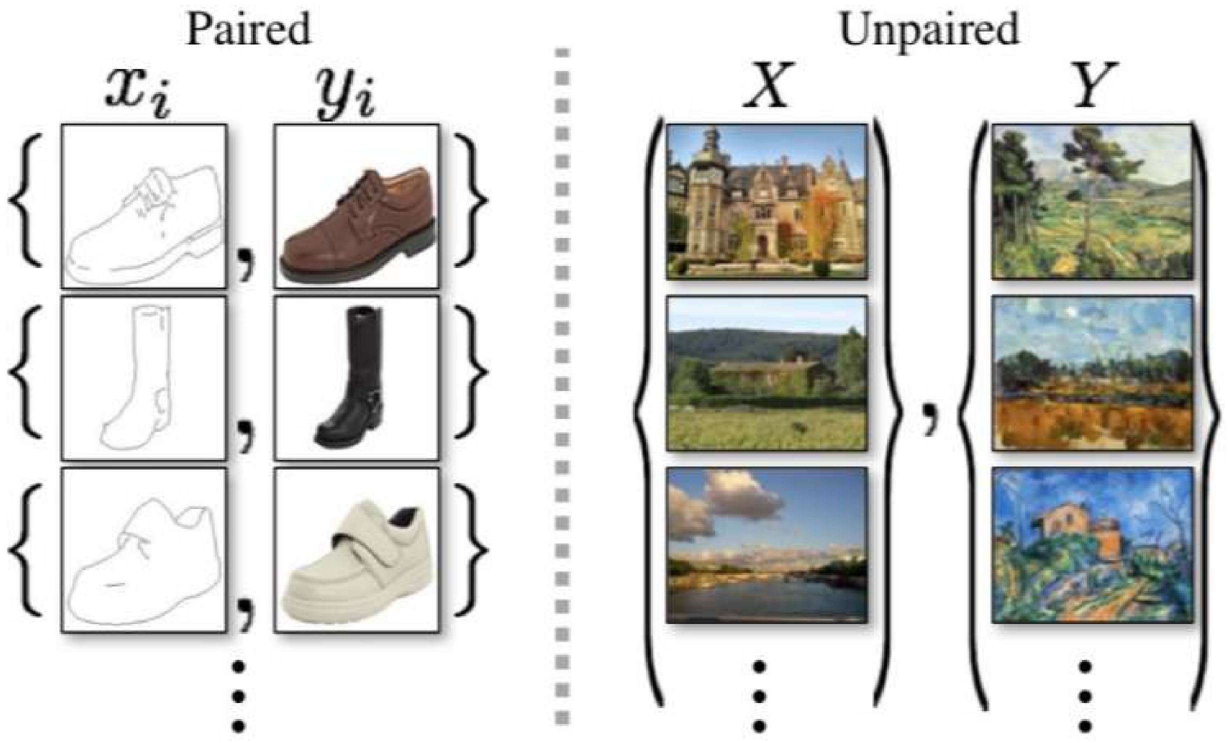
Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

- cGAN objective function:
  $\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x}),\boldsymbol{y}\sim p(\mathbf{y})} \log D(\mathbf{x},\mathbf{y}) + \mathbb{E}_{\boldsymbol{z}\sim p(\mathbf{z}),\boldsymbol{y}\sim p(\mathbf{y})} \log\big(1 - D(G(\mathbf{z},\mathbf{y}),\mathbf{y})\big)$
- L_1 loss function for G:
  $$\mathcal{L}_{L_1} = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x}),\boldsymbol{y}\sim p(\mathbf{y}),\boldsymbol{z}\sim p(\mathbf{z})} \big|\big|\mathbf{x} - G(\mathbf{z},\mathbf{y})\big|\big|_1$$
- Optimize the mix of these two
  $$\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L_1}.$$
- U-net (for generator) and PatchGAN (for discriminator) were used.

Unpaired Image to Image Translation: Examples

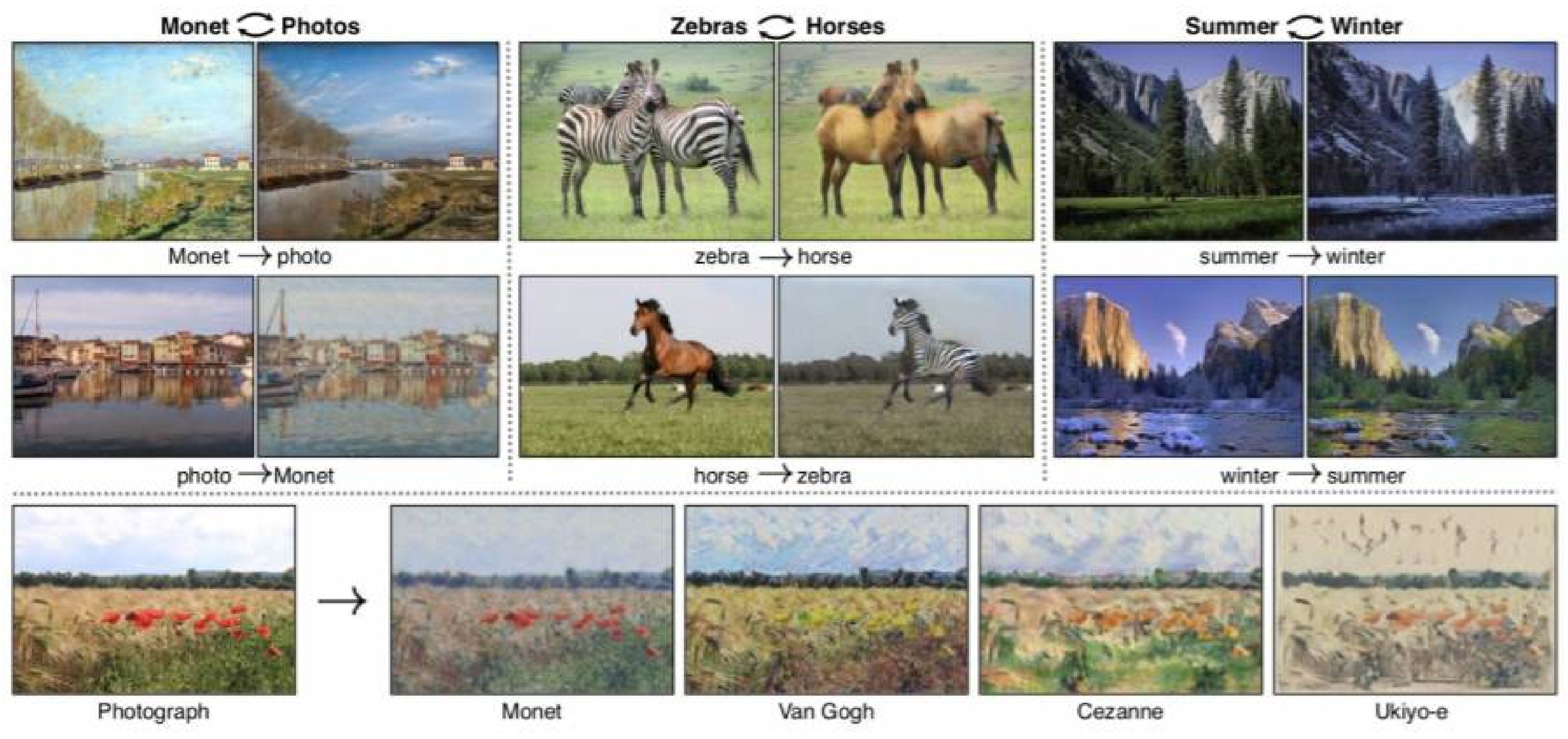
Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros (2017), ”Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks,” ICCV.

- Given any two unordered image collections, it learns to automatically translate an image from one to other and vice versa.



Unpaired Image to Image Translation: Examples

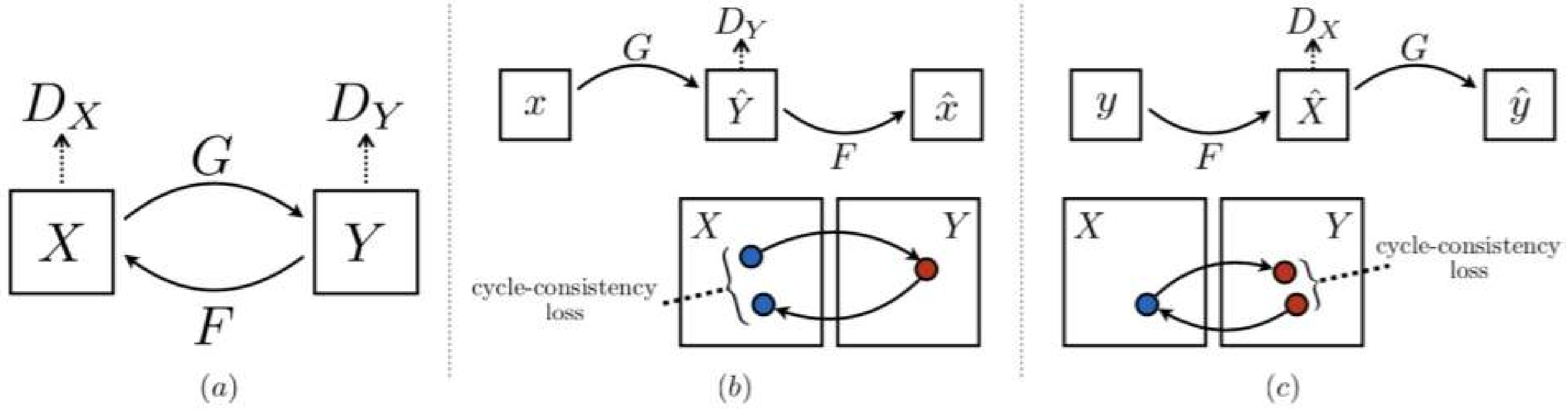
- Given any two unordered image collections, it learns to automatically translate an image from one to other and vice versa.



Unpaired Image to Image Translation: Cycle-Consistency

- Translation should be **cycle consistent**.

$$G: \mathcal{X} \to \mathcal{Y},$$
$$F: \mathcal{Y} \to \mathcal{X},$$
$$F\big(G(\mathbf{x})\big) \approx \mathbf{x} \quad \text{and} \quad F\big(G(\mathbf{y})\big) \approx \mathbf{y}.$$



Adversarial Loss + Cycle Consistency Loss

- **Adversarial loss** for $G: \mathcal{X} \to \mathcal{Y}$ and $F: \mathcal{Y} \to \mathcal{X}$

$$\mathcal{L}_{gan}(G, D_Y, \mathcal{X}, \mathcal{Y}) = \mathbb{E}_{\mathbf{y}\sim p_{data}(\mathbf{y})} \log D_Y(\mathbf{y}) + \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\mathbf{x})} \log\left(1 - D_Y\left(G(\mathbf{x})\right)\right),$$
$$\mathcal{L}_{gan}(F, D_X, \mathcal{Y}, \mathcal{X}) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})} \log D_X(\mathbf{x}) + \mathbb{E}_{\boldsymbol{y}\sim p_{data}(\mathbf{y})} \log\left(1 - D_X\left(F(\mathbf{y})\right)\right).$$

- **Cycle consistency loss**

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})} \left|\left|F\left(G(\mathbf{x})\right) - \mathbf{x}\right|\right|_1 + \mathbb{E}_{\boldsymbol{y}\sim p_{data}(\mathbf{y})} \left|\left|G\left(F(\mathbf{y})\right) - \mathbf{y}\right|\right|_1.$$

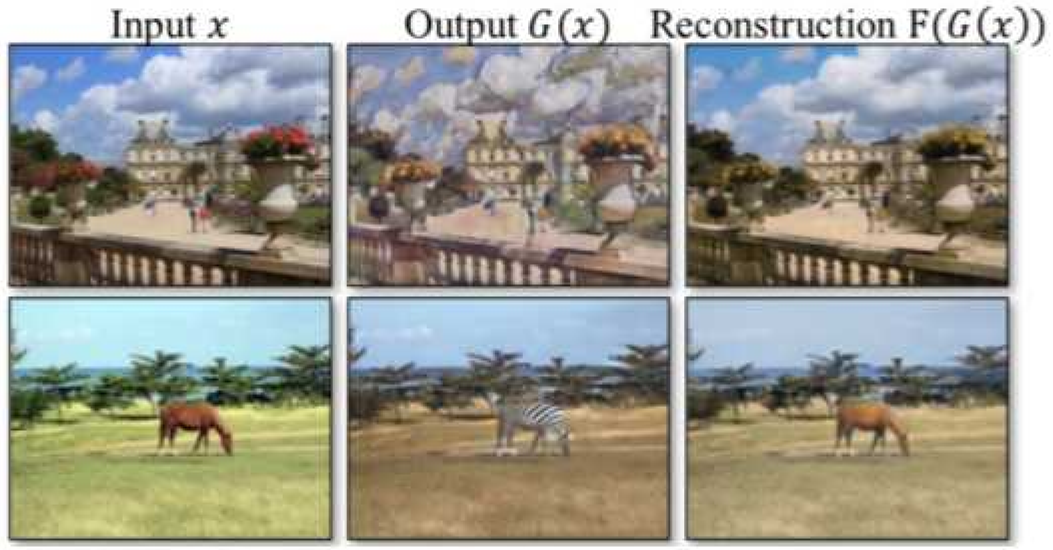


Figure: photo ↔ Cezanne, horses ↔ zebras.

Unpaired Image to Image Translation: Training

- The objective function is given by

$$\mathcal{L}(G, F, D_X, D_Y)$$
$$= \underbrace{\mathcal{L}_{gan}(G, D_Y, \mathcal{X}, \mathcal{Y}) + \mathcal{L}_{gan}(F, D_Y, \mathcal{Y}, \mathcal{X})}_{\text{adversarial loss}} + \lambda \underbrace{\mathcal{L}_{cyc}(G, F)}_{\text{cycle consistency loss}} .$$

- Determine two mapping function G and F by solving the following optimization

$$G^*, F^* = arg \min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) .$$

Experiments:

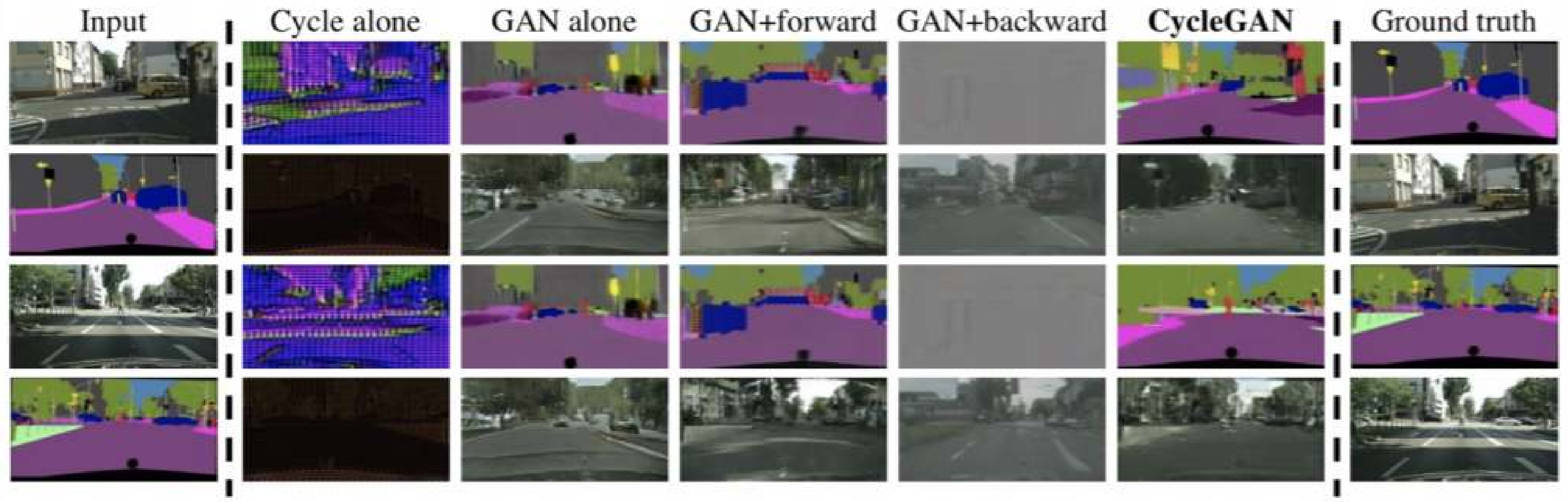


Figure 7: Different variants of our method for mapping labels↔photos trained on cityscapes. From left to right: input, cycle-consistency loss alone, adversarial loss alone, GAN + forward cycle-consistency loss ($F(G(x)) \approx x$), GAN + backward cycle-consistency loss ($G(F(y)) \approx y$), CycleGAN (our full method), and ground truth. Both *Cycle alone* and *GAN + backward* fail to produce images similar to the target domain. *GAN alone* and *GAN + forward* suffer from mode collapse, producing identical label maps regardless of the input photo.

Experiments:

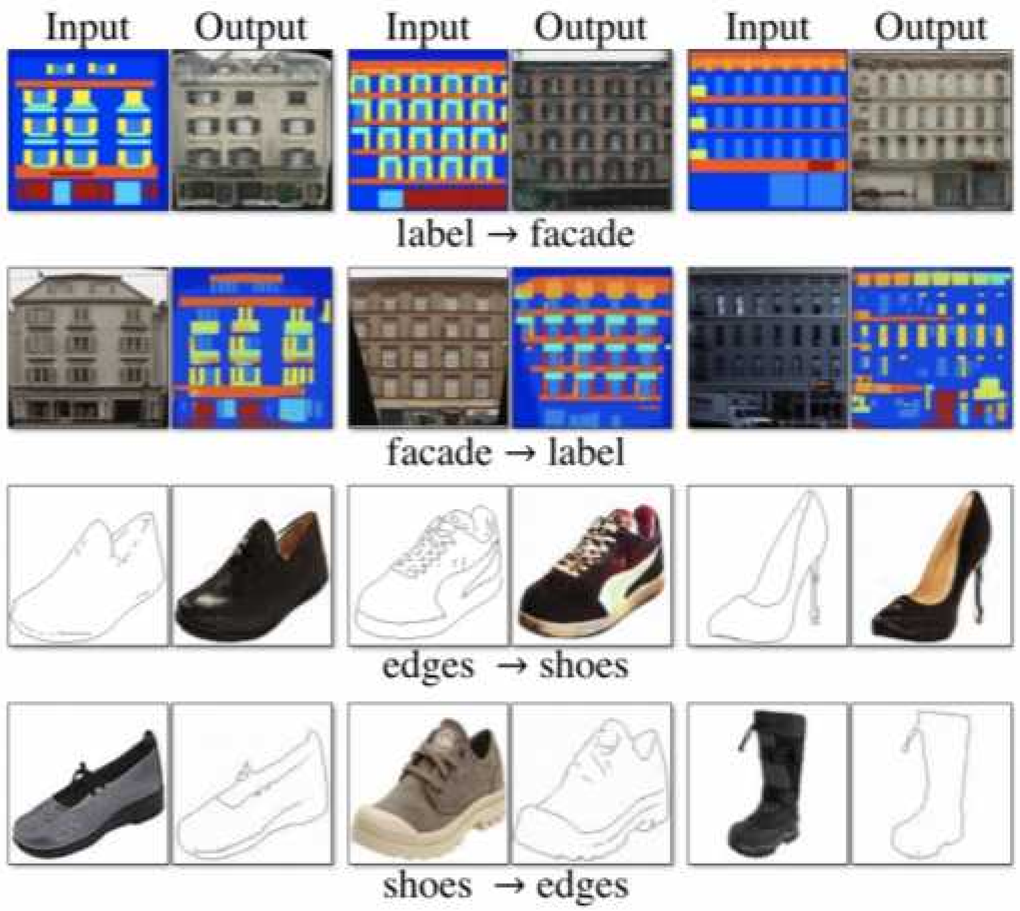
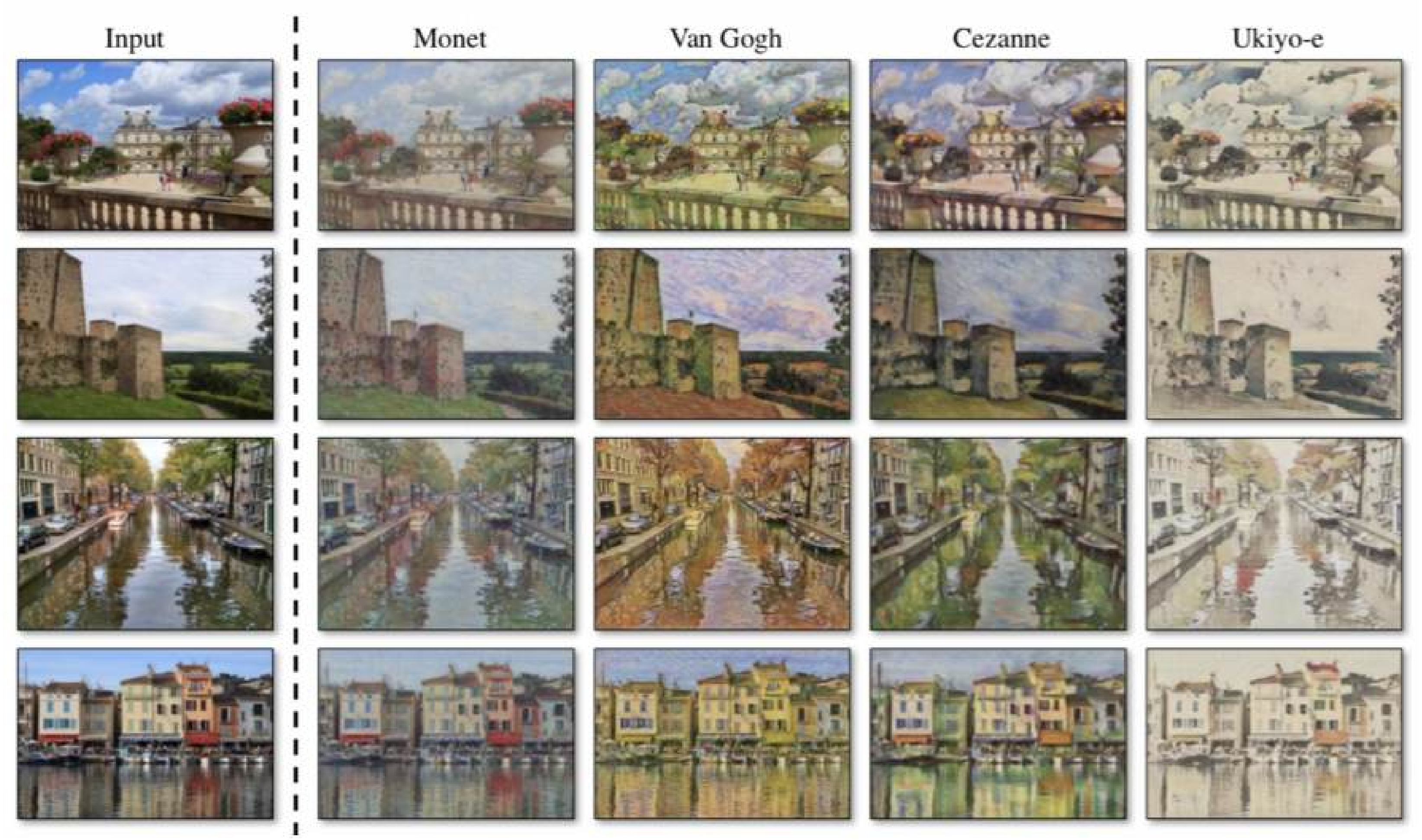


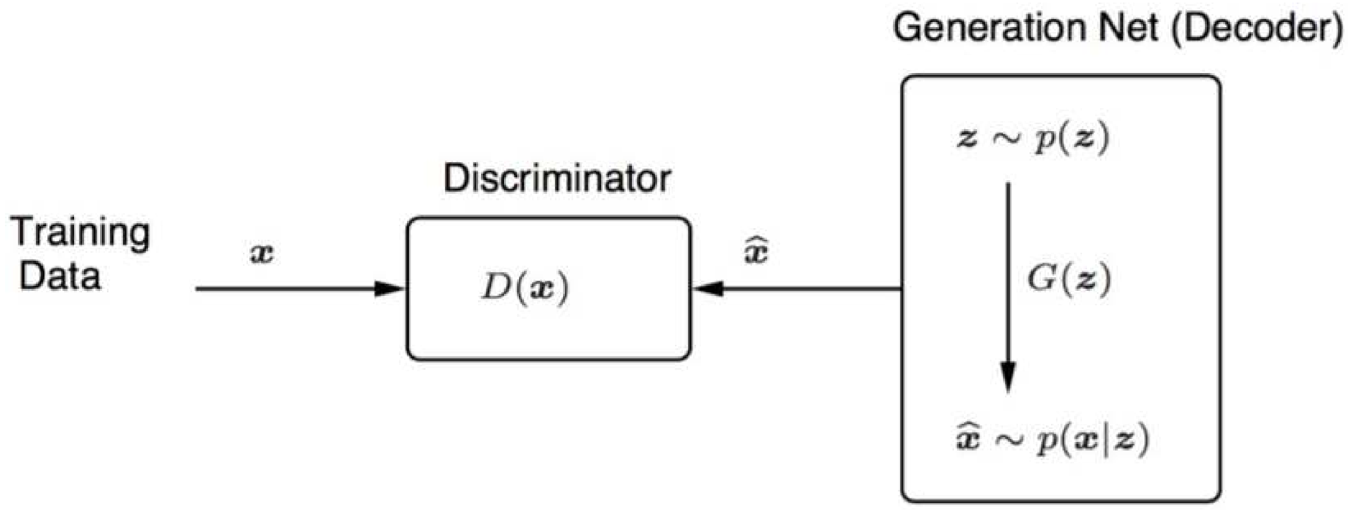
Figure 8: Example results of CycleGAN on paired datasets used in "pix2pix" [22] such as architectural labels↔photos and edges↔shoes.

Experiments: Style Transfer

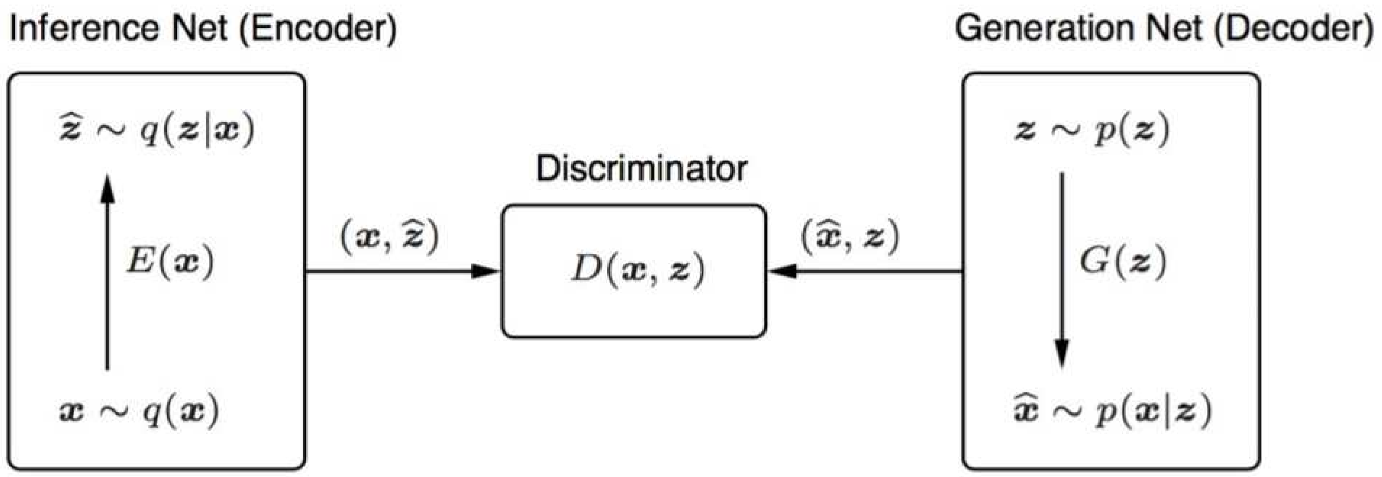


GAN with Encoder Networks: GAN Revisited

ALI: Vincent Dumoulin et al. (2017), ”Adversarially learned inference,” ICLR.
BiGAN: J. Donahue, P. Kr¨ahenb¨uhl, and T. Darrell (2017), ”Adversarial feature learning,” ICLR.



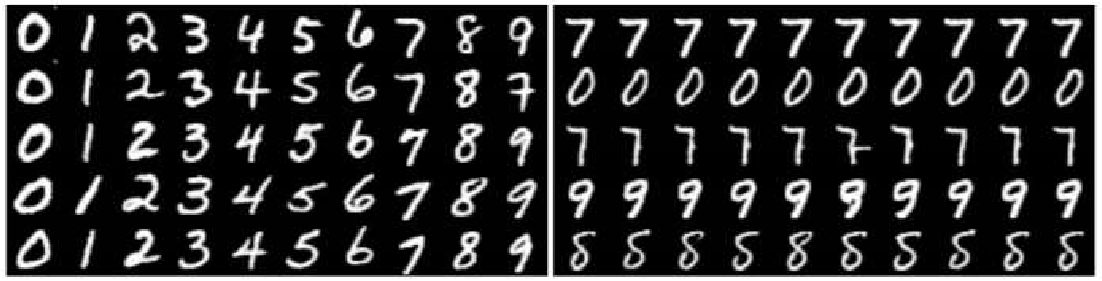
Adversarially Learned Inference



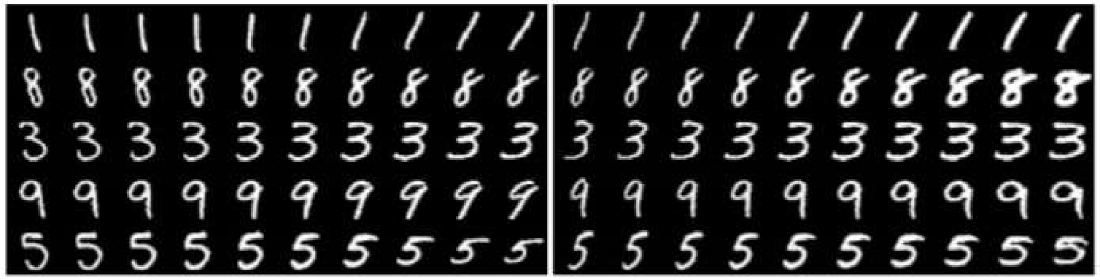
- Encoder joint distribution: $q(\mathbf{x}, \mathbf{z}) = q(\mathbf{x})q(\mathbf{z}|\mathbf{x})$
- Decoder joint distribution: $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})q(\mathbf{x}|\mathbf{z})$
- Match these two joint distributions
- The minimax game:

$$\min_G \max_D \mathbb{E}_{q(\mathbf{x})}[\log D(\mathbf{x}, E(\mathbf{x}))] + \mathbb{E}_{p(\mathbf{z})}\big[\log\big(1 - D(G(\mathbf{z}), \mathbf{z})\big)\big].$$

InfoGAN



(a) Varying c_1 on InfoGAN (Digit type) (b) Varying c_1 on regular GAN (No clear meaning)



(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation) (d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Figure 2: **Manipulating latent codes on MNIST:** *In all figures of latent code manipulation, we will use the convention that in each one latent code varies from left to right while the other latent codes and noise are fixed. The different rows correspond to different random samples of fixed latent codes and noise. For instance, in (a), one column contains five samples from the same category in c_1, and a row shows the generated images for 10 possible categories in c_1 with other noise fixed.* In (a), each category in c_1 largely corresponds to one digit type; in (b), varying c_1 on a GAN trained without information regularization results in non-interpretable variations; in (c), a small value of c_2 denotes left leaning digit whereas a high value corresponds to right leaning digit; in (d), c_3 smoothly controls the width. We reorder (a) for visualization purpose, as the categorical code is inherently unordered.

$$c_1 \sim \mathrm{Cat}(K = 10, p = 0.1), \quad c_2, c_3 \sim \mathrm{Unif}(-1, 1).$$

[Figure source: X. Chen et. al., 2016]

InfoGAN

X. Chen et al. (2016), ”InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” NIPS.

Decompose the input noise vector into two parts (structured noise vector):

- z: treated as source of incompressible noise;
- c: latent code which will target the salient features of the data distribution

$$p(c_1, c_2, \ldots, c_K) = \prod_{i=1}^{K} p(c_i)$$

InfoGAN: Generator is of the form G(z, c) and involves information-regularized minimax game,

$$\min_{G} \max_{D} \mathcal{V}_{GAN} - \lambda I\,(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$$