Neural Networks and Deep Learning

Generative Models II

Nicholas Dronen

Department of Computer Science
dronen@colorado.edu

March 4, 2019

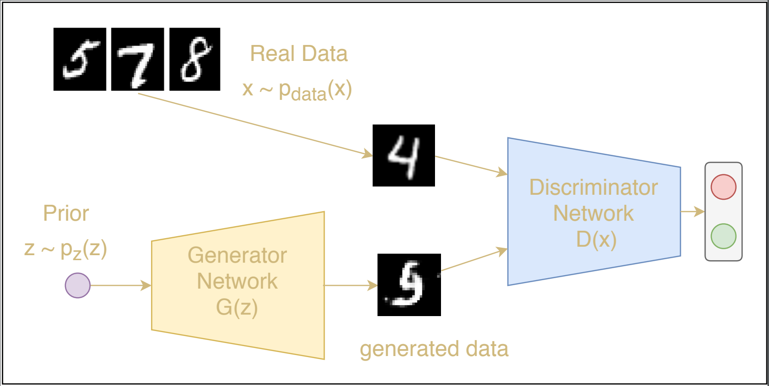
A GAN is a zero-sum game between two adversaries, a generator (G) and a discriminator (D).

G generates samples from a learned distribution p_G and tries to trick D into believing they are from p_{data}, the true data distribution.

D tries not to be deceived.

	Generator	Discriminator
Input	A random vector	A sample from p_G or p_{data}
Output	Sample generated from p_G	Probability that input $\sim p_{data}$
Task	Make p_G close to p_{data}	Distinguish between p_G and p_{data}

G and D are neural networks – typically, though not necessarily, ConvNets.



Generative Adversarial Networks Mark Chang

Within a training iteration, repeat the following k times to optimize the weights of the discriminator D.

Given

- a minibatch of m noise samples $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$, and
- a minibatch of m examples from $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{data}(\mathbf{x})$

update D with gradient *ascent*:

$$\nabla_{d_\theta} \frac{1}{m} \sum_{i=1}^{m} \left[logD\left(\mathbf{x}^{(\mathbf{i})}\right) + log\left(1 - D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right)\right].$$

Within a training iteration, do the following *once* to optimize the weights of the generator G.

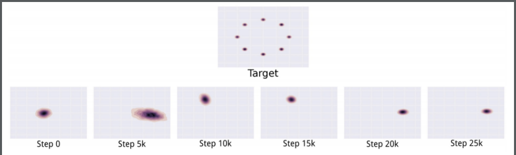
Given a minibatch of m noise samples $\{\, \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(m)} \,\}$ from noise prior $p_g(\mathbf{z})$, update G with gradient *descent*:

$$\nabla_{g_\theta} \frac{1}{m} \sum_{i=1}^{m} log\Big(1 - D\Big(G\Big(\mathbf{z}^{(i)}\Big)\Big)\Big).$$

$$min_G max_D V(D, G) = \mathbb{E}_{x \sim P_{data}}[log D(x)] + \mathbb{E}_{z \sim P_{noise}}[log(1 - D(G(z)))]$$


a)
b)
c)
d)

- Non-convergence - in the zero-sum game played by the generator and discriminator, the equilibrium can be evasive. The progress made by one player may, in turn and repeatedly, be undone by the other player.
- Mode collapse, mode dropping. Real data are multimodal. Mode collapse occurs when the generator settles into a state where it outputs samples from one or a small number of modes. The effect is that the generator creates samples that are far less diverse than those found in the real data.



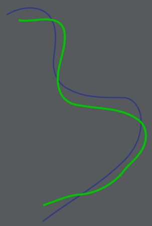
Goodfellow, 2016

It is common to add noise during training of generative models.

Manifolds of p_{data} and p_g

How does noise affect the manifolds?

It is common to add noise during training of generative models.

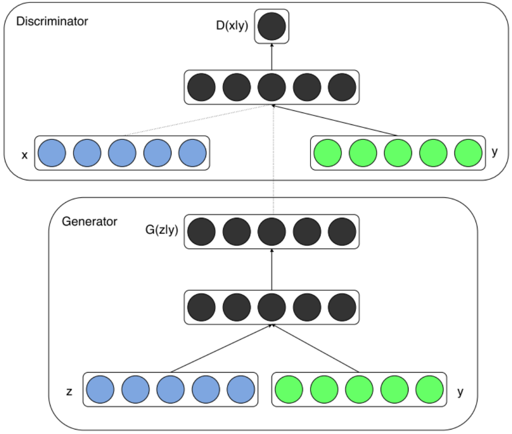


Manifolds of p_{data} and p_g

Manifolds of p_{data} and p_g with noise

- Eliminates lack of common support between p_{data} and p_g.
- Makes D perform worse (initially), so gradients of D are non-zero
- Ensures that KL-divergence is defined and the GAN convergence proof holds (modulo comment at end of original GAN proof)

See Sonderby et al, 2017

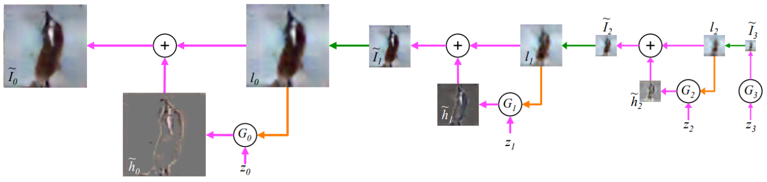


Mizra and Osindero, 2014

(a) Synthesized result
(b) Application: Change label types
(c) Application: Edit object appearance

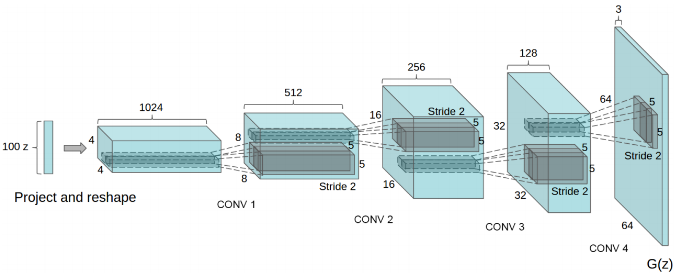
Wang et al, 2018

First flavor of GANs to scale to "high resolution" images (64×64).



Sampling procedure for LAPGAN Denton et al, 2015

Training procedure for LAPGAN Denton et al, 2015



Generator for deep convolutional generative adversarial network Radford et al, 2015

Increase quality of generator G, by

- adding batch normalization layers to G and D
- optimizing using Adam instead of SGD

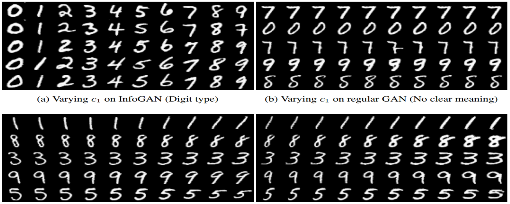
Generated bedrooms after one training pass through the dataset Radford et al, 2015

GAN loss

$$min_G max_D V(D,G) = \mathbb{E}_{x\sim P_{data}}[logD(x)] + \mathbb{E}_{z\sim P_{noise}}[log(1-D(G(z)))]$$

InfoGAN loss

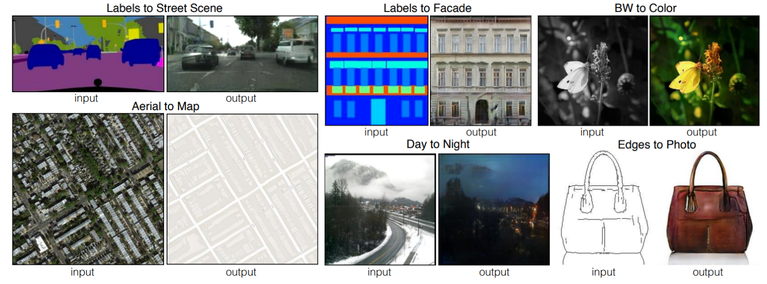
$$min_G max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$


(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)

Chen et al, 2016

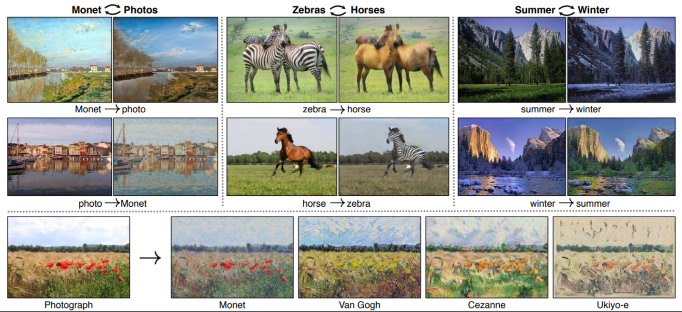
Image-to-image translation using conditional GANs and paired images.

An online demo illustrates the basic approach well – particularly the building facades.



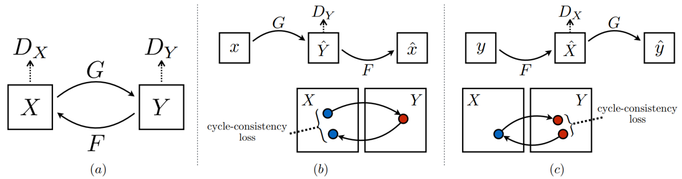
Isola et al, 2016

In CycleGAN, the authors use the GAN framework and corpora of unpaired images to learn to translate salient features between domains.



Zhu et al, 2017

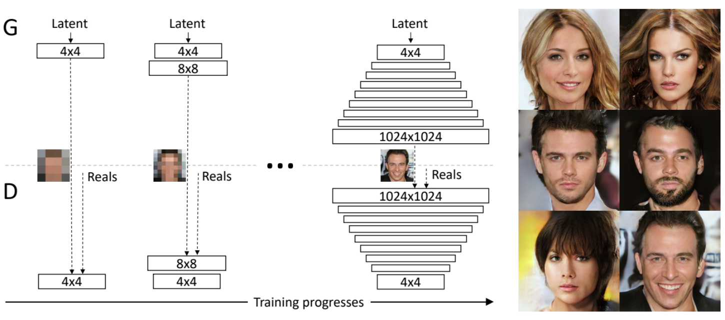
The pair of generators G and F map between domains (i.e. $G: X \to Y$ and $F: Y \to X$).



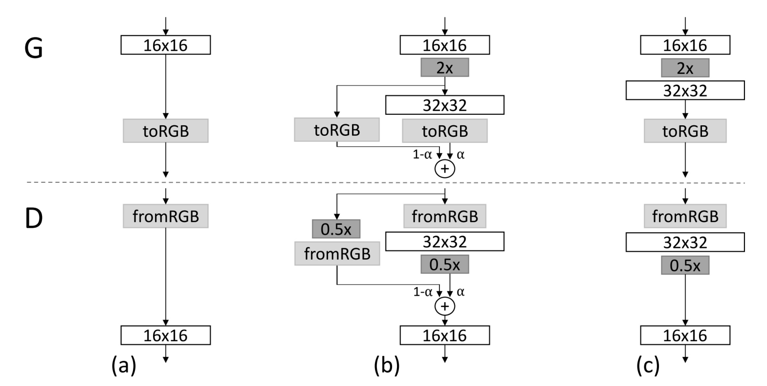
Zhu et al, 2017

The study showed by ablation that a cycle consistency loss that ensured $F(G(X)) \sim X$ and $(G(F(Y)) \sim Y$ substantially improved the quality of generated images.

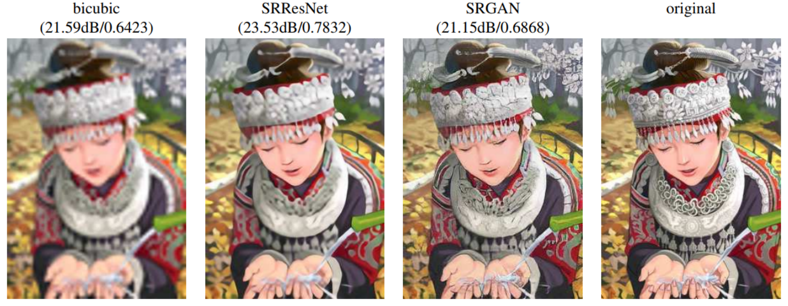
Progressively adding layers of the generator and discriminator allows scaling up to images of size 1024×1024.



Training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Karras et al, 2017



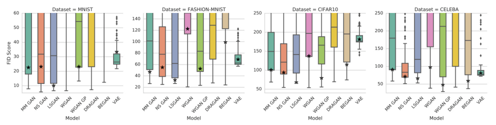
Karras et al, 2017



Ledig et al, 2017

Some methods for measuring generator quality. See Lucic et al, 2018.

- *Inception Score* takes into account the entropy of the distribution of labels (i.e. softmax output) of generated samples ($p(y|x)$) and the variance of the classes using an Inception model trained on ImageNet.
- *Fréchet Inception Distance* is the Fréchet distance between two multivariate Gaussians, $\mathcal{N}(\mu_x, \Sigma_x)$ and $\mathcal{N}(\mu_g, \Sigma_g)$, where the parameters of the distributions are estimates from the Inception embeddings of the real and generated data.



Sensitivity of GANs to hyperparameters Lucic et al, 2018

	MNIST	FASHION	CIFAR	CELEBA
MM GAN	9.8 ± 0.9	29.6 ± 1.6	72.7 ± 3.6	65.6 ± 4.2
NS GAN	6.8 ± 0.5	26.5 ± 1.6	58.5 ± 1.9	55.0 ± 3.3
LSGAN	7.8 ± 0.6*	30.7 ± 2.2	87.1 ± 47.5	53.9 ± 2.8*
WGAN	6.7 ± 0.4	21.5 ± 1.6	55.2 ± 2.3	41.3 ± 2.0
WGAN GP	20.3 ± 5.0	24.5 ± 2.1	55.8 ± 0.9	30.0 ± 1.0
DRAGAN	7.6 ± 0.4	27.7 ± 1.2	69.8 ± 2.0	42.3 ± 3.0
BEGAN	13.1 ± 1.0	22.9 ± 0.9	71.4 ± 1.6	38.9 ± 0.9
VAE	23.8 ± 0.6	58.7 ± 1.2	155.7 ± 11.6	85.7 ± 3.8

Best performance of GANs on various datasets Lucic et al, 2018