



Deep learning of generative models

Jayanta Mukhopadhyay
Dept. of Computer Science and Engg.

Courtesy: K. Sairam



References

Most content are adapted from:

- Deep Learning by Ian Goodfellow, Yoshua Bengio & Aaron Courville, An MIT Press Book
- CS231n: Convolutional Neural Networks for Visual Recognition by Fei Fei Li
- NIPS 2016 Tutorial: Generative Adversarial Networks, Ian Goodfellow
- Recent Research Papers
 - Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014.



Generative Models

- Given training data (x) , generate new samples x in model from same distribution.
- Want to learn $p_{\text{model}}(x)$ similar to $p_{\text{data}}(x)$.

Discriminative models: Learn conditional distribution $p(y|x)$

Generative models: Learn joint distribution $p(x,y)$ or alone $p(x)$.



Why Generative Models?

- To interpret underlying structure of input data even when there are no labels.
- To enable inference of latent representations
- Generation of realistic samples for artwork, super-resolution, colorization, etc.
- Generative models of time-series data can be used for simulation and planning (reinforcement learning applications!).
- Can generate missing data or datasets.
- Multimodal output like Next video frame prediction.



Generative Adversarial Networks (GANs)

- **Generative**

- Learn a generative model

- **Adversarial**

- Trained in an adversarial setting

- **Networks**

- Use Deep Neural Networks

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn a transformation to training distribution.



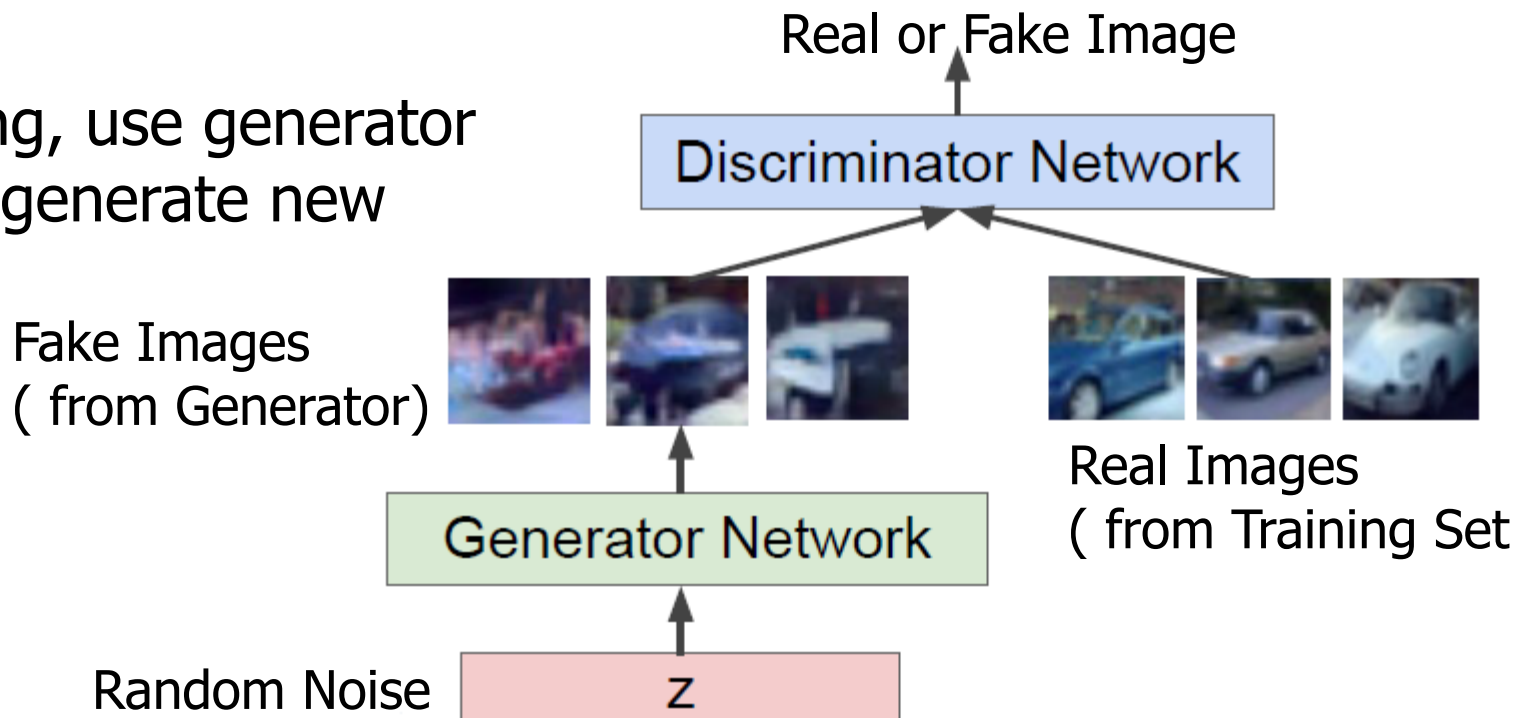
GAN Applications

- Image generation
- High-resolution image generation
- Interactive image generation
- Image inpainting
- Semantic segmentation
- Text-to-image generation
- Image-to-image translation
- Video generation

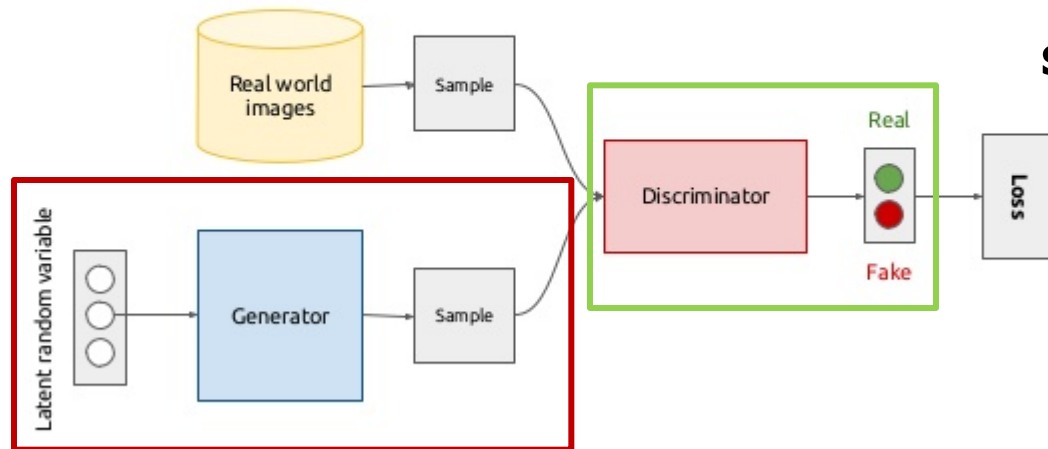
Training GANs: Two-player Game

- **Generator network:** try to fool the discriminator by generating real-looking images
- **Discriminator network:** try to distinguish between real and fake images

After training, use generator network to generate new images.



Training GANs: Two-player Game



simultaneously train G and D

- Generator G picks a point z from noise distribution, and produces a sample.
- $G(z)$ represents a mapping from noise distribution to the input space.
- $G(\cdot)$ is a network

Discriminator D takes input from both real-world images and generated image $G(z)$, and assigns the correct label to both real-world image data and generated image.



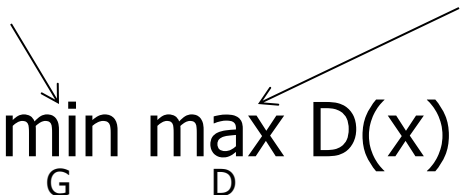
Training GANs: Two-player Game

Generator G tries to:

- a) maximize $D(G(z))$, or, minimize $(1 - D(G(z)))$
- b) minimize $D(x)$
- c) combine a and b as minimize $D(x) + (1 - D(G(z)))$

Discriminator D tries to:

- a) maximize $D(x)$ (x is real)
- b) minimize $D(G(z))$, or, maximize $(1 - D(G(z)))$ (z is fake)
- c) combine a and b as maximize $D(x) + (1 - D(G(z)))$


$$\arg \min_G \max_D D(x) + (1 - D(G(z)))$$

Discriminator wants to **maximize objective** such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake)

Generator wants to **minimize objective** such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real)

Remains **unchanged** at equilibrium point - NASH equilibrium point



Training GANs: Two-player Game

MinMax Objective Function: (Take expectation to analyse **average** behaviour)

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate Training between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

When sample is likely fake, want to learn from it to improve generator.

But gradient in this region is relatively flat!

Optimizing this generator objective does not work well!



Training GANs: Two-player Game

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

MinMax Objective Function: (Take expectation to analyse **average** behaviour)

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate Training between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

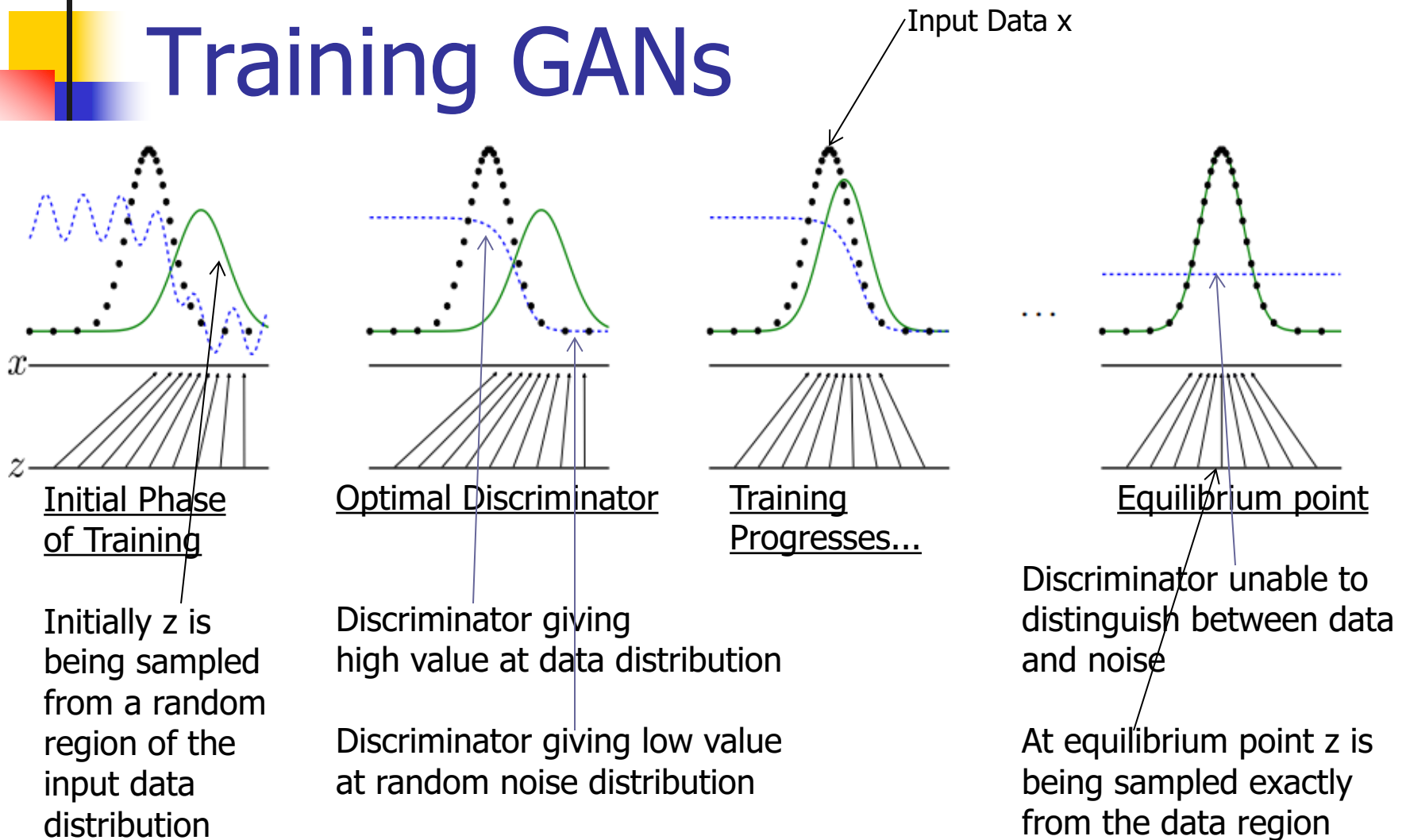
2. **Gradient descent** on generator on different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.

Same objective of fooling discriminator, but now higher gradient signal for bad samples which works much better! Standard in practice.

Training GANs



Pitfalls of GANs

- No guarantee to equilibrium
 - Mode collapsing
 - Oscillation
 - No indicator when to finish

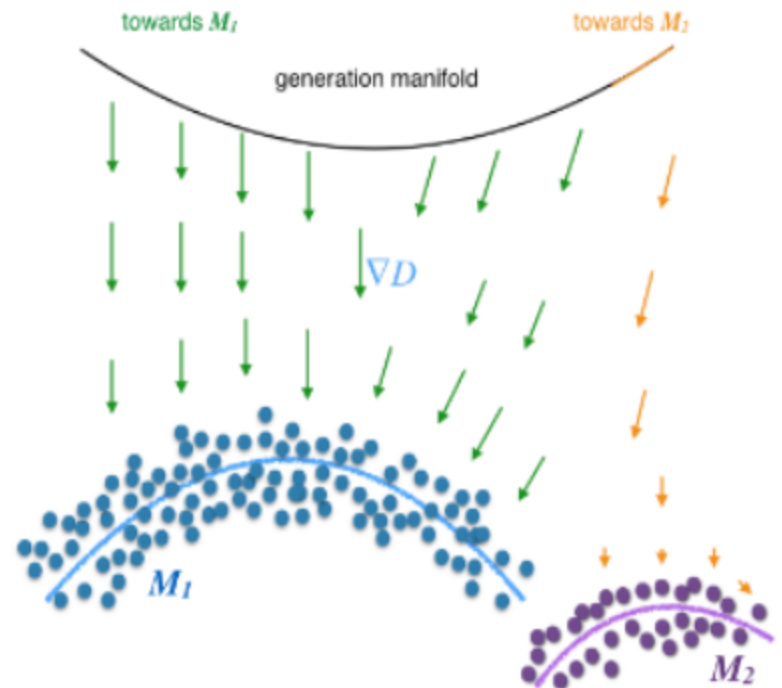



Illustration of missing modes problem



GANs: Convolutional Architectures

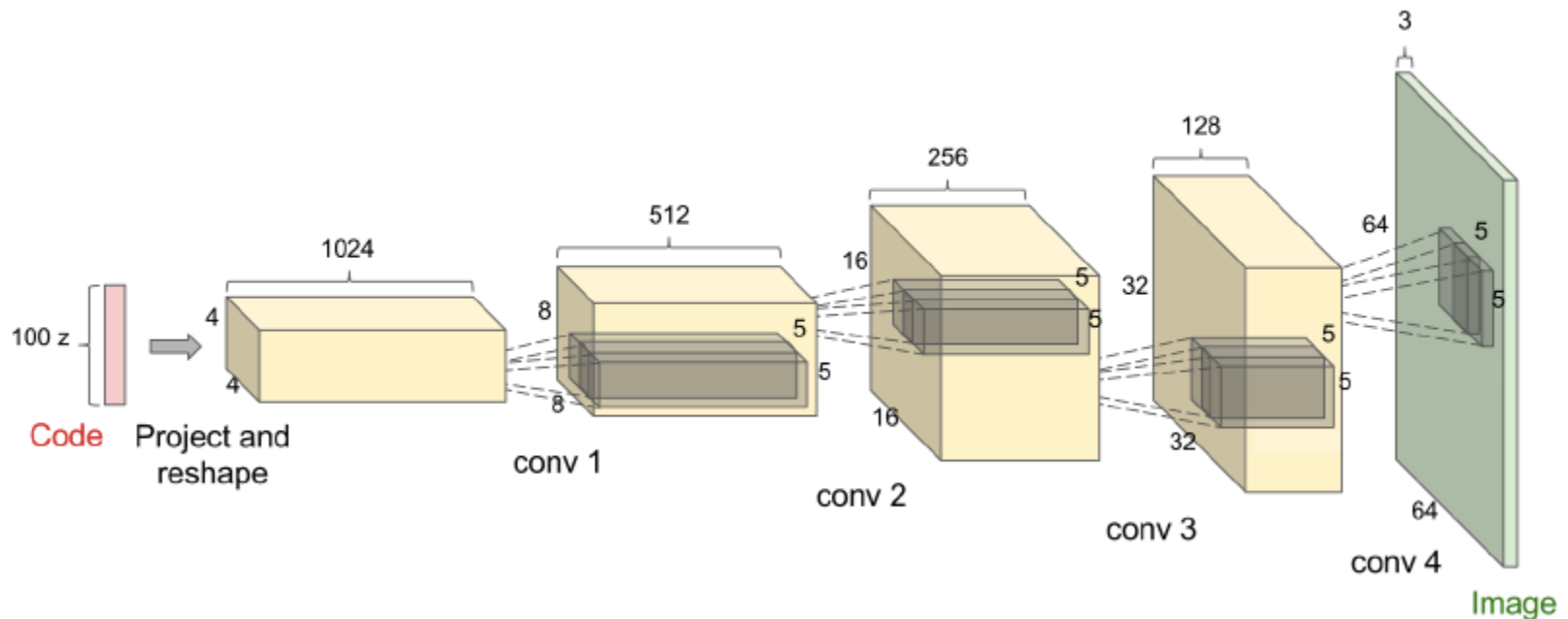
Generator is an upsampling network with fractionally-strided convolutions

Discriminator is a convolutional network

Guidelines for stable Deep Convolutional GANs (DC-GANS):

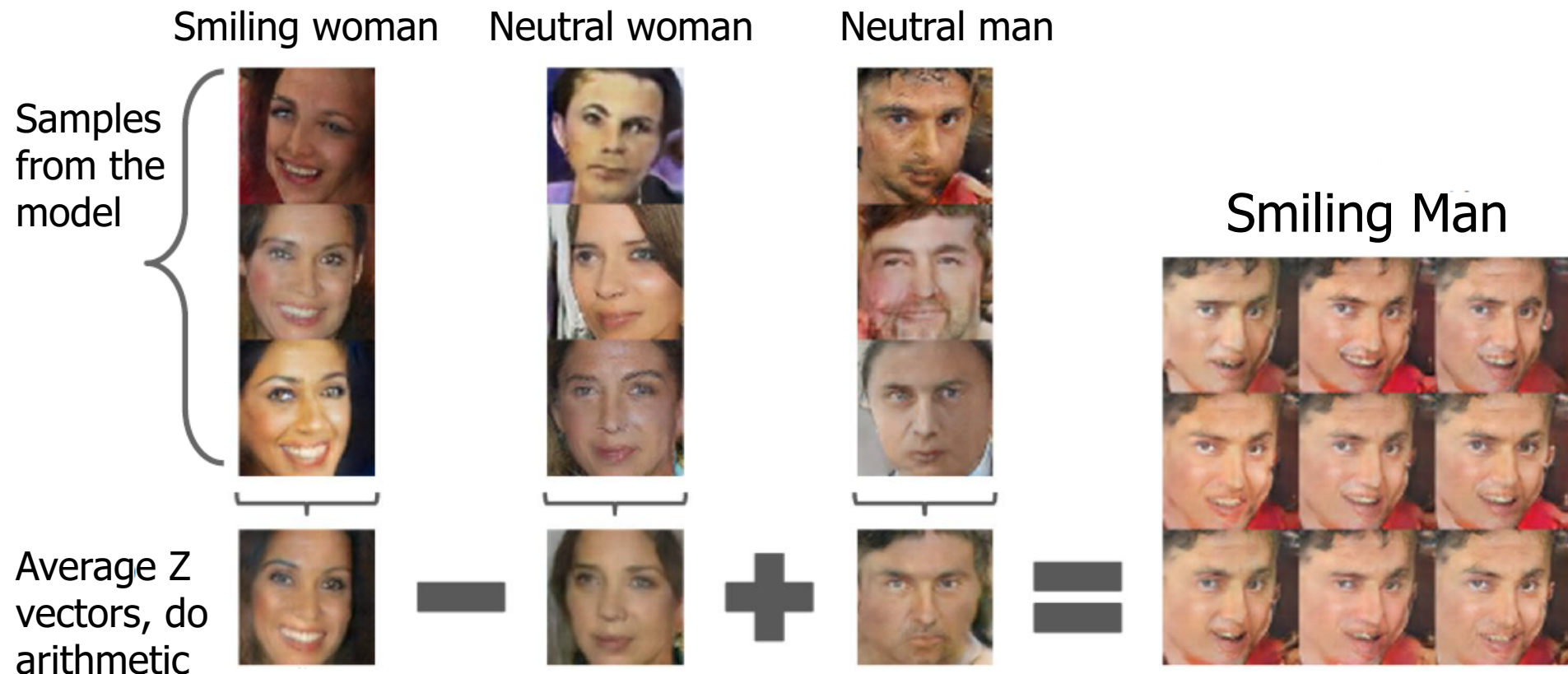
- Replace any pooling layers with strided convolutions (Discriminator) and fractional strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove FC layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses **tanh**.
- Use LeakyReLU activation in discriminator for all layers.

DCGAN (Image Generation)



Generator Network in DCGAN

Interpretable Vector Math

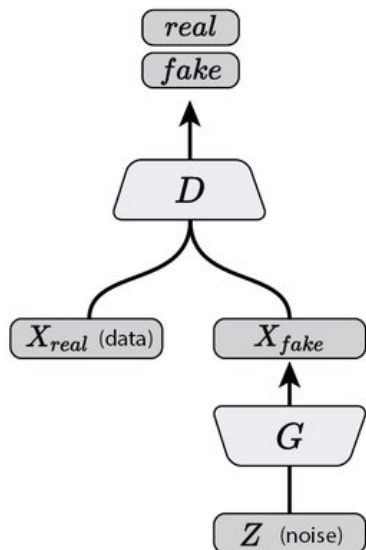


Variations of GANs

Vanilla GAN

Vanilla GAN

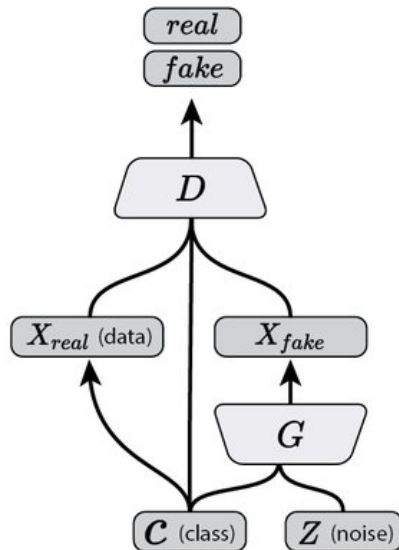
(Goodfellow, et al., 2014)



Discriminator Looks at Latent Variables

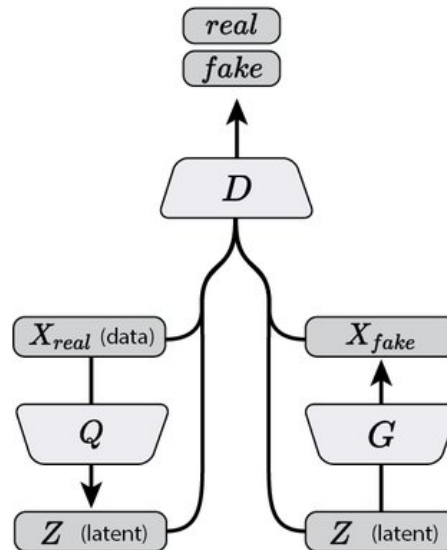
Conditional GAN

(Mirza & Osindero, 2014)



Bidirectional GAN

(Donahue, et al., 2016; Dumoulin, et al., 2016)



Conditional GANs:

Able to generate samples taking into account external information (class label, text, another image). Force G to generate a particular type of output.

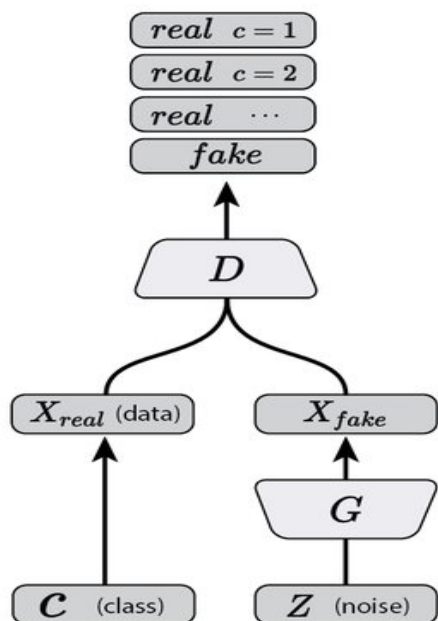
Variations of GANs

InfoGAN – Approximate the data distribution and learn interpretable, useful vector representations of data

Discriminator Predicts Latent Variables

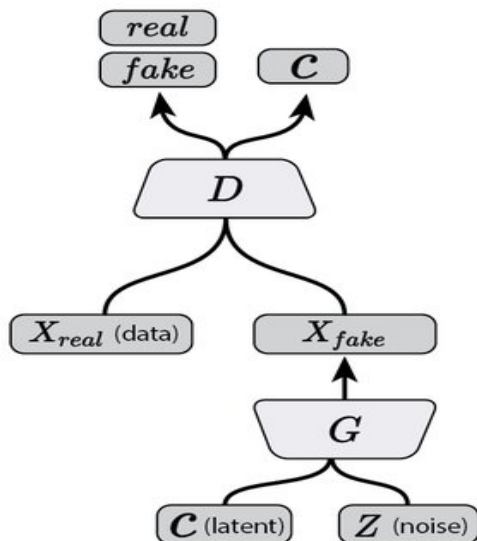
Semi-Supervised GAN

(Odena, 2016; Salimans, et al., 2016)



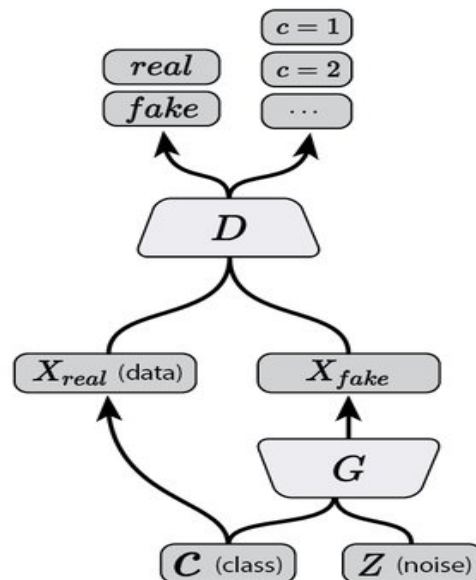
InfoGAN

(Chen, et al., 2016)



Auxiliary Classifier GAN

(Odena, et al., 2016)



Super-Resolution (SRGAN)

How do we get a high resolution (HR) image from just one

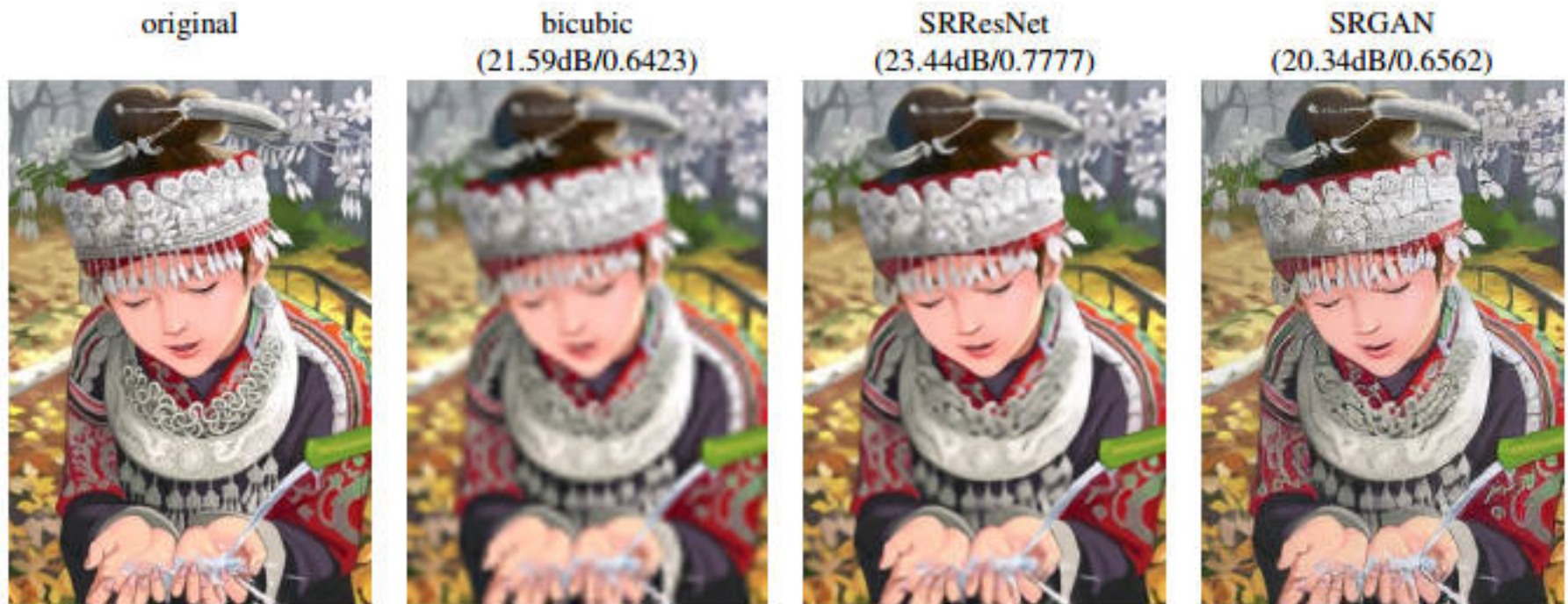


Figure 2: Illustration of performance of different SR approaches with downsampling factor: 4 \times . From left to right: original HR image, bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception. Corresponding PSNR and SSIM are shown in brackets.



SRGAN

- **G**: generator that takes a low-res image I^{LR} and outputs its HR counterpart I^{SR}
- θ_G : parameters of G, $\{W_{1:L}, b_{1:L}\}$
- l^{SR} : loss function measures the difference between the 2 HR image

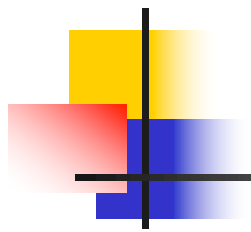
$$\hat{\theta}_G = \arg \min_{\theta_G} \frac{1}{N} \sum_{n=1}^N l^{SR}(G_{\theta_G}(I_n^{LR}), I_n^{HR})$$

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR} \sim p_{\text{train}}(I^{HR})} [\log D_{\theta_D}(I^{HR})] + \\ \mathbb{E}_{I^{LR} \sim p_G(I^{LR})} [\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR})))]$$



GANs

- Don't work with an explicit density function
- Take game-theoretic approach: learn to generate from training distribution through 2-player game
- Pros:
 - Many State-of-the-art samples!
- Cons:
 - Tricky and more unstable to train
 - Can't solve inference queries such as $p(x)$, $p(z|x)$
- Active areas of research:
 - Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
 - Conditional GANs, GANs for all kinds of applications



Thank you!