Generative Adversarial Nets

Ian J.Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio

Presenters
Do Quang Phong - 15021797
Nguyen Tuan Anh - 15020971
Nguyen Hoai Nam - 18020929
Nguyen Minh Dat - 18020288

Outline

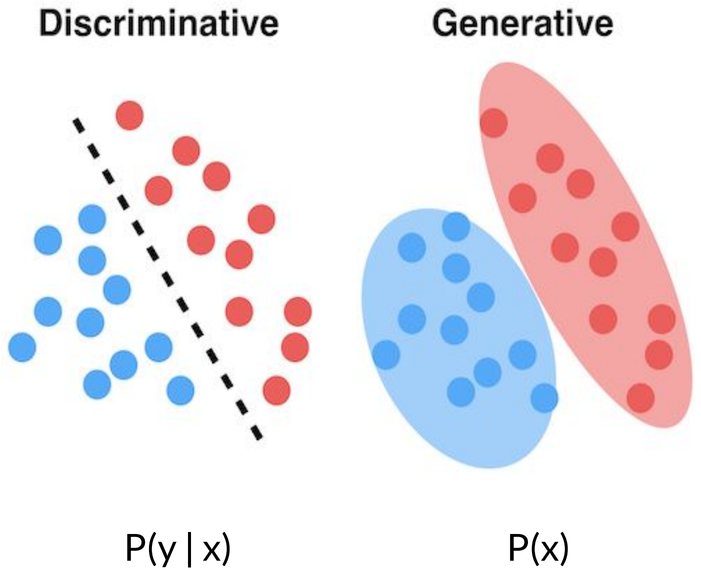
- Introduction
- Generative Adversarial Nets
- Advancements
- Applications
- Conclusion

Introduction

Generative vs Discriminative Models

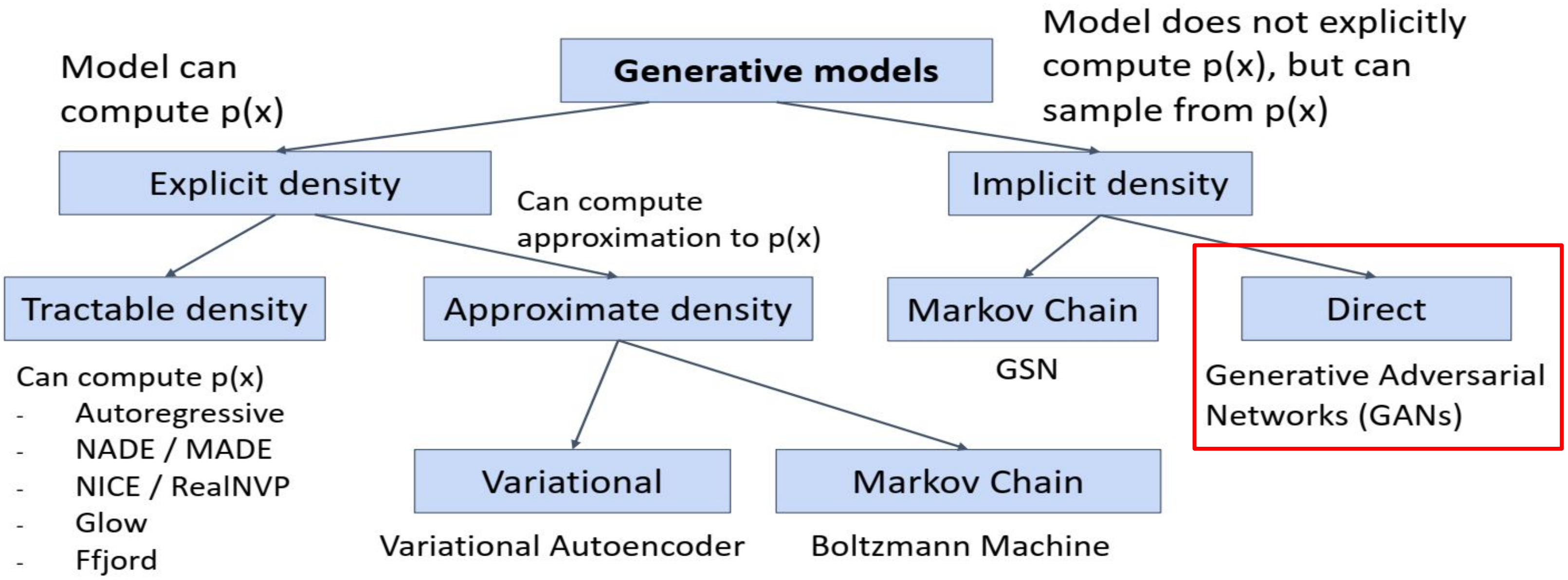
- Given a data sample x, Discriminative model aims at predicting its label y, hence its model by the posterior distribution $P(y|x)$

- Generative models instead model the distribution $P(x)$ defined over the datapoints x.



Generative Adversarial Networks

Taxonomy of Generative Models



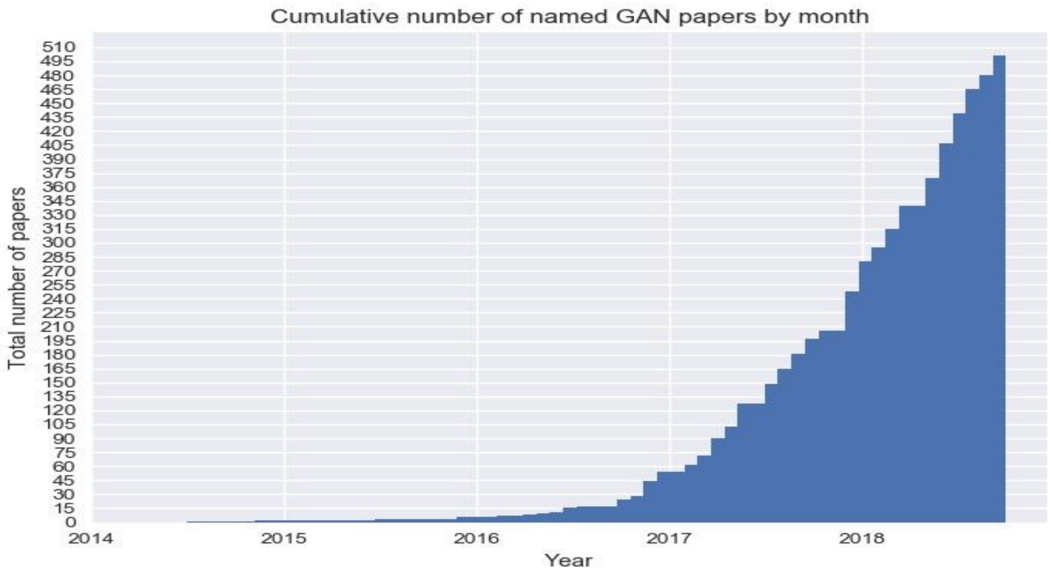
GANs Over Time

2014 - Black - White

2018 - Colored
Photo Realistic

4.5 years of GAN progress on face generation

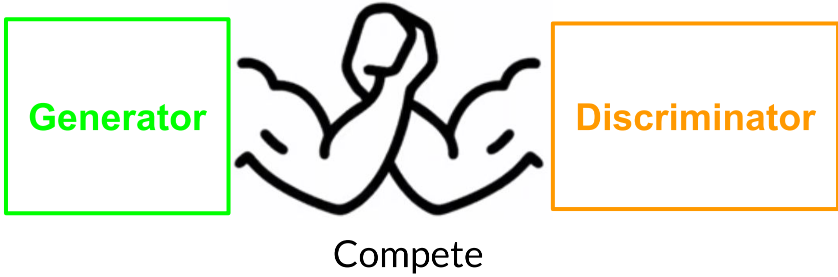
GANs Over Time



Generative Adversarial Networks (GANs)

- An introduction to Generative Adversarial Networks (GANs Goodfellow) accepted in NIPS 2014.
    - Generator, discriminator.
    - Some GAN variants

Ian Goodfellow



Generative Adversarial Networks (GANs)

Generator

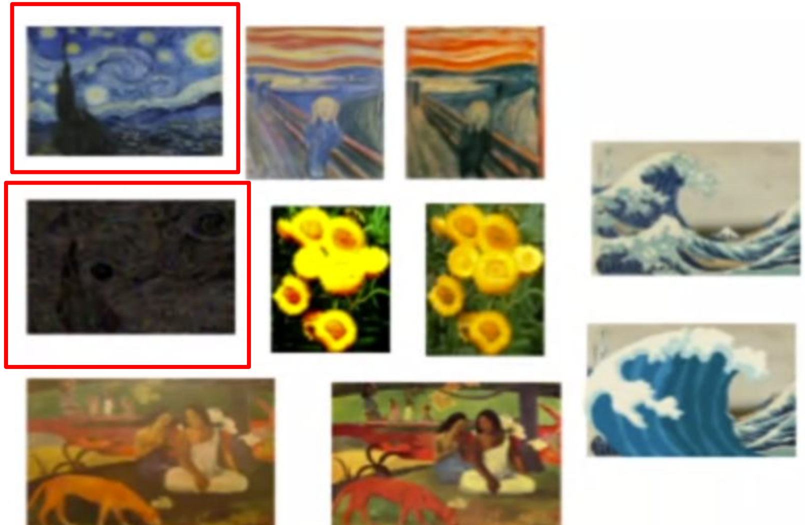
Generator learns to make **fakes** that look like **real**

Fake

Discriminator

Discriminator learns to distinguish **real** from **fake**

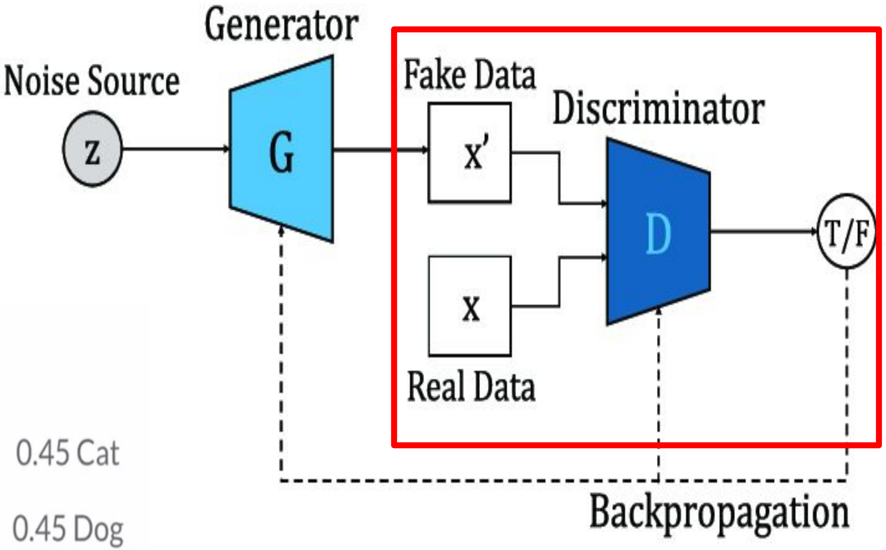
Real

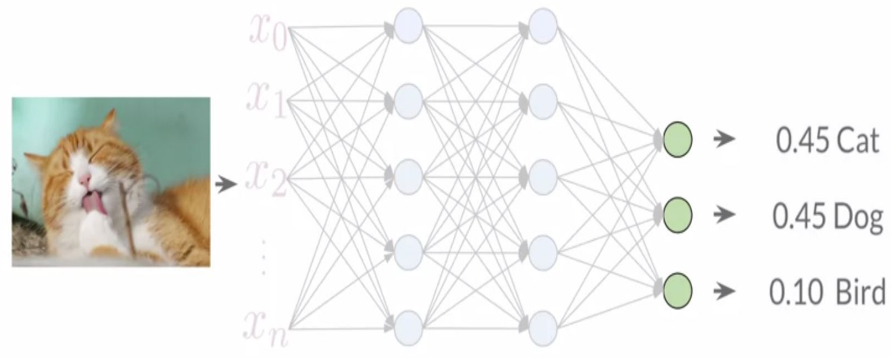


Fake

Discriminator

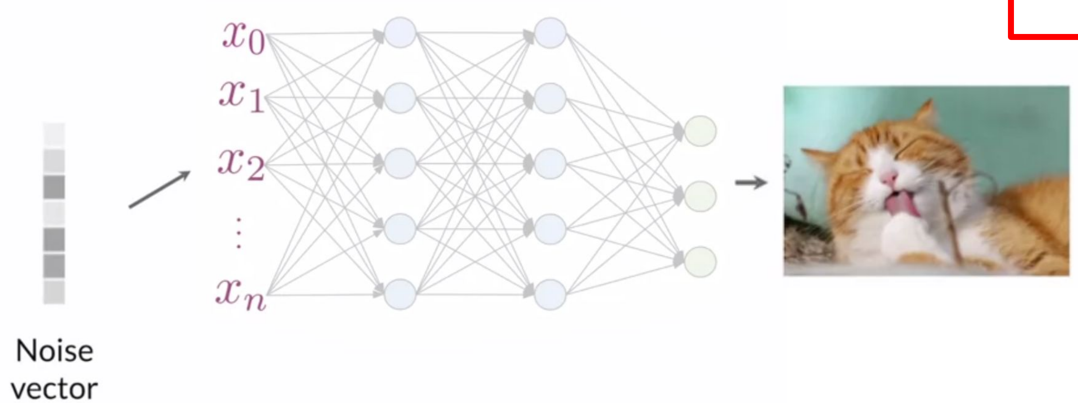
- The discriminator is a classification system
- It learns the probability of class Y (real or fake) given by features X.
- The probabilities are the feedback for generator

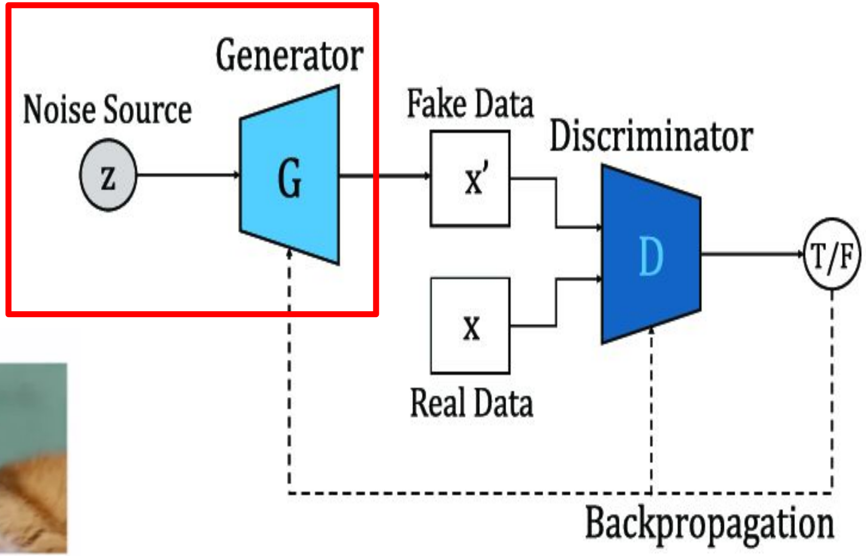




Generator

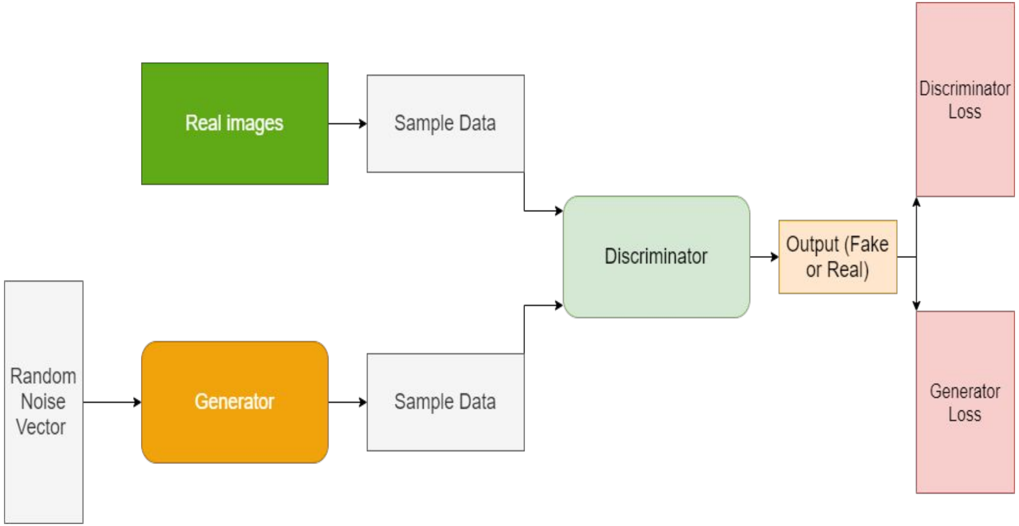
- The generator produces fake data
- It learns about the probability of features X
- The generator takes input as noise (random features)





Putting It All Together

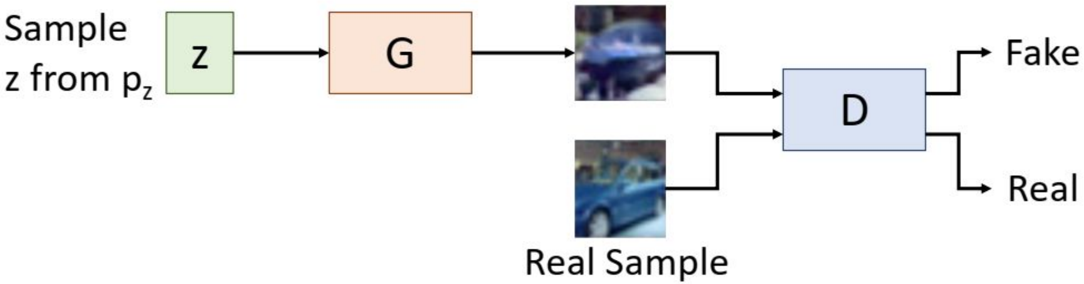
- The Binary Cross Entropy Loss is used for training model.
- GANs training in an alternating fashion
- Train Generator & Discriminator together, like a minimax game (adversarial training)



GAN Training Phase

In each iteration, alternately train

- ➢ Discriminator
  - ○ Learn a real sample : x
  - ○ Learn a fake sample : G(z)
- ➢ Generator
  - ○ Optimize G to fool D
  - ○ G(z) -> Real
- ➢ Object Function



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

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.

  • Update the discriminator by ascending its stochastic gradient:

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

  end for

  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Update the generator by descending its stochastic gradient:

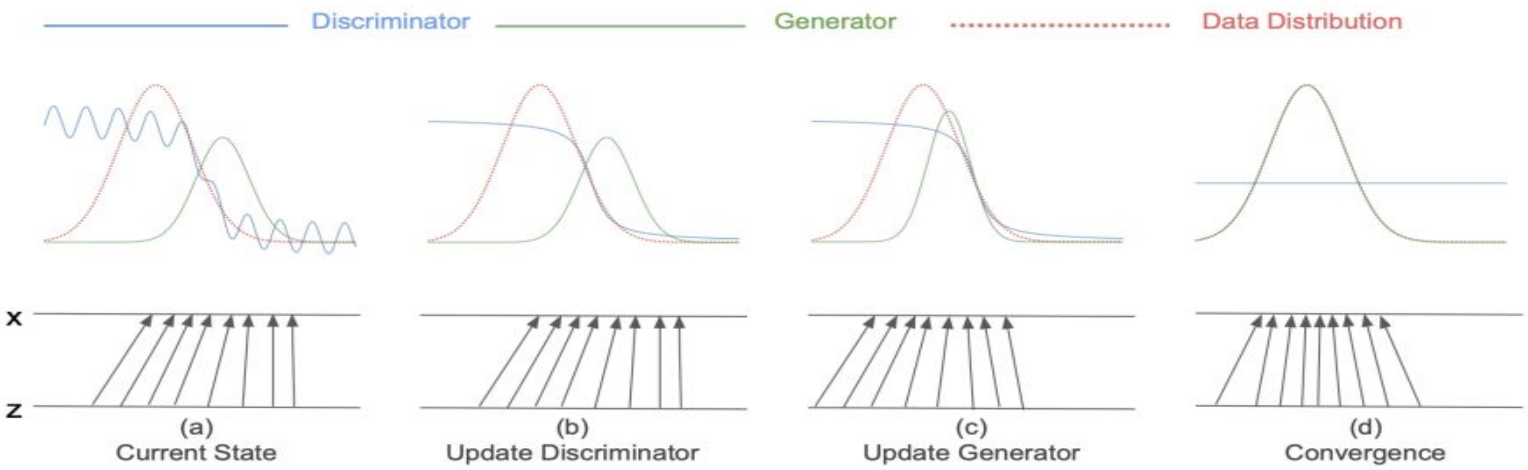
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

GAN Training Phase

Alternating optimization in GANs



Theoretical Analysis

- For a given generator, the optimal discriminator is:

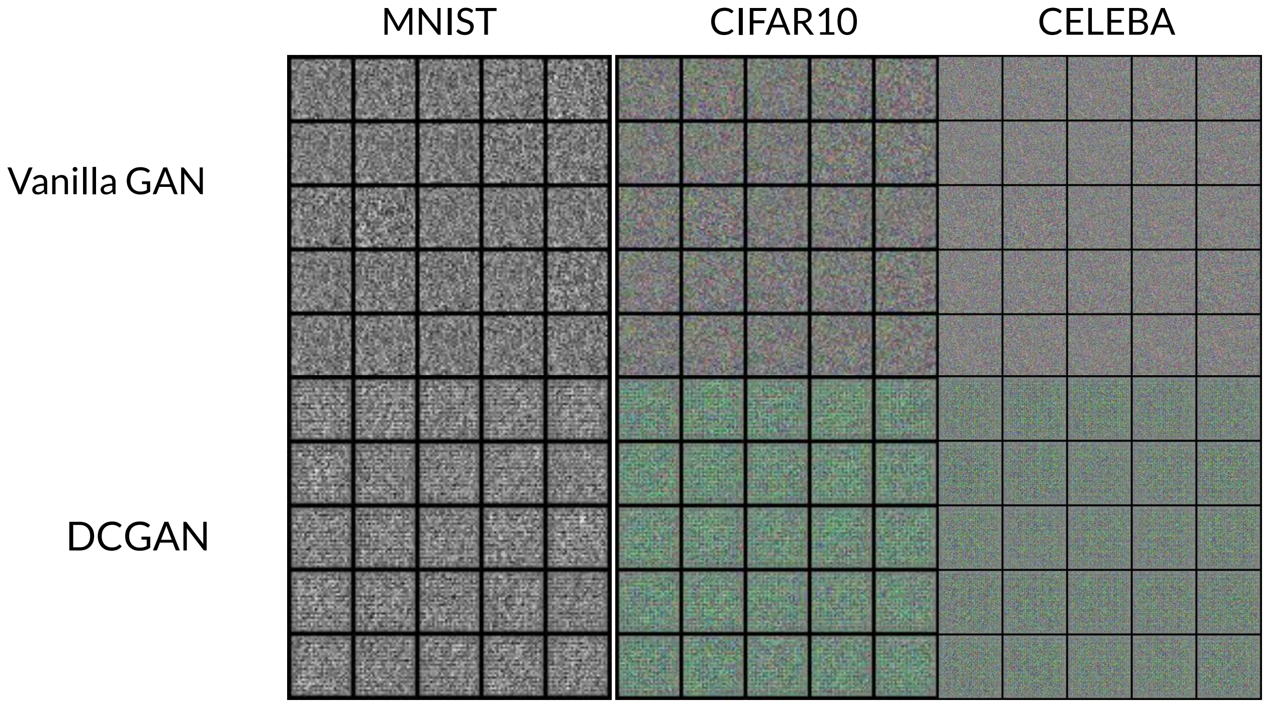
$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{3}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \rightarrow a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. $\square$

Experiments

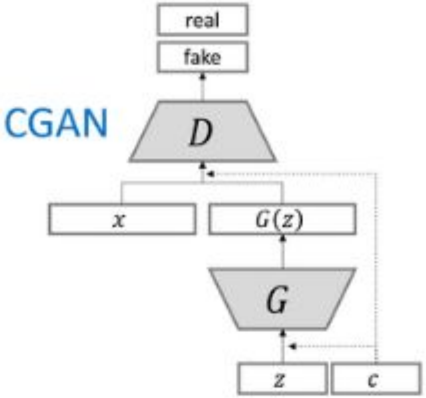


Advancements

Conditional GAN (cGAN)

- Generate image x with known label y
- Input noise vector with one-hot presentation of y to both G & D
- Discriminator loss by class

Result on MNIST



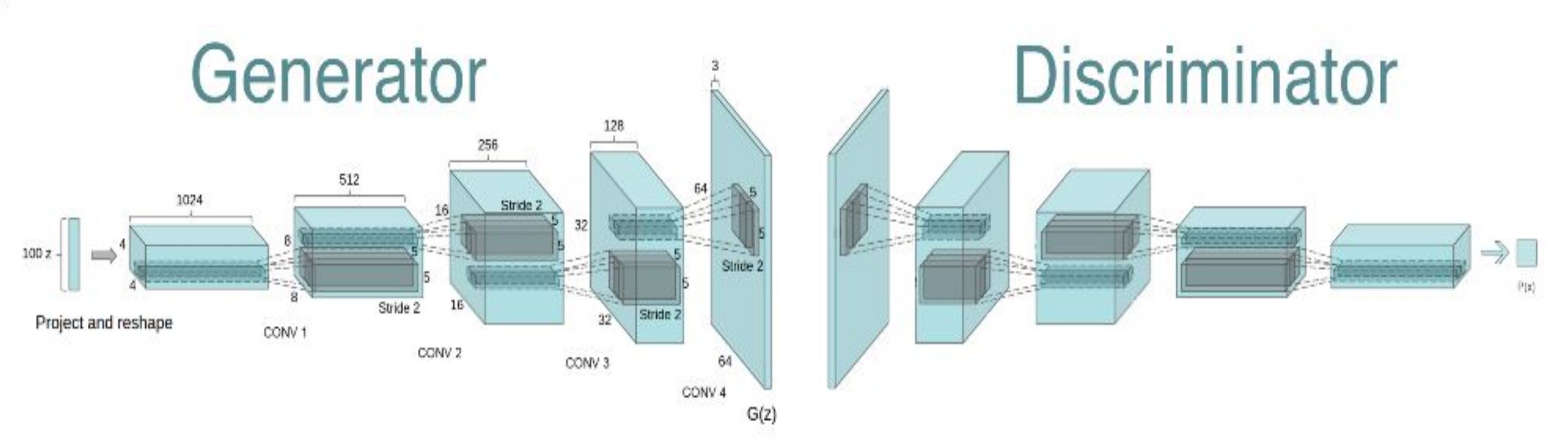
Deep Convolutional GAN (DCGAN)

- First working CNN-based GAN
- Notably, the generator uses "*transposed convolutional layers*" also informally called "*Deconvolution layers*"

Architecture guidelines for stable Deep Convolutional GANs

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

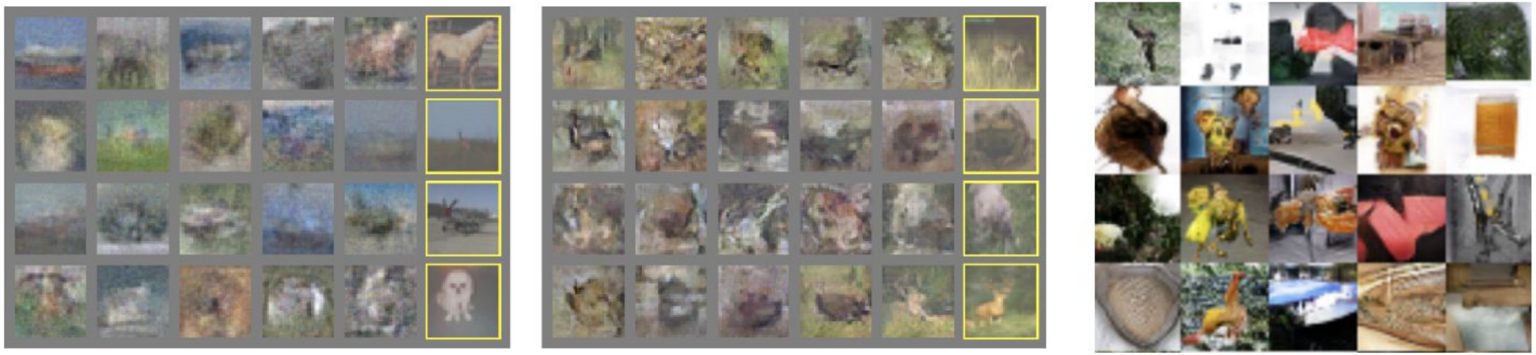
Deep Convolutional GAN (DCGAN)



Deep Convolutional GAN (DCGAN)

Original GAN (CIFAR-10)

DCGAN (ImageNet)



No convolution

One convolutional layer

Many convolutional layers (Radford et al, 2015)

Wasserstein GAN (WGAN)

GAN Issues:

- In practice, can optimize the discriminator easier than the generator
- If distribution q far away from the ground truth p, the discriminator can saturate early & the generator barely learns anything.
- Research has indicated that if your discriminator is too good, then generator training can fail due to vanishing gradients.

Solution : Change loss function => More stable training

	Discriminator/Critic	**Generator**
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(z^{(i)}\right)\right)\right)$
WGAN	$\nabla_{w} \frac{1}{m} \sum_{i=1}^{m} \left[D(x^{(i)}) - D(G(z^{(i)}))\right]$	$\nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} D(G(z^{(i)}))$

Applications

Applications

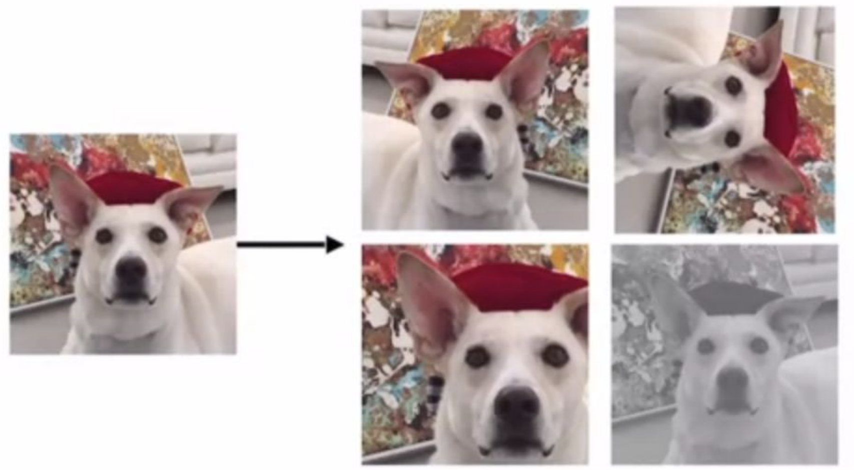
- Huge opportunity to work in applications
- Many companies are using GANs for their work
- Some applications such as :
    - Data Augmentation
    - Image Translation
    - Adversarial Machine Learning
    - Face Reenactment

Data Augmentation

- Training data issues :
  - Not enough data to achieve the best performance
  - Too imbalanced
- Supplement data when real data is ...
  - Too expensive
  - Too rare

Data Augmentation

Can mix the data augmentation techniques !



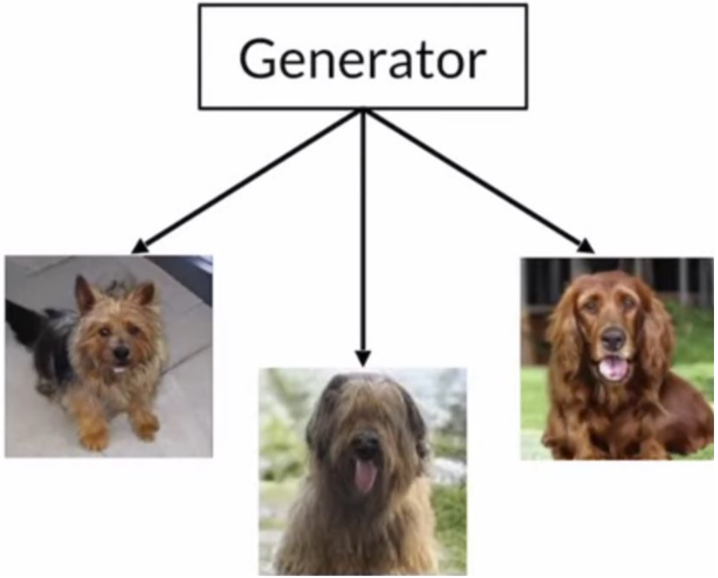


Image Translation

CycleGAN

- The most common unpaired training image-to-image translation
- Two similar generators
  - G: x → y   (forward)
  - F: y → x   (backward)
- Two similar discriminators DX and DY
- Cycle-consistency loss

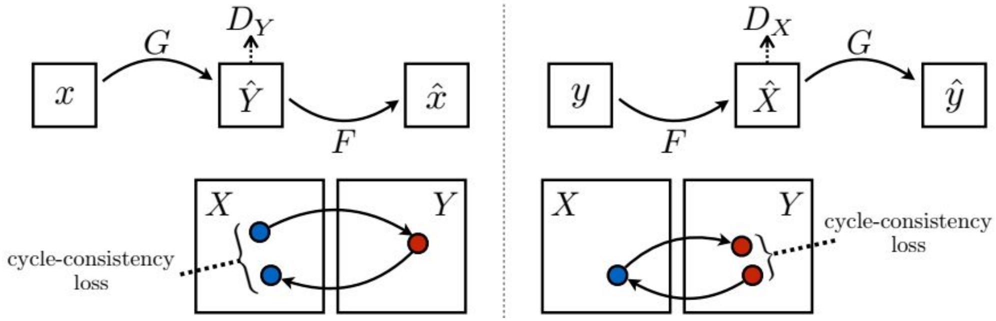


Image Translation

From one domain to another

CycleGAN

Image Translation

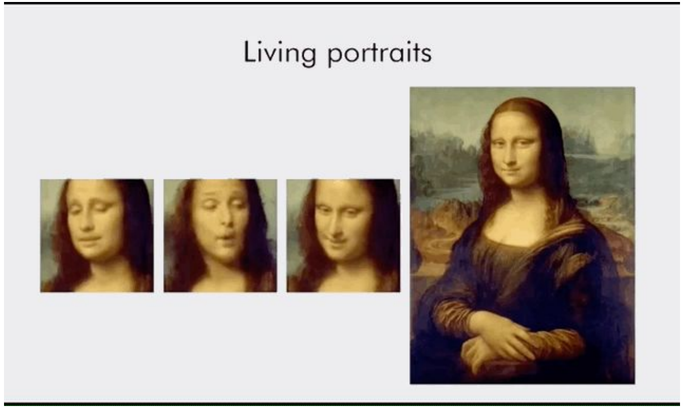
GauGAN

Doodles

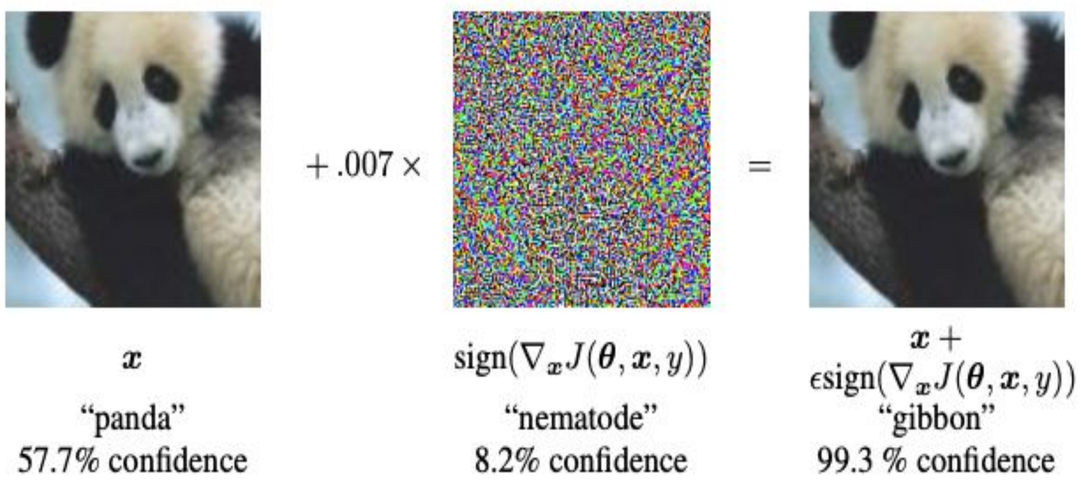
↓

Pictures

Face Reenactment



Adversarial Research Areas



Adversarial examples

Companies using GANs

Next - gen
Photoshop

Image Filters

Text
Generation

Data
Augmentation

Conclusion

- GANs have been successfully applied to several domain and tasks
- However, working with GANs can be very challenging in practice
  - Unstable optimization
  - Mode collapse
  - Evaluation
- Besides, many cool applications with GANs are using in real life.

Acknowledgement

- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, “Generative Adversarial Networks”. In NIPS 2014.
- A. Radford, L. Metz, and S. Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In ICLR 2016.
- M. Arjovsky, S. Chintala, and L. Bottou. “Wasserstein generative adversarial networks”. In ICML 2017
- CVPR 2018 Tutorials on GANs
- Generative Adversarial Networks (GANs) Specialization
- https://github.com/hindupuravinash/the-gan-zoo

Further Reading

- https://github.com/goodfeli/adversarial
- http://nvidia-research-mingyuliu.com/gaugan/
- Improved Training of Wasserstein GANs
- https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490
- https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
- CS236G Generative Adversarial Networks
- https://blog.paperspace.com/nvidia-gaugan-introduction/

Thank you for your attention