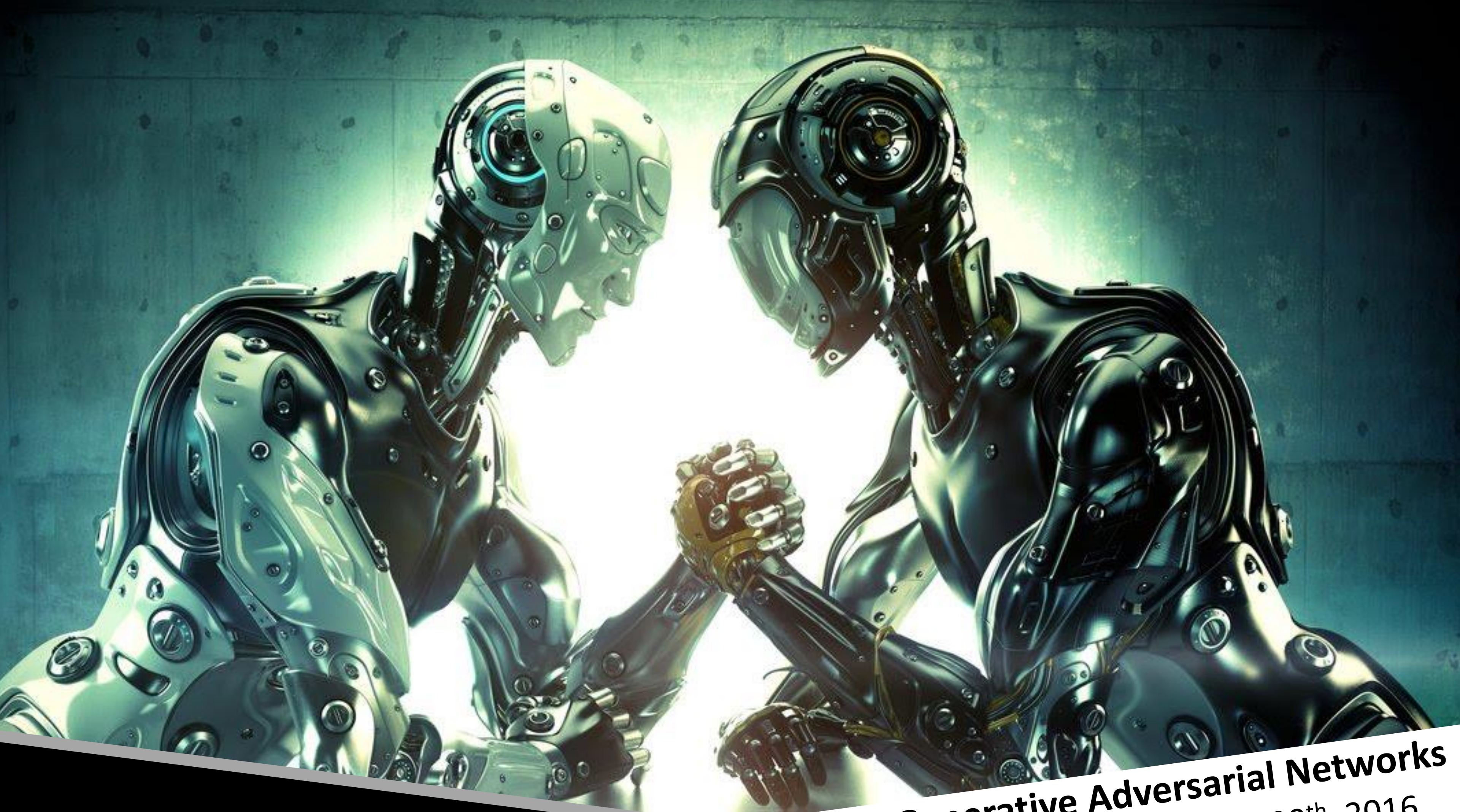


Generative Adversarial Networks

Dec 29th, 2016

Applied Deep Learning

YUN-NUNG (VIVIAN) CHEN

WWW.CSIE.NTU.EDU.TW/~YVCHEN/F105-ADL

National Taiwan University

Slide credit from Hung-Yi Lee

Review

Generative Model

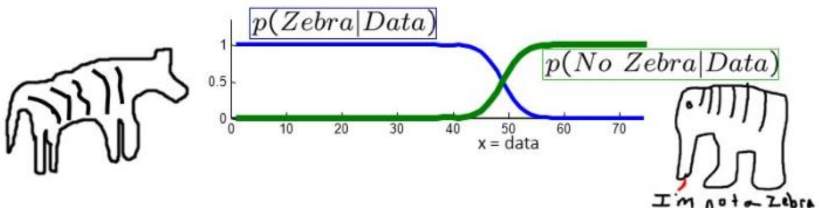
Discriminative v.s. Generative Models

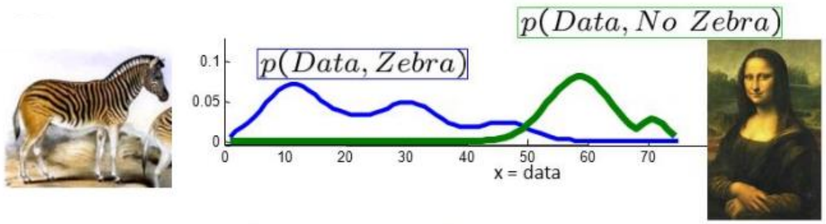
Discriminative

- learns a function that maps the input data (*x*) to some desired output class label (*y*)
  - directly learn the conditional distribution *P(y|x)*

Generative

- tries to learn the joint probability of the input data and labels simultaneously, i.e. *P(x,y)*
  - can be converted to *P(y|x)* for classification via Bayes rule

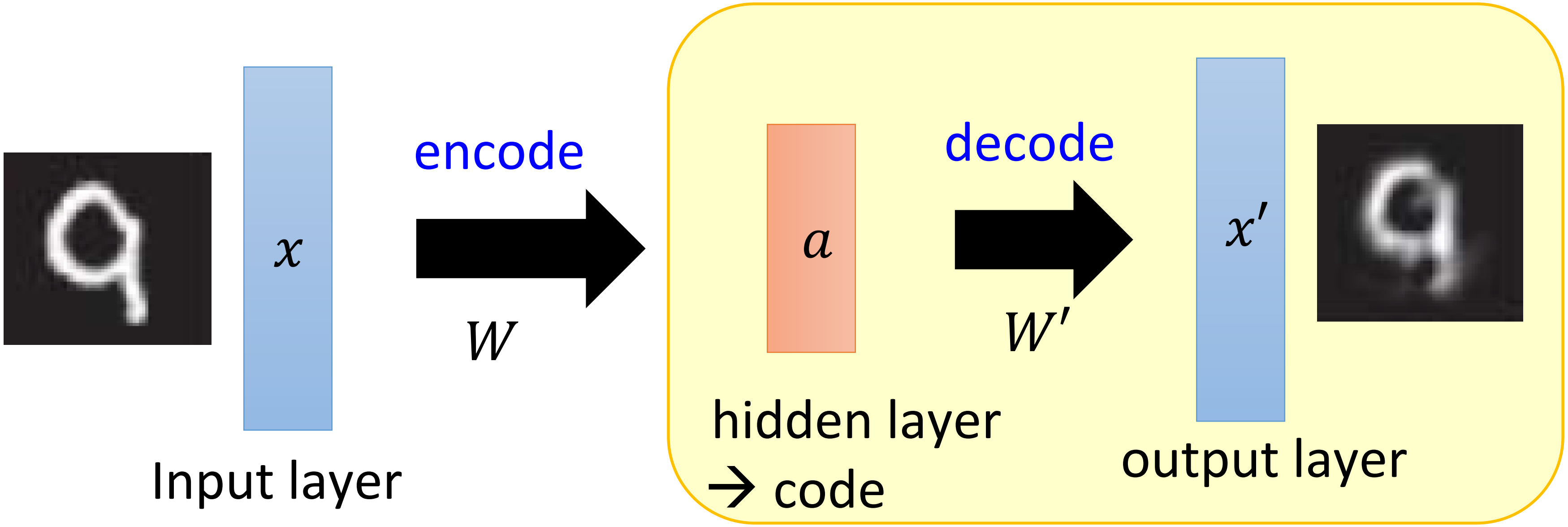




Advantage: generative models have the potential to <u>understand and explain the underlying structure</u> of the input data even when there are no labels

Generator

Decoder from autoencoder as generator



The generator is to generate the data from the code

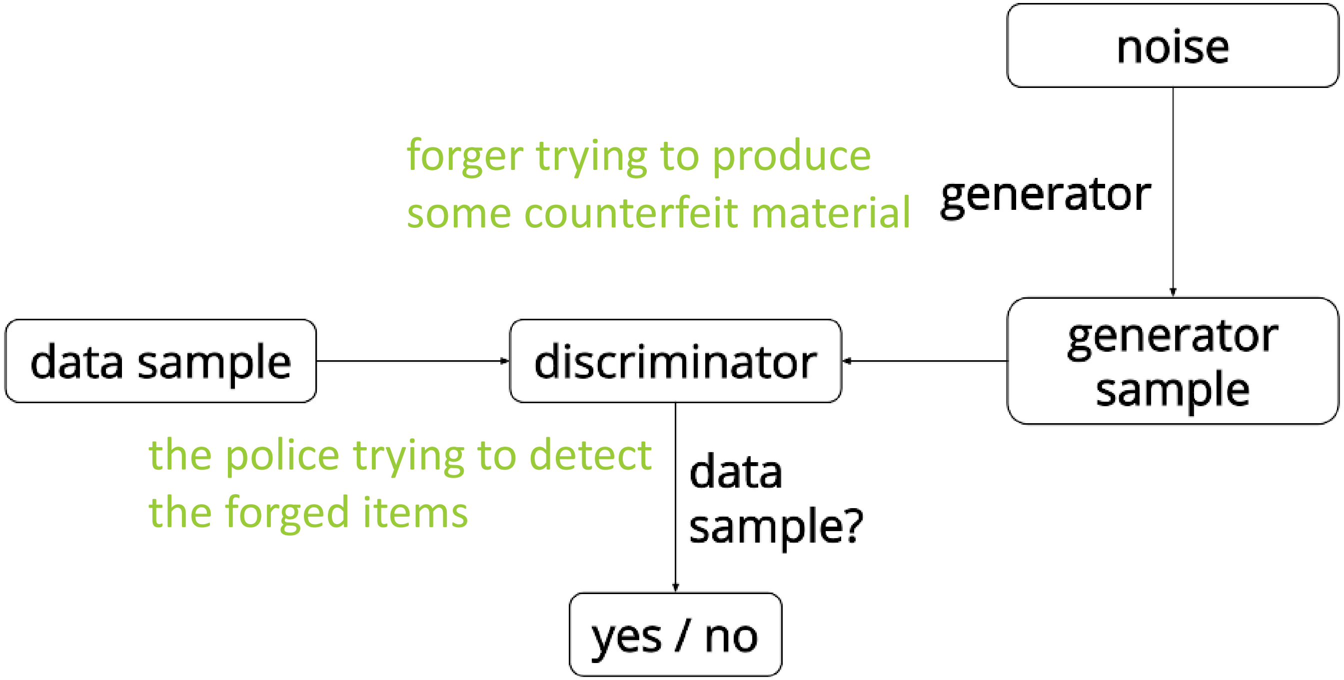
Generative Adversarial Network (GAN)

Representation Learning

"There are many interesting recent development in deep learning…The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion." – **Yann LeCun**

Generative Adversarial Networks (GAN)

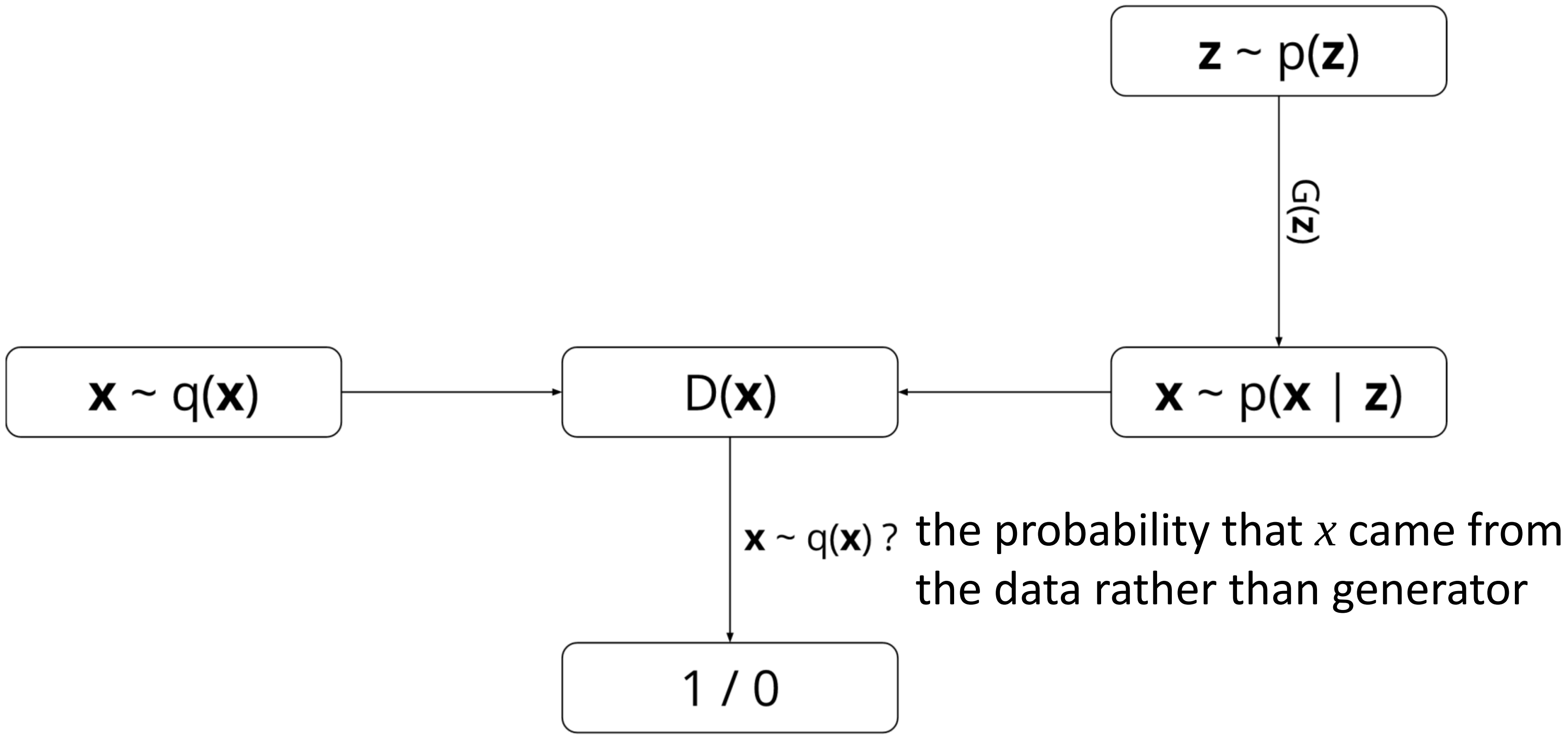
Two competing neural networks: generator & discriminator



Training two networks jointly → the generator knows how to adapt its parameters in order to produce output data that can fool the discriminator

Goodfellow, et al., “Generative adversarial networks,” in *NIPS*, 2014.
http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/

Generative Adversarial Networks (GAN)



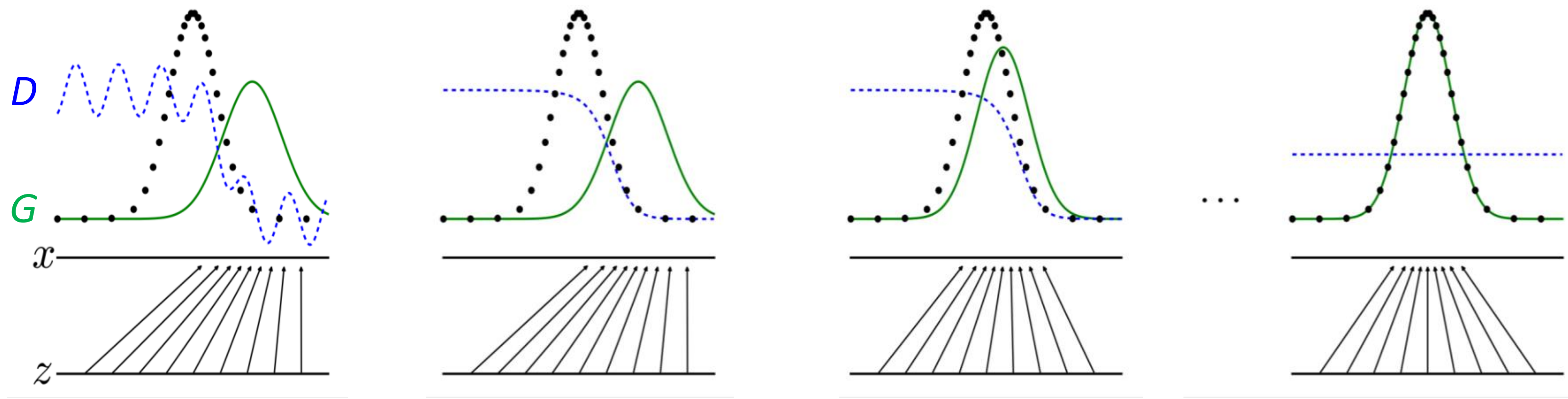
GAN Objective

$$\min_G \max_D V(D, G)$$

D(*x*): the probability that *x* came from the data rather than generator

$$= \mathbb{E}_{q(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

$$= \int q(\mathbf{x}) \log(D(\mathbf{x})) d\mathbf{x} + \iint p(\mathbf{z}) p(\mathbf{x} \mid \mathbf{z}) \log(1 - D(\mathbf{x})) d\mathbf{x} d\mathbf{z}$$



GAN Training Algorithm

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**

Discriminator

  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

Generator

  - 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.

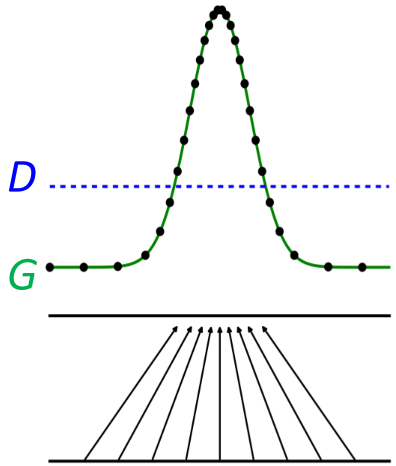
Goodfellow, et al., "Generative adversarial networks," in *NIPS*, 2014.

GAN Nash Equilibrium

Global optimality

- Discriminator

$$D^*(\mathbf{x}) = \frac{q(\mathbf{x})}{q(\mathbf{x}) + p(\mathbf{x})}$$



- Generator

$$G^*(\mathbf{z}) \text{ s.t. } p(\mathbf{z}) = q(\mathbf{x})$$

Two competing networks are trained towards global optimality

GAN-Generated Bedrooms

Applications of Generative Models

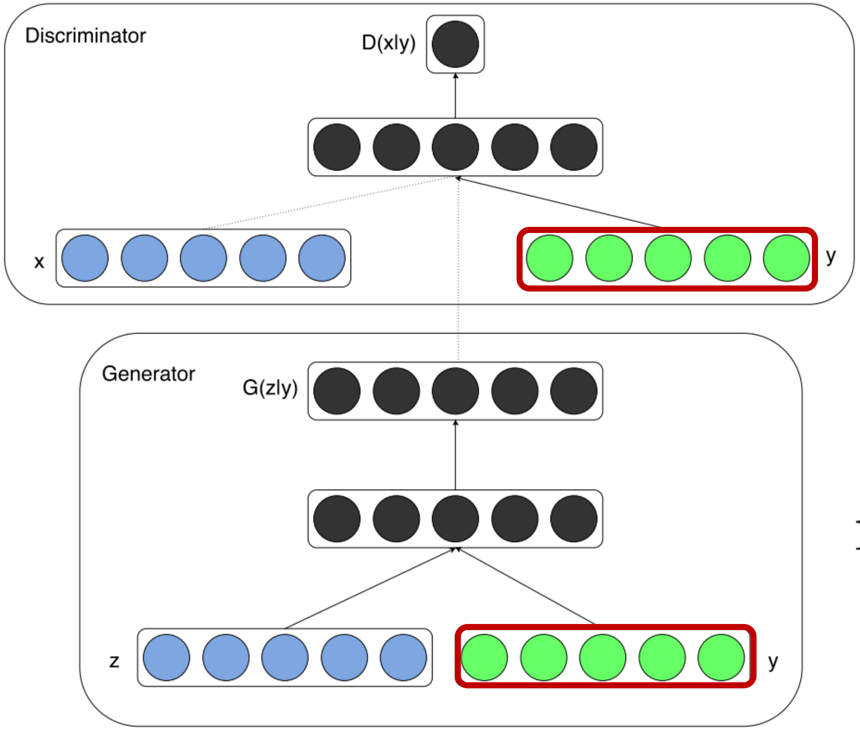
Semi-supervised learning

- few training samples with annotations
- generate more training data using GAN

Conditional GAN

Generator Conditioned on Labels

Conditional GAN



GAN

$$\min_G \max_D V(D, G)$$

$$= \mathbb{E}_{q(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

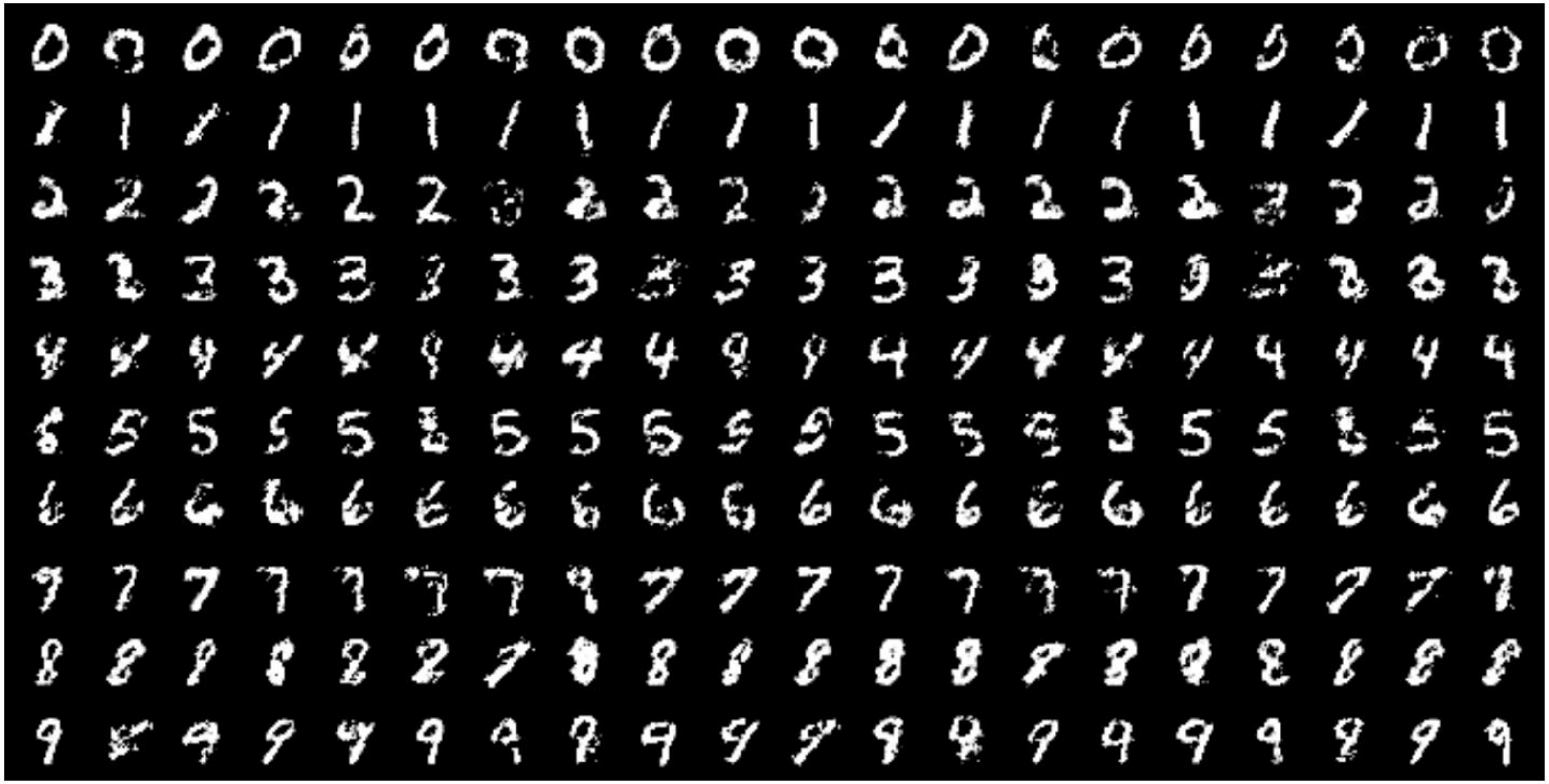
Conditional GAN

$$\min_G \max_D V(D, G)$$

$$= \mathbb{E}_{q(\mathbf{x})}[\log(D(\mathbf{x} \mid \mathbf{y}))] + \mathbb{E}_{p(\mathbf{z})}[\log(1 - D(G(\mathbf{z} \mid \mathbf{y})))]$$

Mirza and Osindero, "Conditional Generative Adversarial Nets," arXiv:1411.1784.

Generated Images Conditioned on Label



Adversarially Learned Inference (ALI)

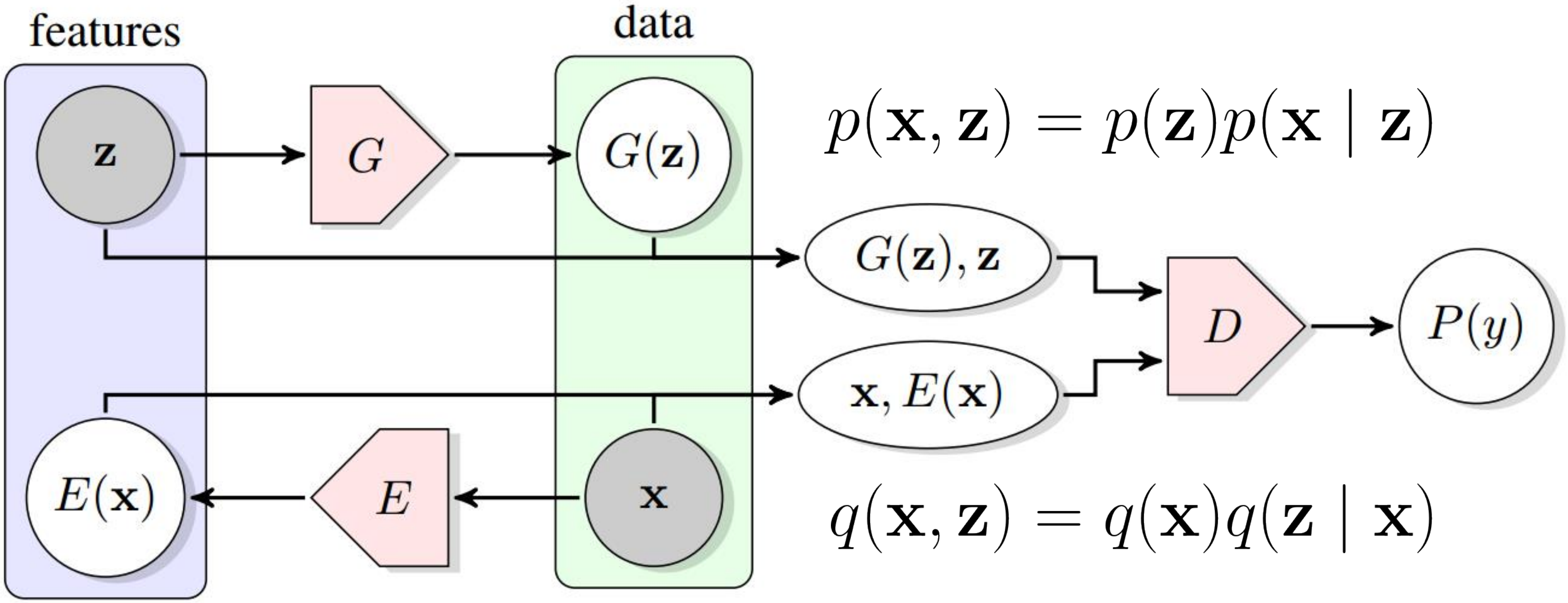
Inference with Latent Variables

Adversarially Learned Inference (ALI) / Bidirectional GAN (BiGAN)

Inference is important but ignored by GAN

Idea: incorporate <u>latent variables</u> for inference

Inference: given x, what z is likely to have produced it



$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z})$$

$$q(\mathbf{x}, \mathbf{z}) = q(\mathbf{x})q(\mathbf{z} \mid \mathbf{x})$$

Dumoulin et al., "Adversarially Learned Inference," arXiv:1606.00704.
Donahue et al., "Adversarial Feature Learning," arXiv:1605.09782.

ALI / BiGAN

$$\min_{E,G} \max_{D} V(D, G, \boxed{E})$$

$$= \mathbb{E}_{q(\mathbf{x})}[\log(D(\mathbf{x}, \boxed{E(\mathbf{x})}))] + \mathbb{E}_{p(\mathbf{z})}[\log(1 - D(G(\mathbf{z}), \mathbf{z}))]$$

$$q(\mathbf{x}, \mathbf{z}) = q(\mathbf{x})q(\mathbf{z} \mid \mathbf{x}) \qquad p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z})$$

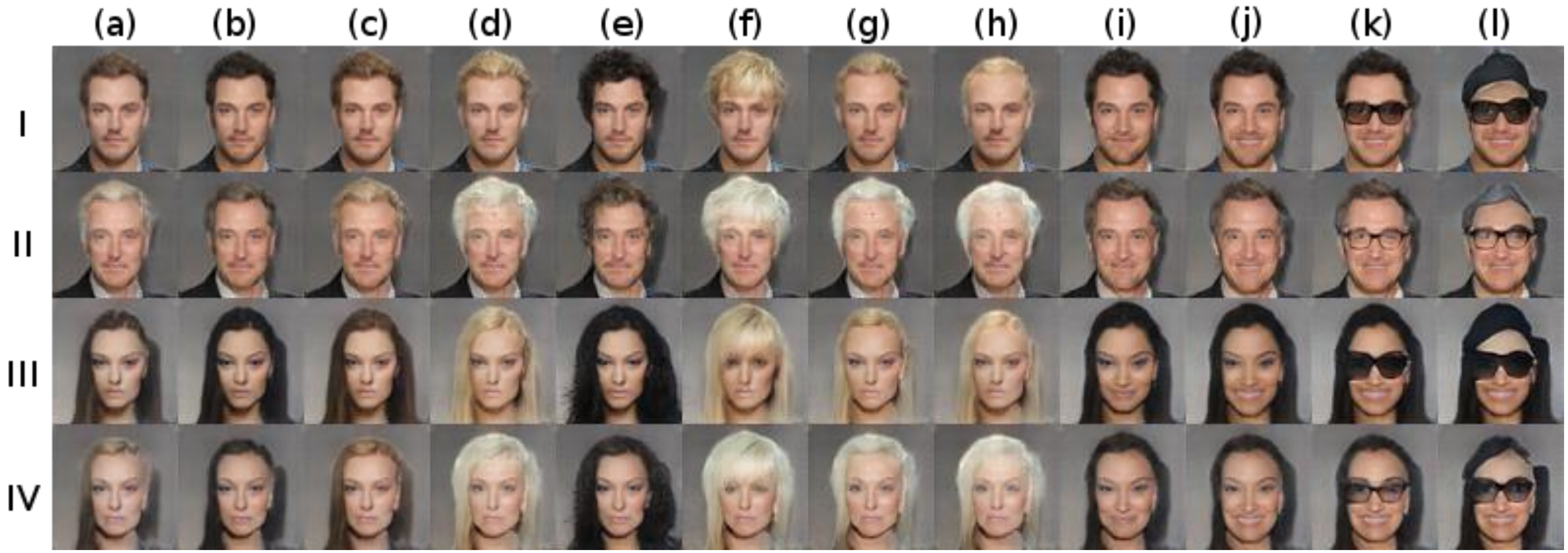
$$= \iint q(\mathbf{x})q(\mathbf{z} \mid \mathbf{x}) \log(D(\mathbf{x}, \mathbf{z}))d\mathbf{x}d\mathbf{z}$$

$$+ \iint p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z}) \log(1 - D(\mathbf{x}, \mathbf{z}))d\mathbf{x}d\mathbf{z}$$

Dumoulin et al., “Adversarially Learned Inference,” arXiv:1606.00704.
Donahue et al., “Adversarial Feature Learning,” arXiv:1605.09782.

Conditional ALI

Conditional generation: providing a conditioning variable y for generator, encoder, discriminator



Latent variables can represent attributes

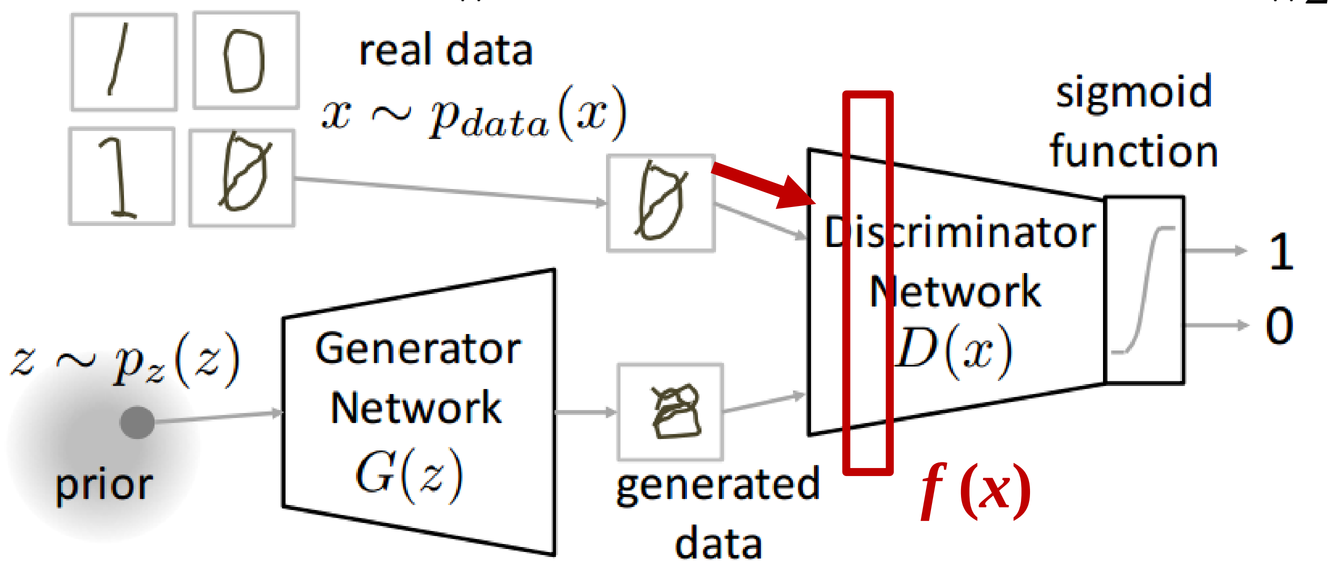
Improvement of Training GAN

Stableness and Robustness

Feature Matching (Generator Objective)

Idea: match the expected values of the features on an intermediate layer of discriminator

Generator's objective: $\left\| \mathbb{E}_{q(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p(\mathbf{z})} f(G(\mathbf{z})) \right\|_2^2$



Generators utilize the features used by discriminators as objective

Unrolled GAN (Generator Objective)

Idea: allow generator to consider discriminator's capability

Iterative optimization procedure

$$\theta_D^0 = \theta_D$$
$$\theta_D^{k+1} = \theta_D^k + \eta^k \frac{\partial f(\theta_G, \theta_D^k)}{\partial \theta_D^k}$$
$$\theta_D^* = \lim_{k\to\infty} \theta_D^k$$

$$f(\theta_G, \theta_D)$$
$$= \mathbb{E}_{q(\mathbf{x})}[\log(D(\mathbf{x};\theta_D))]$$
$$+\mathbb{E}_{p(\mathbf{z})}[\log(1-D(G(\mathbf{z};\theta_G);\theta_D))]$$

Surrogate objective for generator

$$f_K(\theta_G, \theta_D) = f(\theta_G, \theta_D^K(\theta_G, \theta_D))$$

$$\theta_G \leftarrow \theta_G - \eta \frac{\partial f_K(\theta_G, \theta_D)}{\partial \theta_G}$$

$$\theta_D \leftarrow \theta_D - \eta \frac{\partial f(\theta_G, \theta_D)}{\partial \theta_D}$$

Unrolled GAN (Generator Objective)

Idea: allow generator to consider discriminator's capability

Surrogate objective for generator

$$f_K(\theta_G, \theta_D) = f(\theta_G, \theta_D^K(\theta_G, \theta_D))$$

$$\theta_G \leftarrow \theta_G - \eta \frac{\partial f_K(\theta_G, \theta_D)}{\partial \theta_G}$$

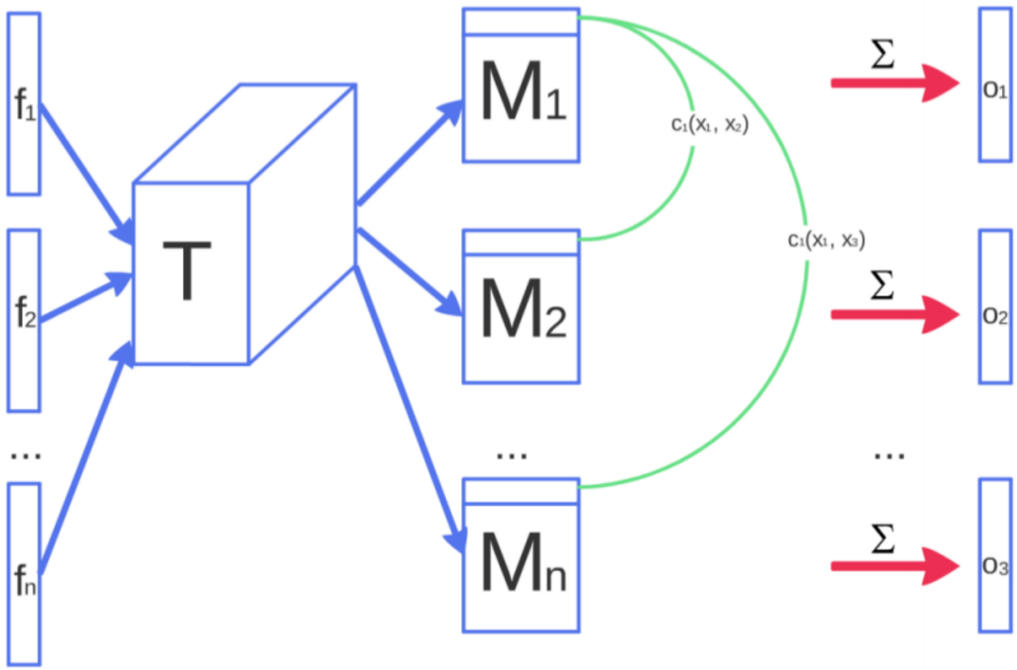
$$\theta_D \leftarrow \theta_D - \eta \frac{\partial f(\theta_G, \theta_D)}{\partial \theta_D}$$

$$\frac{\partial f_K(\theta_G, \theta_D)}{\partial \theta_G} = \frac{\partial f(\theta_G, \theta_D^K(\theta_G, \theta_D))}{\partial \theta_G} + \boxed{\frac{\partial f(\theta_G, \theta_D^K(\theta_G, \theta_D))}{\partial \theta_D^K(\theta_G, \theta_D)} \frac{\partial \theta_D^K(\theta_G, \theta_D)}{\partial \theta_G}}$$

Generators can move towards better directions based on what discriminators tell

Minibatch Discrimination (Discriminator Objective)

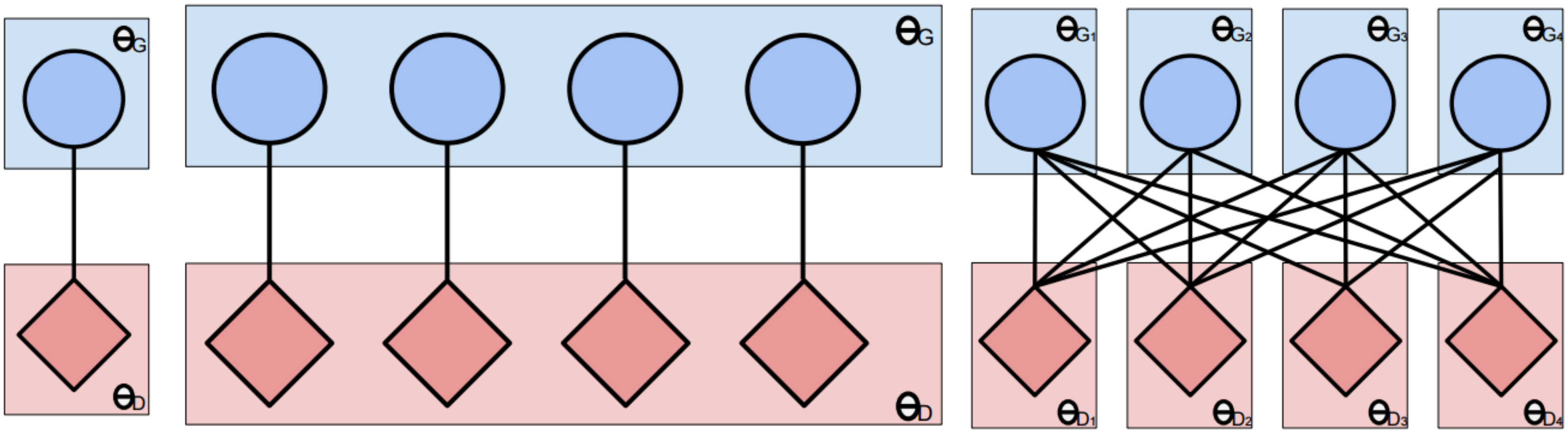
Idea: allow the discriminator to see multiple data examples in combination to avoid collapsing to a single mode



Considering a batch in combination results in better robustness and diversity

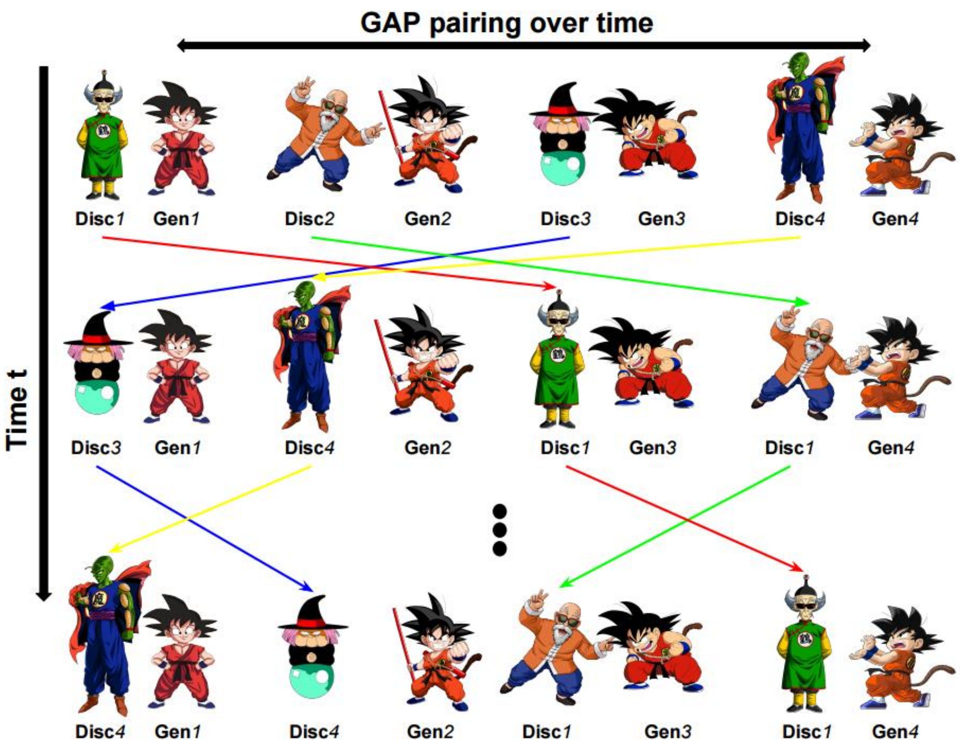
Generative Adversarial Parallelization (GAP)
(Discriminator Objective)

Idea: train multiple GANs and allow different discriminators to discriminate difference generators



(a) GAN (b) GANs trained by data-parallelism (c) GAP

Generative Adversarial Parallelization (GAP)
(Discriminator Objective)



Algorithm 1 Training procedure of GAP.

Let T be total number of weight updates.
Let N be the total number of GANs.
Let K be the swapping frequency.
Let $\mathcal{M} = \{(G_1, D_1), (G_2, D_2), \cdots, (G_N, D_N)\}$.
while $t < T$ **do**
  Update $\mathcal{M}_{i_t} = (G_{i_t}, D_{i_t})\ \forall i = 1 \cdots N$.

  if $t \ \% \ K == 0$ **then**
    Randomly select $\frac{N}{2}$ pairs with indices (i, j) w/o replacement.
    Swap D_i and D_j (G_i and G_j) $\forall i \neq j$.
  end if
end while
Select the best GAN based on GAM evaluation.

Discriminators can have better robustness because seeing different generated modes

Concluding Remarks

Generative adversarial networks (GAN)
- jointly train two competing networks, **generator** and **discriminator**

Adversarially learned inference (ALI) / bidirectional GAN (BiGAN)
- jointly train three networks, **generator**, **encoder**, and **discriminator**
- latent variables can be encoded

Training tricks
- Generator objective: feature matching, unrolled GAN
- Discriminator objective: minibatch discrimination, generative adversarial parallelization (GAP)

Applications
- semi-supervised learning