generate data

Generative Models

AIGC: AI Generated Contents

Distribution learning

. learn a distribution over data
. Sample from distribution

Training Data(CelebA)

Model Samples (Karras et.al., 2018)

4 years of progression on Faces

Brundage et al., 2017

Image credits to Andrej Risteski

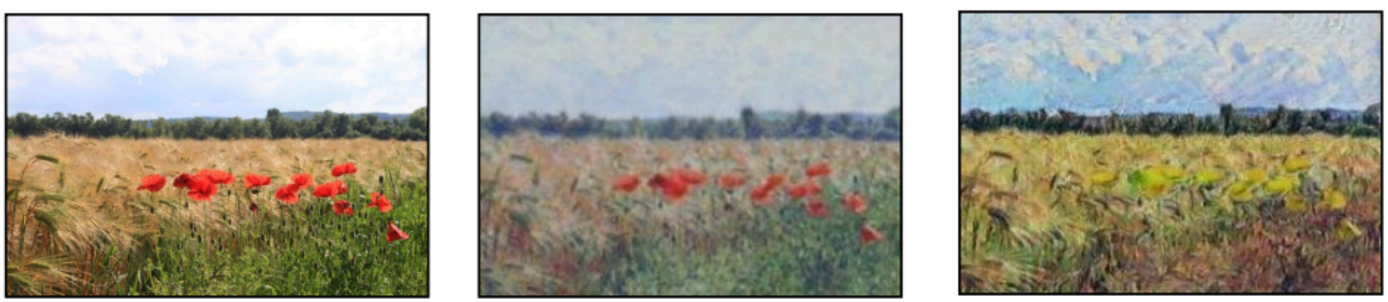
Distribution learning

BigGAN, Brock et al '18

Distribution learning

Conditional generative model P(zebra images| horse images)

Style Transfer



Input Image Monet Van Gogh

Image credits to Andrej Risteski

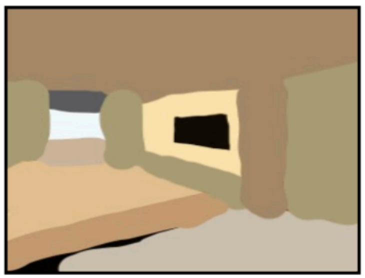
Distribution learning

Source actor

Target actor

Real-time reenactment

Generative model



Generative model
of realistic images

Generate

Stroke paintings to realistic images
[Meng, He, Song, et al., ICLR 2022]

“Ace of Pentacles”

Generative model
of paintings

Generate

Language-guided artwork creation
https://chainbreakers.kath.io @RiversHaveWings

Slides credit to Yang Song

Generative model

$P_\theta(x)$ large

High probability

Generative model of traffic signs

Low probability

Outlier detection

[Song et al., ICLR 2018]

Slide credit to Yang Song

Desiderata for generative models

- **Probability evaluation**: given a sample, it is computationally efficient to evaluate the probability of this sample.

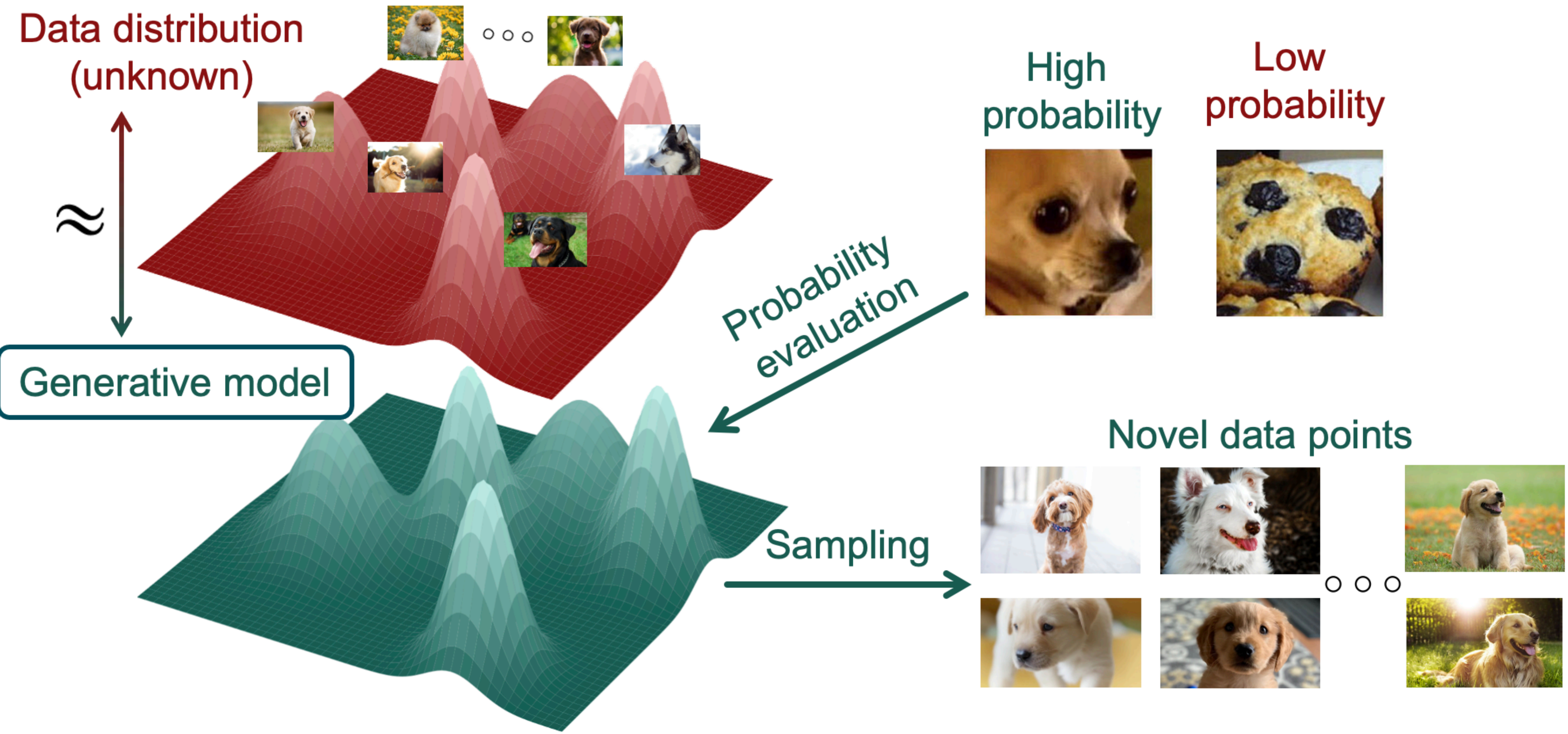
  x, p_θ, $p_\theta(x)$ efficiently

- **Flexible model family:** it is easy to incorporate any neural network models.

  p_θ: CNN, Transformer

- **Easy sampling:** it is computationally efficient to sample a data from the probabilistic model.

Desiderata for generative models



Slide credit to Yang Song

Taxonomy of generative models

$p_\theta(x)$

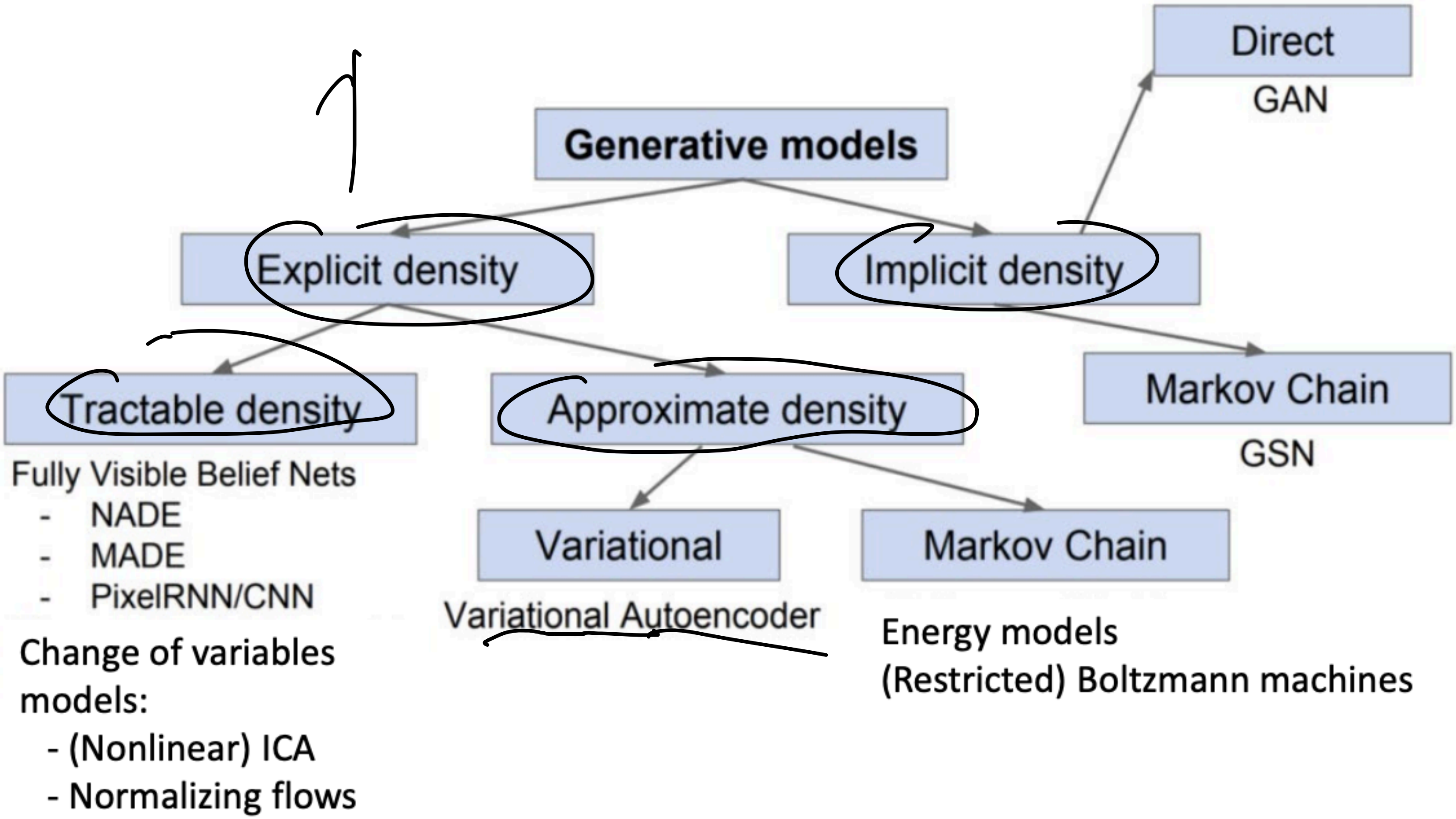
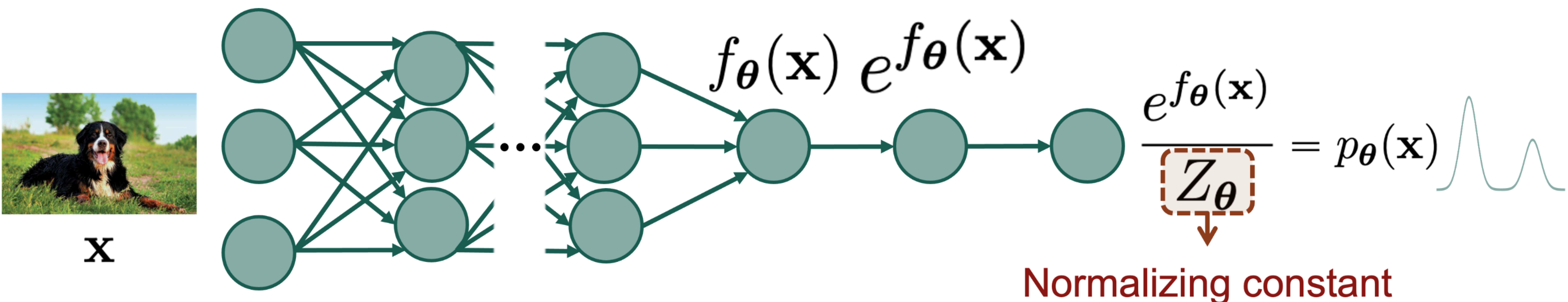


Image credits to Andrej Risteski

Key challenge for building generative models

distribution $\begin{cases} p_\theta(x) \geq 0 \\ \int_x p_\theta(x)\,dx = 1 \end{cases}$



$$\int_x e^{f_\theta(x)}\,dx$$

Slide credit to Yang Song

Key challenge for building generative models

Approximating the normalizing constant

- Variational auto-encoders [Kingma & Welling 2014, Rezende et al. 2014]
- Energy-based models [Ackley et al. 1985, LeCun et al. 2006]

✕ Inaccurate probability evaluation

Using restricted neural network models

- Autoregressive models [Bengio & Bengio 2000, van den Oord et al. 2016]
- Normalizing flow models [Dinh et al. 2014, Rezende & Mohamed 2015]

✕ Restricted model family

Generative adversarial networks (GANs)

- Model the generation process, not the probability distribution [Goodfellow et al. 2014]

✕ Cannot evaluate probabilities

Slide credit to Yang Song

Training generative models

- **Likelihood-based:** maximize the likelihood of the data under the model (possibly using advanced techniques such as variational method or MCMC):

$$\max_{\theta} \sum_{i=1}^{n} \log p_{\theta}(x_i)$$

- Pros:
  - **Easy training**: can just maximize via SGD.
  - **Evaluation**: evaluating the fit of the model can be done by evaluating the likelihood (on test data).

- Cons:
  - **Large models needed**: likelihood objectve is hard, to fit well need very big model.
  - **Likelihood entourages averaging:** produced samples tend to be blurrier, as likelihood encourages "coverage" of training data.

Training generative models

- **Likelihood-free:** use a **surrogate loss** (e.g., GAN) to train a discriminator to differentiate real and generated samples.

- Pros:
  - **Better objective, smaller models needed:** objective itself is learned - can result in visually better images with smaller models.

- Cons:
  - **Unstable training:** typically min-max (saddle point) problems.
  - **Evaluation:** no way to evaluate the quality of fit.

Generative Adversarial Nets

Implicit Generative Model

- **Goal:** a sampler $g(\cdot)$ to generate images
- A simple generator $g(z;\theta)$:
  - $z \sim N(0,I)$
  - $x = g(z;\theta)$ deterministic transformation

(1) Kullback-Leibler divergence (KL)
(2) Total variation
(3) Wasserstein distance
(4) Jensen-Shannon Divergence
(5) Integral Probability Metric (IPM)

- Likelihood-free training:
  - Given a dataset from some distribution p_{data}
  - Goal: $g(z;\theta)$ defines a distribution, we want this distribution $\approx p_{data}$
  - Training: minimize $D(g(z;\theta), p_{data})$
    - D is some distance metric (not likelihood)
  - Key idea: ***Learn* a differentiable** D

GAN (Goodfellow et al., '14)

- Parameterize the discriminator $D(\cdot\,;\phi)$ with parameter ϕ

- **Goal:** learn ϕ such that $D(x;\phi)$ measures how likely x is from p_{data}
  - $D(x,\phi) = 1$ if $x \sim p_{data}$
  - $D(x,\phi) = 0$ if $x ! \sim p_{data}$
  - a.k.a., a binary classifier
- GAN: use a neural network for $D(\cdot\,;\phi)$

$\{x_1, \ldots x_n\} \overset{i.i.d.}{\sim} p_{data}$

- **Training:** need both negative and positive samples
  - Positive samples: just the training data
  - Negative samples: use our sampler $g(z;\theta)$ (can provide infinite samples).

- **Overall objectives:**
  - Generator: $\theta^* = \max_\theta D(g(z;\theta);\phi)$
  - Discriminator uses MLE Training:

$$\phi^* = \max_\phi \mathbb{E}_{x\sim p_{data}}[\log D(x;\phi)] + \mathbb{E}_{\hat{x}\sim g(\cdot)}[\log(1 - D(\hat{x};\phi))]$$

GAN (Goodfellow et al., '14)

- Generator $g(z;\theta)$ where $z \sim N(0,I)$
  - Generate realistic data
- Discriminator $D(x;\phi)$
  - Classify whether the data is real (from p_{data}) or fake (from g)
- Objective function:

$$L(\theta,\phi) = \min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{data}} \left[\log D(x;\phi)\right] + \mathbb{E}_{\hat{x} \sim g(z,\theta)} \left[\log(1 - D(\hat{x};\phi))\right]$$

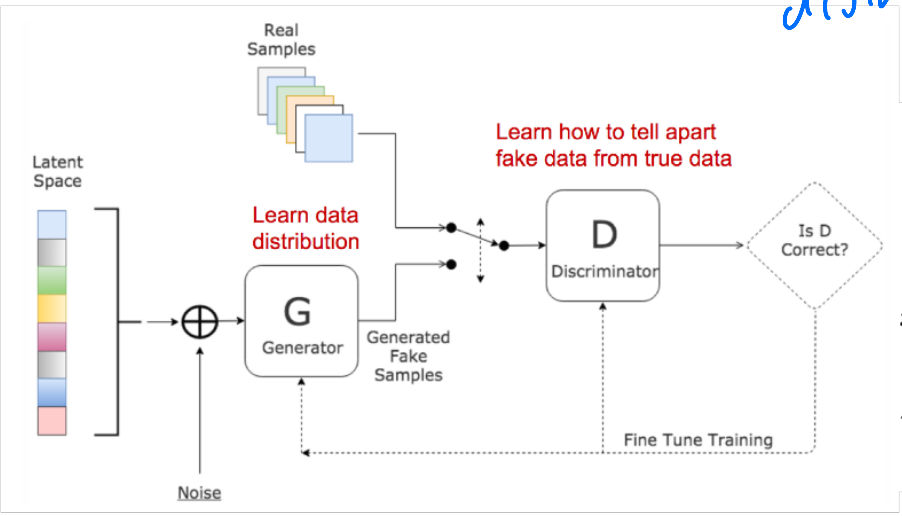
+ label −

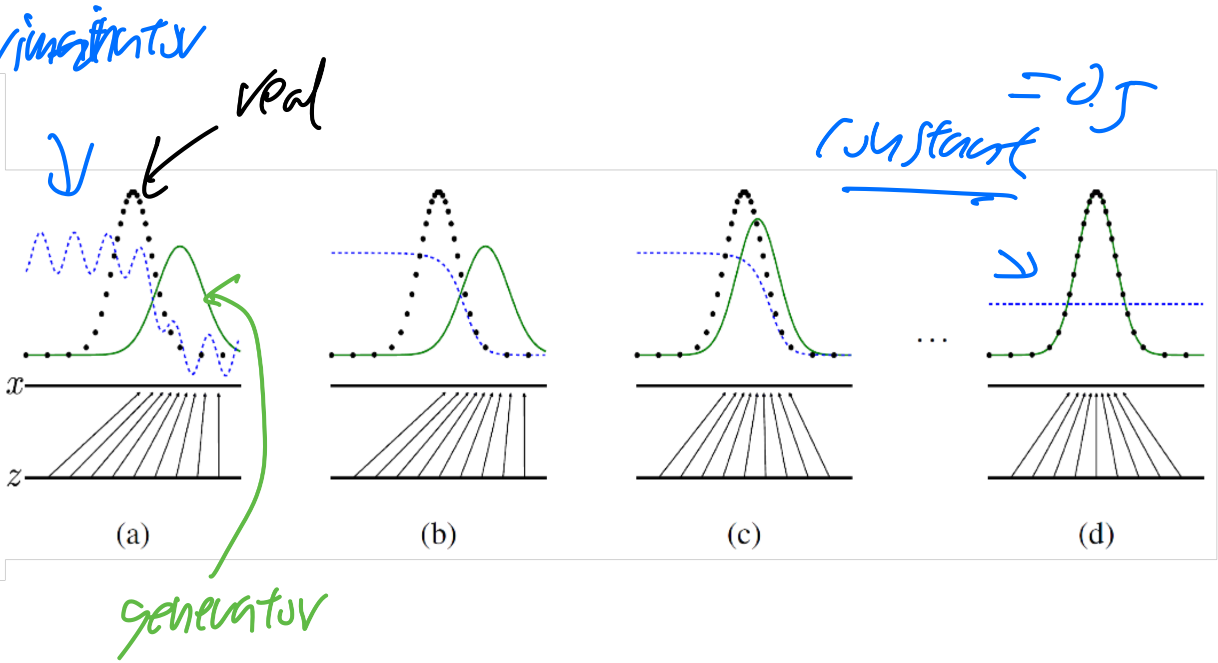
- Training procedure:
  - ~~Collect~~ dataset $\{(x,1) \mid x \sim p_{data}\} \cup \{(\hat{x},0) \sim g(z;\theta)\}$
  - Train discriminator $D : L(\phi) = \mathbb{E}_{x \sim p_{data}} \left[\log D(x;\phi)\right] + \mathbb{E}_{\hat{x} \sim g(z,\theta)} \left[\log(1 - D(\hat{x};\phi))\right]$
  - Train generator $g : L(\theta) = \mathbb{E}_{z \sim N(0,I)} \left[\log D(g(z;\theta),\phi)\right]$
  - Repeat

GAN (Goodfellow et al., '14)

- Objective function:

$$L(\theta, \phi) = \min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{data}} \left[\log D(x; \phi)\right] + \mathbb{E}_{\hat{x} \sim G} \left[\log(1 - D(\hat{x}; \phi))\right]$$





(a) (b) (c) (d)