Generative Models

GAN

Generative Adversarial Nets

Implicit Generative Model

$p_\theta(x)$

- **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

- Likelihood-free training:
  - Given a dataset from some distribution p_{data} $\{x_1, \ldots, x_N\}$
  - 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**

Metrics:
1. KL-divergence
2. Total variation
3. wasserstein dist

$\left(\frac{1}{2}\right)^d$

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 $D(\cdot\,; \phi)$

true $\{(x_1, 1), (x_2, 1) \ldots, (x_N, 1)\}$
artificial $\{(x_1', 0), (x_2', 0) \ldots, (x_N', 0)\}$

- **Training:** need both negative and positive samples
  - Positive samples: just the training data
  - Negative samples: use our sampler $g(\cdot\,; z)$ (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:

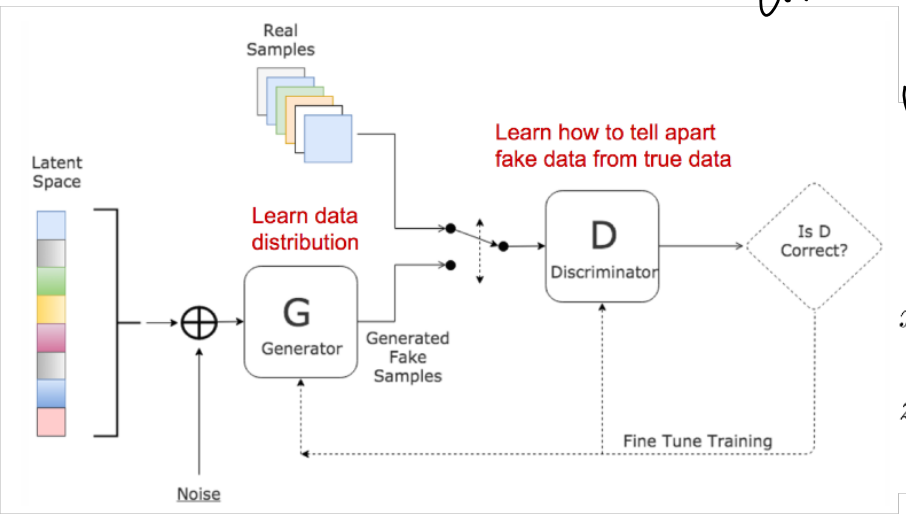
$$L(\theta,\phi) = \min_{\theta} \max_{\phi} \mathbb{E}_{x\sim p_{data}} \left[\log D(x;\phi)\right] + \mathbb{E}_{\hat{x}\sim G} \left[\log(1 - D(\hat{x};\phi))\right]$$

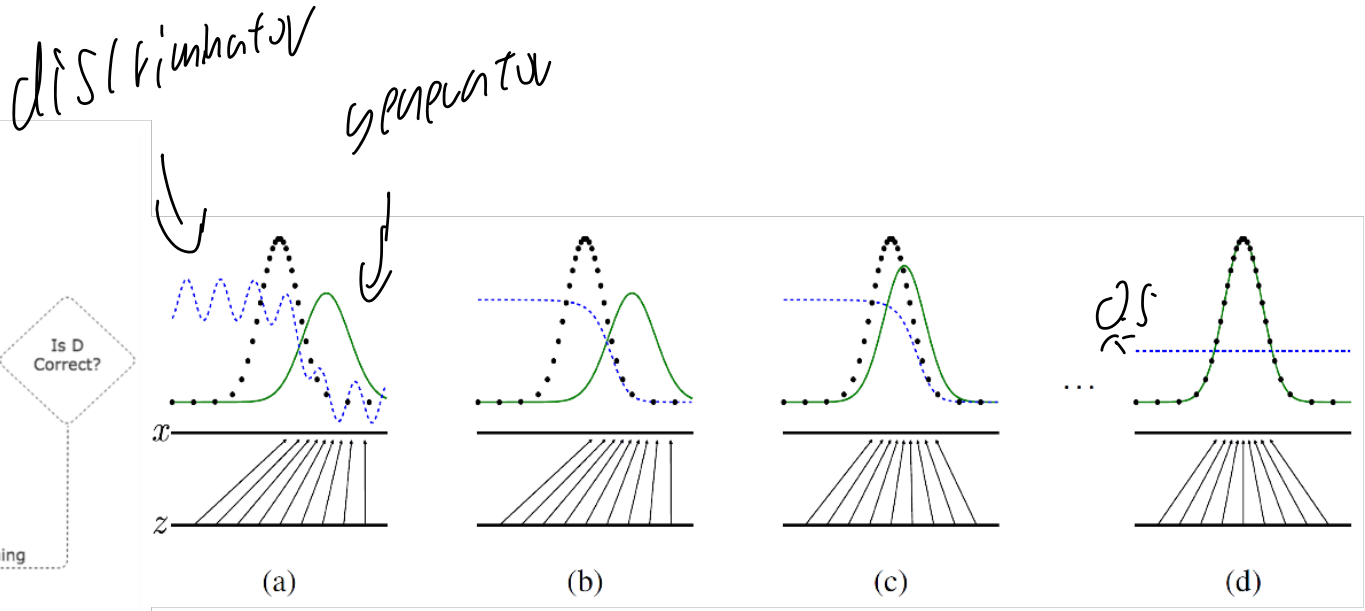
- Training procedure:
  - Collect dataset $\{(x,1) \,|\, x \sim p_{data}\} \cup \{(\hat{x},0) \sim g(z;\theta)\}$
  - Train discriminator
    $D : L(\phi) = \mathbb{E}_{x\sim p_{data}} \left[\log D(x;\phi)\right] + \mathbb{E}_{\hat{x}\sim G} \left[\log(1 - D(\hat{x};\phi))\right]$
  - Train generator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

GAN (Goodfellow et al., '14)

- Objective function:

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





Math Behind GAN

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

Let D^*, g^* be the solution to

- Optimal D^*, for a given X

$$L_X(D) = p_{data}(x) \cdot \log(D(x)) + p_g(x) \cdot \log(1 - D(x))$$

$$\Rightarrow \frac{\partial L}{\partial D} = 0, \text{ first-order condition}$$

$$\Rightarrow \frac{p_{data}(x)}{D^*(x)} - \frac{p_g(x)}{1 - D^*(x)} = 0$$

$$\Rightarrow D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

⊗ if $p_{data} = p_g \Rightarrow D^*(x) = 0.5$

Math Behind GAN

Consider optimal g^*, given optimal D^*

$$L(\theta,\phi) = \mathbb{E}_{x\sim p_{data}}\left[\log \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right] + \mathbb{E}_{x\sim g}\left[\log \frac{p_g(x)}{p_{data}(x)+p_g(x)}\right]$$

$$= \mathbb{E}_{x\sim p_{data}}\left[\log \frac{p_{data}(x)}{\left(\frac{p_{data}(x)+p_g(x)}{2}\right)}\right] - \log 2$$

$$+ \mathbb{E}_{x\sim g}\left[\log \frac{p_g(x)}{\left(\frac{p_{data}(x)+p_g(x)}{2}\right)}\right] - \log 2$$

$$= KL\left(p_{data} \,\middle|\, \frac{1}{2}(p_{data}+p_g)\right) + KL\left(p_g \,\middle|\, \frac{1}{2}(p_{data}+p_g)\right) - \log 4$$

$KL(p\|q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right]$

2. Jensen Shannon Divergence

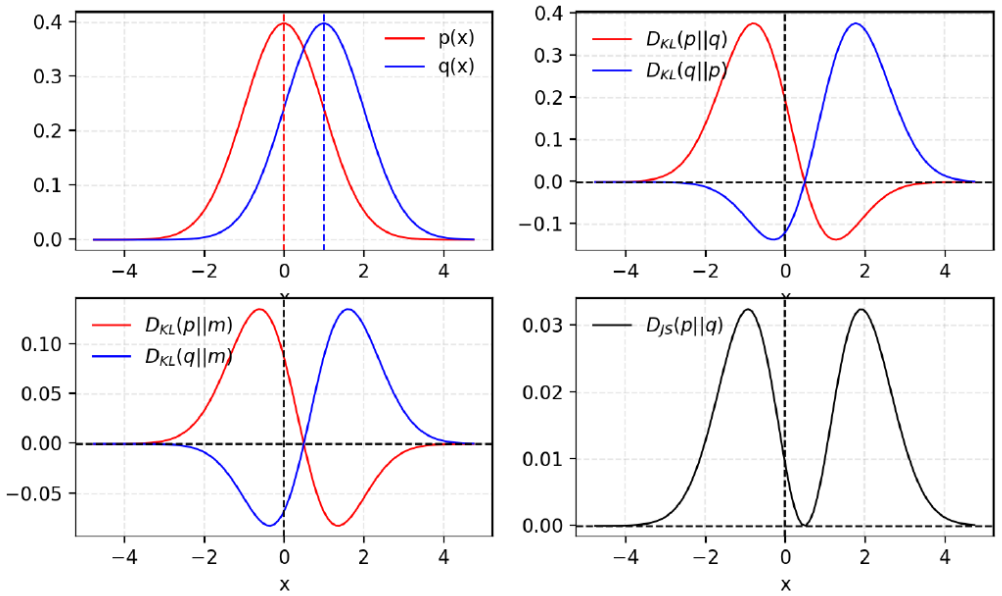
$JSD(p_{data}, p_g)$

KL-Divergence and JS-Divergence

- $JSD \geq 0$

if $JSD(p||q) = 0$

$\Leftrightarrow p = q$



Math Behind GAN

$\Rightarrow$ Given D^*

$$\min_G L(G) = 2 \cdot JSD(P_g \| P_{data}) - \log 4$$

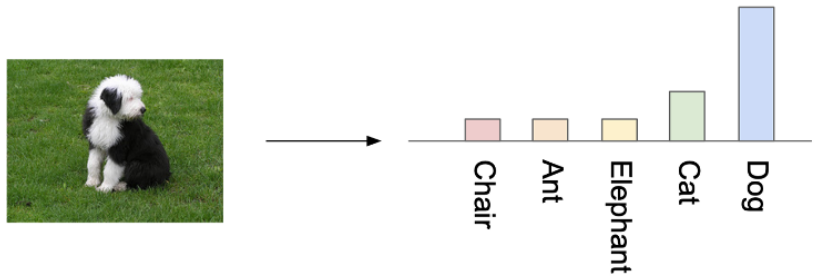
if G^* is optimal

$\Rightarrow$ $P_{G^*} = P_{data}$

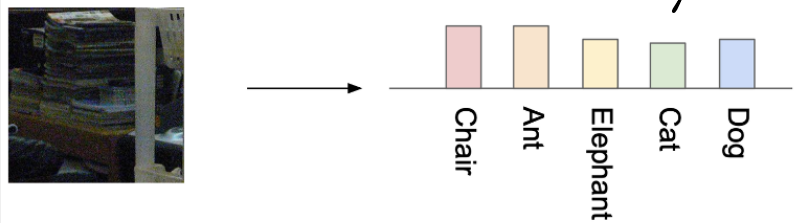
Evaluation of GAN

- No $p(x)$ in GAN.
- Idea: use a trained classifier $f(y \mid x)$:
- If $x \sim p_{data}$, $f(y \mid x)$ should have low entropy
  - Otherwise, $f(y \mid x)$ close to uniform.
- Samples from G should be diverse:
  - $p_f(y) = \mathbb{E}_{x \sim G}[f(y \mid x)]$ close to uniform.

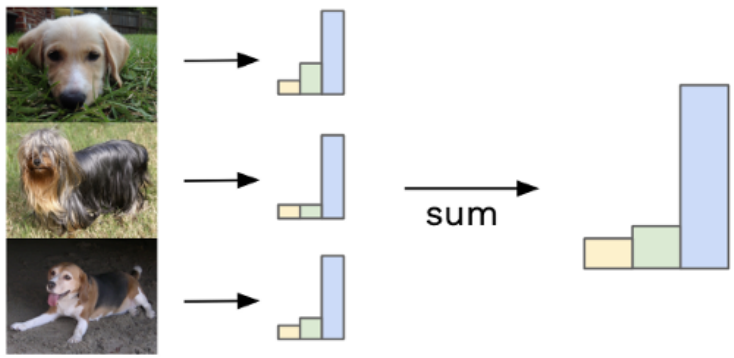
low entropy



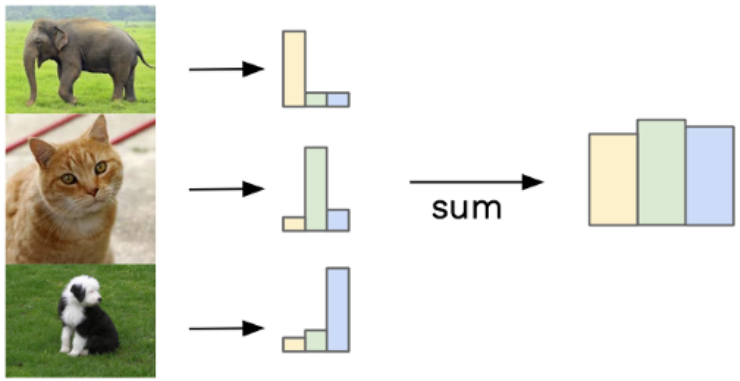
high entropy



Similar labels sum to give focussed distribution



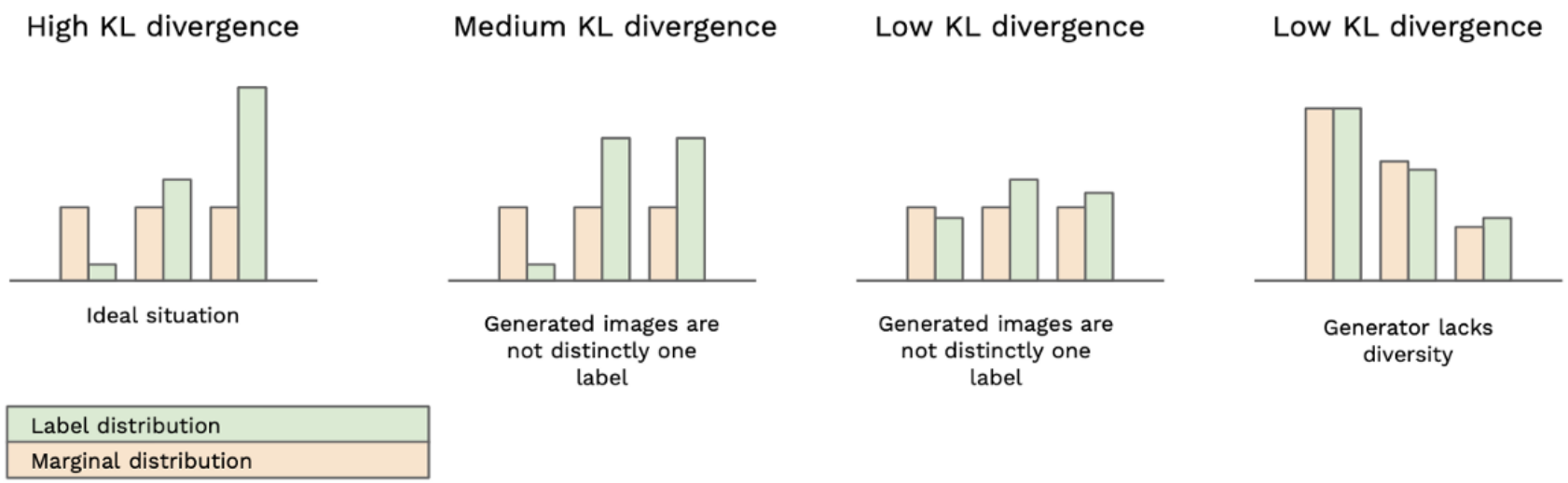
Different labels sum to give uniform distribution



Evaluation of GAN

- **Inception Score** (IS, Salimans et al. '16)
  - Use Inception V3 trained on ImageNet as $f(y \mid x)$
  - $IS = \exp\left(\mathbb{E}_{x \sim G}\left[KL(f(y \mid x) \,||\, p_f(y))\right]\right)$
  - Higher the better

individual marginal



Comments on GAN

- Other evaluation metrics:
  - Fréchet Inception Distance (FID): Wasserstein distance between Gaussians

- Mode collapse:
  - The generator only generate a few type of samples.
  - Or keep oscillating over a few modes.

- Training instability:
  - Discriminator and generator may keep oscillating
  - Example: $-xy$, generator x, discriminator y. NE: $x = y = 0$ but GD oscillates.
  - No stopping criteria.
  - Use Wsserstein GAN (Arjovsky et al. '17):

    $$\min_{G} \max_{f:\mathrm{Lip}(f)\leq 1} \mathbb{E}_{x\sim p_{data}}\left[f(x)\right] - \mathbb{E}_{\hat{x}\sim p_G}[f(\hat{x})]$$

  - And need many other tricks…

Energy-Based Models

Energy-based Models

- Goal of generative models:
  - a probability distribution of data: $P(x)$

- Requirements
  - $P(x) \geq 0$ (non-negative)
  - $\int_x P(x)dx = 1$

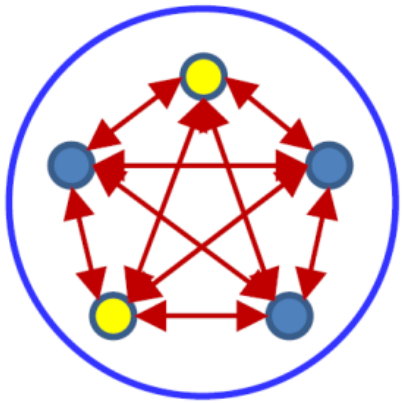
- Energy-based model:
  - Energy function: $E(x;\theta)$, parameterized by θ
  - $P(x) = \frac{1}{z}\exp(-E(x;\theta))$ (why exp?)
  - $z = \int_x \exp(-E(x;\theta))dx$

Boltzmann Machine

$y \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

- Generative model
  - $E(y) = -\frac{1}{2} y^\top W y$
  - $P(y) = \frac{1}{z} \exp(-\frac{E(y)}{T})$, T: temperature hyper-parameter
  - W: parameter to learn
- When y_i is binary, patterns are affecting each other through W

$y_1, \cdots, y_i, \cdots, y_d$



$$z_i = \frac{1}{T} \sum_j w_{ji} s_j$$

$s = y$

$$P(s_i = 1 | s_{j \neq i}) = \frac{1}{1 + e^{-z_i}}$$

Boltzmann Machine: Training

$y \in \{0,1\}^d$

$y_i \in \{-1,1\}$

- Objective: maximum likelihood learning (assume T =1):
  - Probability of one sample:

$$P(y) = \frac{\exp(\frac{1}{2} y^\top W y)}{\sum_{y'} \exp(y'^\top W y')}$$

2^d values of y

  - Maximum log-likelihood:

$$\max_W \quad L(W) = \frac{1}{N} \sum_{y \in D} \frac{1}{2} y^\top W y - \log \sum_{y'} \exp(\frac{1}{2} y'^\top W y')$$

$D = \{y^1, \ldots y^N\}$

$W : d \times d$

Boltzmann Machine: Training

$$L(W) = \frac{1}{N} \sum_{y \in D} \frac{1}{2} y^T W y - \log \sum_{y'} \exp\left(\frac{1}{2} y'^T W y'\right)$$

$$\nabla_{w_{ij}} L = \frac{1}{N} \sum_{y \in D} y_i y_j - \sum_{y'} \frac{\exp\left(\frac{1}{2} y'^T W y'\right) y_i' y_j'}{\sum_{y''} \exp\left(\frac{1}{2} y''^T W y''\right)}$$

$P(y')$

$\mathbb{E}_{y'}[y_i' \cdot y_j']$

S = sample $\{y^1, y^2, \ldots, y^N\}$

$$\Rightarrow \approx \frac{1}{N} \sum_{y \in D} y_i y_j - \frac{1}{N} \sum_{y' \in S} y_i' \cdot y_j'$$

Boltzmann Machine: Training

Sampling : MCMC

initial $y(0) \in \{0,1\}^d$

for $t = 1, \ldots T$

iterate $j \in \{1, \ldots, d\}$ conditional sampling

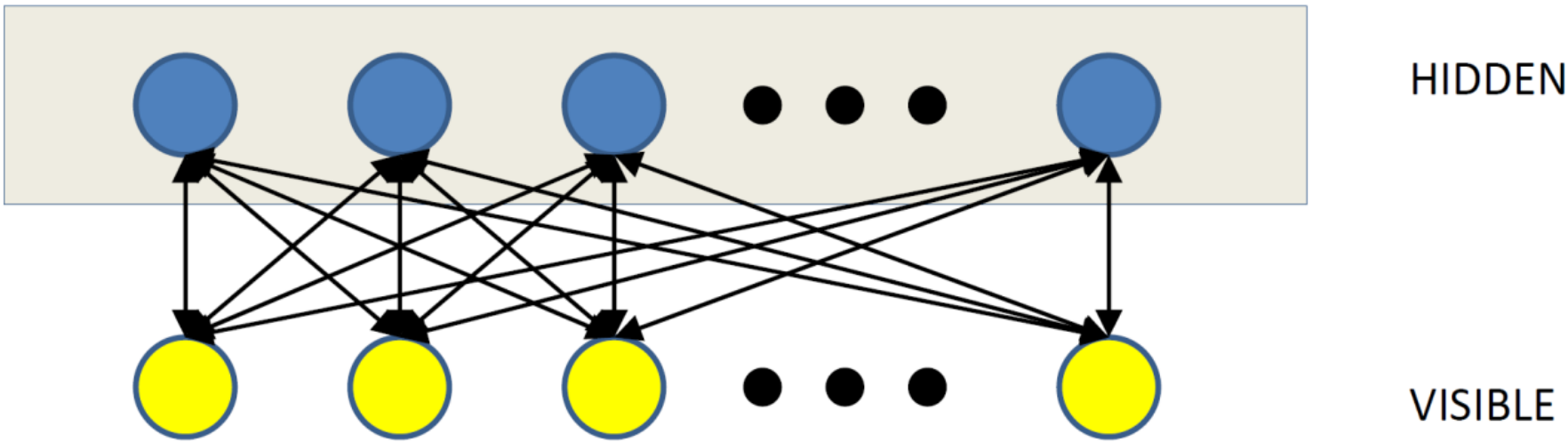
$y_j(t) \sim P(\cdot \mid y_{\neq j}(t-1))$

Theorem $T \to \infty$ $\{y(1), \ldots y(T)\}$

$\longrightarrow$ generator

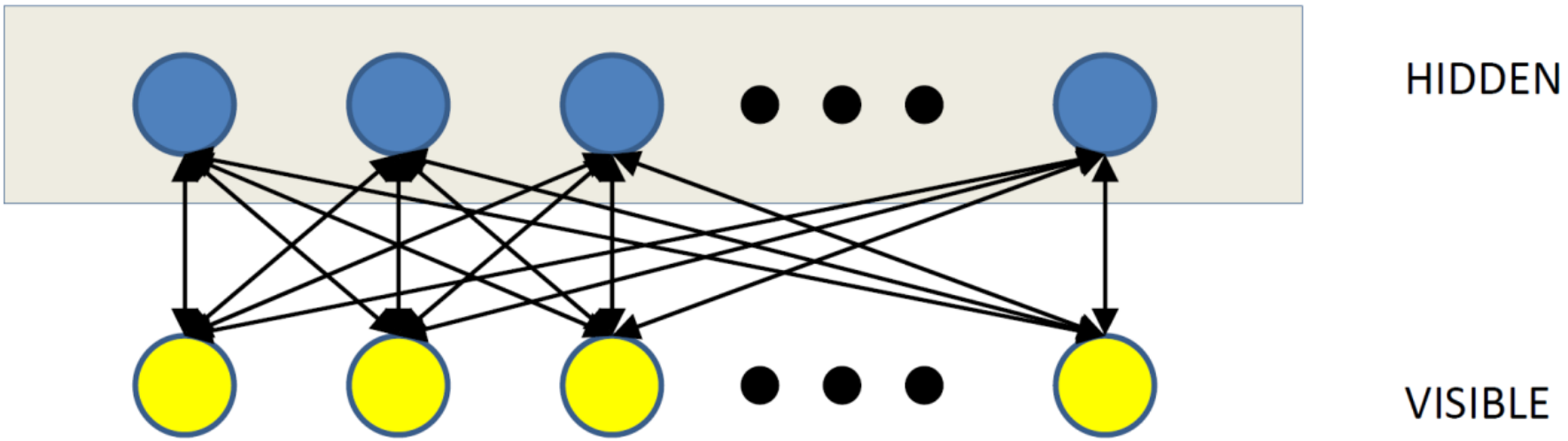
Restricted Bolzmann Machine

- A structured Boltzmann Machine
  - Hidden neurons are only connected to visible neurons
  - No intra-layer connections
  - Invented by Paul Smolensky in ’89
  - Became more practical after Hinton invested fast learning algorithms in mid 2000



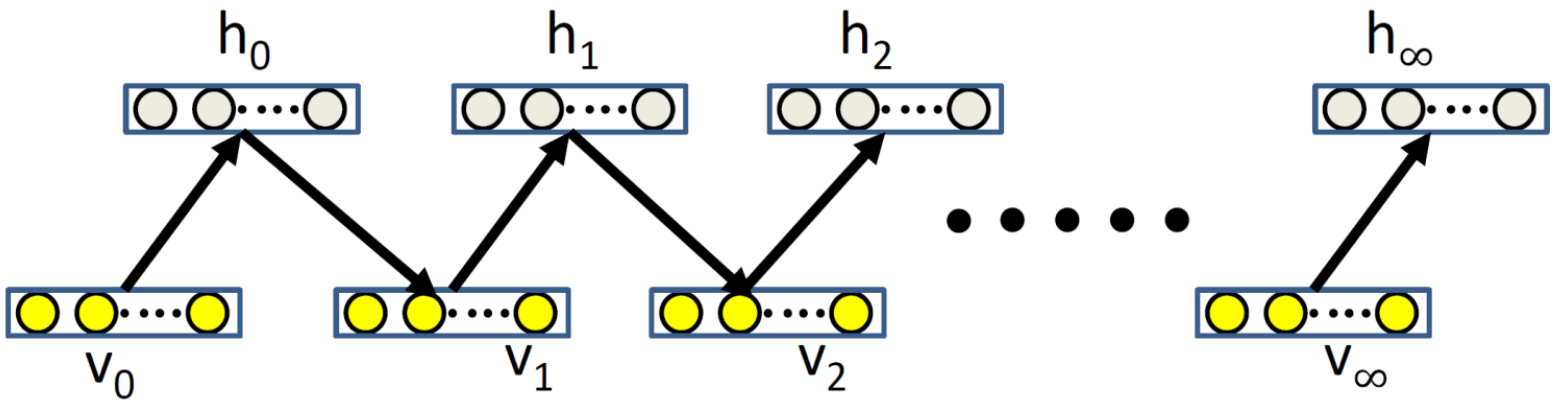
Restricted Bolzmann Machine

- Computation Rules
  - Iterative sampling
  - Hidden neurons h_i: $z_i = \sum_j w_{ij} v_j$, $P(h_i \mid v) = \dfrac{1}{1 + \exp(-z_i)}$
  - Visible neurons v_j: $z_j = \sum_i w_{ij} h_i$, $P(v_j \mid h) = \dfrac{1}{1 + \exp(-z_j)}$



Restricted Bolzmann Machine

- Sampling:
  - Randomly initialize visible neurons v_0
  - Iterative sampling between hidden neurons and visible neurons
  - Get final sample (v_∞, h_∞)

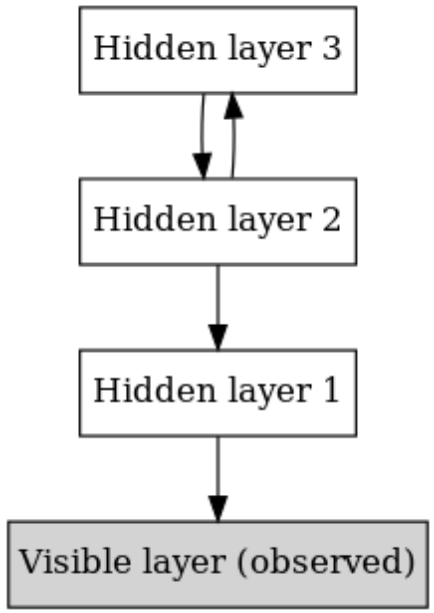


Deep Bolzmann Machine

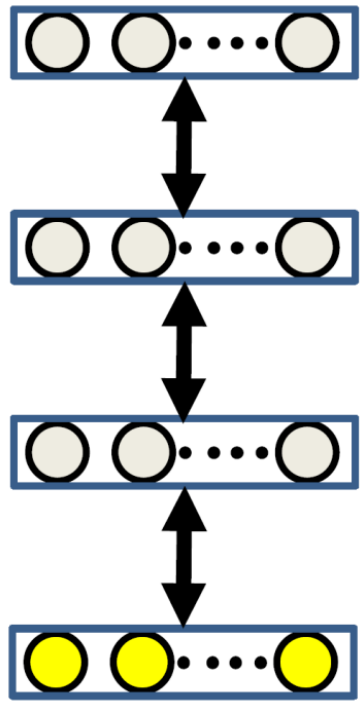
- Can we have a **deep** version of RBM?
  - Deep Belief Net (’06)
  - Deep Boltzmann Machine (’09)

- Sampling?
  - Forward pass: bottom-up
  - Backward pass: top-down

- Deep Bolzmann Machine
  - The very first deep generative model
  - Salakhudinov & Hinton

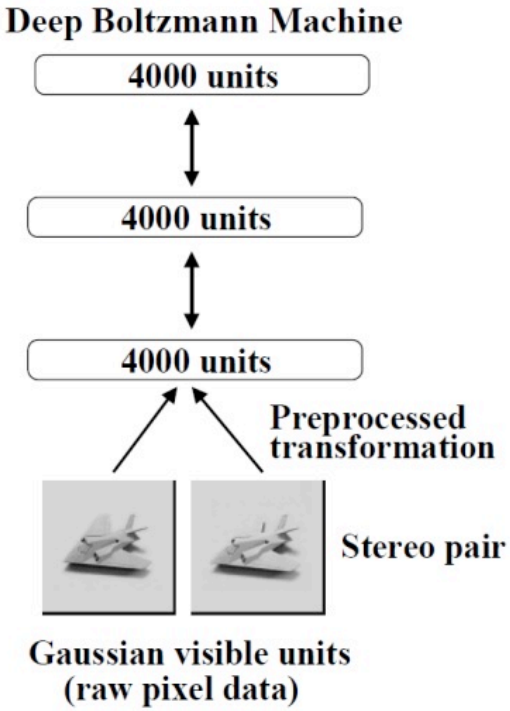


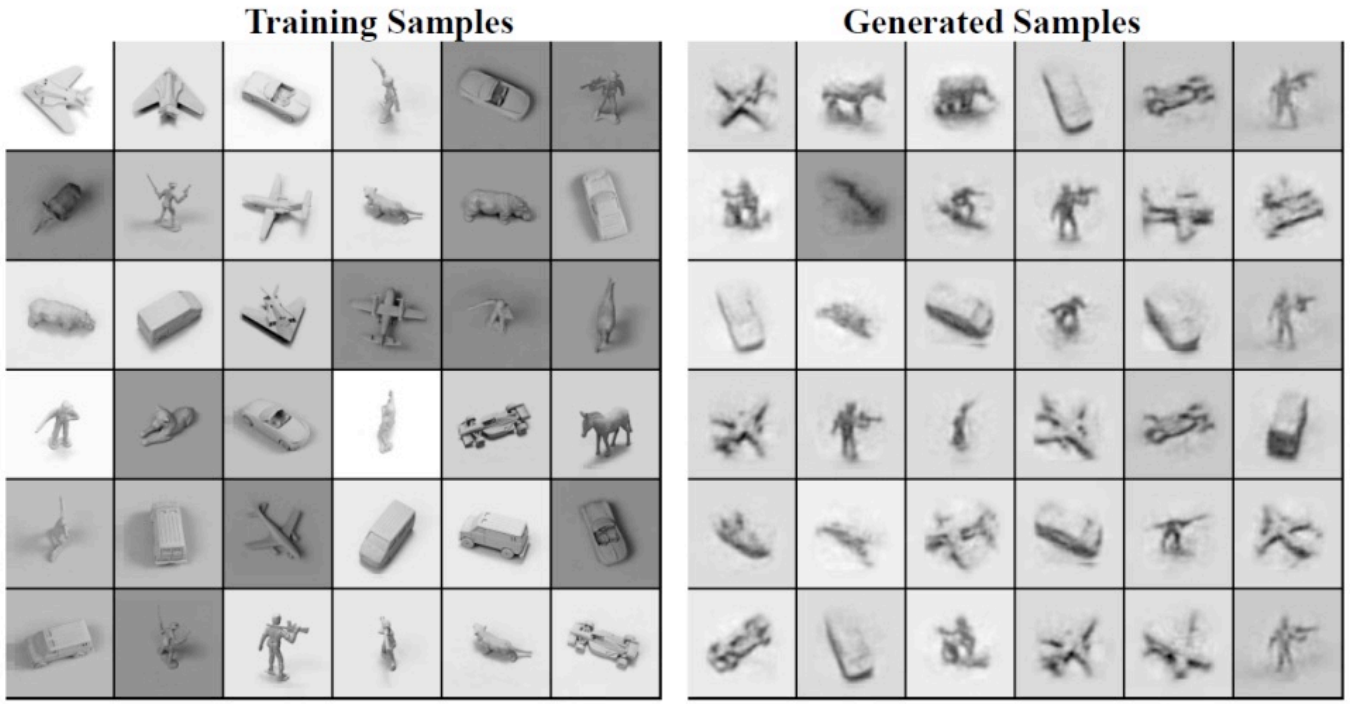
deep belief net



Deep Boltzmann Machine

Deep Bolzmann Machine





Summary

- Pros: powerful and flexible
  - An arbitrarily complex density function $p(x) = \frac{1}{z} \exp(-E(x))$
- Cons: hard to sample / train
  - Hard to sample:
    - MCMC sampling
  - Partition function
    - No closed-form calculation for likelihood
    - Cannot optimize MLE loss exactly
    - MCMC sampling