

# Inference and Representation

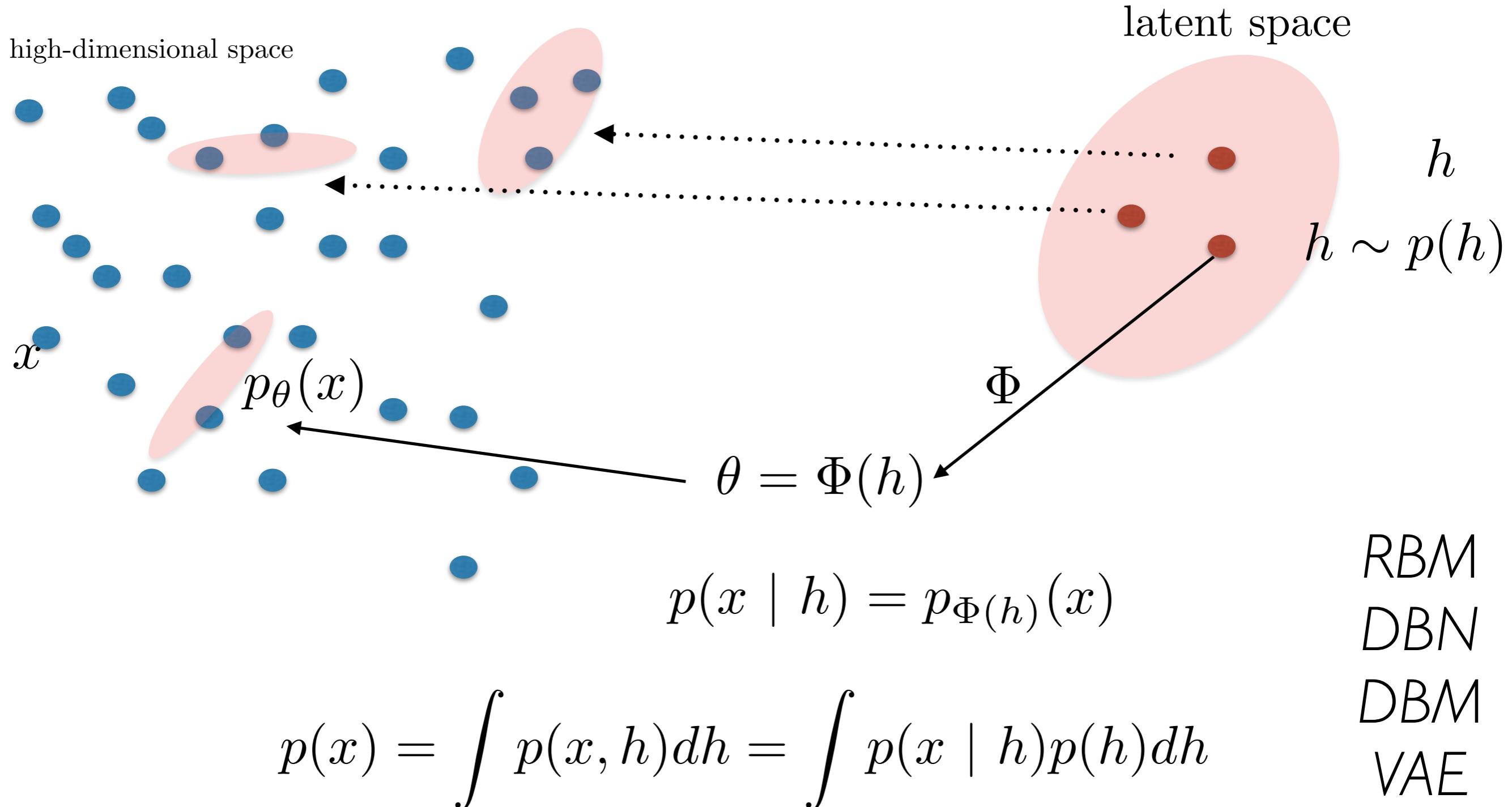
## Lecture 11

Joan Bruna  
Courant Institute, NYU



# Latent Graphical Models

- Latent Graphical Models or *Mixtures*.



Model: additive combination of simple parametric models

# Objectives lectures 11 and 12

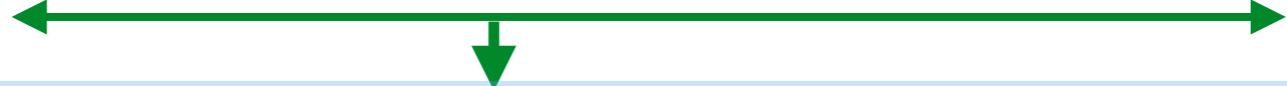
- Variational Flows
- Generative Adversarial Networks
- Autoregressive Models
- Open Research

# Variational Autoencoders

[Kingma & Welling'14, Rezende et al.'14]

- Recall the variational lower bound:

$$\log p(X \mid \theta) = \mathbb{E}_{q(z|\beta)} \{ \log(p(X, Z \mid \theta)) \} + H(q(z \mid \beta)) + D_{KL}(q(z|\beta) \parallel p(z|x, \theta))$$



$$\log p(X \mid \theta) = \mathcal{L}(\theta, \beta, X) + D_{KL}(q(z|\beta) \parallel p(z|X, \theta))$$

- Can we optimize jointly both generative and variational parameters efficiently?
- For appropriate posterior approximations, we can reparametrize samples as

$$Z \sim q(z|x, \beta) \Rightarrow Z \stackrel{d}{=} g_\beta(\epsilon, x) , \quad \epsilon \sim p_0$$

$$\left( \text{e.g. } q(z|x, \beta) = \mathcal{N}(z; \mu(x), \Sigma(x)) \leftrightarrow z = \mu(x) + \Sigma(x)^{1/2}\epsilon , \quad \epsilon \sim \mathcal{N}(0, 1) \right)$$

# Variational Autoencoders

- It results that

$$\mathcal{L}(\theta, \beta, X) = -D_{KL}(q_\beta(z|X)||p_\theta(z)) + \mathbb{E}_{q_\beta(z|X)}\{\log p(X|z, \theta)\}$$

can be estimated via Monte-Carlo by

$$\widehat{\mathcal{L}(\theta, \beta, X)} = -D_{KL}(q_\beta(z|X)||p_\theta(z)) + \frac{1}{S} \sum_{s \leq S} \log p(X|z^{(s)}, \theta)$$

$$z^{(s)} = g_\beta(X, \epsilon^{(s)}) \text{ and } \epsilon^{(s)} \sim p_0.$$

- First term acts as a regularizer: limits the capacity of the encoder
- Second term is a reconstruction error.

# Incorporate MCMC to posterior approx.

“Markov Chain Monte Carlo and Variational Inference: Bridging the Gap”, Salimans et al’15

- We saw in Lecture 7 how to use Markov Chains to approximate intractable posteriors.

$$p(z \mid x) \stackrel{d}{=} \lim_{T \rightarrow \infty} q_0(z_0 \mid x) \prod_{t < T} q(z_t \mid z_{t-1}, x) .$$

- For fixed  $T$ , this can be seen as another variational approximation, by considering  $y = z_1, \dots, z_{T-1}$  as extra hidden variables.
- The resulting Variational Lower bound becomes

$$\begin{aligned} \mathcal{L}_{MCMC} &= \mathcal{L} - \mathbb{E}_{q(z_T \mid x)} \{ D_{KL}(r(y|z_T, x) \parallel q(y \mid z_T, x)) \} \\ &\leq \mathcal{L} \leq \log p(x) . \end{aligned}$$

$r(y|x, z_T)$ : auxiliary variational approximation

- If we choose  $r$  to be an inverse Markov chain, we obtain

$$\mathcal{L}_{aux} = \mathbb{E}_q \{ \log p(x, z_T) - \log q(z_0|x) \} + \sum_{t=1}^T (\log r_t(z_{t-1}|x, z_t) - \log q_t(z_t|x, z_{t-1}))$$

# Variational inference with Importance Sampling

“Importance Weighted Autoencoders”

Burda et al’16

- Another mechanism to improve the variational lower bound is to use importance sampling.
- For each  $k$ , we define

$$\mathcal{L}_k(x) = \mathbb{E}_{z_1, \dots, z_k \sim q(z|x)} \left[ \log \frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right].$$

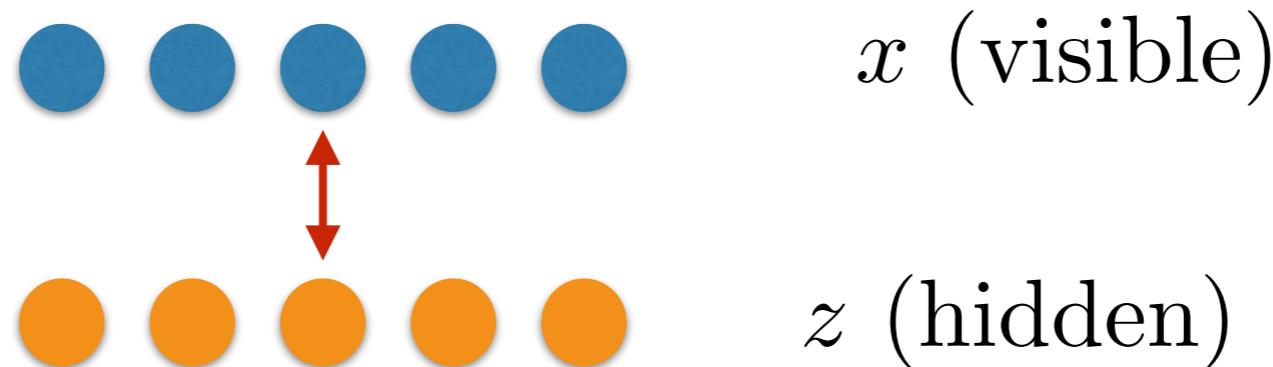
- It results that

$$\forall k, \log p(x) \geq \mathcal{L}_{k+1}(x) \geq \mathcal{L}_k(x), \text{ and}$$

$$\lim_{k \rightarrow \infty} \mathcal{L}_k(x) = \log p(x) \text{ if } \frac{p(x, z)}{q(z|x)} \text{ is bounded}.$$

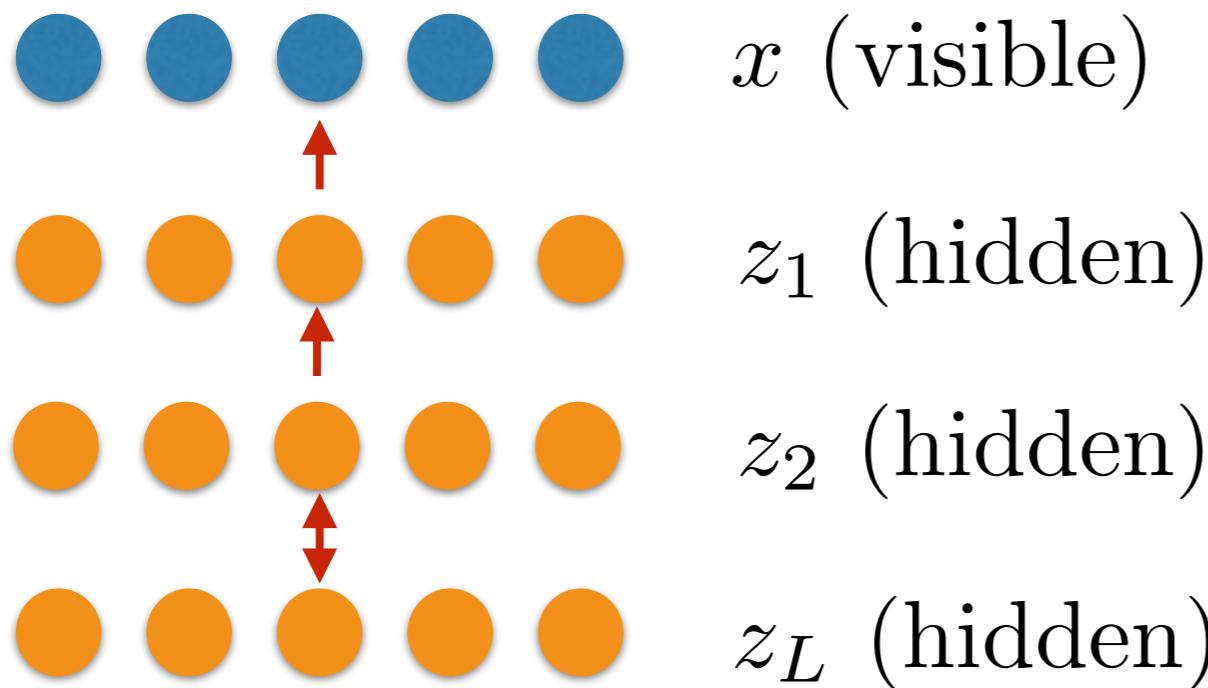
# Other directed models

- Restricted Boltzmann Machines [Smolenski'86, Hinton, '02] are undirected graphical models with binary variables



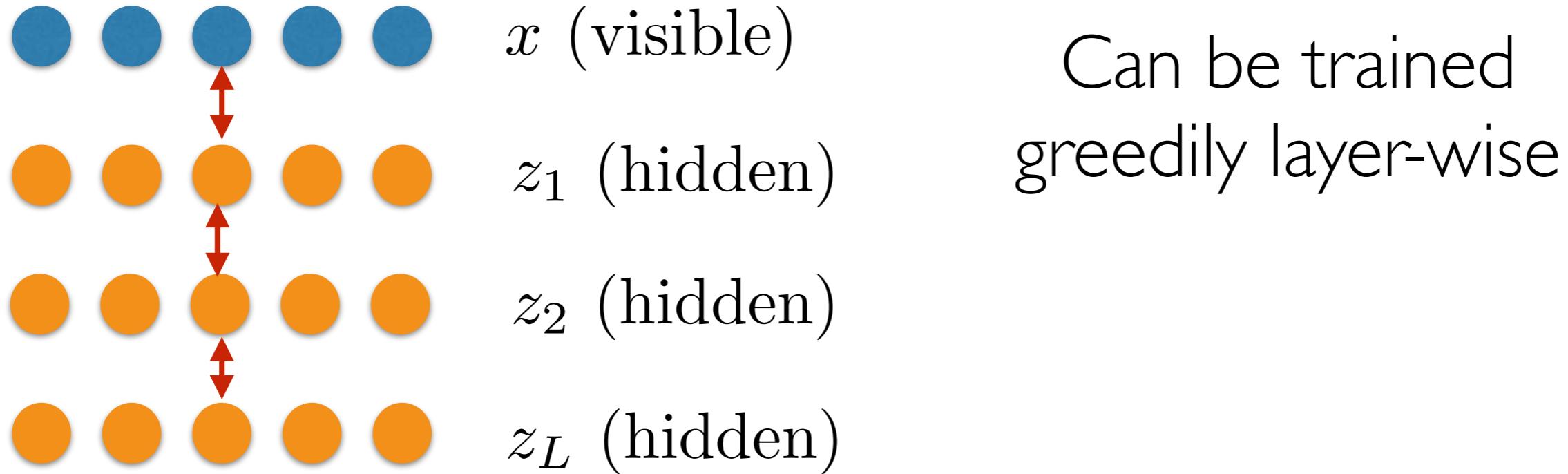
$$p(x, z) = \exp \left( \langle \theta_1, xz^T \rangle + \langle \theta_2, x \rangle + \langle \theta_3, z \rangle - \log A(\theta) \right)$$

- Deep Belief Networks [Hinton et al'02]



# Other directed models

- Deep Boltzmann Machines [Saladutnikov & Hinton, '09]



Can be trained  
greedily layer-wise

- See also:
  - Wake-Sleep [Hinton et al'95]
  - Generative Stochastic Networks [Bengio, '13].

•

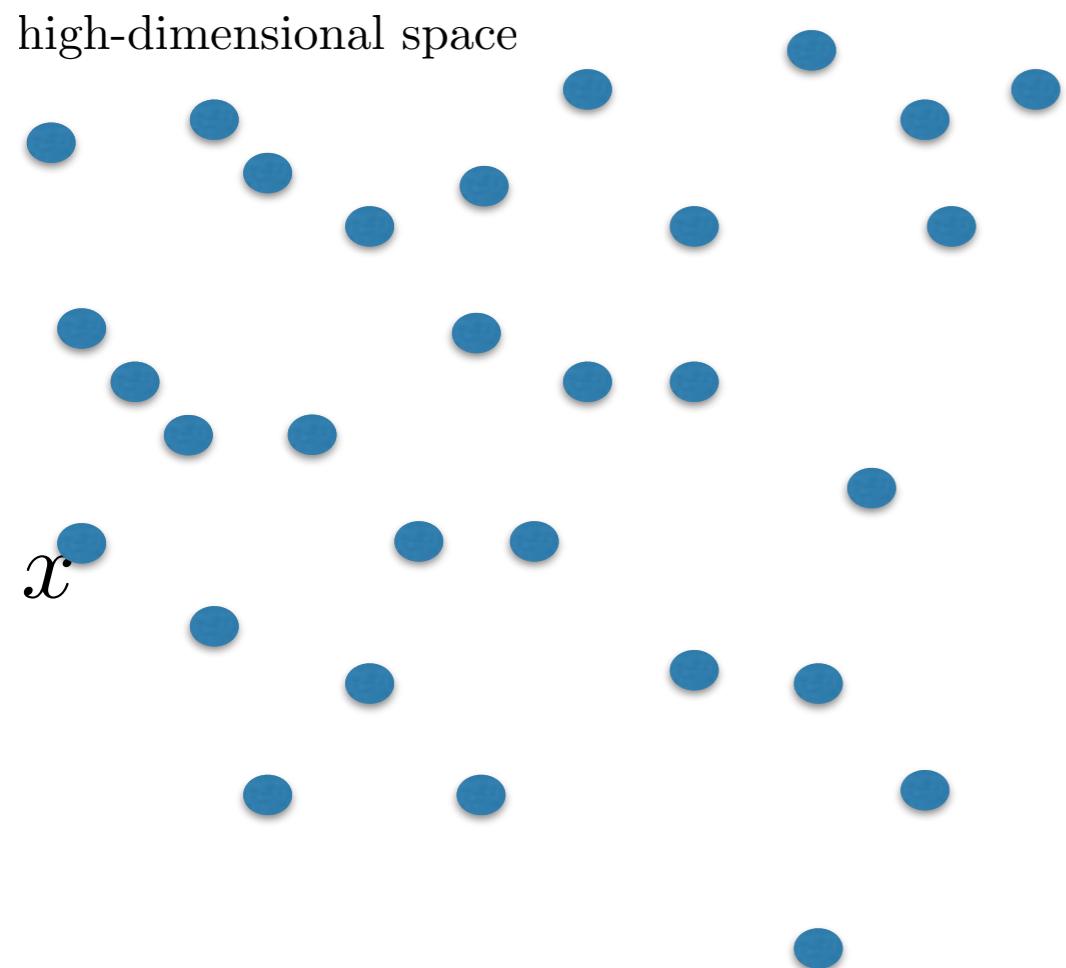
...

# Limits of Mixture Models

- Inference is computationally expensive for large models.
- The modeling  $p(x)$  is reduced to the task of modeling  $p(x|z)$
- Q: How to account for image variability?
  - $p(x|z) = \mathcal{N}(\Phi(z), \Sigma(z))$  corresponds to a model of *additive variability*:
$$x = \Phi(z) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \Sigma(z))$$
$$-\log p(x|z) \propto \|\Sigma(z)^{-1/2}(x - \Phi(z))\|^2$$
  - In particular, can we guarantee that  $|p(x_\tau) - p(x)| \lesssim \|\tau\|$  with a mixture model?
  - Gaussian likelihoods tend to suffer from regression to the mean.

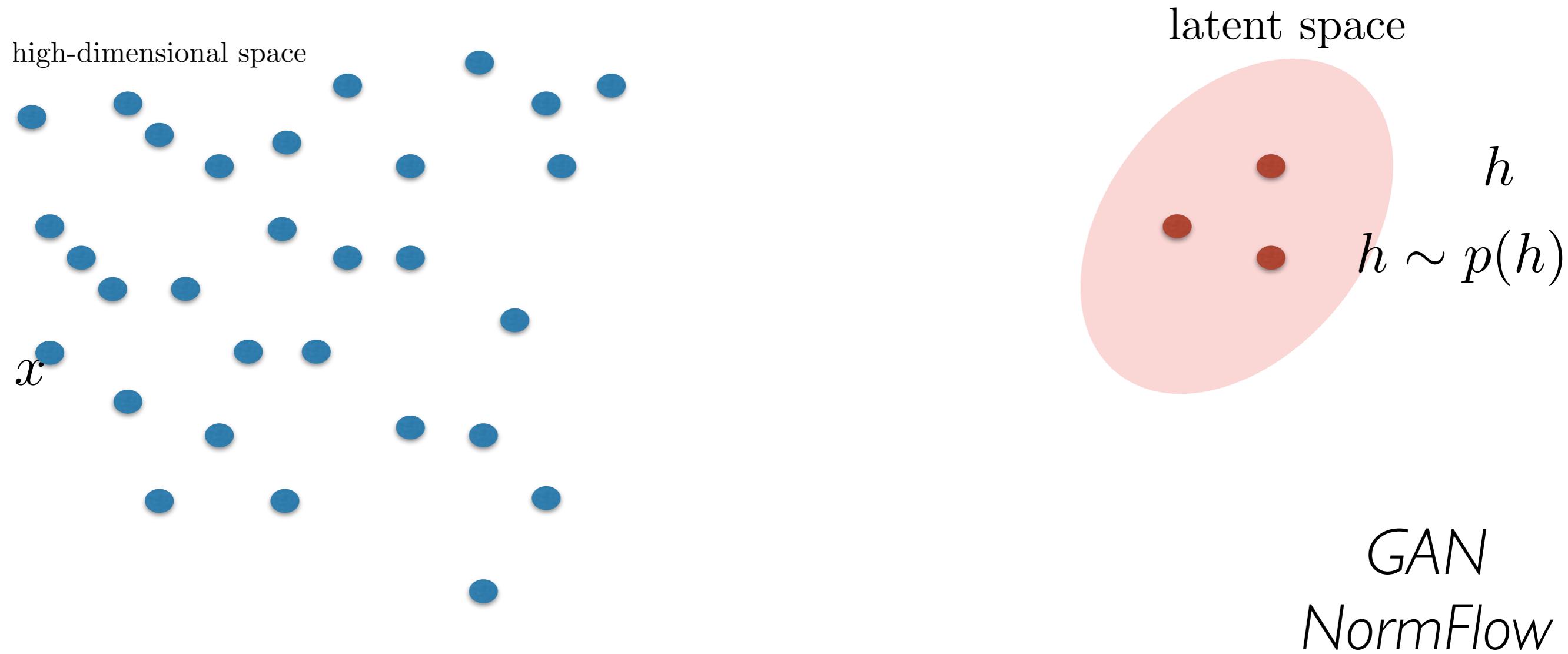
# Generative Models of Complex data

- Flows or Transports of Measure:



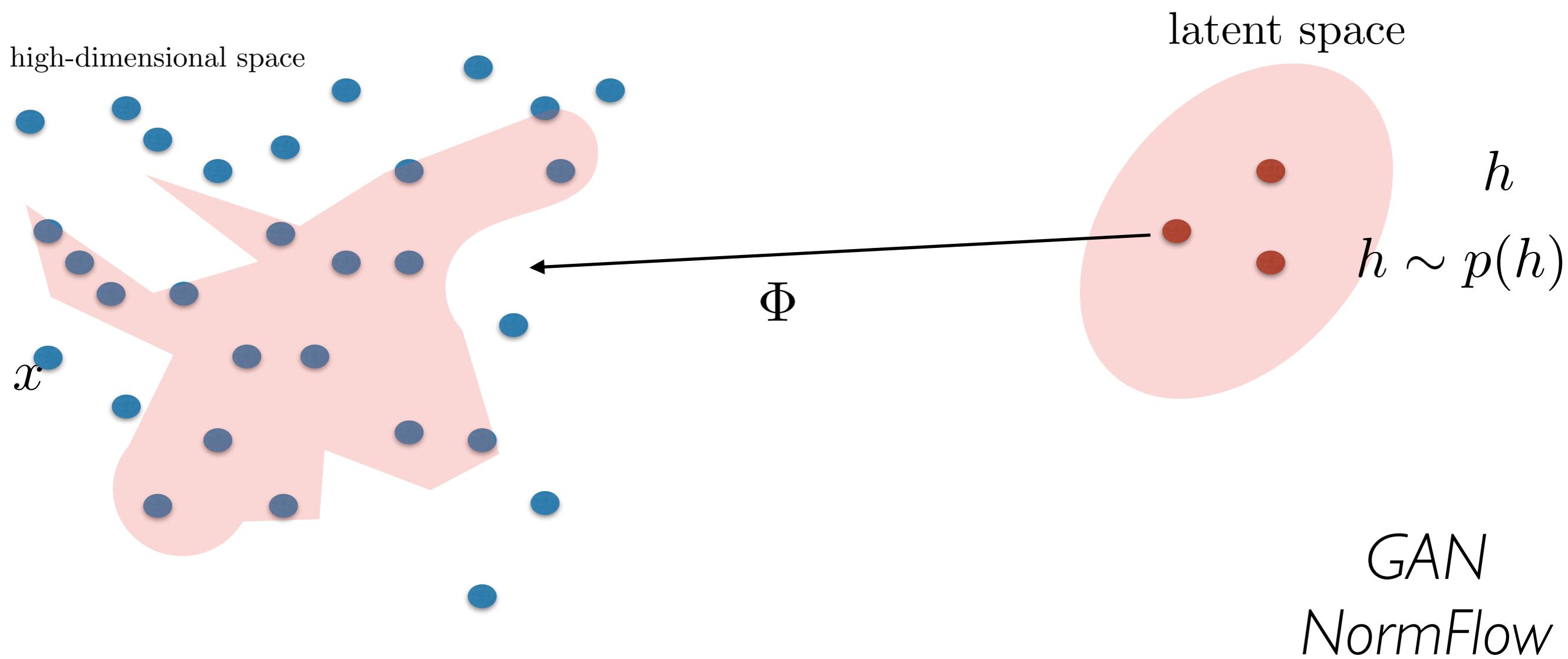
# Generative Models of Complex data

- Flows or Transports of Measure:



# Generative Models of Complex data

- Flows or Transports of Measure



$p(x)$  defined implicitly with

$$\int f(x)p(x)dx = \int f(\Phi(h))p(h)dh , \quad \forall f \text{ measurable}$$

...

GAN

NormFlow

# Measure Transports

- How to train the transport  $\Phi$ ?
- We will see two methods:
  - Directly by optimizing data log-likelihood [Normalizing Flows]
  - Using a Discriminative Model [Generative Adversarial Networks]

# Normalizing Flows

[Variational Inference with Normalizing Flows, Rezende & Mohamed'15]

- Consider a diffeomorphism  $\Phi : \mathbb{R}^N \rightarrow \mathbb{R}^N$ . [Tabak et al.'10]
- If  $Z \in \mathbb{R}^N$  is a random variable with density  $q(z)$ , what is the density of  $Z' = \Phi(Z)$  ?

# Normalizing Flows

[Variational Inference with Normalizing Flows, Rezende & Mohamed'15]

- Consider a diffeomorphism  $\Phi : \mathbb{R}^N \rightarrow \mathbb{R}^N$ . [Tabak et al.'10]
- If  $Z \in \mathbb{R}^N$  is a random variable with density  $q(z)$ , what is the density of  $Z' = \Phi(Z)$  ?
- We have, for any measurable  $f$ ,

$$\begin{aligned}\mathbb{E}_{z \sim q}(f(z')) &= \int f(z')q(z)dz \\ &= \int f(\Phi(z))q(z)dz = \int f(z)q(\Phi^{-1}(z))|\det(\nabla\Phi^{-1}(z))|dz \\ &= \int f(z)\tilde{q}(z)dz = \mathbb{E}_{z' \sim \tilde{q}}(f(z')) , \text{ with}\end{aligned}$$

$$\tilde{q}(z') = q(z) |\det \nabla\Phi(z)|^{-1} , \quad z = \Phi^{-1}(z') .$$

# Normalizing Flows

- The density  $q_K(z)$  obtained by transporting a base measure  $q_0$  through a cascade of  $K$  diffeomorphisms  $\Phi_1, \dots, \Phi_K$  is

$$z_K = \Phi_K \circ \dots \circ \Phi_1(z_0) , \text{ with } z_0 \sim q_0(z)$$

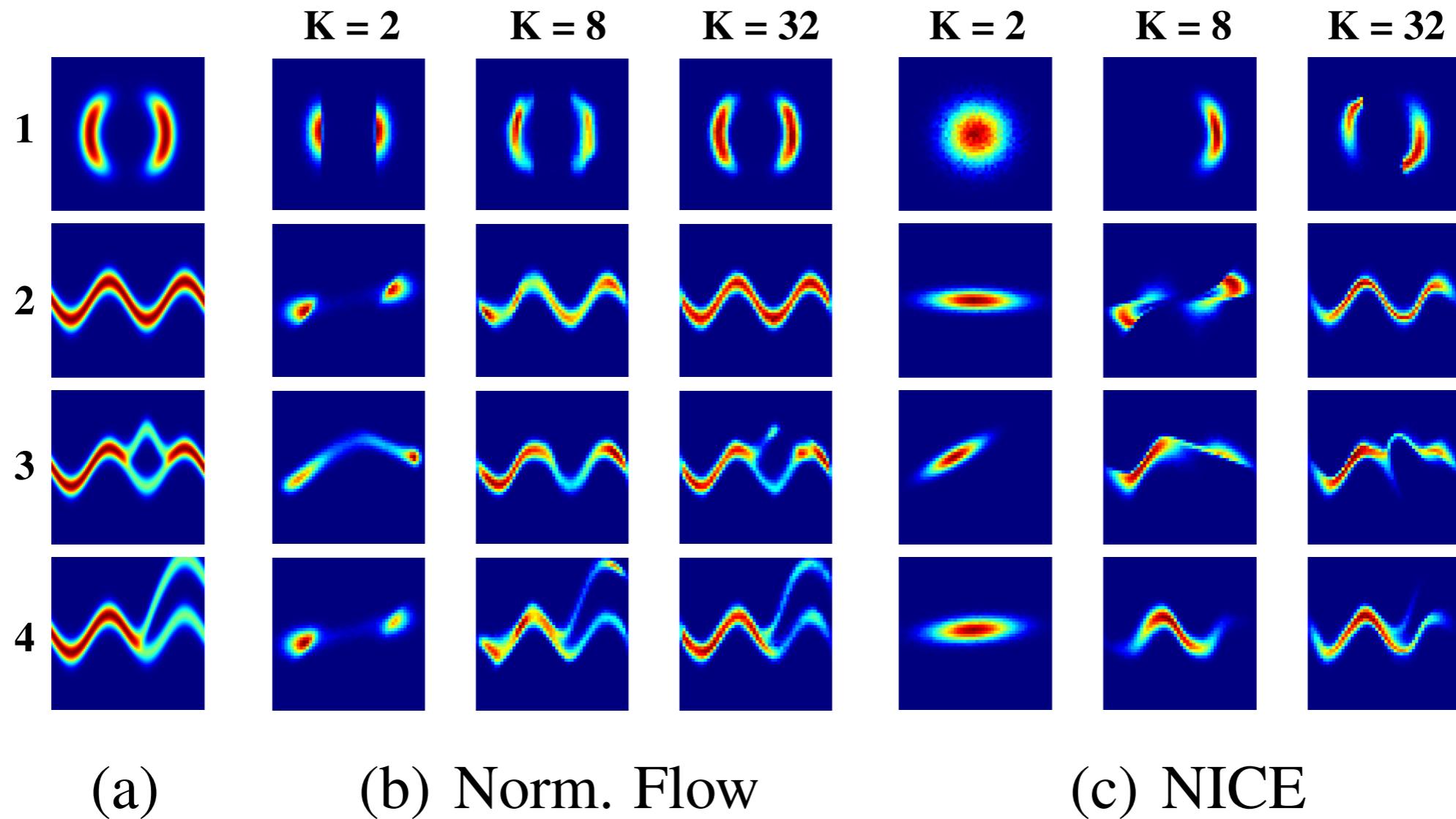
$$\log q_K(z) = \log q_0(z_0) - \sum_{k \leq K} \log |\det \nabla_{z_k} \Phi_k| .$$

- One can parametrize invertible flows and use them within the variational inference to improve the variational approximation. [Rezende et al.'15]
- Also considered in ["NICE", Dinh et al'15].
- Special case: *Inverse Autoregressive Flows* (i.e. Jacobian triangular) explored in "Variational Inference with Inverse Autoregressive Flows", by [Kingma, Salimans & Welling, NIPS'16].

# Normalizing Flows

- Some low-dimensional transport results:

[Rezende et al. '15]



# Diffusion and Non-equilibrium Thermodynamics

[Sohl-Dickstein et al.'15]

- We can also consider *infinitesimal* flows:

$$\frac{\partial q_t(z)}{\partial t} = \mathcal{F}(q_t(z)) , \quad q_0(z) = p_0(z) .$$

$\mathcal{F}$  describes the dynamics.

# Diffusion and Non-equilibrium Thermodynamics

[Sohl-Dickstein et al.'15]

- We can also consider *infinitesimal* flows:

$$\frac{\partial q_t(z)}{\partial t} = \mathcal{F}(q_t(z)) , \quad q_0(z) = p_0(z) .$$

$\mathcal{F}$  describes the dynamics.

- For  $\mathcal{F} = -\Delta$  we have Gaussian diffusion.

It defines a Markov diffusion kernel that successively transforms data distribution  $p_0(x)$  into a tractable distribution  $\pi(x)$ :

$$\pi(x) = \int T_\pi(x|x')\pi(x')dx'$$

$$q(x^{(t+1)}|x^{(t)}) = T_\pi(x^{(t+1)}|x^{(t)}, \beta_t) \quad \beta_t: \text{diffusion rate.}$$

# Diffusion and Non-equilibrium Thermodynamics

[Sohl-Dickstein et al.'15]

- The "forward" trajectory diffuses the data distribution into a tractable distribution, eg Gaussian.

# Diffusion and Non-equilibrium Thermodynamics

[Sohl-Dickstein et al.'15]

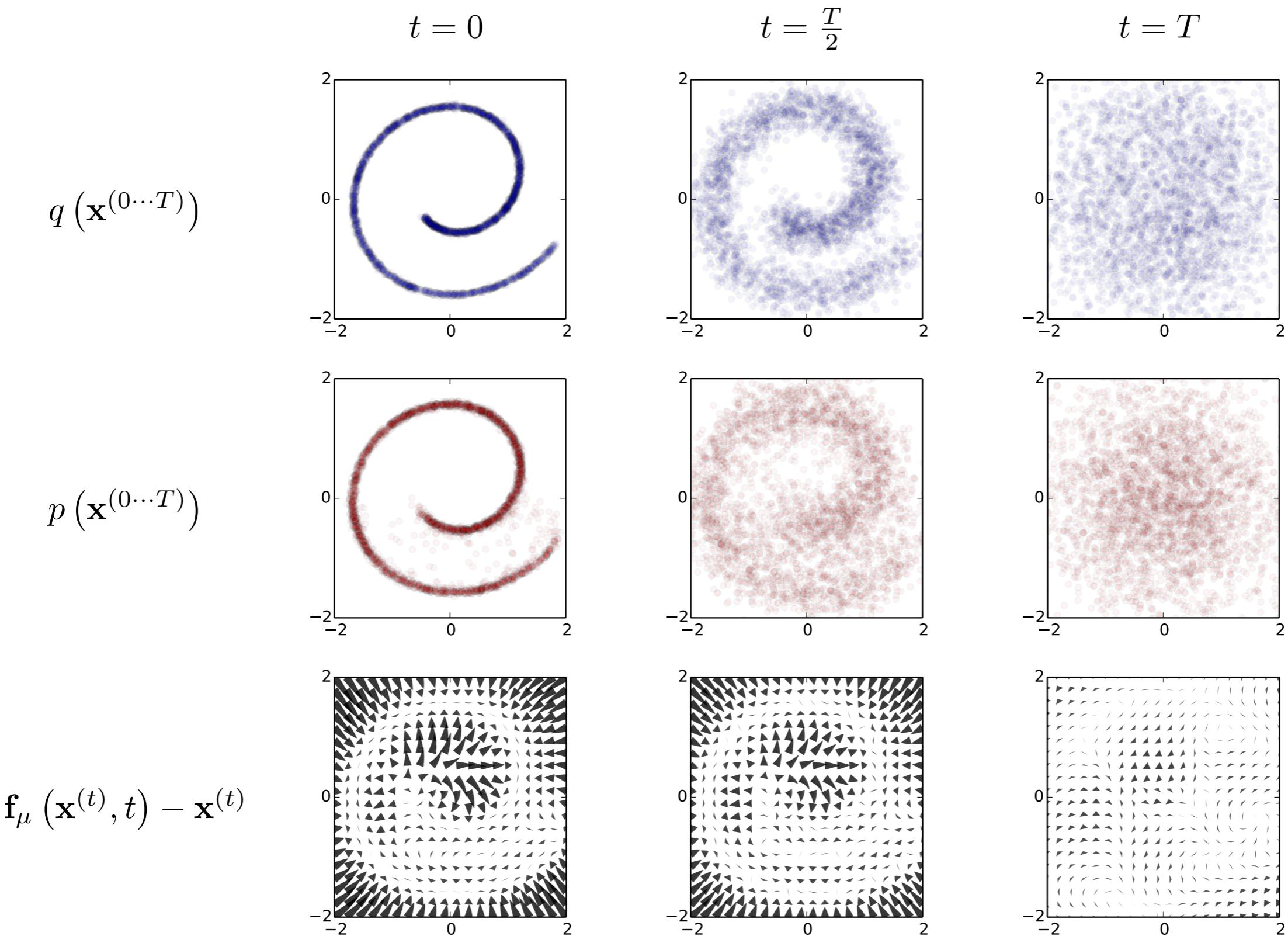
- The “forward” trajectory diffuses the data distribution into a tractable distribution, eg Gaussian.
- The generative model learns how to reverse the diffusion:

$$p(x^{(0\dots T)}) = p(x^{(T)}) \prod_{t \leq T} p(x^{(t-1)} | x^{(t)}) .$$

- in the limit of infinitesimal diffusion, the forward and backward kernel have the same functional form (Gaussian).
- The parameters of the model are  $\{\mu(x^{(t)}, t), \Sigma(x^{(t)}, t)\}_{t \leq T}$
- The data likelihood admits lower bound that can be evaluated efficiently using annealed importance sampling.

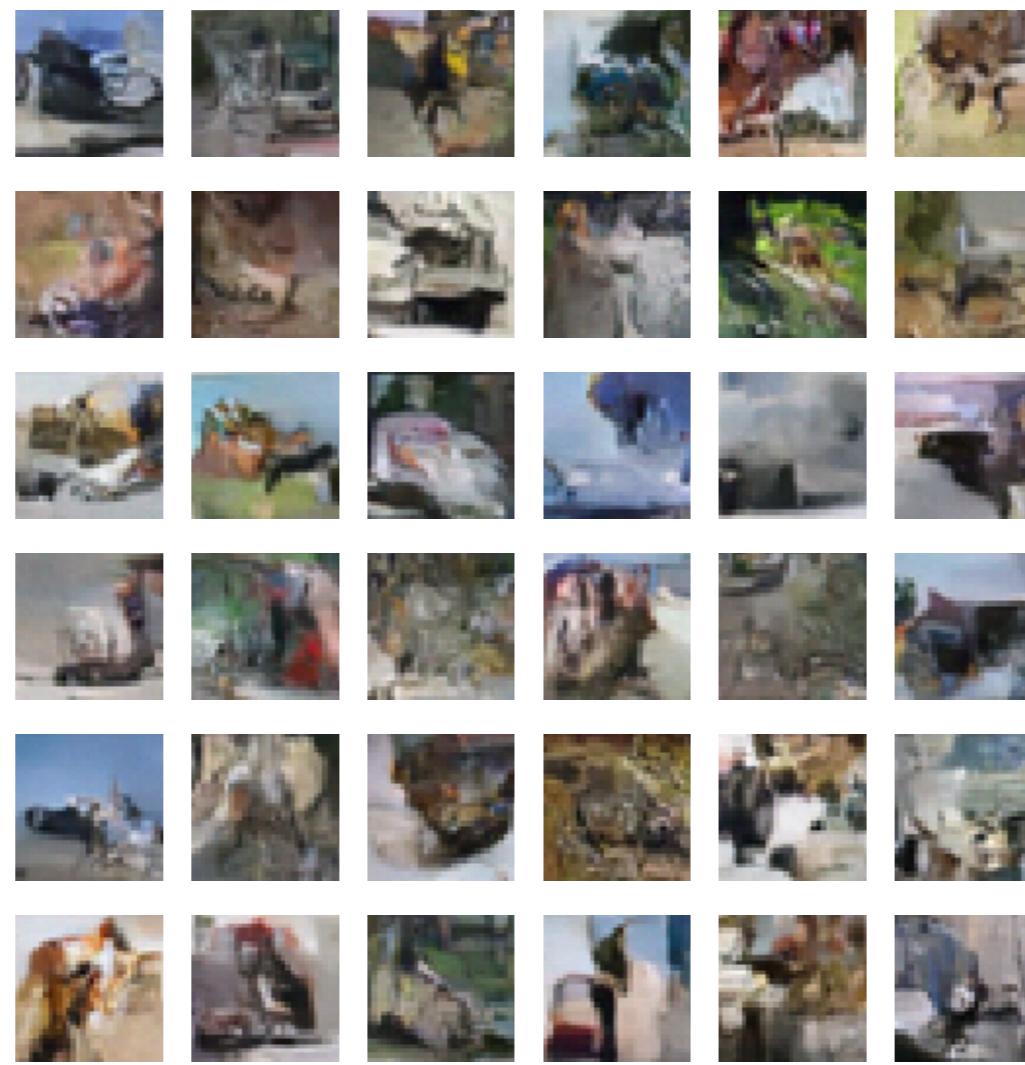
# Diffusion and Non-equilibrium Thermodynamics

[Sohl-Dickstein et al.'15]

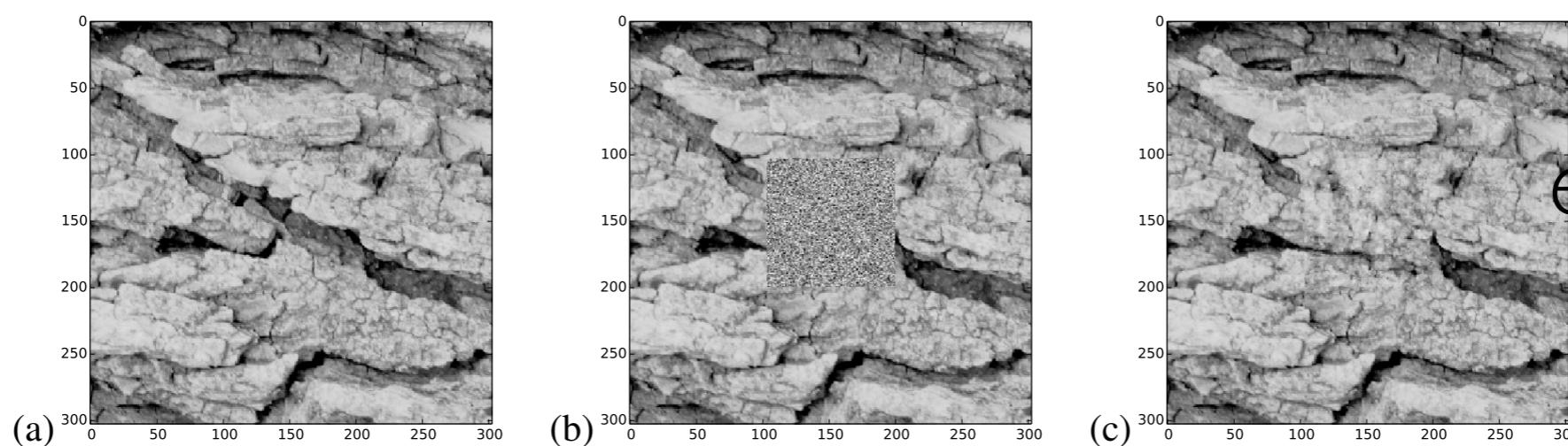


# Diffusion and Non-equilibrium Thermodynamics

[Sohl-Dickstein et al.'15]



samples  
from the model  
trained on  
CIFAR-10

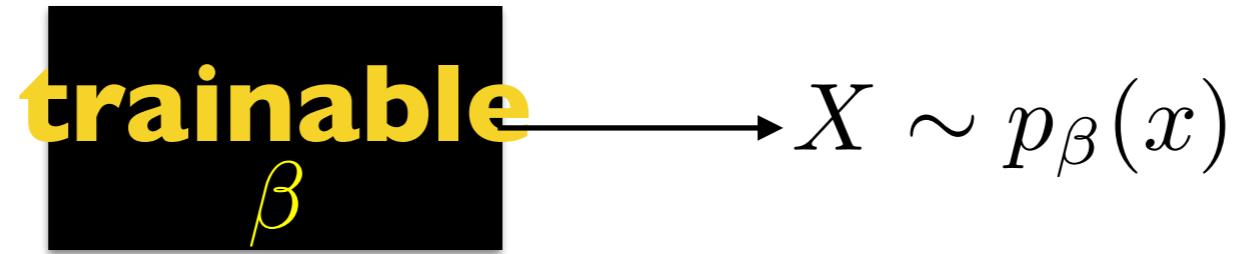


inpainting  
experiments

# Generative Adversarial Networks

[Goodfellow et al., '14]

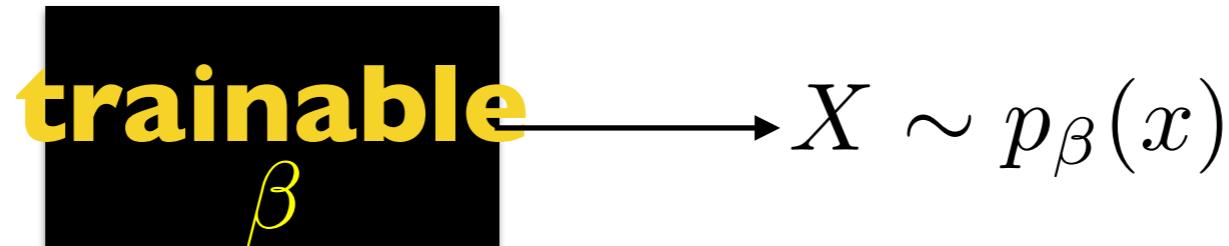
- Suppose we have a *trainable* black box generator:



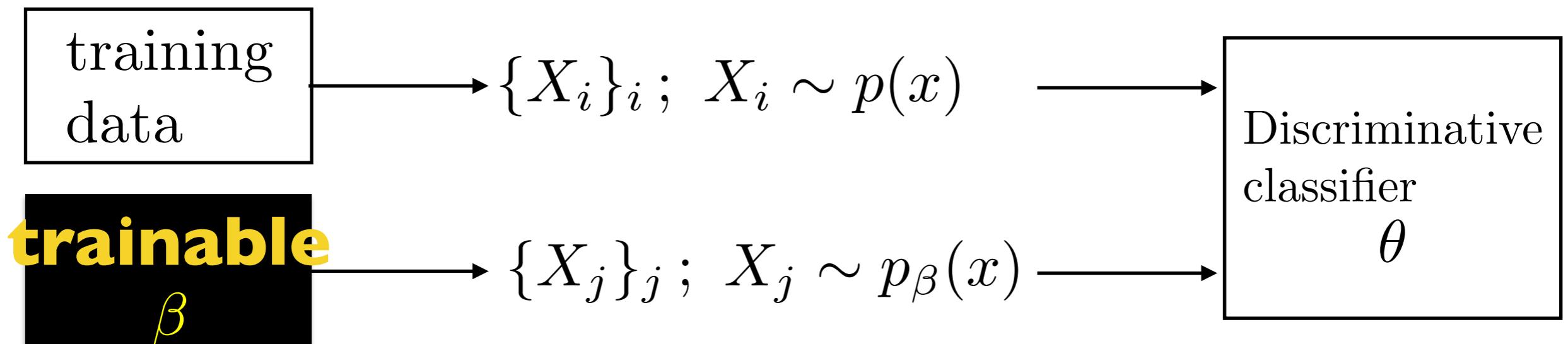
# Generative Adversarial Networks

[Goodfellow et al., '14]

- Suppose we have a *trainable* black box generator:



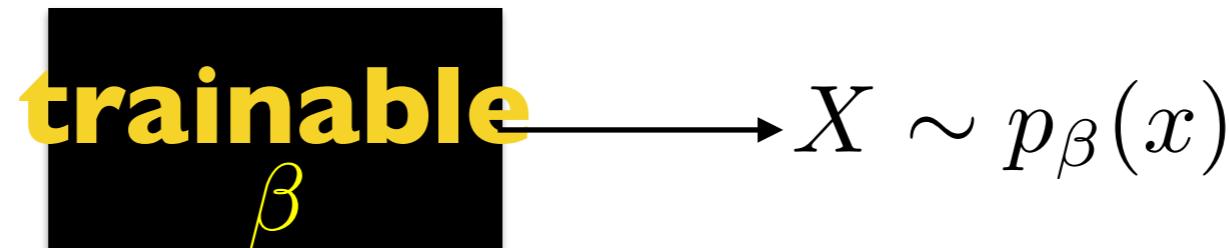
- Given observed data  $\{X_i\}_i$ ;  $X_i \sim p(x)$  how to force our generator to produce samples from  $p(x)$  ?



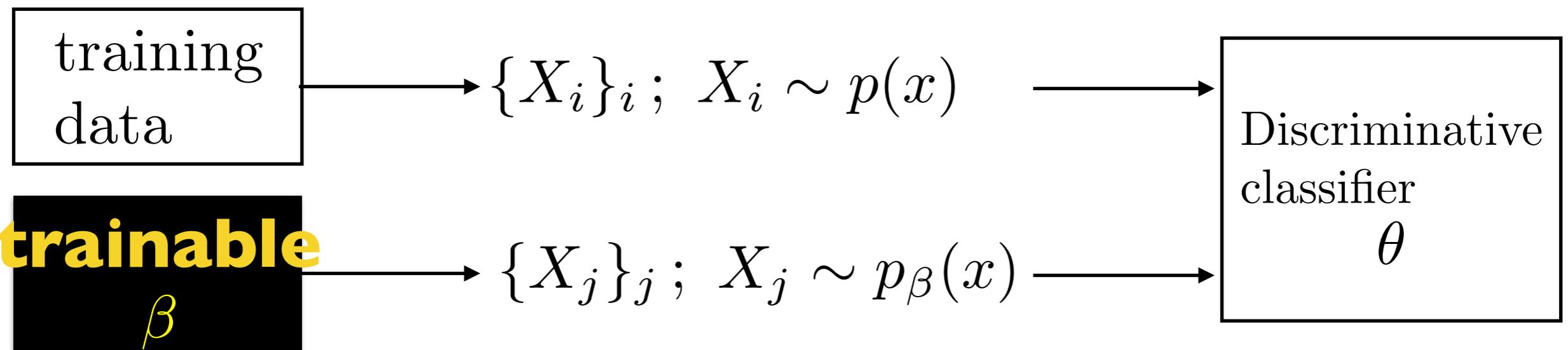
# Generative Adversarial Networks

[Goodfellow et al., '14]

- Suppose we have a *trainable* black box generator:



- Given observed data  $\{X_i\}_i$ ;  $X_i \sim p(x)$ , how to force our generator to produce samples from  $p(x)$ ?

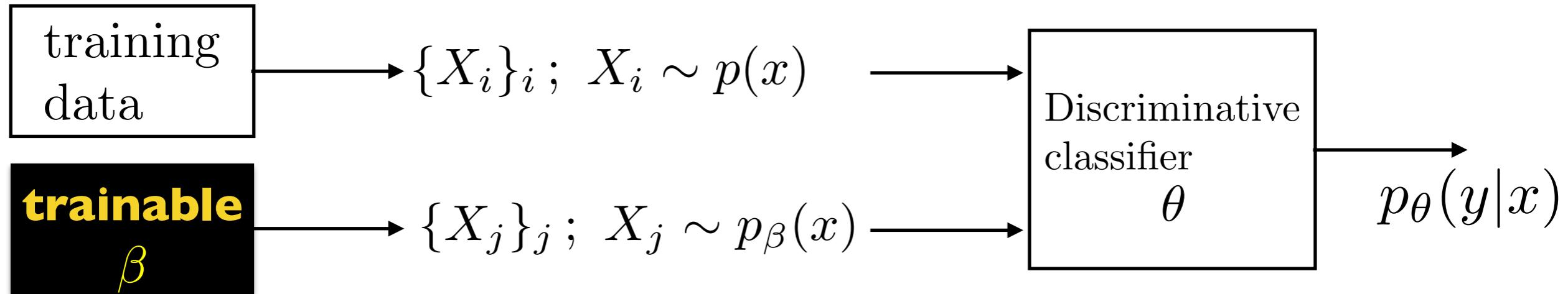


- The generator should make the classification task as hard as possible for any *discriminator*.

# Generative Adversarial Networks

[Goodfellow et al., '14]

- Train generator and discriminator in a minimax setting:



$y = 1$ : “real” samples

$y = 0$ : “fake” samples

$$\min_{\beta} \max_{\theta} \left( \mathbb{E}_{x \sim p_{data}} \log p_{\theta}(y=1|x) + \mathbb{E}_{x \sim p_{\beta}} \log p_{\theta}(y=0|x) \right) .$$

# Generative Adversarial Networks

- Q: Do we have consistency? (in the limit of infinite capacity)

# Generative Adversarial Networks

- Q: Do we have consistency? (in the limit of infinite capacity)

Given current  $p_\beta$  and  $p_{\text{data}}$ , the optimum discriminator is given by

$$D(x) = p(y = 1|x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\beta(x)} .$$

# Generative Adversarial Networks

- Q: Do we have consistency? (in the limit of infinite capacity)

Given current  $p_\beta$  and  $p_{\text{data}}$ , the optimum discriminator is given by

$$D(x) = p(y = 1|x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\beta(x)} .$$

For each  $x$ ,

$$p_{\text{data}}(x) \log D(x) + p_\beta(x) \log(1 - D(x)) = (p_{\text{data}}(x) + p_\beta(x)) (\alpha \log \gamma + (1 - \alpha \log(1 - \gamma))) ,$$

$$\alpha = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\beta(x)} , \quad \gamma = D(x) .$$

# Generative Adversarial Networks

- Q: Do we have consistency? (in the limit of infinite capacity)

Given current  $p_\beta$  and  $p_{\text{data}}$ , the optimum discriminator is given by

$$D(x) = p(y = 1|x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\beta(x)} .$$

For each  $x$ ,

$$p_{\text{data}}(x) \log D(x) + p_\beta(x) \log(1 - D(x)) = (p_{\text{data}}(x) + p_\beta(x)) (\alpha \log \gamma + (1 - \alpha) \log(1 - \gamma)) ,$$

$$\alpha = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\beta(x)} , \quad \gamma = D(x) .$$

But

$$\alpha \log \gamma + (1 - \alpha) \log(1 - \gamma) = -H(\bar{\alpha}) - D_{KL}(\bar{\alpha}||p(y|x)) \leq -H(\bar{\alpha})$$

# Generative Adversarial Networks

- It results that

$\min -H(\bar{\alpha})$  is attained when  $\alpha = 1/2$ , thus

$$p_\beta(x) = p_{data}(x)$$

- In practice, however, we parametrize both generator and discriminator using neural networks.
- Optimize the cost using gradient descent.

# Generative Adversarial Training

- Challenge: it is unfeasible to optimize fully in the inner discriminator loop:

$$\min_{\beta} \max_{\theta} F(\beta, \theta)$$

$$F(\beta, \theta) = (\mathbb{E}_{x \sim p_{data}} \log p_{\theta}(y=1|x) + \mathbb{E}_{x \sim p_{\beta}} \log p_{\theta}(y=0|x)) .$$

$$\theta^*(\beta) = \arg \max_{\theta} F(\beta, \theta) . \quad G(\beta) := F(\beta, \theta^*(\beta))$$

# Generative Adversarial Training

- Challenge: it is unfeasible to optimize fully in the inner discriminator loop:

$$\min_{\beta} \max_{\theta} F(\beta, \theta)$$

$$F(\beta, \theta) = (\mathbb{E}_{x \sim p_{data}} \log p_{\theta}(y=1|x) + \mathbb{E}_{x \sim p_{\beta}} \log p_{\theta}(y=0|x)) .$$

- Indeed,  $\theta^*(\beta) = \arg \max_{\theta} F(\beta, \theta) . \quad G(\beta) := F(\beta, \theta^*(\beta))$

$$\frac{\partial G(\beta)}{\partial \beta} = 0 \quad w.h.p.$$

- Numerical approach: alternate  $k$  steps of discriminator update with 1 step of generator update.
- Also, heuristic uses different false positive and false negative losses to improve numerical gradient computations.

# LAPGAN

[Denton, Chintala et al.'15]

- Initial GAN models were hard to scale to large input domains.
- Laplacian Pyramid of Adversarial Networks significantly improved quality by generating independently at each scale.
- Laplacian Pyramids are invertible linear multi-scale decompositions:

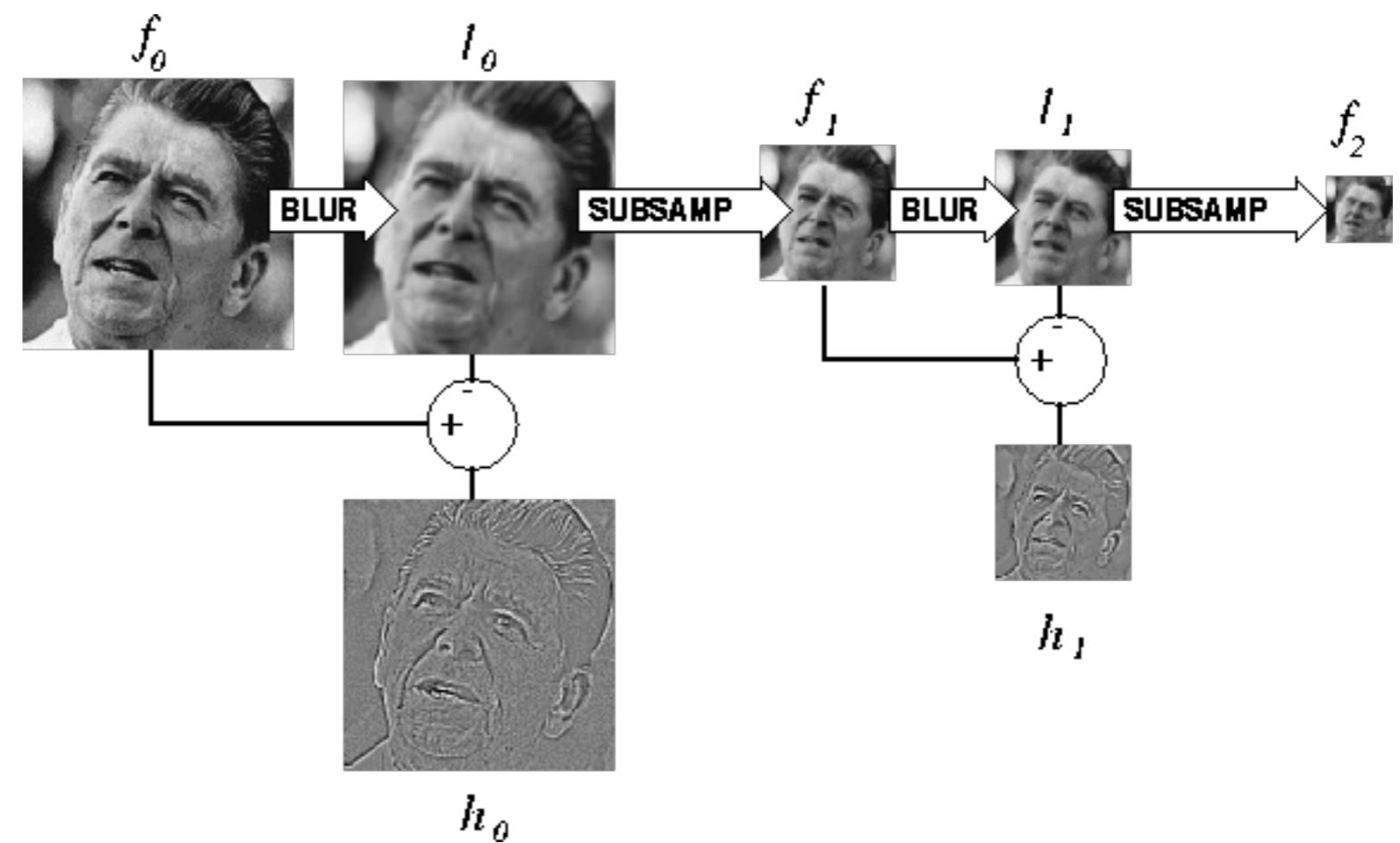
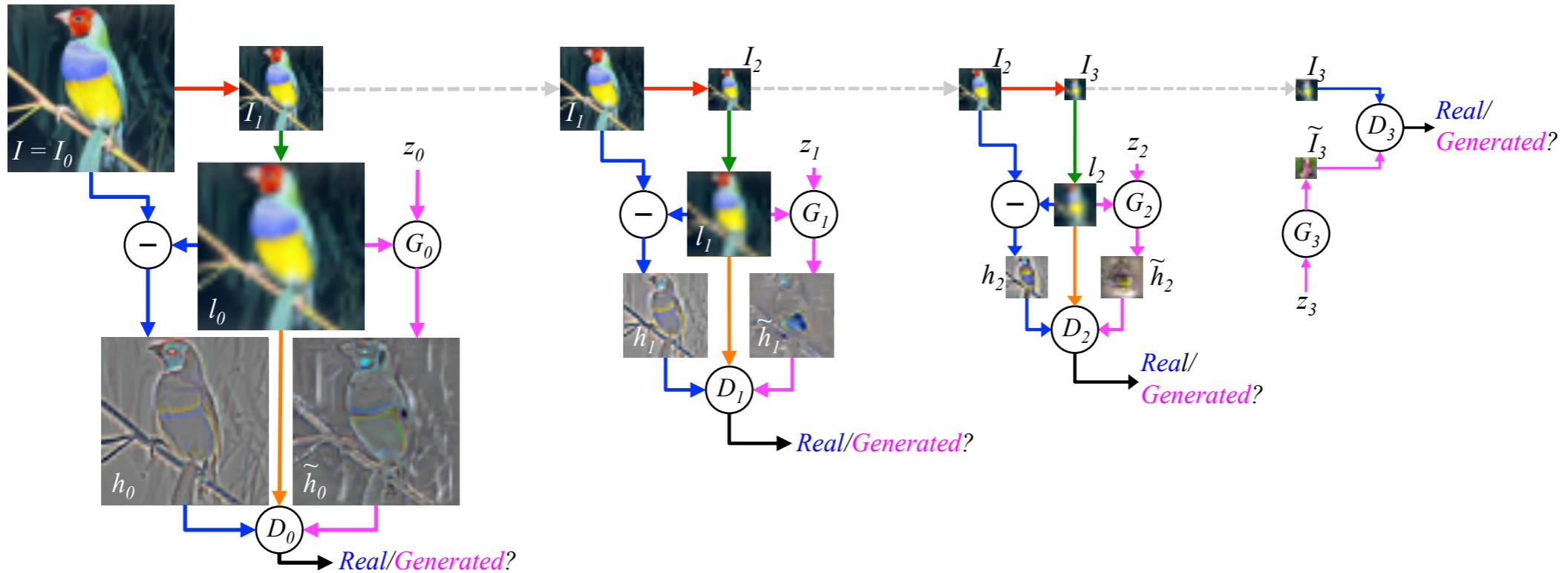


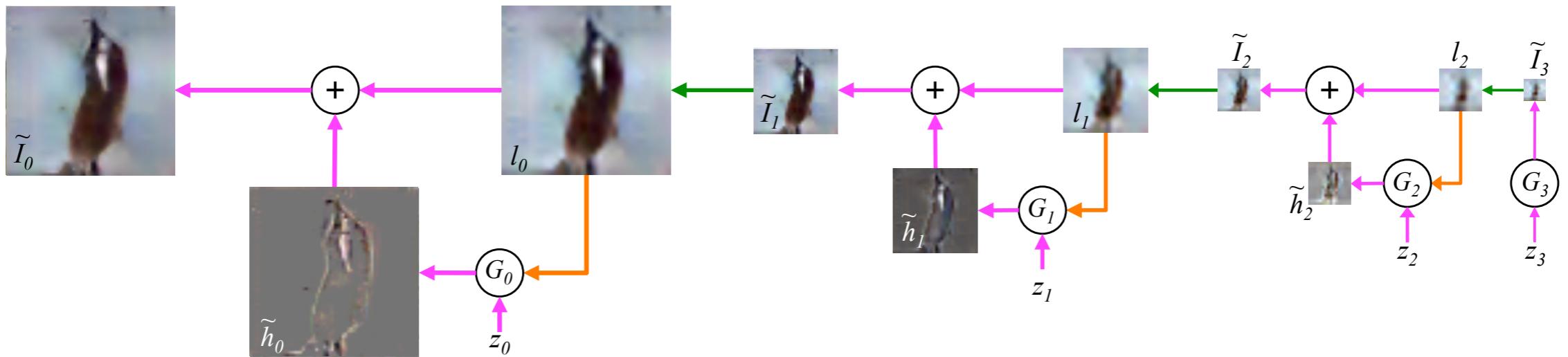
figure source: <http://sepwww.stanford.edu>

# LAPGAN

- Training procedure:

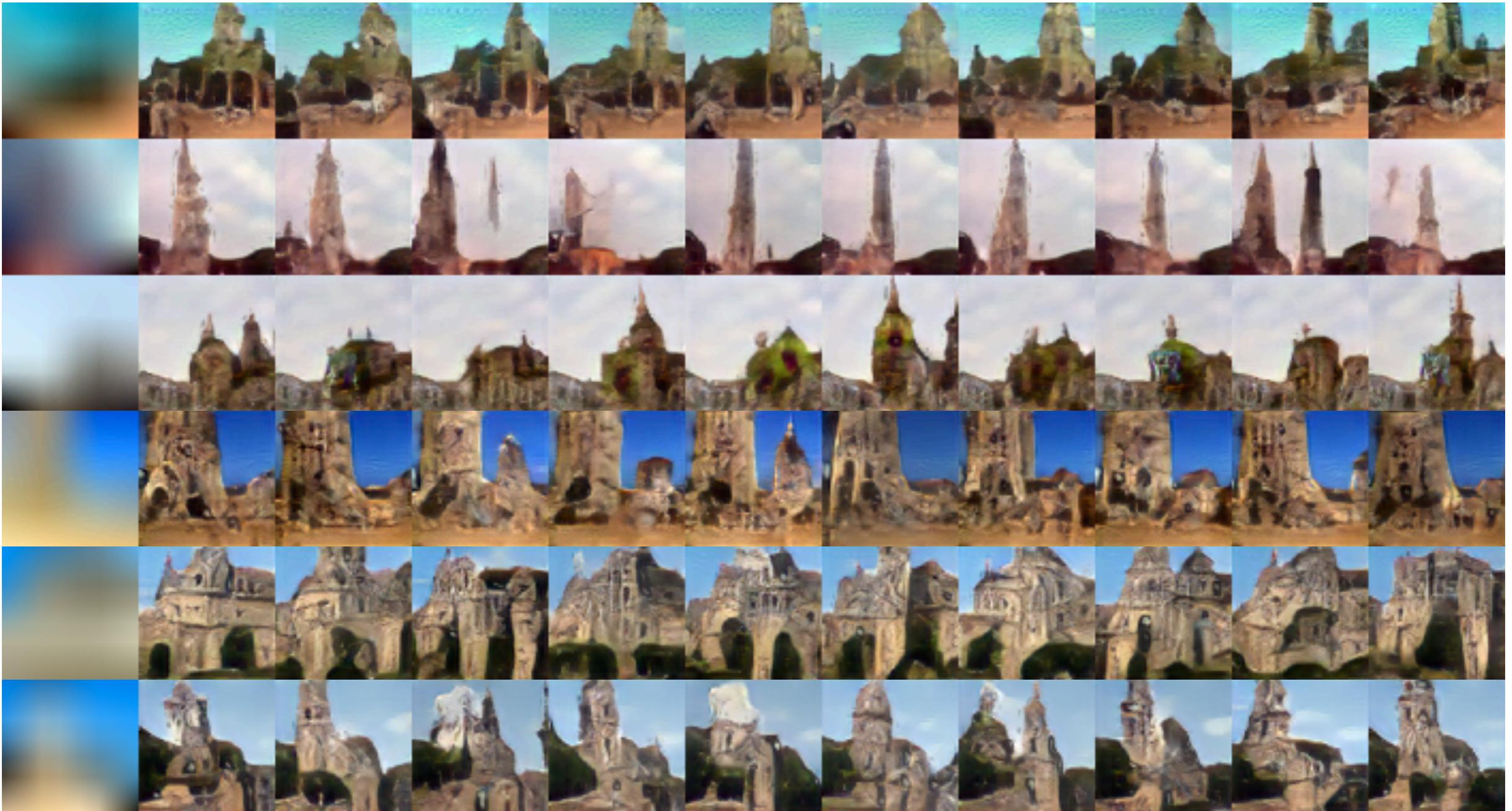


- Sampling procedure:



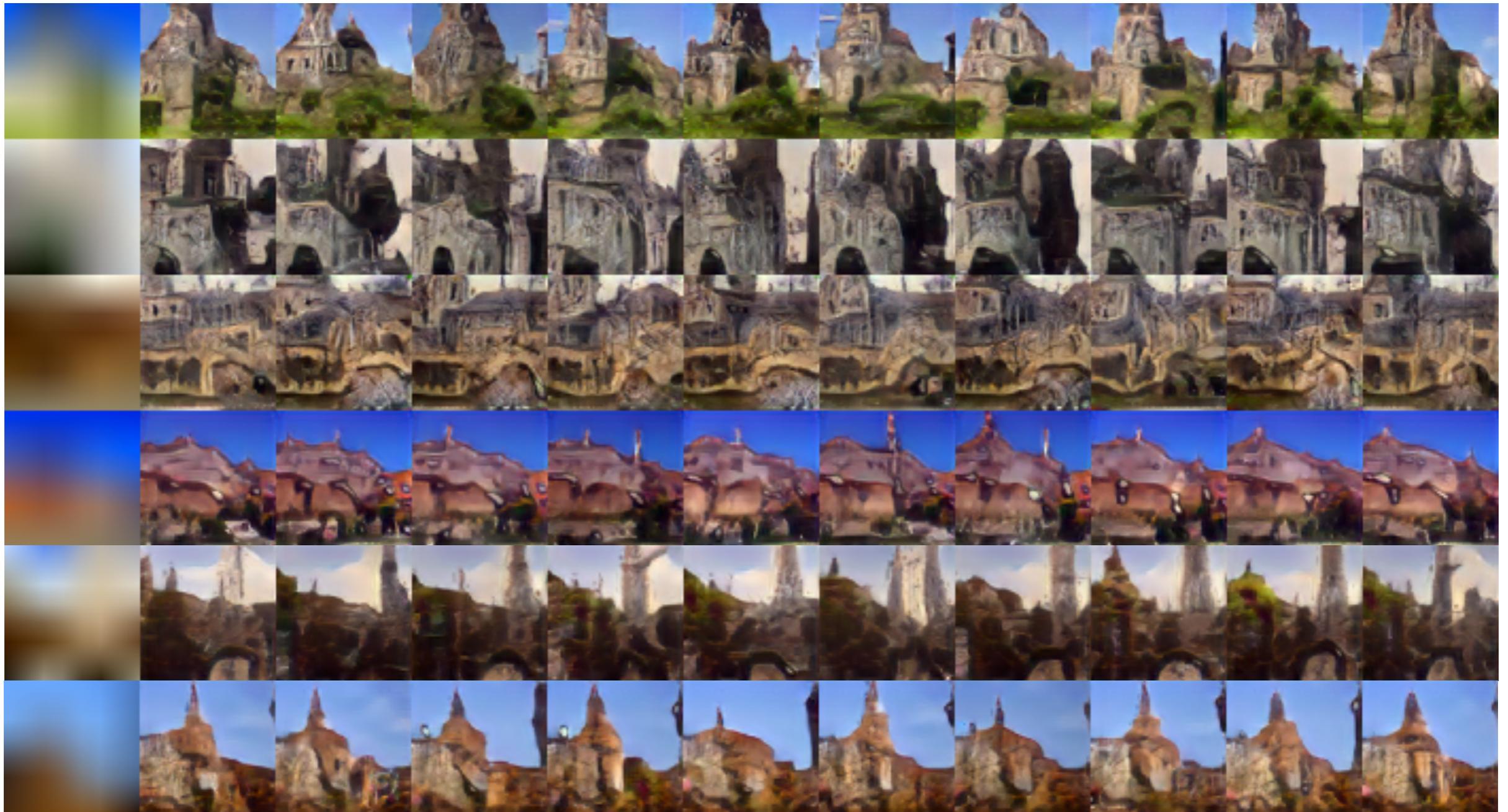
# LAPGAN

- Samples generated from the model:



# LAPGAN

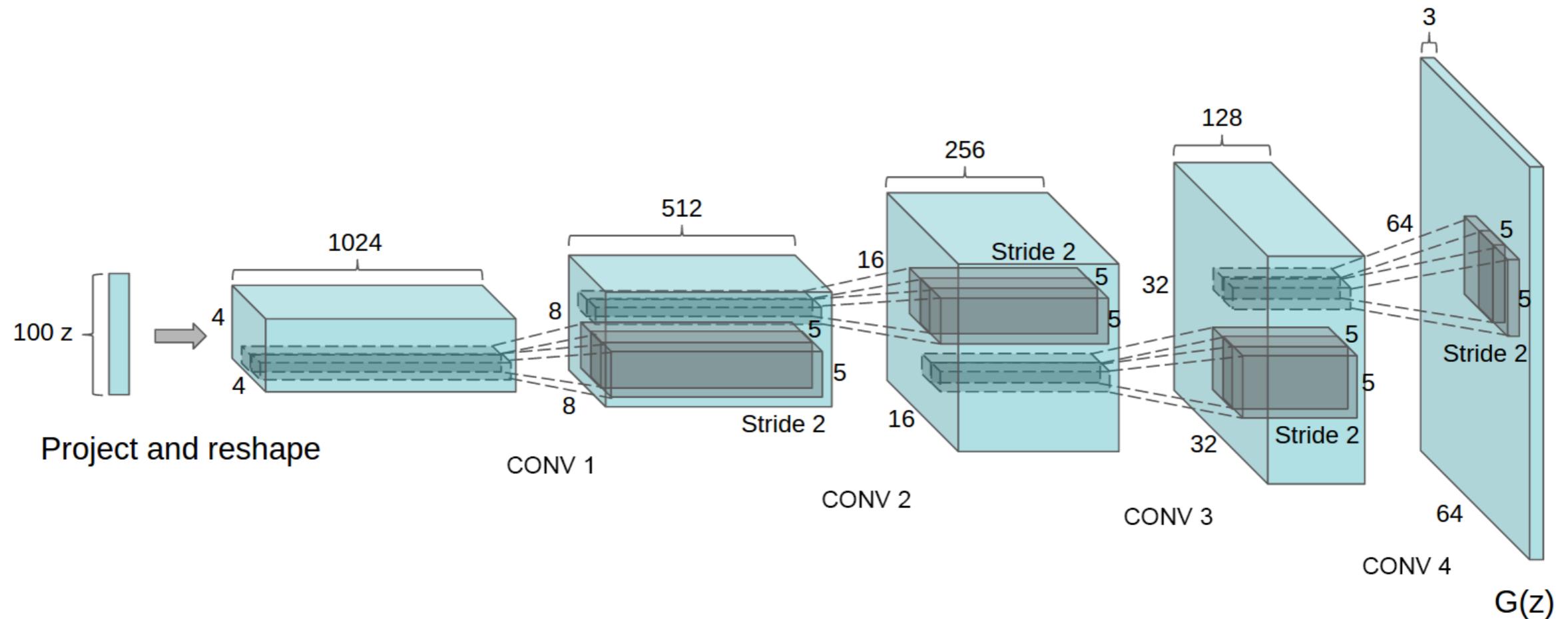
- Samples generated from the model:



# DC-GAN

[Radford et al.'16]

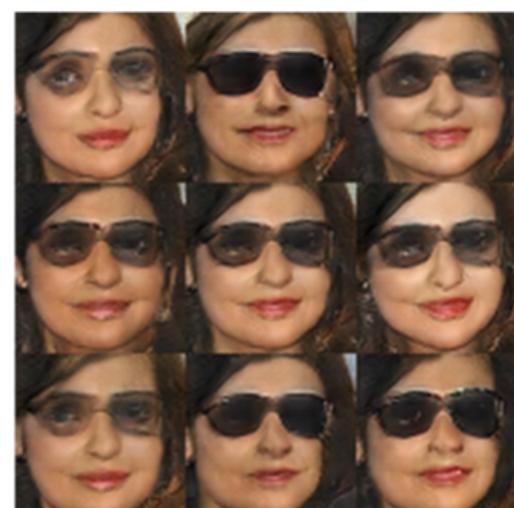
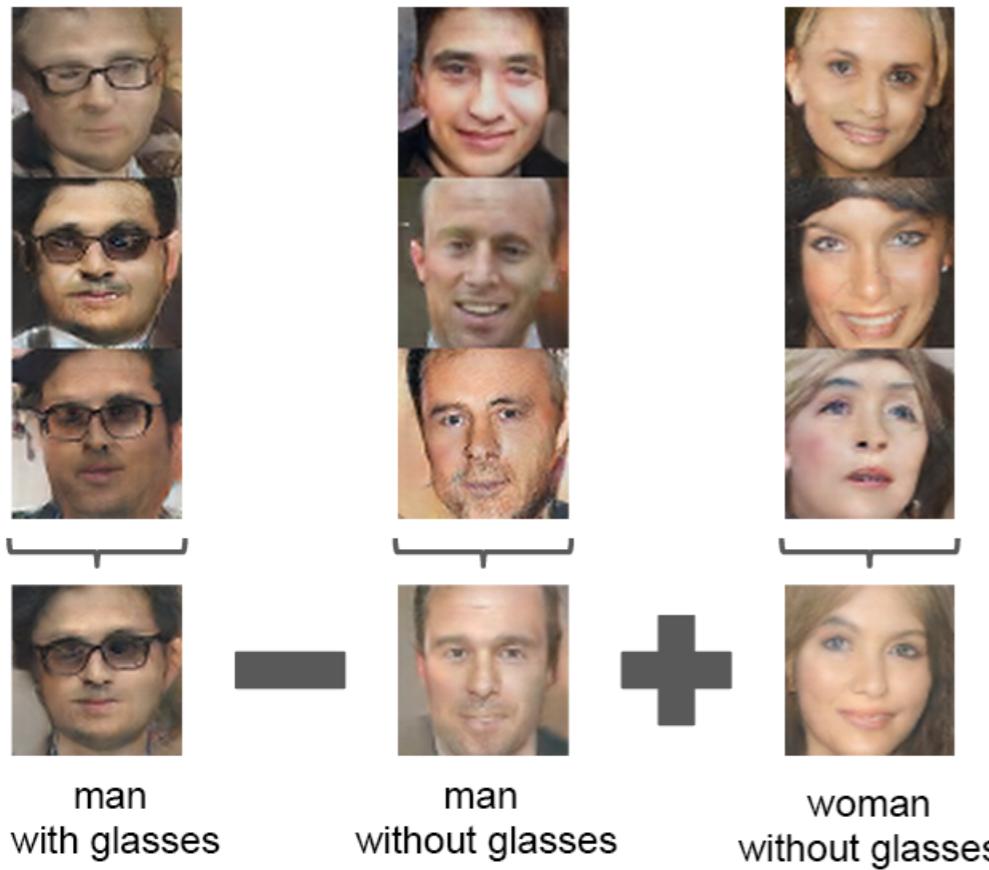
- Improved multi-scale architecture and Batch-Normalization:



# DC-GAN

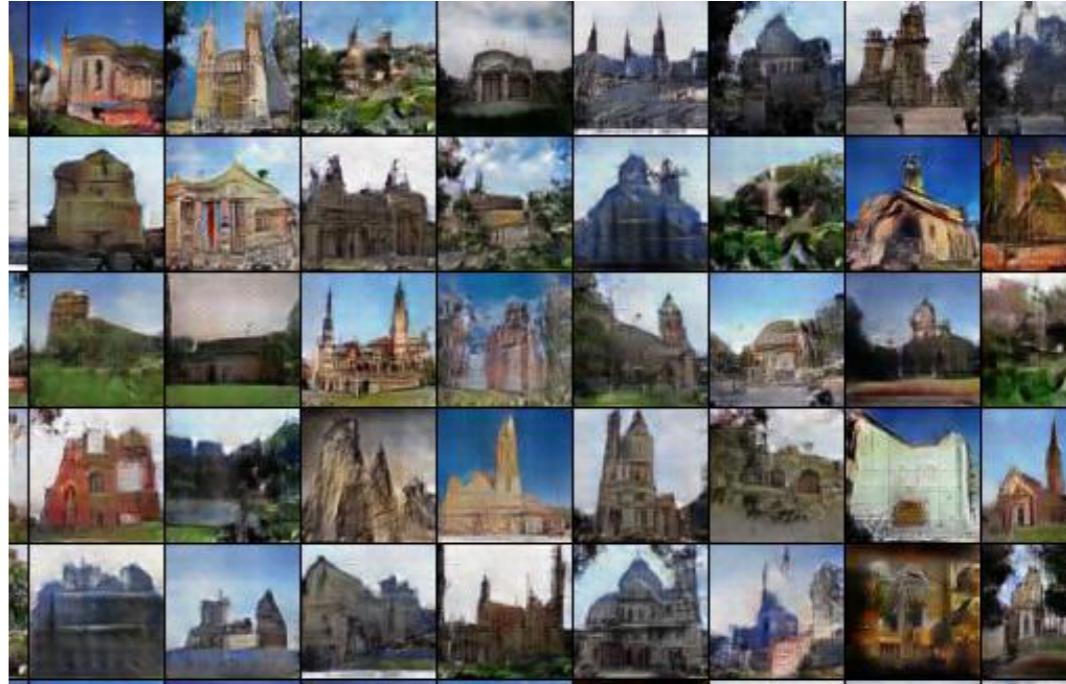
[Radford et al.'16]

- Improved multi-scale architecture and Batch-Normalization:



# Generative Adversarial Networks

- GRAN [Generative Recurrent Adversarial Nets, Im et al.'16]



- Video Prediction [Mathieu et al.'16]

- CNN Reconstruction [Brox et al.'16]

- More recently:

- Improved Techniques for GAN [OpenAI]
- Energy-based GANs [Zhao & Mathieu]
- ... (lots of papers!)

# GANs and Optimal Transport

- So far, we have measured/trained density models using log-likelihood:

**Model :**  $p(x; \theta)$

$$E(\theta) = \frac{1}{L} \sum_{l \leq L} \log p(x_l; \theta).$$

- This requires ability to measure model likelihoods.
  - (or at least a good lower bound as in variational autoencoders).

# GANs and Optimal Transport

- So far, we have measured/trained density models using log-likelihood:

**Model :**  $p(x; \theta)$

$$E(\theta) = \frac{1}{L} \sum_{l \leq L} \log p(x_l; \theta).$$

- This requires ability to measure model likelihoods.
  - (or at least a good lower bound as in variational autoencoders).
- The underlying “distance” in the space of distributions is the Kullback-Liebler divergence

$$KL(p_r \parallel p_m) = \int \log \left( \frac{p_r(x)}{p_m(x)} \right) p_r(x) d\mu(x) ,$$

- Estimated from finite data sample with

$$\widehat{KL}(p_r \parallel p_m) = \frac{1}{L} \sum_{l \leq L} \left( \frac{p_r(x_l)}{p_m(x_l; \theta)} \right) \propto -E(\theta) + C .$$

# GANs and Optimal Transport

- Original GANs are associated with Jensen-Shannon divergence

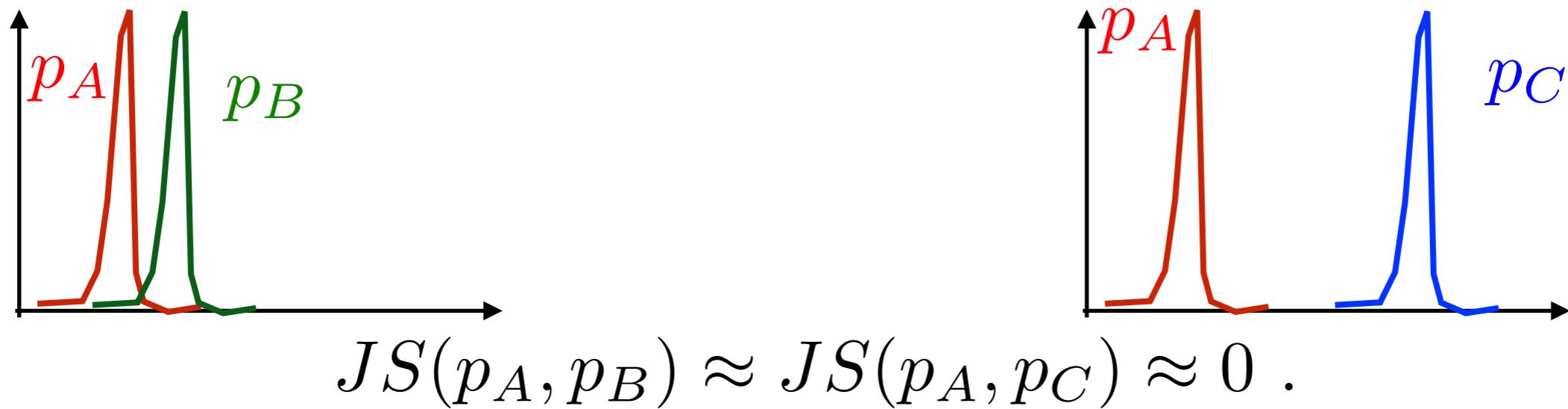
$$JS(p_r \parallel p_m) = KL(p_r \parallel p_m) + KL(p_m \parallel p_r)$$

# GANs and Optimal Transport

- Original GANs are associated with Jensen-Shannon divergence

$$JS(p_r \parallel p_m) = KL(p_r \parallel p_m) + KL(p_m \parallel p_r)$$

- Limitations:
  - Expression of densities not always available.
  - These metrics require the distributions to be absolutely continuous with respect to each other, e.g. by having densities wrt Lebesgue.
  - May be too ‘strong’ a distance for many applications:



# Earth-Mover Distance

- Alternatively, we can consider the Earth-Mover Distance (EMD)

$$W(P_r, P_m) = \inf_{\gamma \in \Pi(P_r, P_m)} \mathbb{E}_{(x,y) \sim \gamma} \{ \|x - y\| \},$$

$\Pi(P_r, P_m)$ : Set of joint distributions in  $(x, y)$  whose marginals are  $P_r, P_m$ .

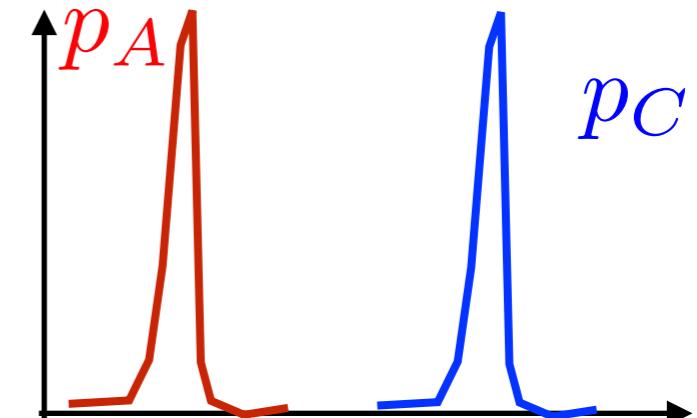
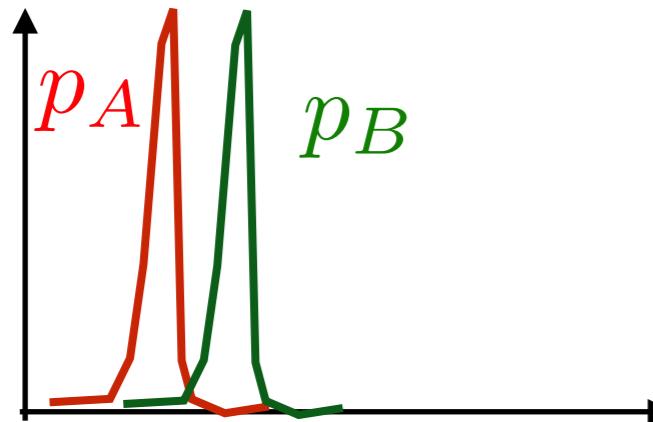
# Earth-Mover Distance

- Alternatively, we can consider the Earth-Mover Distance (EMD)

$$W(P_r, P_m) = \inf_{\gamma \in \Pi(P_r, P_m)} \mathbb{E}_{(x,y) \sim \gamma} \{ \|x - y\| \},$$

$\Pi(P_r, P_m)$ : Set of joint distributions in  $(x, y)$  whose marginals are  $P_r, P_m$ .

- Measures how much mass needs to be transported from  $P_r$  to  $P_m$ , measured with the Euclidean metric here (Wasserstein-1 distance).
- It defines a “weaker” topology in the space of probability distributions:



$$W(p_A, p_B) \ll W(p_A, p_C)$$

# Kantorovich Duality

- In general, computing such “couplings” is hard.
- However, in the Wasserstein-1 case, we have the following dual representation:

$$W(p_r, p_m) = \sup_{\text{Lip}(f) \leq 1} \mathbb{E}_{x \sim p_r} \{f(x)\} - \mathbb{E}_{x \sim p_m} \{f(x)\} .$$

$f : \mathcal{X} \rightarrow \mathbb{R}$  continuous,

$\text{Lip}(f)$ : Lipschitz constant of  $f$ :

$$\forall x, x' , |f(x) - f(x')| \leq \text{Lip}(f) \|x - x'\| .$$

# Kantorovich Duality

- In general, computing such “couplings” is hard.
- However, in the Wasserstein-1 case, we have the following dual representation:

$$W(p_r, p_m) = \sup_{\text{Lip}(f) \leq 1} \mathbb{E}_{x \sim p_r} \{f(x)\} - \mathbb{E}_{x \sim p_m} \{f(x)\} .$$

$f : \mathcal{X} \rightarrow \mathbb{R}$  continuous,

$\text{Lip}(f)$ : Lipschitz constant of  $f$ :

$$\forall x, x' , |f(x) - f(x')| \leq \text{Lip}(f) \|x - x'\| .$$

- Recall the discriminator loss of the “classic” GAN:

$$-\mathbb{E}_{x \sim p_r} \{\log D(x)\} + \mathbb{E}_{x \sim p_m} \{\log(1 - D(x))\}$$

# Wasserstein GAN [Arjovsky et al]

- In practice, we approximate the supremum over Lipschitz functions with a class of functions parametrized by a neural network:

$$W(p_r, p_m) = \sup_{\theta \in \mathcal{K}} \mathbb{E}_{x \sim p_r} \{f_\theta(x)\} - \mathbb{E}_{x \sim p_m} \{f_\theta(x)\} .$$

- Lipschitz bounds are enforced in [Arjovsky et al.] by simply clipping the weights ( $\theta \in \mathcal{K}$ ).
- Better control of Lipschitz regularity in, e.g. [Gulrajani et al].

# Limitations of GAN Modeling

- We are attempting to fit a distribution  $p_m$  to the "real" distribution  $p_r$  using a distance/divergence criteria  $\rho$  :

$$\inf_{\theta} \rho(p_r, p_m(\theta)) .$$

# Limitations of GAN Modeling

- We are attempting to fit a distribution  $p_m$  to the "real" distribution  $p_r$  using a distance/divergence criteria  $\rho$  :

$$\inf_{\theta} \rho(p_r, p_m(\theta)) .$$

- However, we do not have access to  $p_r$ , only to the *empirical measure*  $\hat{p}_{r,L}$ :

$$\hat{p}_{r,L}(x) = \frac{1}{L} \sum_{l \leq L} \delta(x - x_l) .$$

# Limitations of GAN Modeling

- We are attempting to fit a distribution  $p_m$  to the "real" distribution  $p_r$  using a distance/divergence criteria  $\rho$  :

$$\inf_{\theta} \rho(p_r, p_m(\theta)) .$$

- However, we do not have access to  $p_r$ , only to the *empirical measure*  $\hat{p}_{r,L}$ :

$$\hat{p}_{r,L}(x) = \frac{1}{L} \sum_{l \leq L} \delta(x - x_l) .$$

- Triangle Inequality:

$$\rho(p_r, p_m(\theta)) \leq \rho(p_r, \hat{p}_{r,L}) + \rho(\hat{p}_{r,L}, p_m(\theta)) .$$

↑                              ↑

**Statistical Error    Modeling Error**

# Limitations of GAN Modeling

- Thus, we need to regularize the density estimation problem to avoid fitting the empirical measure instead of the underlying real distribution.
- In the case of Wasserstein distance, we have the curse of dimensionality: if input space is  $\mathcal{X} = \mathbb{R}^d$ , then
$$\mathbb{E}\{\rho(p_r, \hat{p}_{r,L})\} \simeq L^{-\frac{1}{d}}$$
  - so we need a number of samples exponential in the dimension to make sampling error disappear.

# Limitations of GAN Modeling

- Thus, we need to regularize the density estimation problem to avoid fitting the empirical measure instead of the underlying real distribution.
- In the case of Wasserstein distance, we have the curse of dimensionality: if input space is  $\mathcal{X} = \mathbb{R}^d$ , then
$$\mathbb{E}\{\rho(p_r, \hat{p}_{r,L})\} \simeq L^{-\frac{1}{d}}$$
  - so we need a number of samples exponential in the dimension to make sampling error disappear.
- Intrinsic hardness of modeling densities in high dimensions.
- Entropic regularization: enforce  $p_m$  to be smooth by enforcing large  $H(p_m)$  (computationally hard in general).

# Some Recent Extensions

- InfoGAN [Chen & Rocky Duan et al., NIPS'16]
- *Image-to-Image Translation with Conditional Adversarial Networks* [Isola et al., '16]
- “Plug and Play Generative Networks” [Nguyen et al, '16]
- CycleGAN
- Progressive GAN [Karras et al]

# InfoGAN [Chen, Duan et al, '16]

- Recall the notion of smoothness in generative models:
  - **Maximum Entropy**
- If  $\mathbf{x}$  is a random vector of the form  $z \sim F_0$  ,  $x | z = G(z, \varphi)$  , how to compute its entropy?

# InfoGAN [Chen, Duan et al, '16]

- Recall the notion of smoothness in generative models:
  - **Maximum Entropy**
- If  $\mathbf{x}$  is a random vector of the form  $z \sim F_0, x | z = G(z, \varphi)$ , how to compute its entropy?
- If  $z \mapsto G(z, \varphi)$  is injective, then
$$p_G(x) = F_0(G^{-1}(x)) \cdot |DG^{-1}(x)|, \text{ and}$$
$$\mathbb{E}_{z \sim F_0} f(G(z)) = \mathbb{E}_{x \sim p_G} f(x) \quad \forall f \text{ measurable}.$$

# InfoGAN [Chen, Duan et al, '16]

- Recall the notion of smoothness in generative models:

- **Maximum Entropy**

- If  $\mathbf{x}$  is a random vector of the form  $z \sim F_0, x | z = G(z, \varphi)$ , how to compute its entropy?

- If  $z \mapsto G(z, \varphi)$  is injective, then

$$p_G(x) = F_0(G^{-1}(x)) \cdot |DG^{-1}(x)|, \text{ and}$$
$$\mathbb{E}_{z \sim F_0} f(G(z)) = \mathbb{E}_{x \sim p_G} f(x) \quad \forall f \text{ measurable}.$$

- Since  $H(p) = -\mathbb{E}_{x \sim p} [\log p(x)]$ , we have

$$\begin{aligned} H(p_G) &= -\mathbb{E}_{x \sim p_G} \log p_G(x) \\ &= -\mathbb{E}_{x \sim p_G} [\log F_0(G^{-1}(x)) + \log |DG^{-1}(x)|] \\ &= -\mathbb{E}_{z \sim F_0} [\log F_0(z) - \log |DG(z)|] \\ &= H(F_0) + \mathbb{E}_{z \sim F_0} \log |DG(z)|. \end{aligned}$$

# InfoGAN [Chen, Duan et al, '16]

- Recall the notion of smoothness in generative models:

- **Maximum Entropy**

- If  $\mathbf{x}$  is a random vector of the form  $z \sim F_0, x | z = G(z, \varphi)$ , how to compute its entropy?

- If  $z \mapsto G(z, \varphi)$  is injective, then

$$p_G(x) = F_0(G^{-1}(x)) \cdot |DG^{-1}(x)|, \text{ and}$$
$$\mathbb{E}_{z \sim F_0} f(G(z)) = \mathbb{E}_{x \sim p_G} f(x) \quad \forall f \text{ measurable}.$$

- Since  $H(p) = -\mathbb{E}_{x \sim p} [\log p(x)]$ , we have

$$\begin{aligned} H(p_G) &= -\mathbb{E}_{x \sim p_G} \log p_G(x) \\ &= -\mathbb{E}_{x \sim p_G} [\log F_0 G^{-1} x + \log |DG^{-1}(x)|] \\ &= -\mathbb{E}_{z \sim F_0} [\log F_0(z) - \log |DG(z)|] \\ &= H(F_0) + \mathbb{E}_{z \sim F_0} \log |DG(z)|. \end{aligned}$$

- Challenge: computing  $\mathbb{E}_{z \sim F_0} \log |DG(z)|$  is hard!

# InfoGAN

- Moreover, in some problems, we have prior information on the latent structure of variability, e.g.
  - Digits have a latent discrete (10 class) structure, plus continuous low-dim structure encoding style.
  - Faces have illumination and pose, identity, muscular movement, etc.

# InfoGAN

- Moreover, in some problems, we have prior information on the latent structure of variability, e.g.
  - Digits have a latent discrete (10 class) structure, plus continuous low-dim structure encoding style.
  - Faces have illumination and pose, identity, muscular movement, etc.
- Q: Can we combine max-entropy regularization with more control on the latent structure?

# InfoGAN

- Moreover, in some problems, we have prior information on the latent structure of variability, e.g.
  - Digits have a latent discrete (10 class) structure, plus continuous low-dim structure encoding style.
  - Faces have illumination and pose, identity, muscular movement, etc.
- Q: Can we combine max-entropy regularization with more control on the latent structure?
- An alternative is to use mutual information:

$$I(X, Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) .$$

- i.e. how much uncertainty remains in  $X$  after observing  $Y$  (and vice versa).
- It can be plugged-in the GAN objective as follows

$$\min_G \max_D \{ \mathbb{E}_{x \sim \hat{P}} [\log D(x)] + \mathbb{E}_{z \sim F_0} [\log(1 - D(G(z)))] \} - \lambda I(c; G(z, c)) ,$$

# InfoGAN

- Computing the mutual information requires access to the posterior  $p(c | x)$
- Although in the GAN we don't have this term, we have something that can approximate it: the discriminator network!

# InfoGAN

- Computing the mutual information requires access to the posterior  $p(c | x)$
- Although in the GAN we don't have this term, we have something that can approximate it: the discriminator network!
- Variational Information Maximization:

$$\begin{aligned} I(c, G(z, c)) &= \mathbb{E}_{x \sim G(z, c)} \mathbb{E}_{c' \sim p(c|x)} [\log p(c'|x)] + H(c) \\ &= \mathbb{E}_{x \sim G(z, c)} (D_{KL}(p(c'|x) || q(c'|x)) + \mathbb{E}_{c' \sim p(c|x)} \log q(c'|x)) + H(c) \\ &\geq \mathbb{E}_{x \sim G(z, c)} \mathbb{E}_{c' \sim p(c|x)} \log q(c'|x) + H(c) \\ &= \mathbb{E}_{(z, c) \sim P(c, z)} \log q(c'|G(z, c)) + H(c) . \end{aligned}$$

# InfoGAN



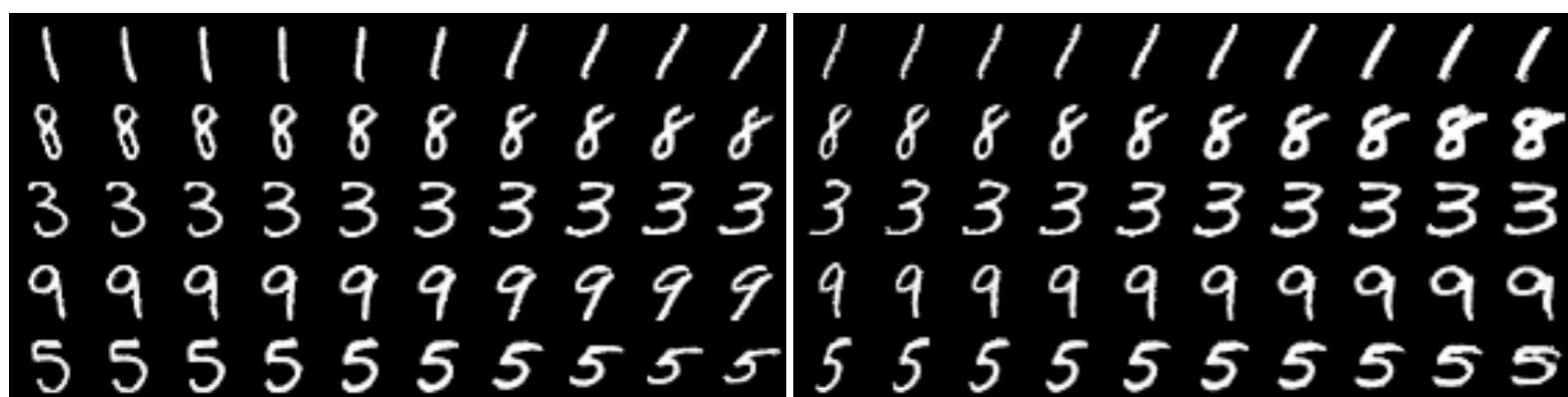
(a) Varying  $c_1$  on InfoGAN (Digit type)

(b) Varying  $c_1$  on regular GAN (No clear meaning)

$v$ -dimensional data.

o overfit the training data.

! VARIOING SINGULAR DISTRIBUTIONS have no well-defined entropy!



(c) Varying  $c_2$  from  $-2$  to  $2$  on InfoGAN (Rotation)

(d) Varying  $c_3$  from  $-2$  to  $2$  on InfoGAN (Width)



(a) Rotation

(b) Width

# Other Recent GAN work

- Conditional GANS for computer vision tasks

Labels to Street Scene

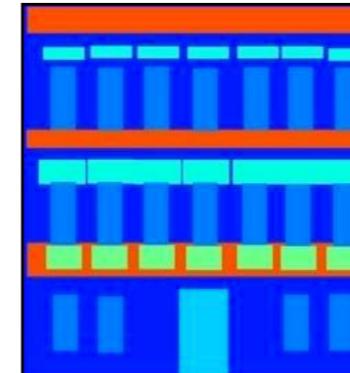


input



output

Labels to Facade



input



output

BW to Color



input

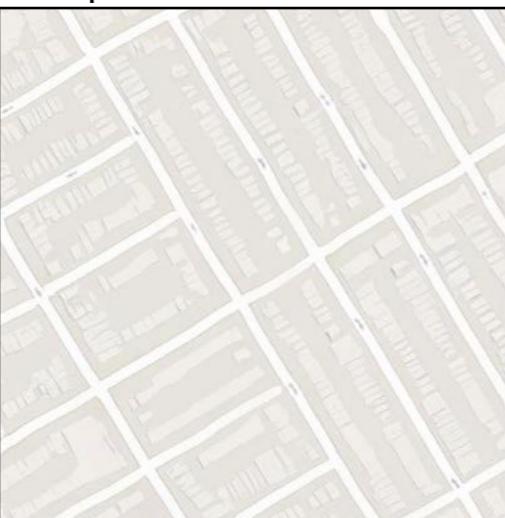


output

Aerial to Map



input



output

Day to Night



input



output

Edges to Photo



input



output

"Image-to-Image translation with Conditional Adversarial Networks", Isola et al.'16

# Cycle GAN [Zhu, Park, Isola, Efros]

- Image-to-Image translation without aligned datasets.

**Monet**  **Photos**



Monet → photo

**Zebras**  **Horses**



zebra → horse

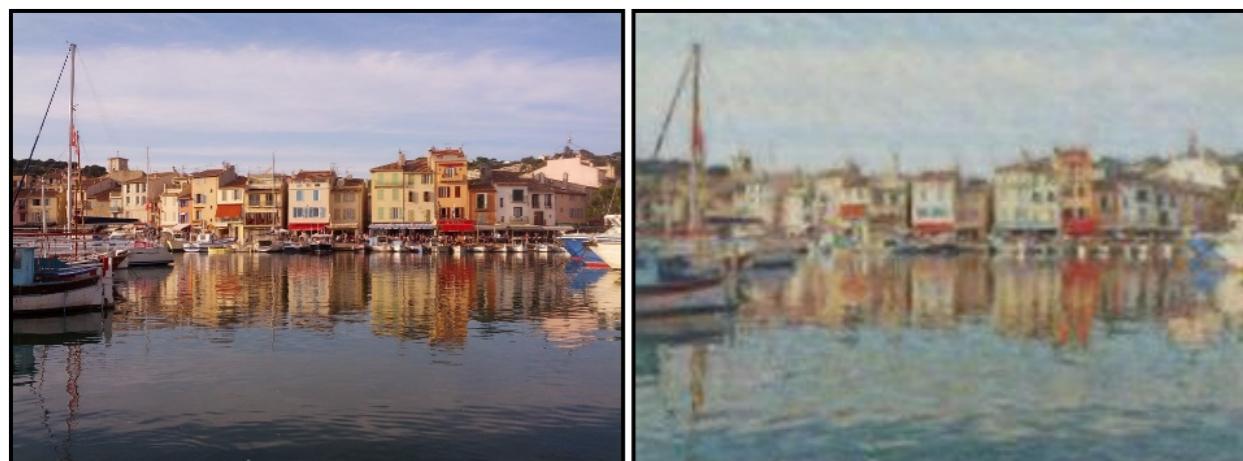
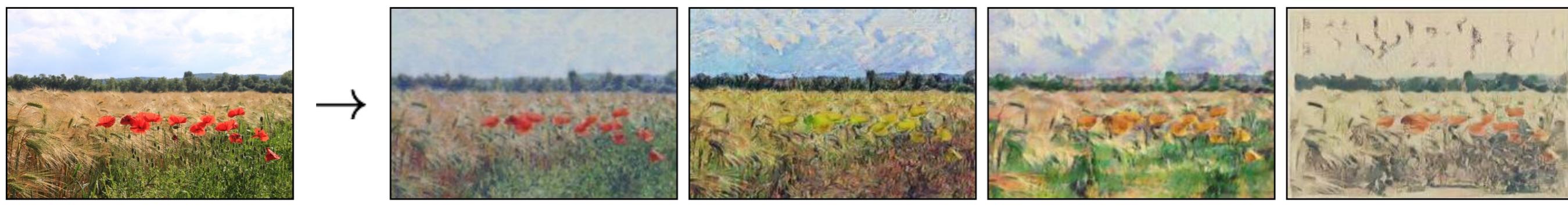


photo → Monet



horse → zebra



Photograph

Monet

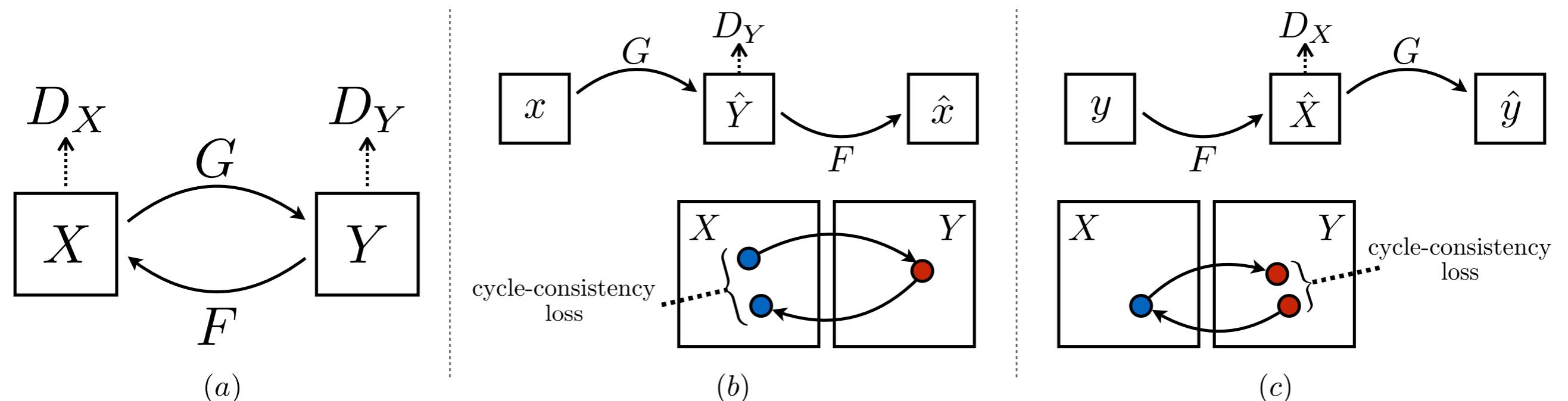
Van Gogh

Cezanne

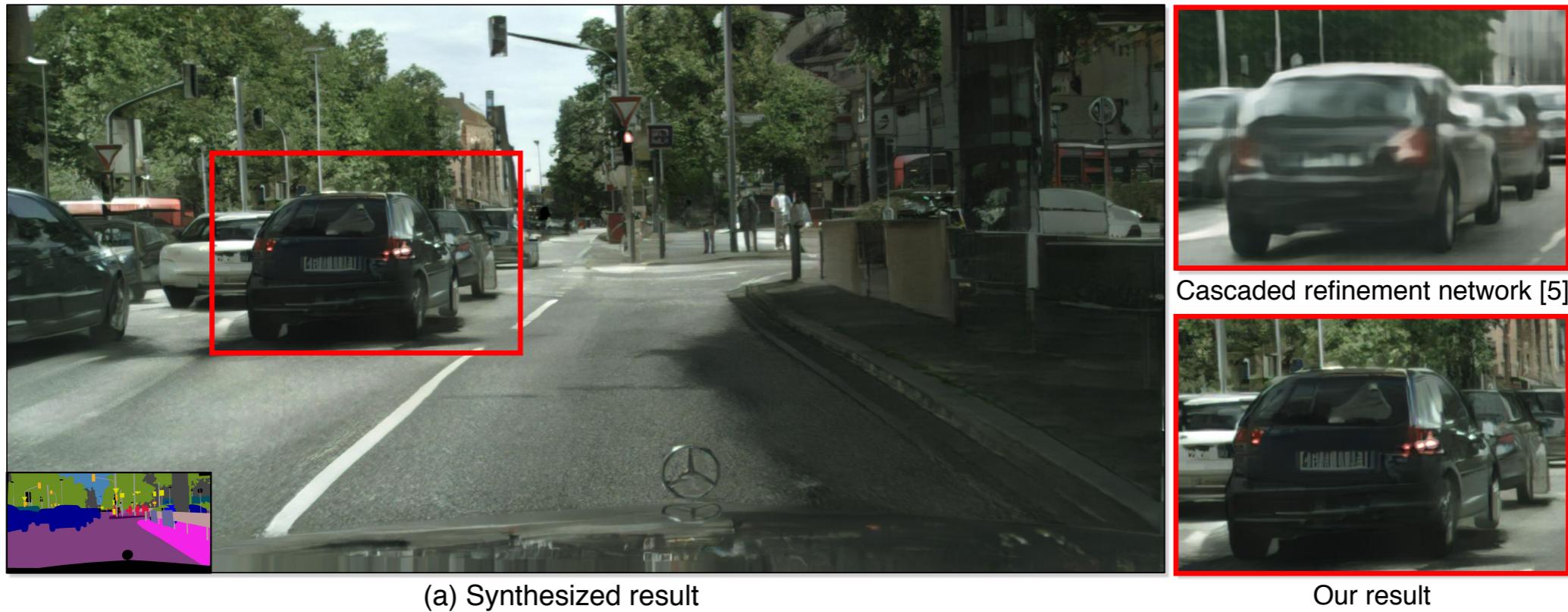
Ukiyo-e

# Cycle GAN

- Cycle Consistency Loss:



# Photo Realistic GAN [Wang et al]

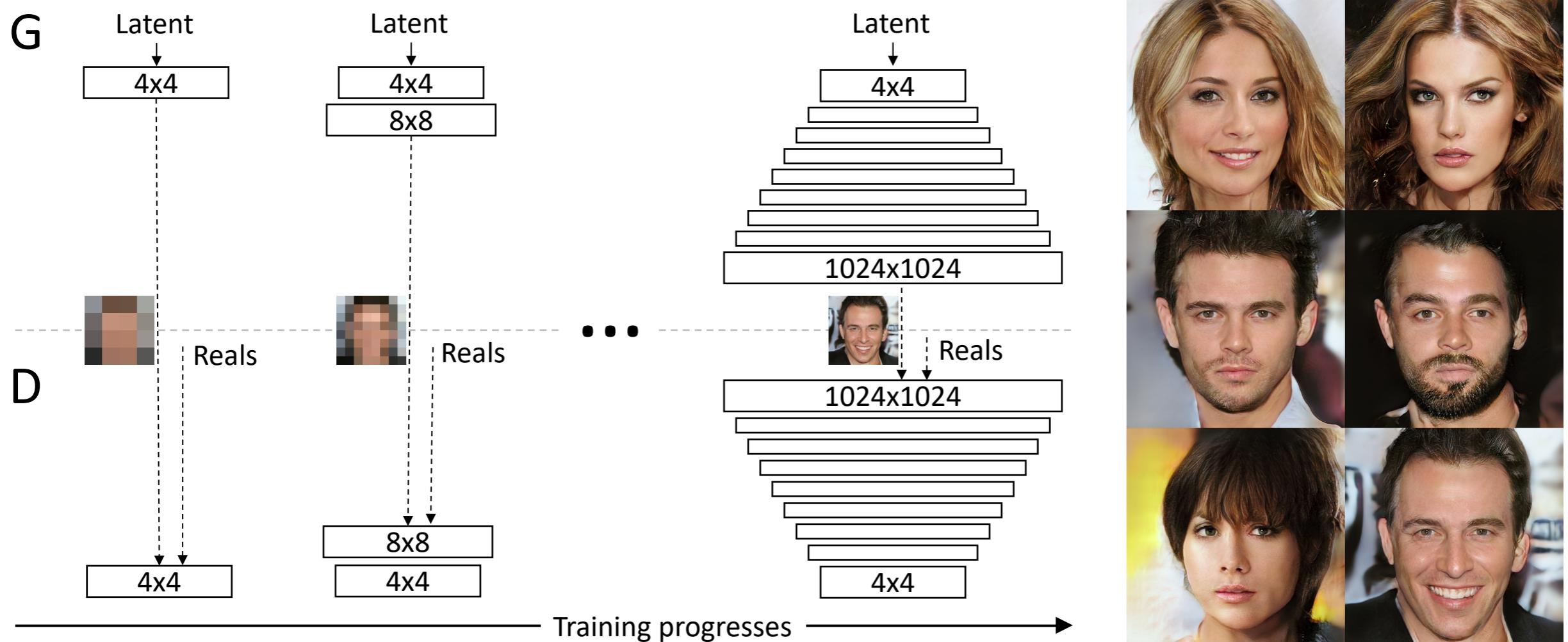


# Photo-Realistic GAN

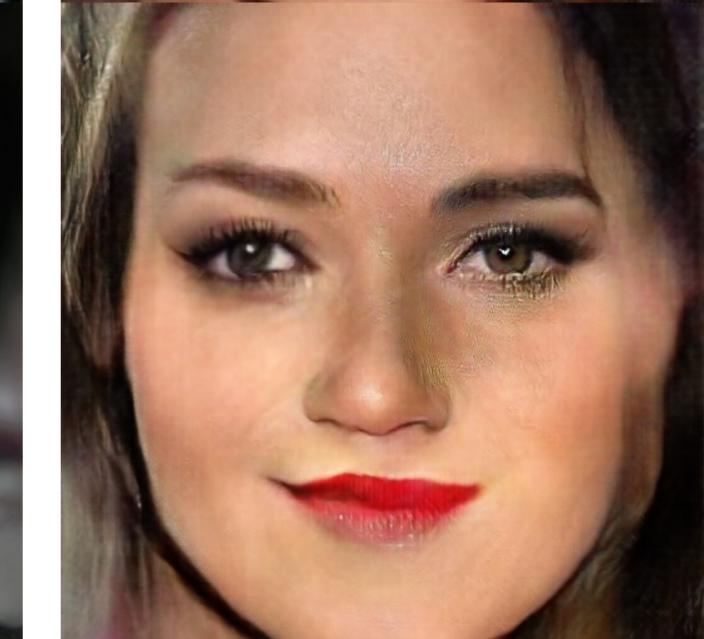
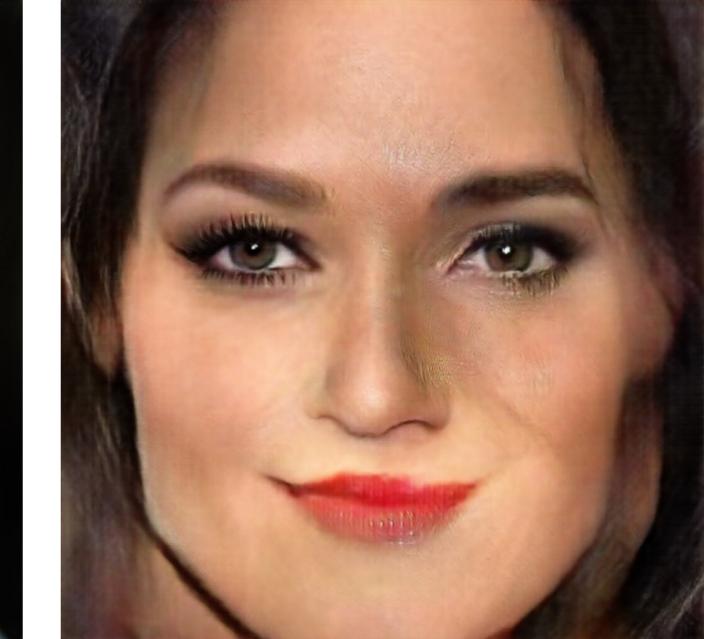
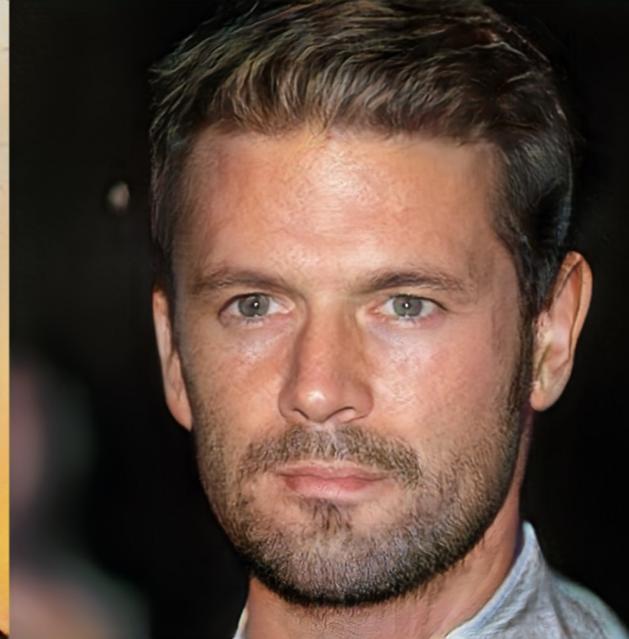


Figure 8: Comparison on the NYU dataset [40]. Our method generates more realistic and colorful images than the other methods.

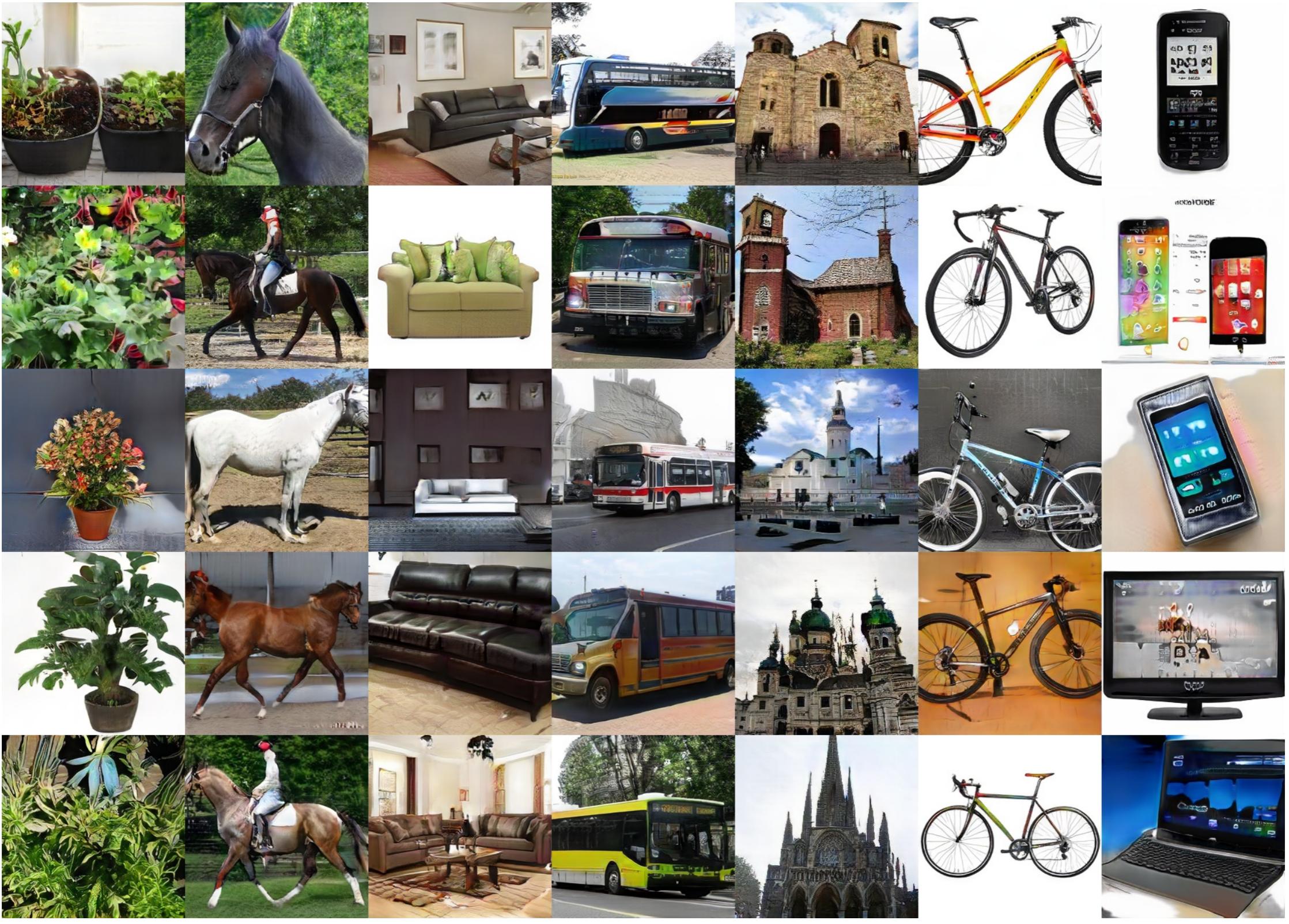
# Progressive GANs [Karras et al]



# Progressive GAN [Karras et al]



# Progressive GANs



POTTEDPLANT

HORSE

SOFA

BUS

CHURCHOUTDOOR

BICYCLE

TVMONITOR

# Autoregressive Models

- So far, we have seen models that attempt to estimate a density of the input domain  $x \in \mathbb{R}^n$

$$p(x) = \int p(h)p(x|h)dh , \quad p(x|h) = \exp(\langle \theta_h, \Phi(x) \rangle - A(\theta_h))$$

$$p(x) = p_0(\Phi(x)) \cdot |\det \nabla \Phi(x)|^{-1}$$

# Autoregressive Models

- So far, we have seen models that attempt to estimate a density of the input domain

$$x \in \mathbb{R}^n$$

$$p(x) = \int p(h)p(x|h)dh , \quad p(x|h) = \exp(\langle \theta_h, \Phi(x) \rangle - A(\theta_h))$$

$$p(x) = p_0(\Phi(x)) \cdot |\det \nabla \Phi(x)|^{-1}$$

- Chained Bayes Rule: for any ordering  $(x_{\sigma(1)}, \dots, x_{\sigma(n)})$  of the coordinates we have

$$p(x) = \prod_{i \leq n} p(x_{\sigma(i)} | x_{\sigma(1)} \dots x_{\sigma(i-1)})$$

# Autoregressive Models

- So far, we have seen models that attempt to estimate a density of the input domain  $\mathbf{x} \in \mathbb{R}^n$

$$p(\mathbf{x}) = \int p(h)p(\mathbf{x}|h)dh , \quad p(\mathbf{x}|h) = \exp(\langle \theta_h, \Phi(\mathbf{x}) \rangle - A(\theta_h))$$

$$p(\mathbf{x}) = p_0(\Phi(\mathbf{x})) \cdot |\det \nabla \Phi(\mathbf{x})|^{-1}$$

- Chained Bayes Rule: for any ordering  $(x_{\sigma(1)}, \dots, x_{\sigma(n)})$  of the coordinates we have

$$p(\mathbf{x}) = \prod_{i \leq n} p(x_{\sigma(i)} | x_{\sigma(1)} \dots x_{\sigma(i-1)})$$

- Q: In which situations is it better to use the factorized?

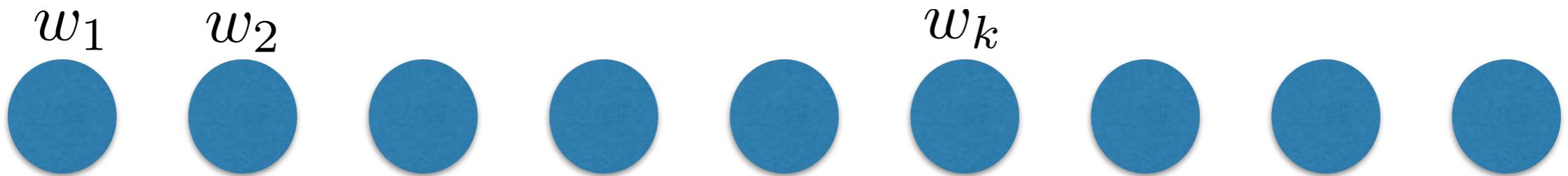
# Autoregressive Models

- Time Series
  - Speech, Music
  - Video
  - Language
  - Other time series (Weather, Finance, ...)
- Spatially ordered data, Multi-Resolution data
  - Images
- Learning is thus reduced to the problem of conditional prediction.

$$p(x) \rightarrow \{p(x_i | x_{N(i)})\}_i$$

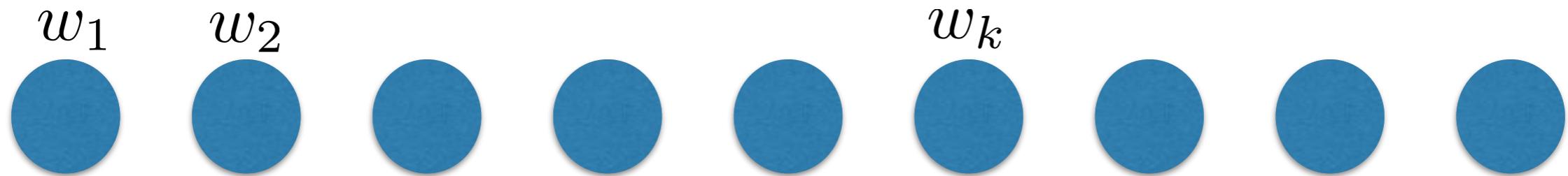
# Word2vec [Mikolov et al.'13].

- Unsupervised learning “success story”.



# Word2vec [Mikolov et al.'13].

- Unsupervised learning “success story”.

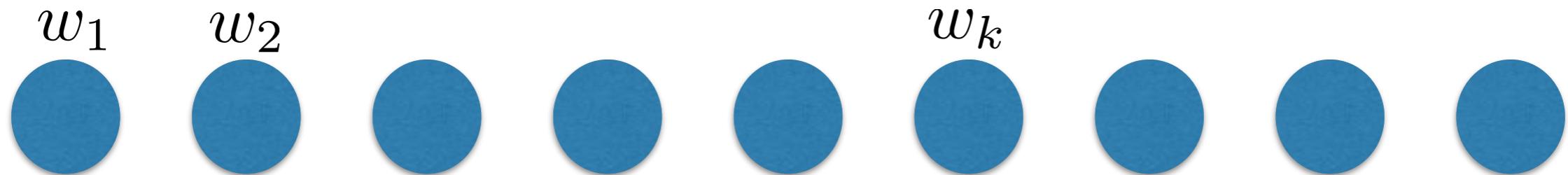


- Language creates a notion of similarity between words:

words  $w_1$ ,  $w_2$  are similar if they are “exchangeable”  
i.e., they appear often within the same context.

# Word2vec [Mikolov et al.'13].

- Unsupervised learning “success story”.



- Language creates a notion of similarity between words:

words  $w_1, w_2$  are similar if they are “exchangeable”  
i.e., they appear often within the same context.

- Goal: find a word representation  $\Phi(w_i) \in \mathbb{R}^d$  that expresses this similarity as a dot product

$$\text{sim}(w_i, w_j) \approx \langle \Phi(w_i), \Phi(w_j) \rangle .$$

# Word2vec [Mikolov et al.'13].

- Main idea: Skip-gram with negative sampling.
- Construct a “training set”
  - positive pairs  $\mathcal{D} = \{(w_k, c_k)\}_k$  of (words, contexts) appearing in a huge language corpus.
  - negative pairs  $\mathcal{D}' = \{(w_{k'}, c_{k'})\}_{k'}$  of (words, contexts) not appearing in the corpus.

# Word2vec [Mikolov et al.'13].

- Main idea: Skip-gram with negative sampling.
- Construct a “training set”
  - positive pairs  $\mathcal{D} = \{(w_k, c_k)\}_k$  of (words, contexts) appearing in a huge language corpus.
  - negative pairs  $\mathcal{D}' = \{(w_{k'}, c_{k'})\}_{k'}$  of (words, contexts) not appearing in the corpus.
- Model the probability of a pair  $(w, c)$  being positive as

$$p(D = 1|c, w) = \sigma(\langle v_w, v_c \rangle), \quad v_w, v_c \in \mathbb{R}^d.$$
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

# Word2vec [Mikolov et al.'13].

- Main idea: Skip-gram with negative sampling.
- Construct a “training set”
  - positive pairs  $\mathcal{D} = \{(w_k, c_k)\}_k$  of (words, contexts) appearing in a huge language corpus.
  - negative pairs  $\mathcal{D}' = \{(w_{k'}, c_{k'})\}_{k'}$  of (words, contexts) not appearing in the corpus.
- Model the probability of a pair  $(w, c)$  being positive as
 
$$p(D = 1|c, w) = \sigma(\langle v_w, v_c \rangle), \quad v_w, v_c \in \mathbb{R}^d.$$
- Training with Maximum Likelihood:
 
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

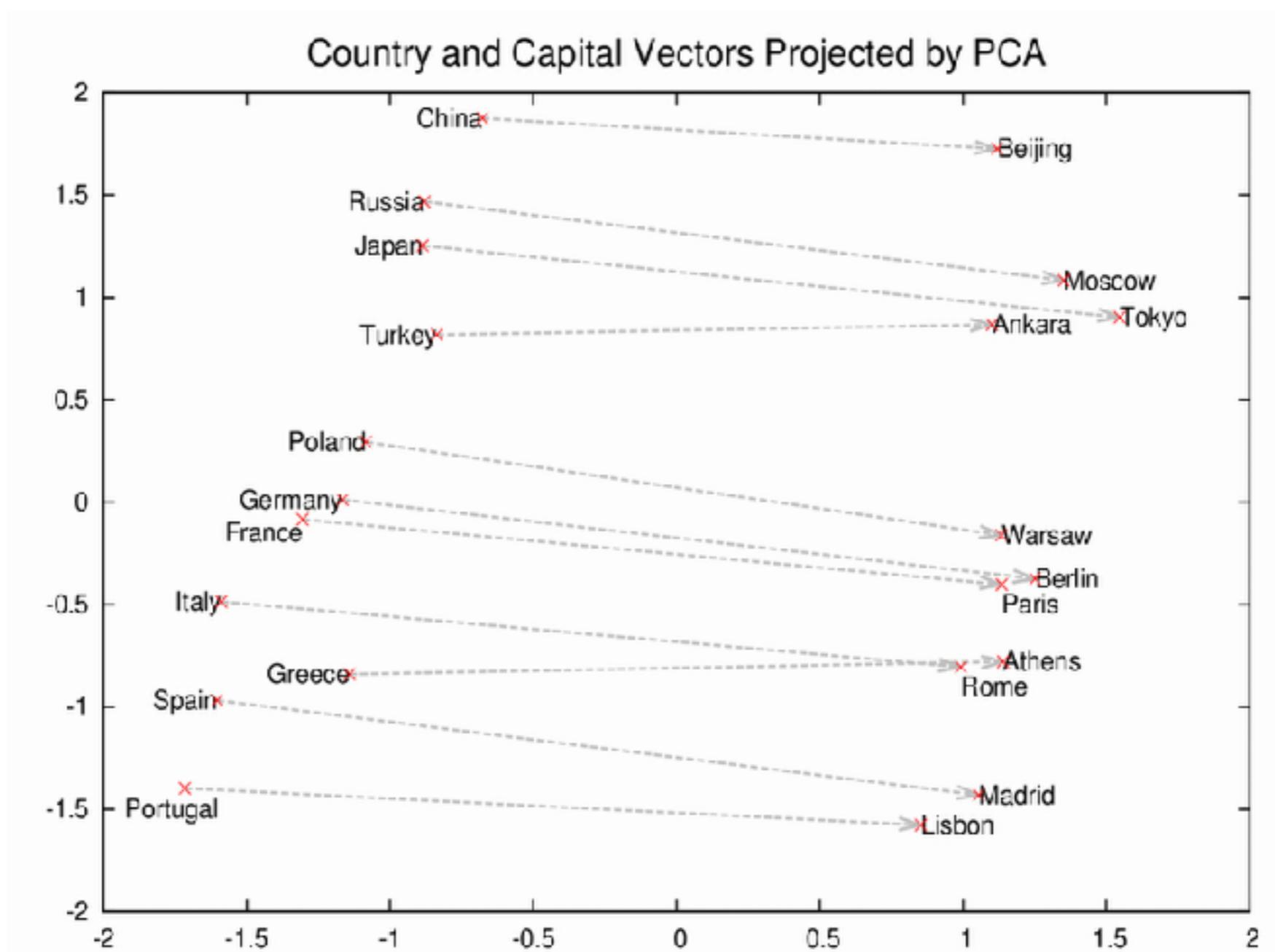
$$\arg \max_{\theta} \prod_{(w,c) \sim \mathcal{D}} p(D = 1|c, w, \theta) \prod_{(w,c) \sim \mathcal{D}'} p(D = 0|c, w, \theta)$$

$$\arg \max_{\theta} \sum_{(w,c) \sim \mathcal{D}} \log \sigma(\langle v_w, v_c \rangle) + \sum_{(w,c) \sim \mathcal{D}'} \log \sigma(-\langle v_w, v_c \rangle)$$

$\mathcal{D}$  : positive contexts       $\mathcal{D}'$  : negative contexts

# Word2vec [Mikolov et al.'13].

- Can be seen as an implicit matrix factorization using a mutual information criteria [Yoav & Goldberg,'14].
- Huge impact on Google's business bottom-line.



# Video Prediction

- Rather than modeling the density of natural images

$$p(x) , \quad x \in \mathbb{R}^d$$

we may be also interested in modeling the conditional distributions

$$p(x_{t+1}|x_1, \dots, x_t)$$

where  $(x_t)_t$  is temporally ordered data.

# Video Prediction

- Rather than modeling the density of natural images

$$p(x) , \quad x \in \mathbb{R}^d$$

we may be also interested in modeling the conditional distributions  
where  $(x_t)_t$  is temporally ordered data.  $p(x_{t+1}|x_1, \dots, x_t)$

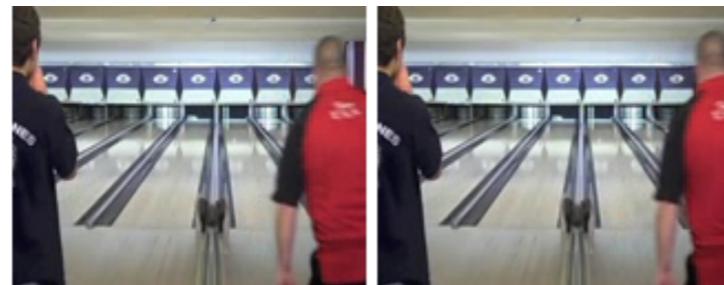
- Similarly, can we find a signal representation  $\Phi(x_t)$  that is consistent with the “video language” metric? i.e.

$$\langle \Phi(x_t), \Phi(x_s) \rangle \approx h(|t - s|)$$

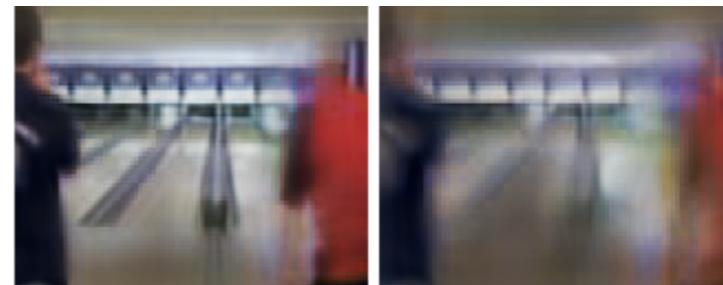
- This is the objective of Slow Feature Analysis [Sejnowski et al'02, Cadieu & Olshausen'10 and many others].

# Video Prediction

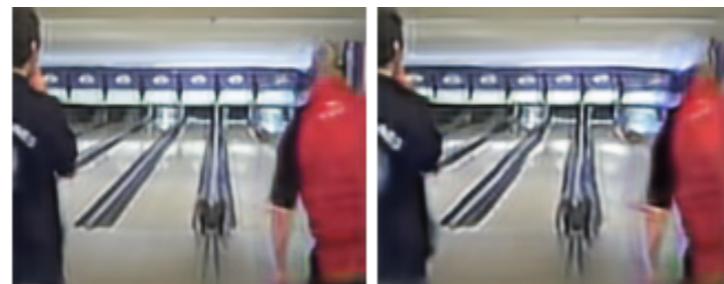
- [Mathieu, Couarie, LeCun, '16]: Conditional video prediction using CNNs and an adversarial cost



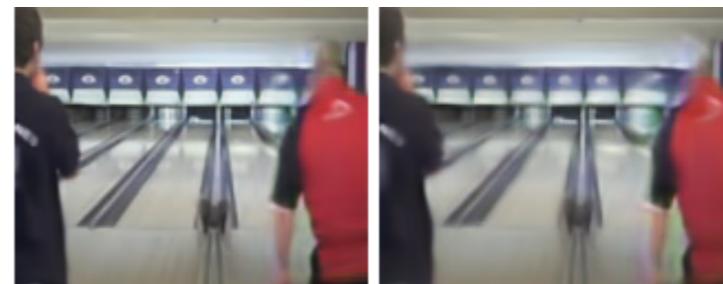
Ground truth



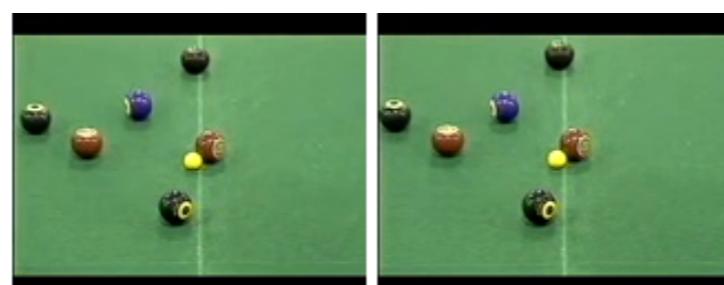
$\ell_2$  result



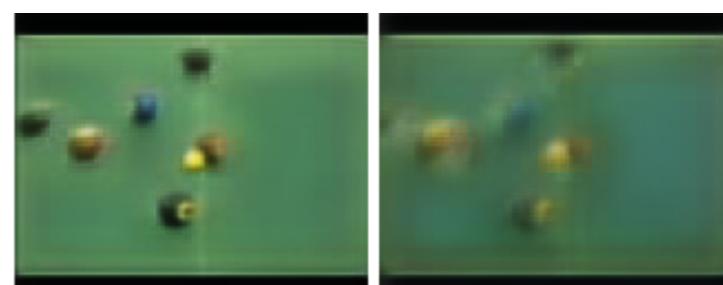
Adversarial result



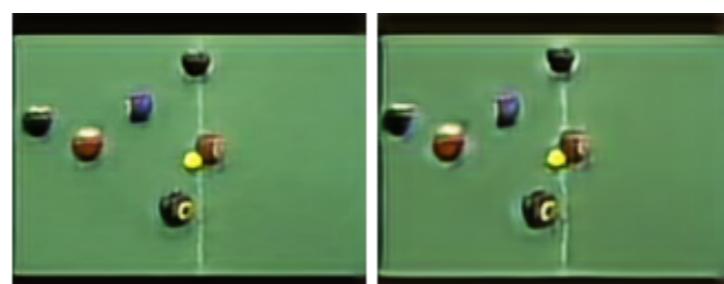
Adversarial+GDL result



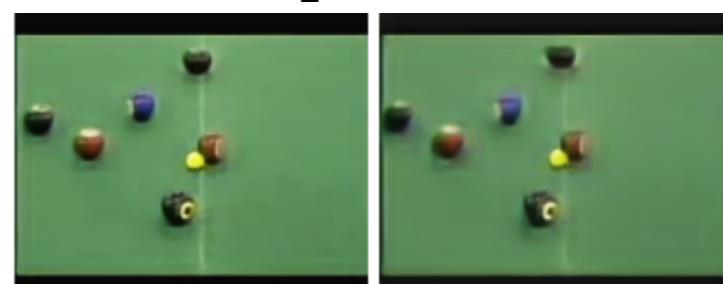
Ground truth



$\ell_2$  result



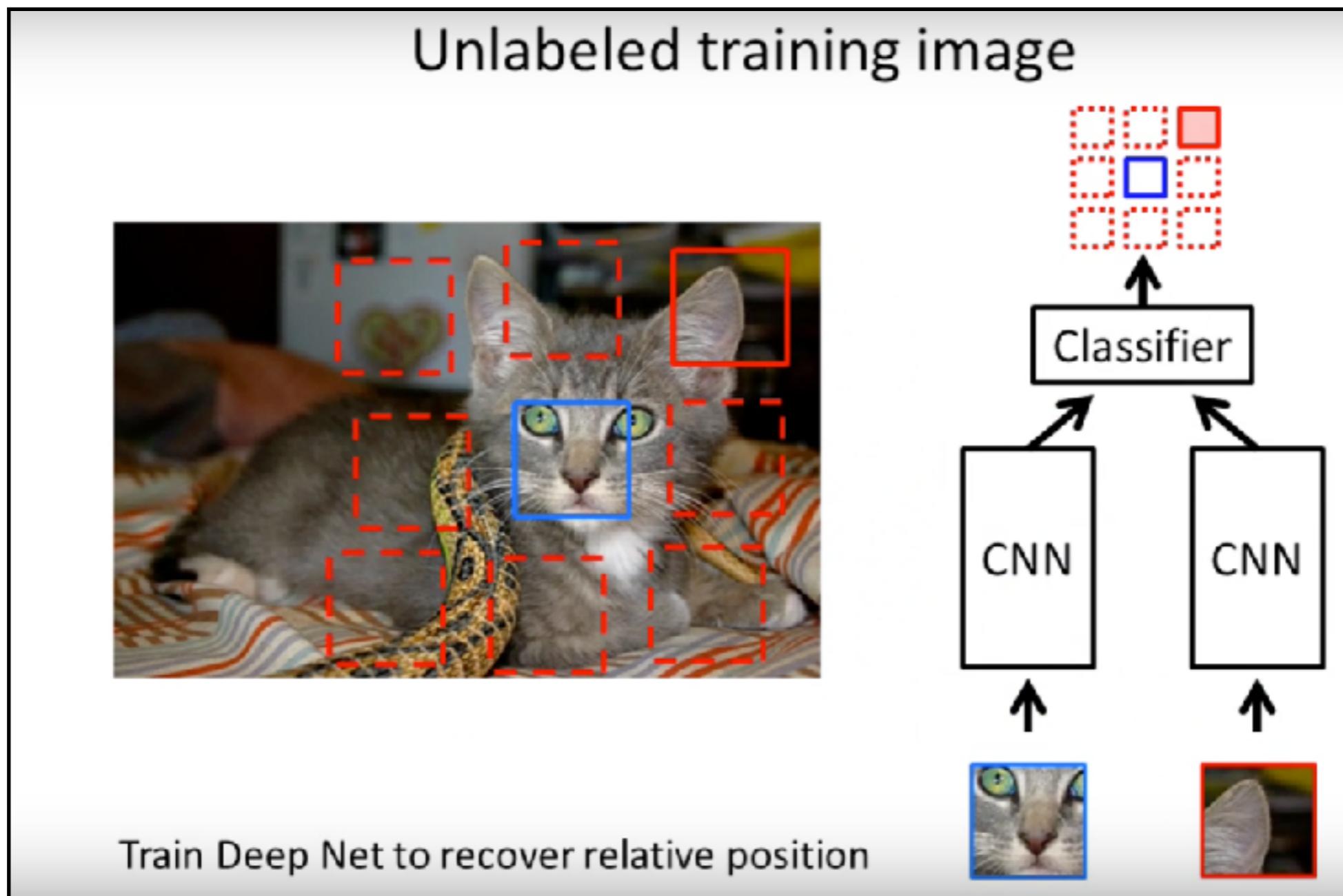
Adversarial result



Adversarial+GDL result

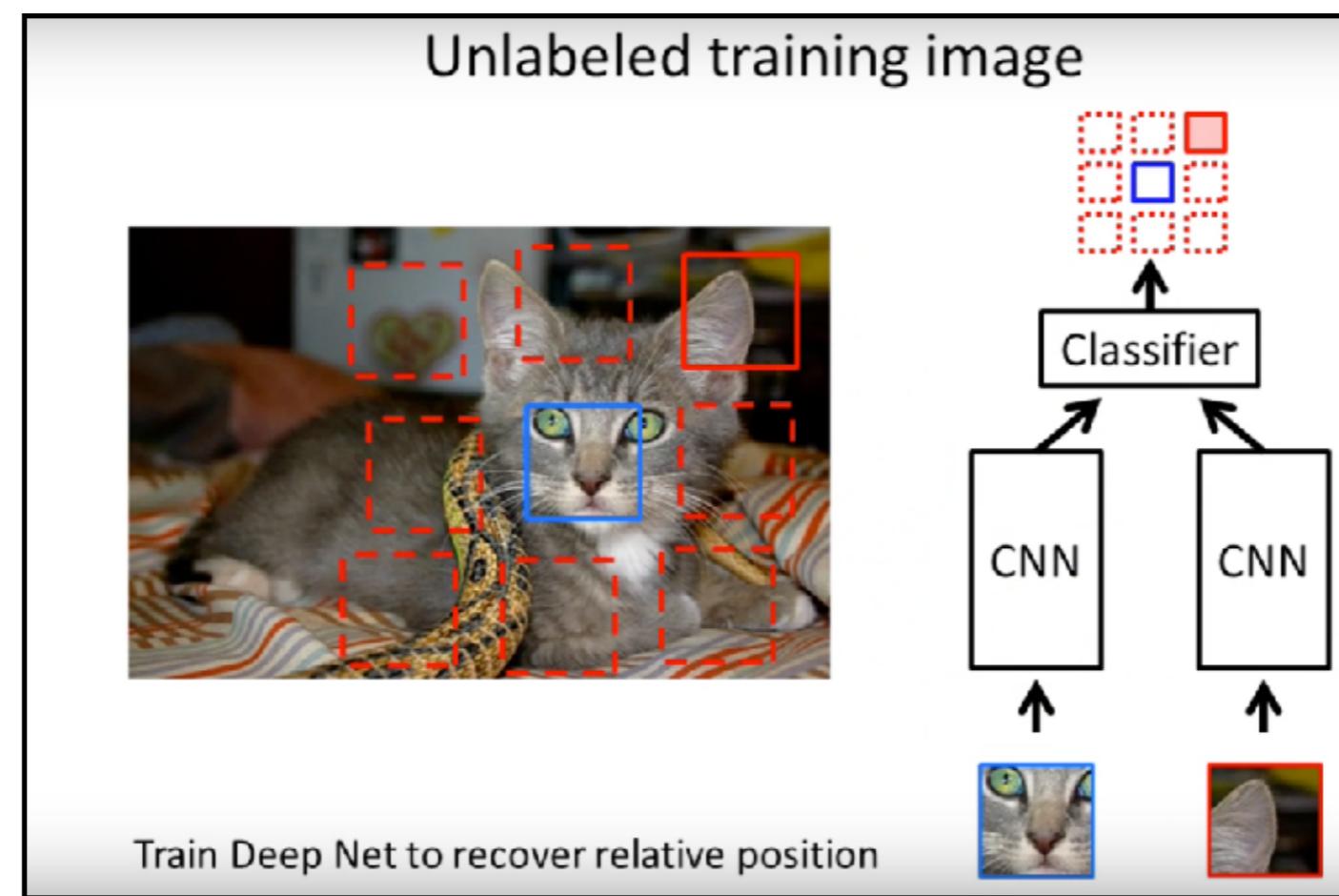
# Patch Relative Configuration [Doerch et al.'15]

- Generalize the idea of positive, negative pairs to a multi-class classification problem about spatial configurations.



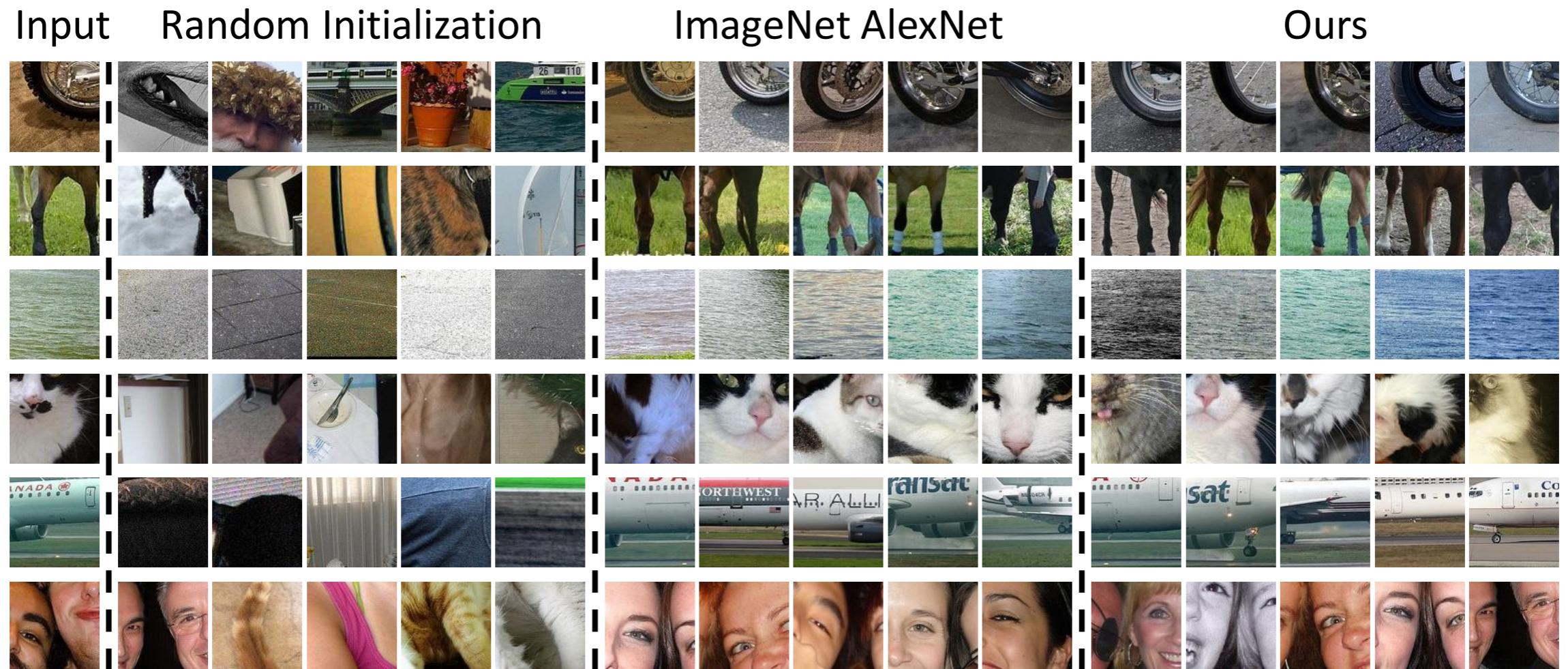
# Patch Relative Configuration [Doerch et al.'15]

- Premise: A patch representation  $\Phi(x)$  that does well in this task indirectly builds object priors.
- The criterion is not generative, but it retains enough information to generalize to other tasks



# Patch Relative Configuration [Doerch et al.'15]

- Retrieval tasks:



- The representation captures visual similarity, leveraged in object detection, retrieval, etc.

# Pixel Recurrent Networks [v.d.Oord et al'16]

- Prediction tasks of the form  $\hat{x}_{t+1} = F(x_1, \dots, x_t)$  require a loss or an associated likelihood

e.g.  $\|\hat{x}_{t+1} - x_{t+1}\|^2 \Leftrightarrow p(x_{t+1}|x_1, \dots, x_t) = \mathcal{N}(F(x_1, \dots, x_t), I)$

- In discrete domains we simply use a multinomial loss, in continuous domains there is no principled choice.

- How about images?

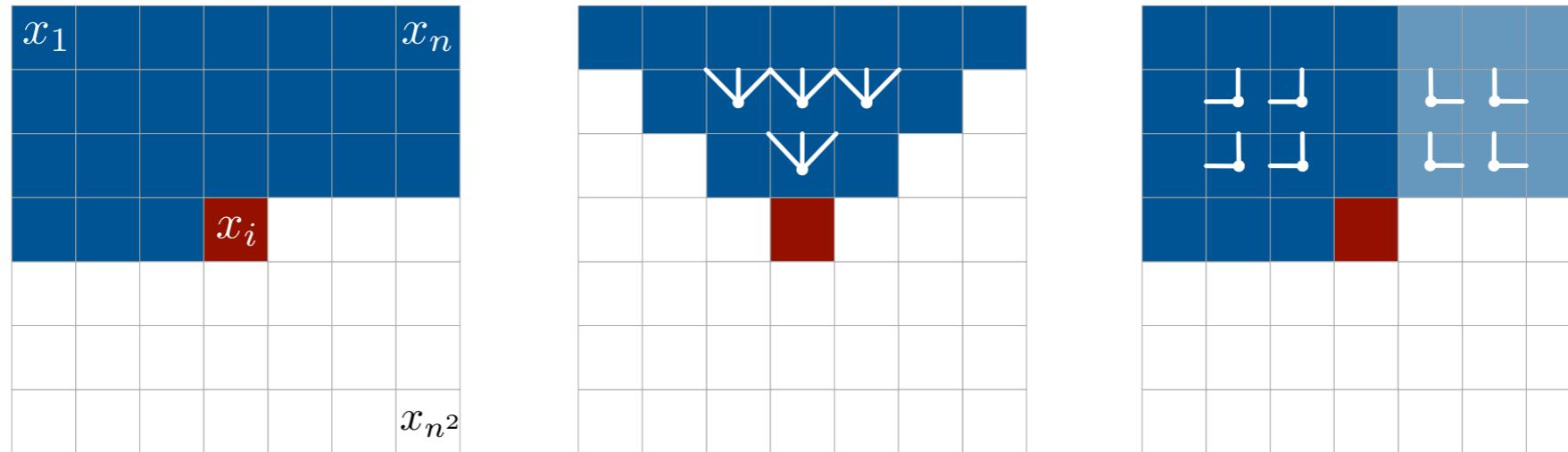
# Pixel Recurrent Networks [v.d.Oord et al'16]

- Prediction tasks of the form  $x_{t+1}^{\hat{}} = F(x_1, \dots, x_t)$  loss or an associated likelihood
  - e.g.  $\|\hat{x}_{t+1} - x_{t+1}\|^2 \Leftrightarrow p(x_{t+1}|x_1, \dots, x_t) = \mathcal{N}(F(x_1, \dots, x_t), I)$
- In discrete domains we simply use a multinomial loss, in continuous domains there is no principled choice.
- How about images?
  - We can treat them as discrete two-dimensional grids  
 $x(u) \in \{0, 255\}$
  - Model each pixel from its “past” context:

$$p(x(u)|x(v); v \in \Omega(u)) = \text{softmax}(\Phi(x, \Omega(u)))$$

# Pixel Recurrent Networks [v.d.Oord et al'16]

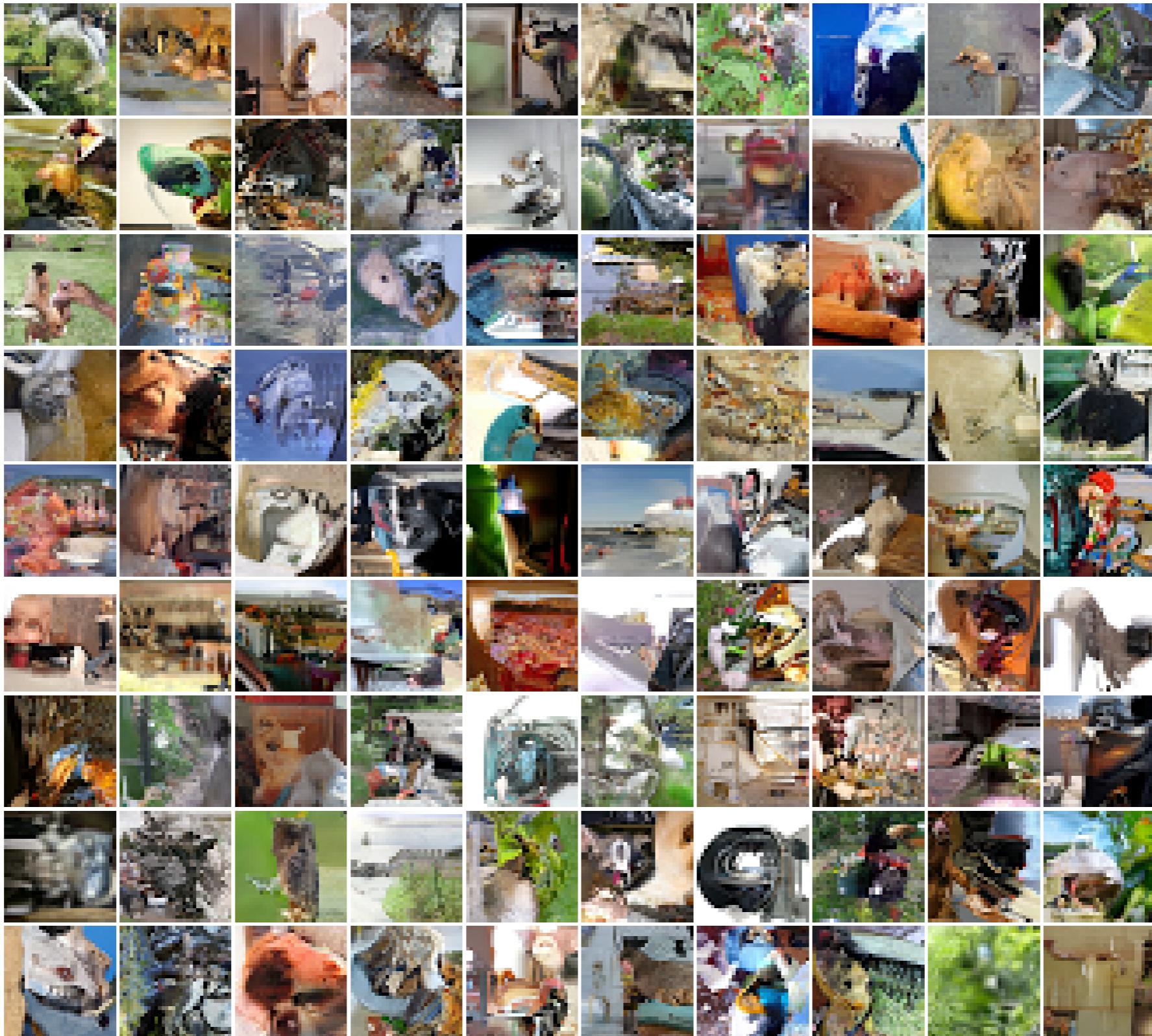
- Contexts are modeled using “diagonal BiLSTMS”.



- Multi-Scale architecture conditions generations upon low-resolution samples (similarly as in LAPGANs).
- Very deep Recurrent Networks (> 10 layers).

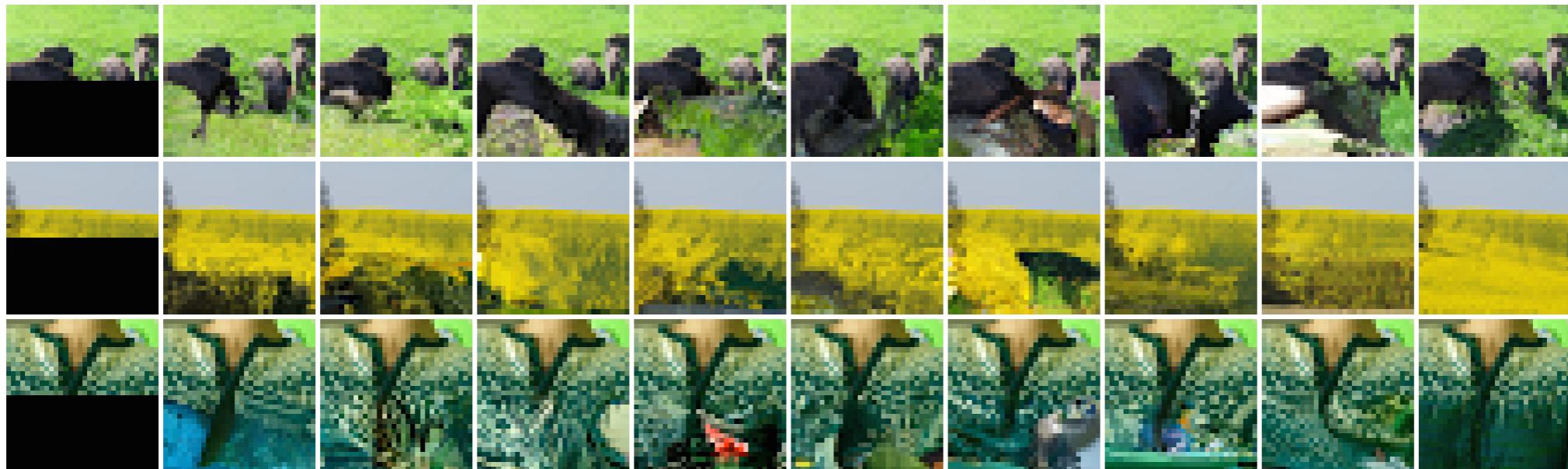
# Pixel Recurrent Networks [v.d.Oord et al'16]

- state-of-the-art image generation and modeling.



# Pixel Recurrent Networks [v.d.Oord et al'16]

occluded



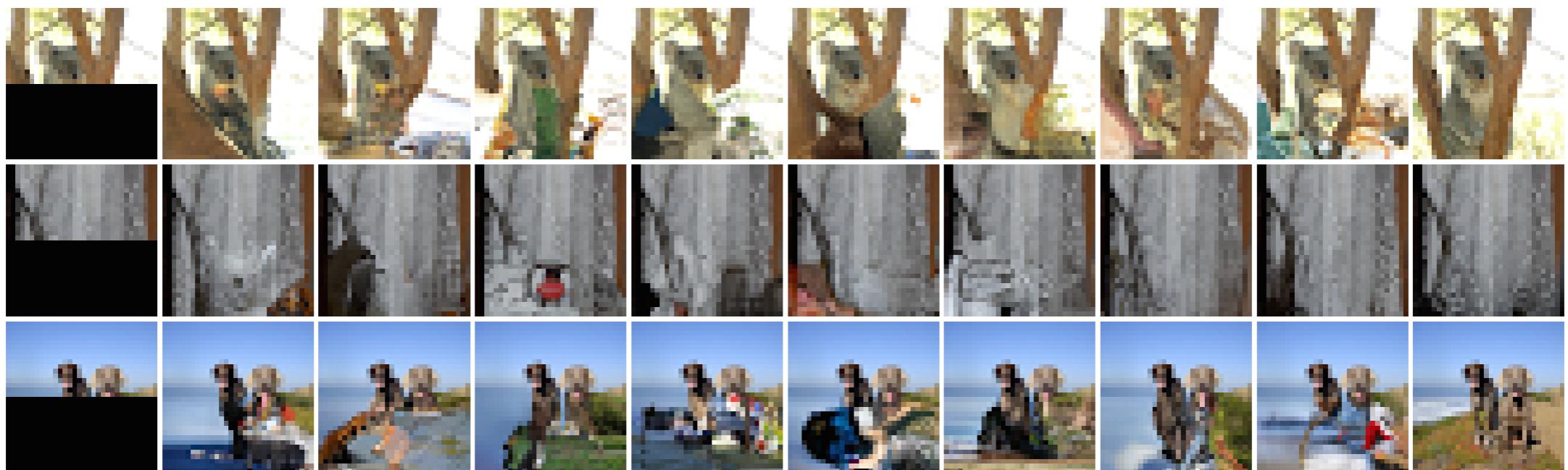
completions

original

occluded

completions

original



# Pixel Recurrent Networks [v.d.Oord et al'16]

- MNIST and Cifar-10 log-likelihoods:

Model	NLL Test
DBM 2hl [1]:	$\approx 84.62$
DBN 2hl [2]:	$\approx 84.55$
NADE [3]:	88.33
EoNADE 2hl (128 orderings) [3]:	85.10
EoNADE-5 2hl (128 orderings) [4]:	84.68
DLGM [5]:	$\approx 86.60$
DLGM 8 leapfrog steps [6]:	$\approx 85.51$
DARN 1hl [7]:	$\approx 84.13$
MADE 2hl (32 masks) [8]:	86.64
DRAW [9]:	$\leq 80.97$
Diagonal BiLSTM (1 layer, $h = 32$ ):	<b>80.75</b>
Diagonal BiLSTM (7 layers, $h = 16$ ):	<b>79.20</b>

Table 4. Test set performance of different models on MNIST in *nats* (negative log-likelihood). Prior results taken from [1] (Salakhutdinov & Hinton, 2009), [2] (Murray & Salakhutdinov, 2009), [3] (Uria et al., 2014), [4] (Raiko et al., 2014), [5] (Rezende et al., 2014), [6] (Salimans et al., 2015), [7] (Gregor et al., 2014), [8] (Germain et al., 2015), [9] (Gregor et al., 2015).

Model	NLL Test (Train)
Uniform Distribution:	8.00
Multivariate Gaussian:	4.70
NICE [1]:	4.48
Deep Diffusion [2]:	4.20
Deep GMMs [3]:	4.00
RIDE [4]:	3.47
PixelCNN:	3.14 (3.08)
Row LSTM:	3.07 (3.00)
Diagonal BiLSTM:	<b>3.00</b> (2.93)

Table 5. Test set performance of different models on CIFAR-10 in *bits/dim*. For our models we give training performance in brackets. [1] (Dinh et al., 2014), [2] (Sohl-Dickstein et al., 2015), [3] (van den Oord & Schrauwen, 2014a), [4] personal communication (Theis & Bethge, 2015).

- Recently, extension to video and speech (see [deepmind.com](http://deepmind.com) for details).

# Limits of Transportation Models

- Direct learning by Optimizing the flow requires back propagation through a term of the form

$$f(\Theta) = \log \det \nabla \Phi(x_i; \Theta)$$

- Very expensive for generic transformations  $\Phi$
- Highly specific flows affect the flexibility of the model.
- Indirect learning by the Discriminative Adversarial Training is implicit
  - No cheap way to evaluate the density  $p(x)$
  - Also, no cheap way to do inference, e.g.  $p(z|x)$
- How to regularize the density estimation?

# Gibbs Models

- Motivation: Given a collection of discriminative measurements  
 $\Phi(x) = \{\Phi_j(x)\}_j$  how can we build a generative model?

# Gibbs Models

- Motivation: Given a collection of discriminative measurements  $\Phi(x) = \{\Phi_j(x)\}_j$  how can we build a generative model?
- Supervised Learning Setup:

Empirical training distribution  $\hat{p} : \{(x_i, y_i)\} \quad y_i \in \{1, K\}$

Empirical class-conditional moments:

$$\mu_k = \mathbb{E}_{(x,y) \sim \hat{p}}(\Phi(x) | y = k) \quad k = 1 \dots K$$

# Gibbs Models

- Motivation: Given a collection of discriminative measurements ,  $\Phi(x) = \{\Phi_j(x)\}_j$  how can we build a generative model?
- Supervised Learning Setup:  
Empirical training distribution  $\hat{p} : \{(x_i, y_i)\} \quad y_i \in \{1, K\}$   
Empirical class-conditional moments:  
$$\mu_k = \mathbb{E}_{(x,y) \sim \hat{p}}(\Phi(x)|y = k) \quad k = 1 \dots K$$
- Necessary condition:  
Class-conditional models  $p_k(x)$  satisfy  
$$\forall k, \mathbb{E}_{x \sim p_k} \Phi(x) = \mu_k$$
  
Q: Does this completely specify  $p_k$ ?

# Gibbs Models

- Motivation: Given a collection of discriminative measurements ,  $\Phi(x) = \{\Phi_j(x)\}_j$  how can we build a generative model?
- Supervised Learning Setup:  
Empirical training distribution  $\hat{p} : \{(x_i, y_i)\} \quad y_i \in \{1, K\}$   
Empirical class-conditional moments:  
$$\mu_k = \mathbb{E}_{(x,y) \sim \hat{p}}(\Phi(x)|y = k) \quad k = 1 \dots K$$
- Necessary condition:  
Class-conditional models  $p_k(x)$  satisfy  
$$\forall k, \mathbb{E}_{x \sim p_k} \Phi(x) = \mu_k$$
  
Q: Does this completely specify  $p_k$ ?      Clearly not

# Gibbs Models

- Thus, we need a regularization principle.
- A “good” norm for probability distributions is the entropy

$$H(p) = -\mathbb{E}[\log p] = - \int p(x) \log p(x) dx$$

- It captures a form of smoothness for probability distributions
  - On compact domains, the maximum entropy distribution is the uniform measure (maximally smooth)
  - On non-compact domains, the max-entropy distribution might not exist.
- In our problem, we can use it to select, under the constraints , those with maximum uncertainty (maximum smoothness).

$$\forall k, \mathbb{E}_{x \sim p_k} \Phi(x) = \mu_k$$

# Gibbs Models and Maximum Entropy

- We are thus interested in the problem

$$\begin{aligned} & \max_p H(p) \\ s.t. \quad & \mathbb{E}_{x \sim p} \Phi(x) = \mu \in \mathbb{R}^d \end{aligned}$$

- Constrained optimization that we approach using calculus of variations
- Lagrangian of the problem is

$$L(p, \lambda_1, \dots, \lambda_d) = H(p) + \sum_j \lambda_j (\mathbb{E}_{x \sim p} \Phi_j(x) - \mu_j) .$$

$$= - \int p(x) \log(p(x)) dx + \sum_j \lambda_j \left( \int \Phi_j(x) p(x) dx - \mu_j \right)$$

# Gibbs Models and Maximum Entropy

- Thus we have

$$\frac{\partial L}{\partial p(x)} = -\log p(x) - 1 + \sum_j \lambda_j \Phi_j(x) = 0$$

$$\Rightarrow \log p(x) = \lambda_0 + \sum_j \lambda_j \Phi_j(x)$$

$$\Rightarrow p(x) = \frac{\exp\left(\sum_j \lambda_j \Phi_j(x)\right)}{Z}$$

where

$\lambda_j$  are Lagrange multipliers guaranteeing that  $\mathbb{E}_{x \sim p} \Phi_j(x) = \mu_j$ .

$Z$  is a Lagrange multiplier guaranteeing that  $p(x) = 1$

# Gibbs Model

- Thus, given features  $\Phi(x)$ , maximum entropy distributions are in the exponential family given by

$$p(x) = \exp(\langle \lambda, \Phi(x) \rangle - A(\lambda))$$

- In a discriminative setting, the final model is a mixture in this exponential family:

$$k \sim \text{cat}\{1, K\}$$

$$x \sim p_k(x) = \exp(\langle \lambda_k, \Phi(x) \rangle - A(\lambda_k)) , \quad \mathbb{E}_{x \sim p_k} \Phi(x) = \mu_k .$$

- This model has many names:
  - Gibbs, Boltzmann, “Energy-based” Model, MaxEnt, ...

# Gibbs Model, Method of Moments and MLE

- In a parametric model  $p(x|\theta)$  two main estimation techniques:
  - *Method of Moments*: Match empirical moments with parametric moments:

Given  $F_1, \dots, F_L$ ,  
empirical moments:  $\hat{F}_i = \frac{1}{N} \sum_{j \leq N} F_i(x_j)$   
parametric moments:  $\bar{F}_i(\theta) = \mathbb{E}_{x \sim p(x|\theta)} F_i(x)$

$$\hat{F} = \bar{F}(\hat{\theta}_{MM})$$

- Maximum Likelihood Estimate (MLE)

$$\hat{\theta}_{MLE} = \arg \max_{\theta} \frac{1}{N} \sum_{i \leq N} \log p(x_i|\theta)$$

- How different are these estimators in our setting?

# Maximum Entropy vs MLE

- The Maximum Entropy model matches the moments defined by the sufficient statistics  $\Phi(x)$ :

$$\mathbb{E}_{x \sim p(x|\theta)} \Phi(x) = \hat{\mu} .$$

- The maximum likelihood attempts to maximize data log-likelihood:

$$\begin{aligned} \max_{\theta} \frac{1}{N} \sum_{j \leq N} \log p(x_j | \theta) &= \frac{1}{N} \sum_j \langle \theta, \Phi(x_j) \rangle - A(\theta) \\ &= \max_{\theta} \langle \theta, \frac{1}{N} \sum_j \Phi(x_j) \rangle - A(\theta) \\ &= \max_{\theta} \langle \theta, \hat{\mu} \rangle - A(\theta) \end{aligned}$$

# Maximum Entropy vs MLE

- Recall the conjugate duality:

$$A^*(\mu) = \sup_{\theta} (\langle \theta, \mu \rangle - A(\theta))$$

- $A^*(\mu)$  is the entropy of the distribution with  $\mathbb{E}(\Phi(x)) = \mu$ .
- Thus, the maximum entropy and the maximum likelihood estimators are the same under our exponential family.

# Gibbs Model and Fisher Kernels

- Given a generative model  $p(x|\theta)$  recall the associated Fisher Kernel:

$$U_x : \text{Fisher Vector} = \nabla_{\theta} \log p(x|\theta) .$$

$$I : \text{Fisher Information} = \mathbb{E}\{U_x U_x^T\} .$$

$$K(x, x') = \langle U_x, I^{-1} U_{x'} \rangle ,$$

# Gibbs Model and Fisher Kernels

- Given a generative model  $p(x|\theta)$ , recall the associated Fisher Kernel:

$$U_x : \text{Fisher Vector} = \nabla_{\theta} \log p(x|\theta) .$$

$$I : \text{Fisher Information} = \mathbb{E}\{U_x U_x^T\} .$$

- When  $K(x, x') = \langle U_x, I^{-1}U_{x'} \rangle$ , the Fisher vector is

$$p(x|\theta) = \exp(\langle \theta, \Phi(x) \rangle - A(\theta))$$

$$U_x = \Phi(x)$$

$$K(x, x') = \langle \tilde{\Phi}(x), \tilde{\Phi}(x') \rangle , \quad (\tilde{\Phi} : \text{whitened features}) .$$

# Gibbs Model and Fisher Kernels

- Given a generative model  $p(x|\theta)$ , recall the associated Fisher Kernel:

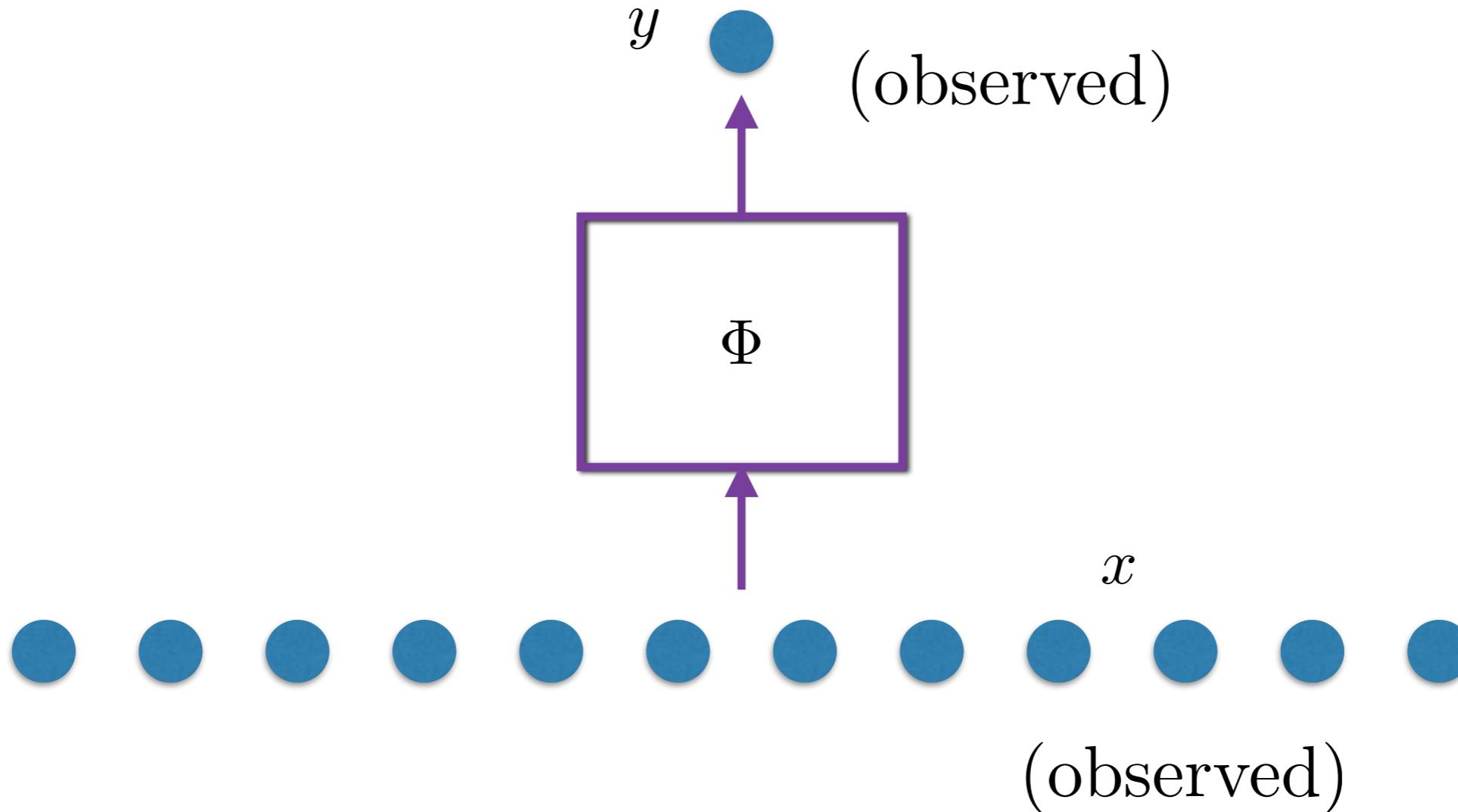
$$U_x : \text{Fisher Vector} = \nabla_{\theta} \log p(x|\theta) .$$

$$I : \text{Fisher Information} = \mathbb{E}\{U_x U_x^T\} .$$

- When  $K(x, x') = \langle U_x, I^{-1} U_{x'} \rangle$ , the Fisher vector is  
$$p(x|\theta) = \exp(\langle \theta, \Phi(x) \rangle - A(\theta))$$
  
$$U_x = \Phi(x) \quad K(x, x') = \langle \tilde{\Phi}(x), \tilde{\Phi}(x') \rangle , \quad (\tilde{\Phi} : \text{whitened features})$$
- Thus the maximum entropy model is the “canonical” generative model associated with linearization features  $\Phi$

# Unsupervised Gibbs Models

- We have derived a fully probabilistic model in a supervised setting:



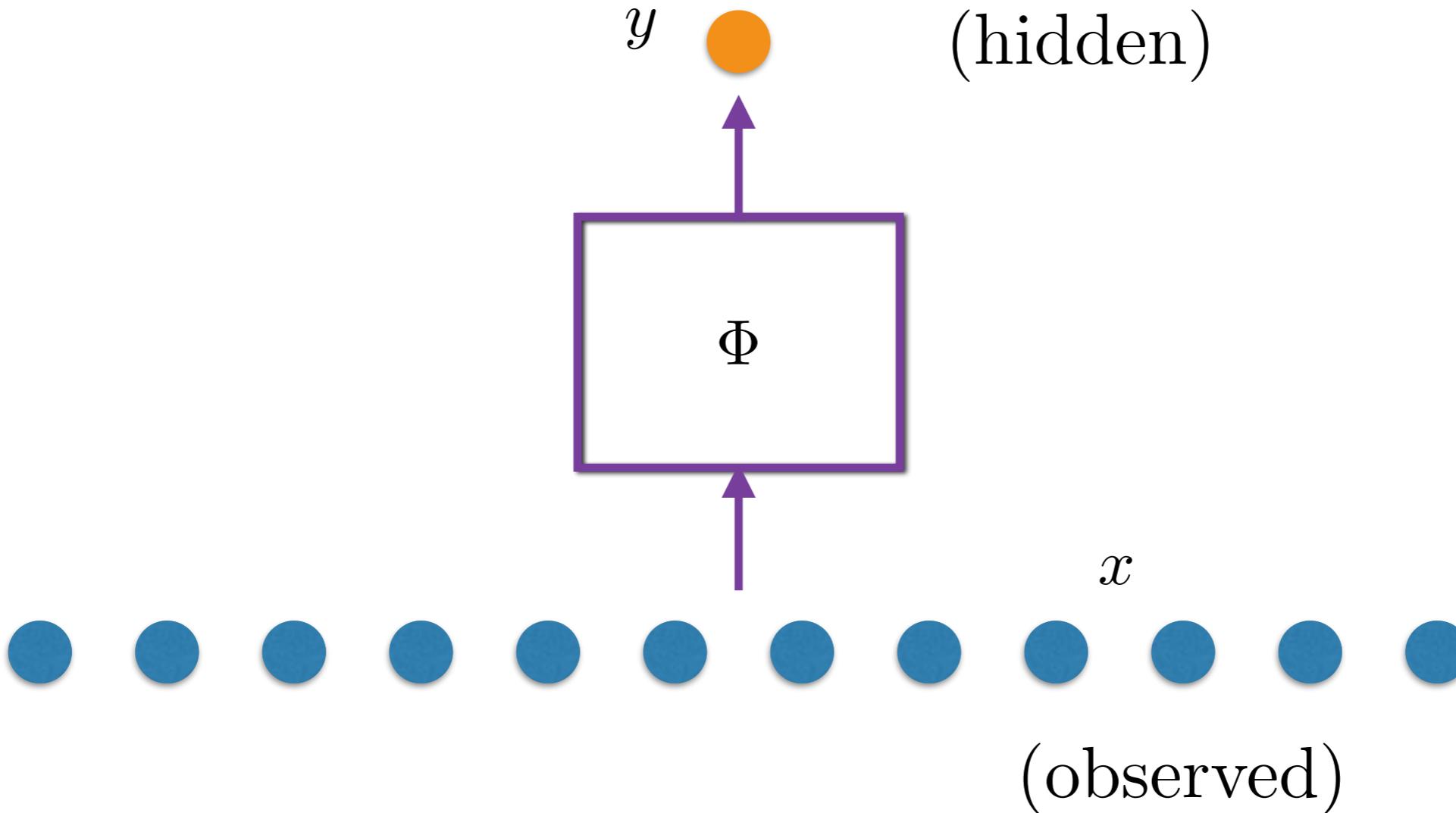
$$y \sim \text{cat}\{1, K\}$$

$$p(x|y) = \exp(\langle \theta_y, \Phi(x) \rangle - A(\theta_y))$$

$$\Rightarrow p(y|x) = \text{softmax}(\{\langle \theta_y, \Phi(x) \rangle\}_y) .$$

# Unsupervised Gibbs Models

- How about when no labels are observed?



$$y \sim \text{mult}(\pi)$$

$$p(x|y) = \exp (\langle \theta_y, \Phi(x) \rangle - A(\theta_y))$$

$$p(x) = \sum_y p(y)p(x|y)$$

# Gibbs Learning

- Q: How to train this model?
  - When adjusting expected values (Method of Moments), find Lagrange multipliers.
  - Equivalently, maximize log-likelihood.
  - Also learn the sufficient statistics?

# Gibbs Learning

- The log-likelihood is  $\log p(x|\theta) = \langle \theta, \Phi(x) \rangle - A(\theta)$
- The gradient with respect to  $\theta$  is

$$\nabla_{\theta} \log p(x|\theta) = \Phi(x) - \nabla_{\theta} A(\theta)$$

# Gibbs Learning

- The log-likelihood is  $\log p(x|\theta) = \langle \theta, \Phi(x) \rangle - A(\theta)$
- The gradient with respect to  $\theta$  is
- We have
  - $\nabla_\theta \log p(x|\theta) = \Phi(x) - \nabla_\theta A(\theta)$
  - $\nabla_\theta A(\theta) = \nabla_\theta \log \int \exp(\langle \theta, \Phi(x) \rangle) dx$
  - $= \frac{\int \nabla_\theta \exp(\langle \theta, \Phi(x) \rangle) dx}{\int \exp(\langle \theta, \Phi(x) \rangle) dx}$
  - $= \frac{\int \Phi(x) \exp(\langle \theta, \Phi(x) \rangle) dx}{\int \exp(\langle \theta, \Phi(x) \rangle) dx}$
  - $= \mathbb{E}_{x \sim p(x|\theta)} \Phi(x)$
- Thus

$$\nabla_\theta \log p(x_i|\theta) = \Phi(x_i) - \mathbb{E}(\Phi(x))$$

# Gibbs and Markov Chain Monte-Carlo

- We estimate the expectation with a finite sample.
- Training is thus reduced to being able to efficiently sample from distributions of the form

$$p(x) = \exp(\langle \theta, \Phi(x) \rangle - A(\theta))$$

- Markov-Chain Monte-Carlo (MCMC) is a broad family of algorithms doing precisely so.