# Inference and Representation
# Lecture 13

Joan Bruna
Courant Institute, NYU

# Review: Variational Autoencoders

- Recall the variational lower bound:

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

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

- Can we optimize jointly both generative and variational parameters efficiently?

- For appropriate posterior approximations, we can reparametrize samples as

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

# Review: 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

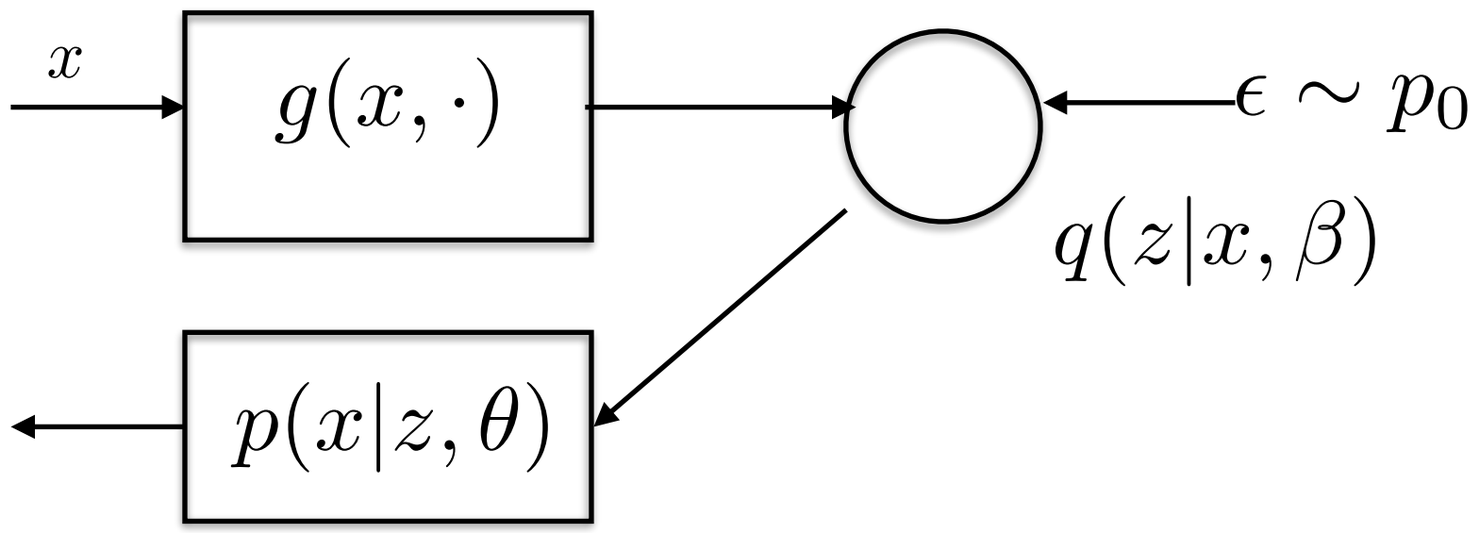
$$\widehat{\mathcal{L}(\theta, \beta, X)} = -D_{KL}(q_\beta(z|X)||p_\theta(z)) + \frac{1}{S}\sum_{s \leq S} \log p(X|z^{(s)}, \theta)$$

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

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

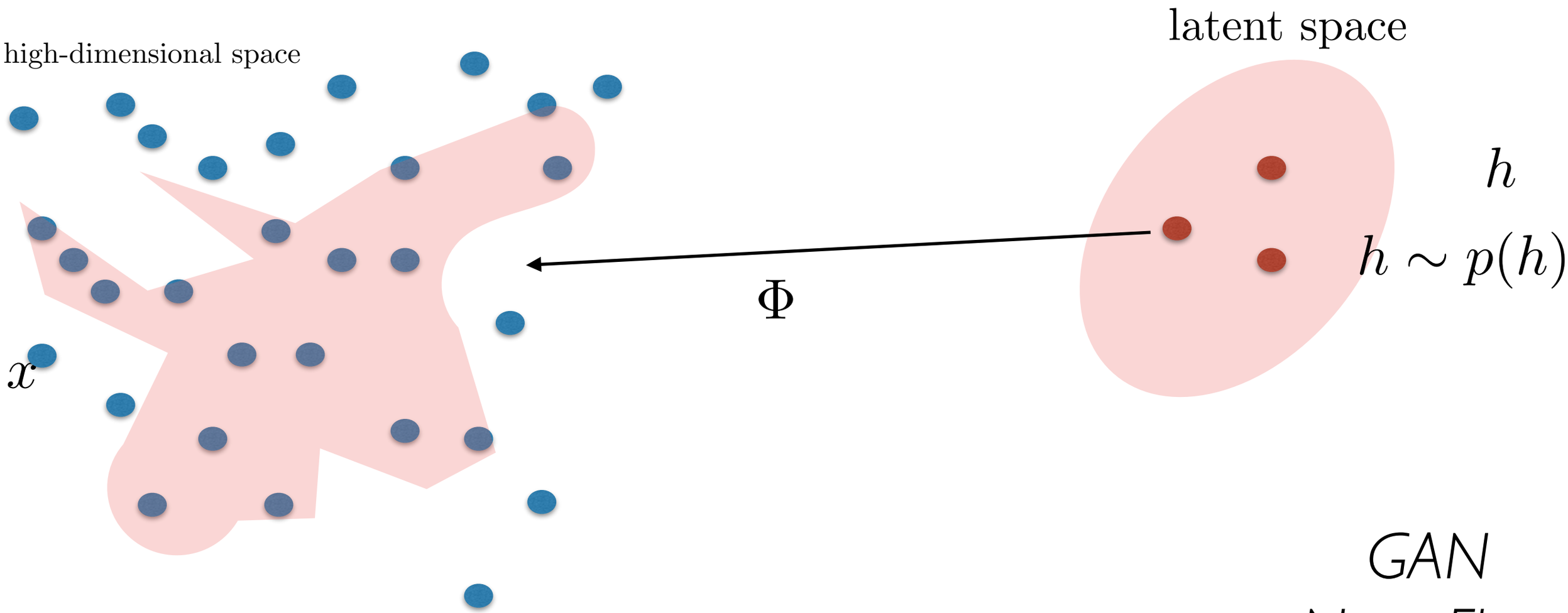
# Review: Variational Autoencoders

- VAE idea: use neural networks to approximate variational and generative parameters.



# Review: Generative Models of Complex data

- Flows or Transports of Measure




*GAN*
*NormFlow*
...

$p(x)$ defined implicitly with

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

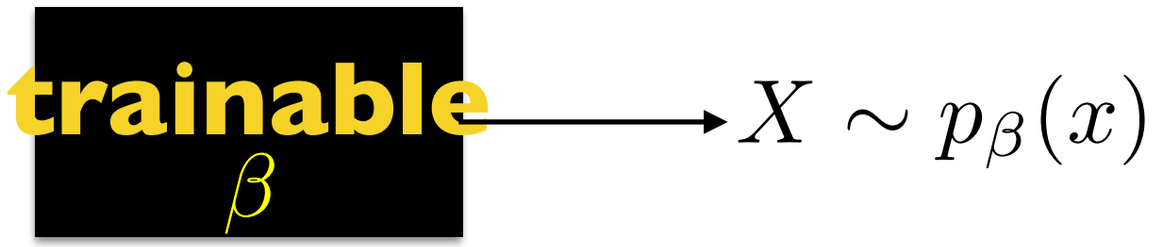
# Today's Lecture

- Generative Adversarial Models
- Unsupervised Gibbs Models
- Autoregressive Models

# Generative Adversarial Networks

*[Goodfellow et al., '14]*

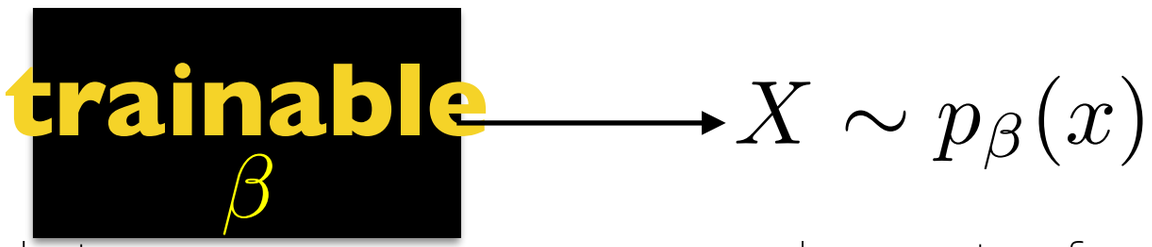
- Suppose we have a *trainable* black box generator:



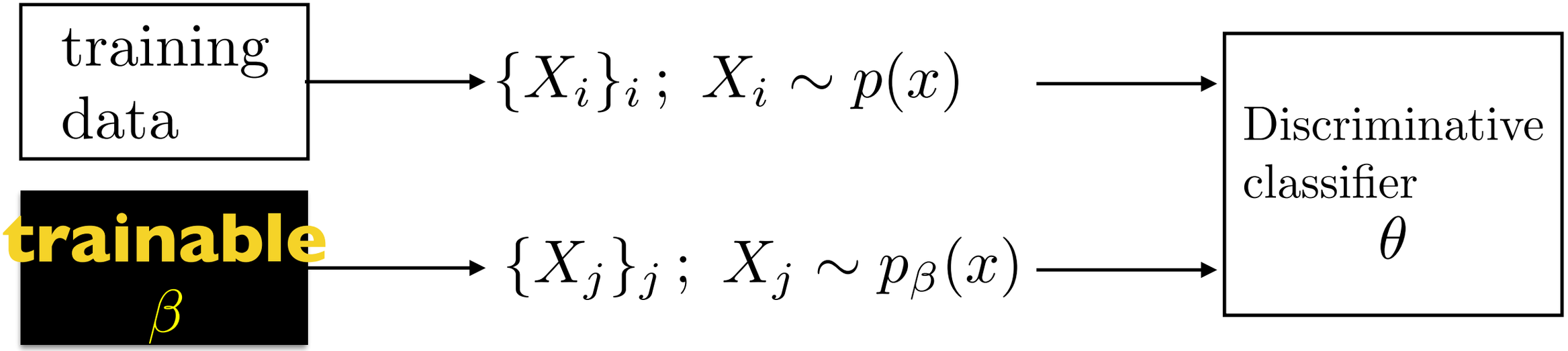
# Generative Adversarial Networks

*[Goodfellow et al., '14]*

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



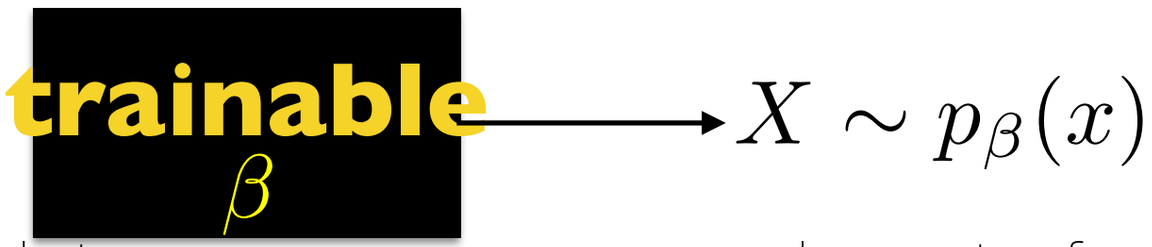

- Given observed data $\{X_i\}_i\,;\;\; X_i \sim p(x)$ how to force our generator to produce samples from $p(x)$ ?



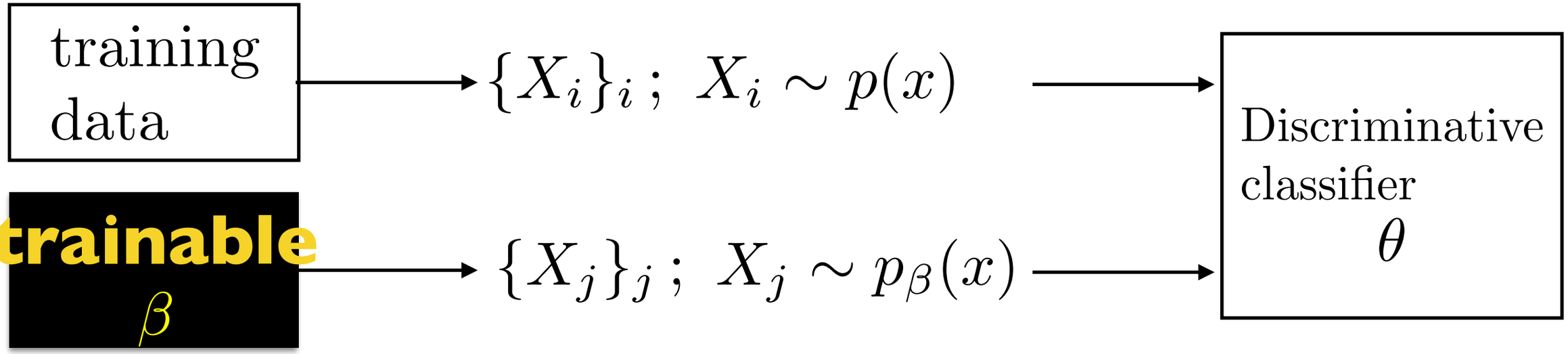
# Generative Adversarial Networks

*[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 $p(x)$

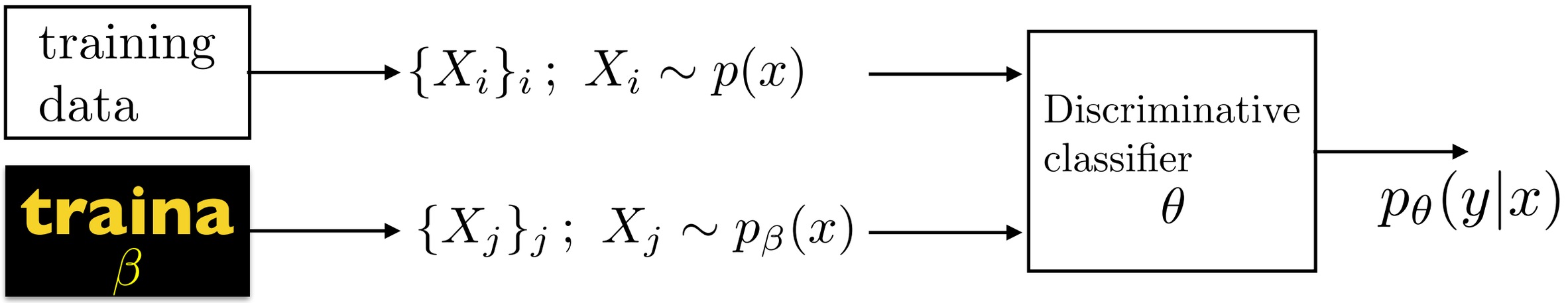



- The generator should make the classification task *as hard as possible for any discriminator.*

# Generative Adversarial Networks

*[Goodfellow et al., '14]*

- Train generator and discriminator in a minimax setting:




$y = 1$: "real" samples
$y = 0$: "fake" samples

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

# 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

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

For each $x$,

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

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

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

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

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

- In practice, however, we parametrize both generator and discriminator using neural networks.

- Optimize the cost using gradient descent.

# Generative Adversarial Training

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

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

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

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

# Generative Adversarial Training

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

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

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

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

- Indeed,

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

- Numerical approach: alternate *k* steps of discriminator update with 1 step of generator update.

# 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:

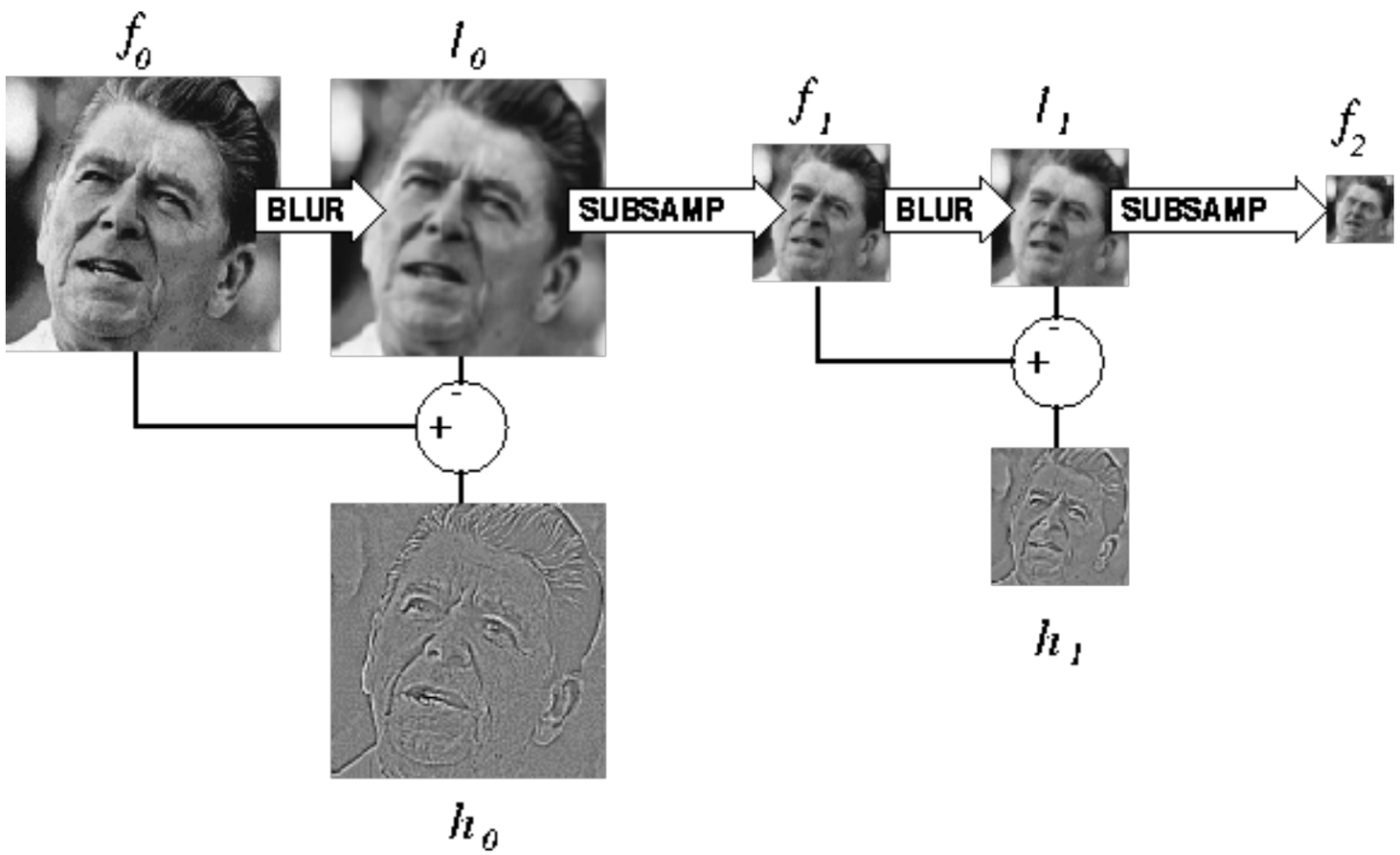
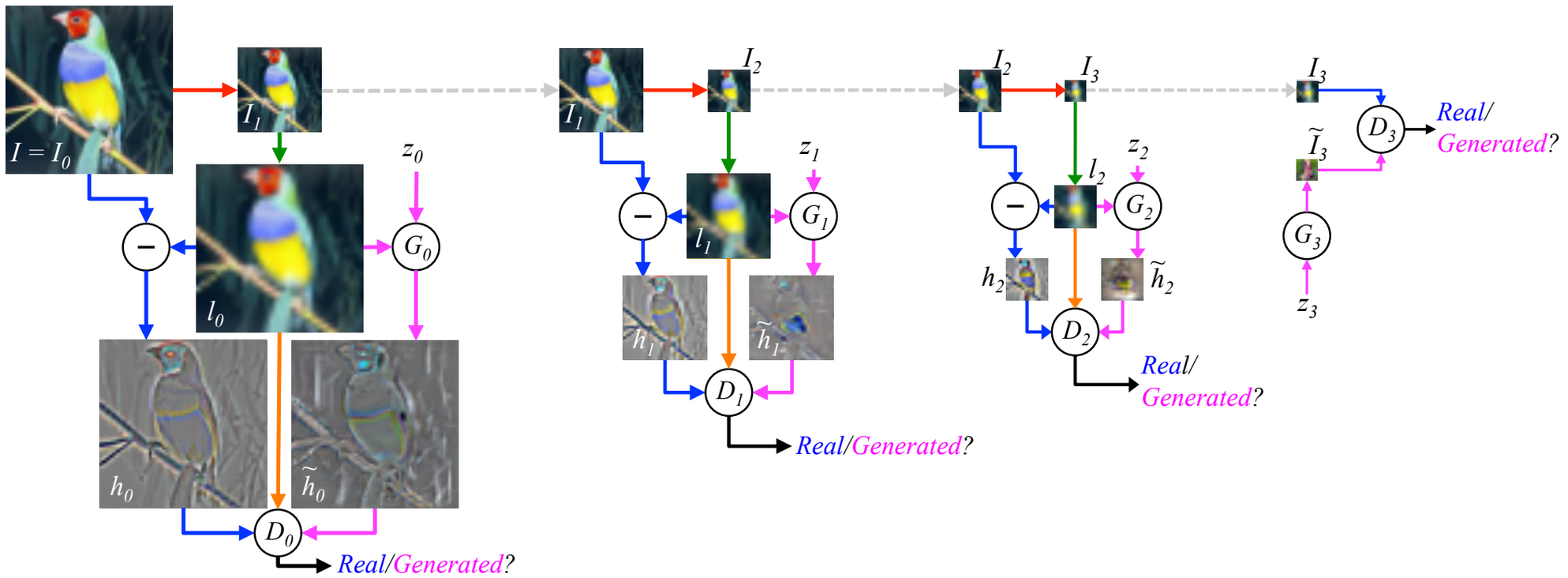


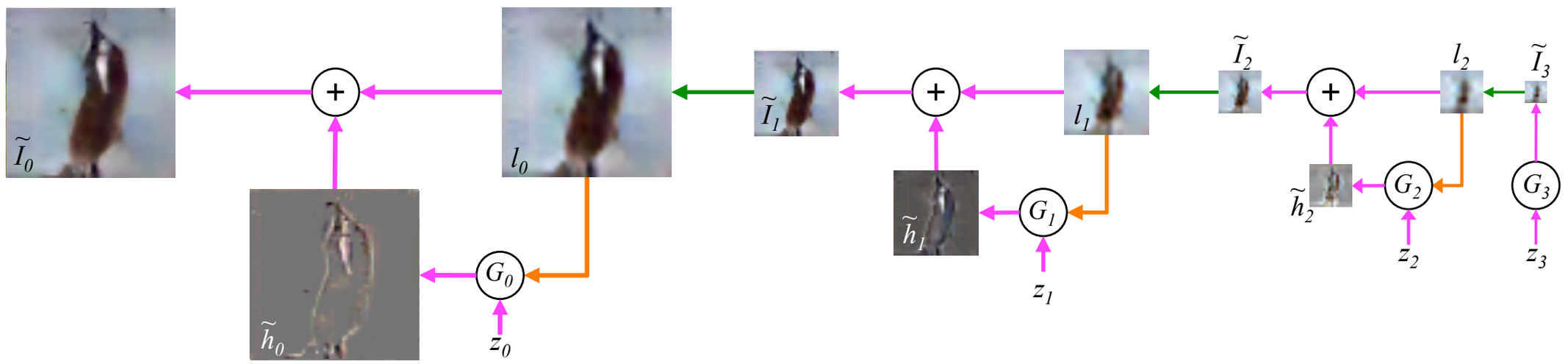

*figure source: http://sepwww.stanford.edu*

# LAPGAN

- Training procedure:

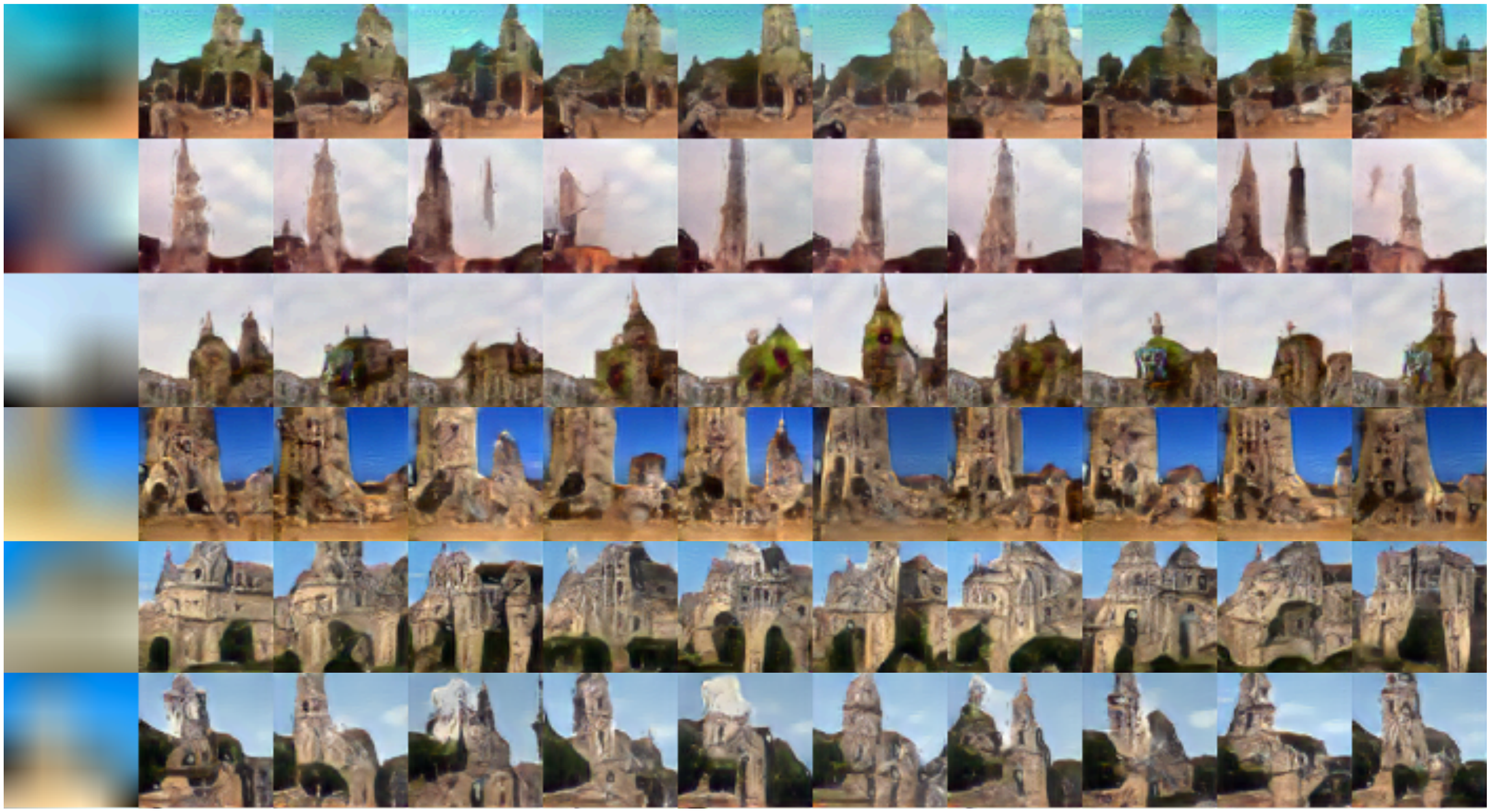



- Sampling procedure:



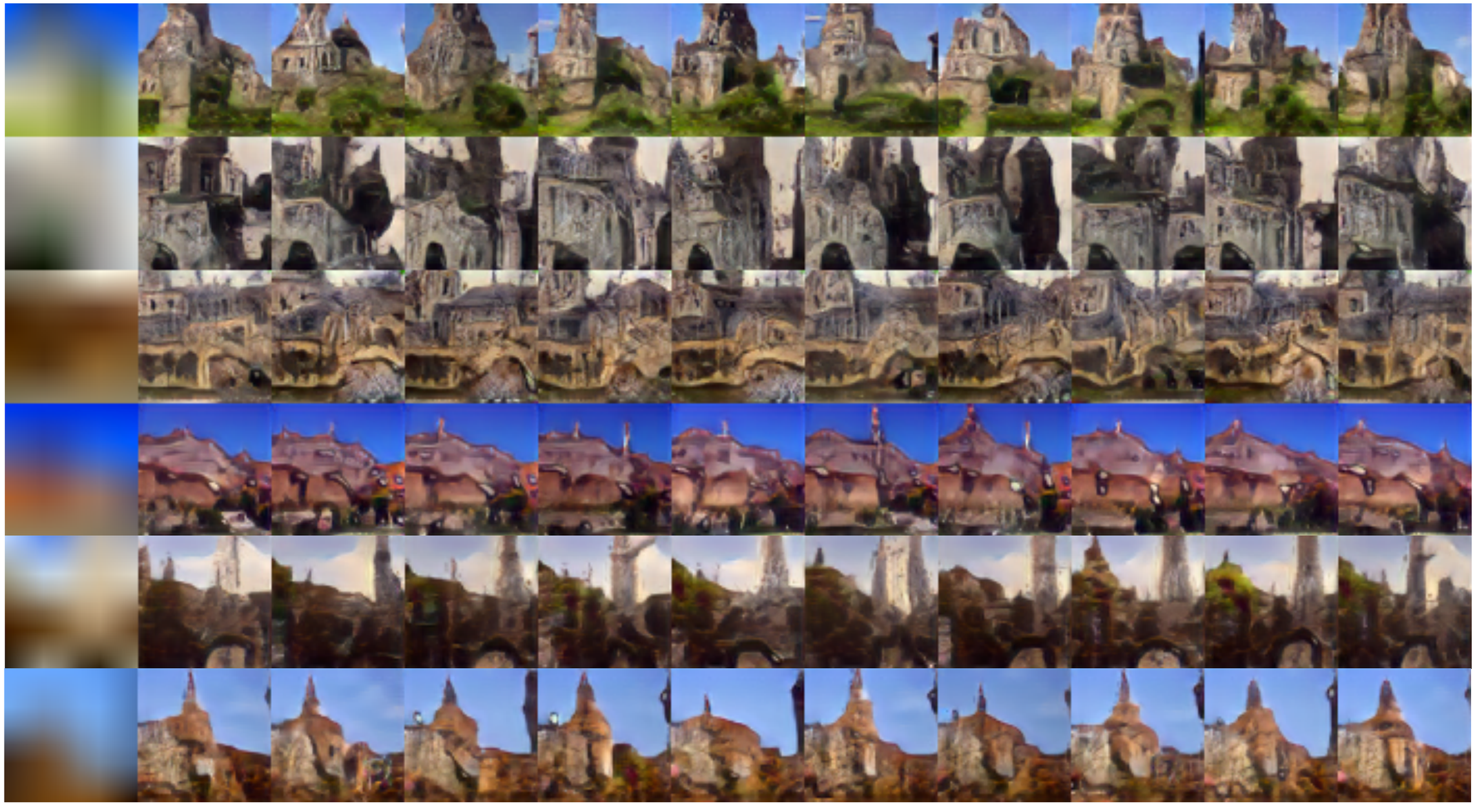
# LAPGAN

- Samples generated from the model:



# LAPGAN

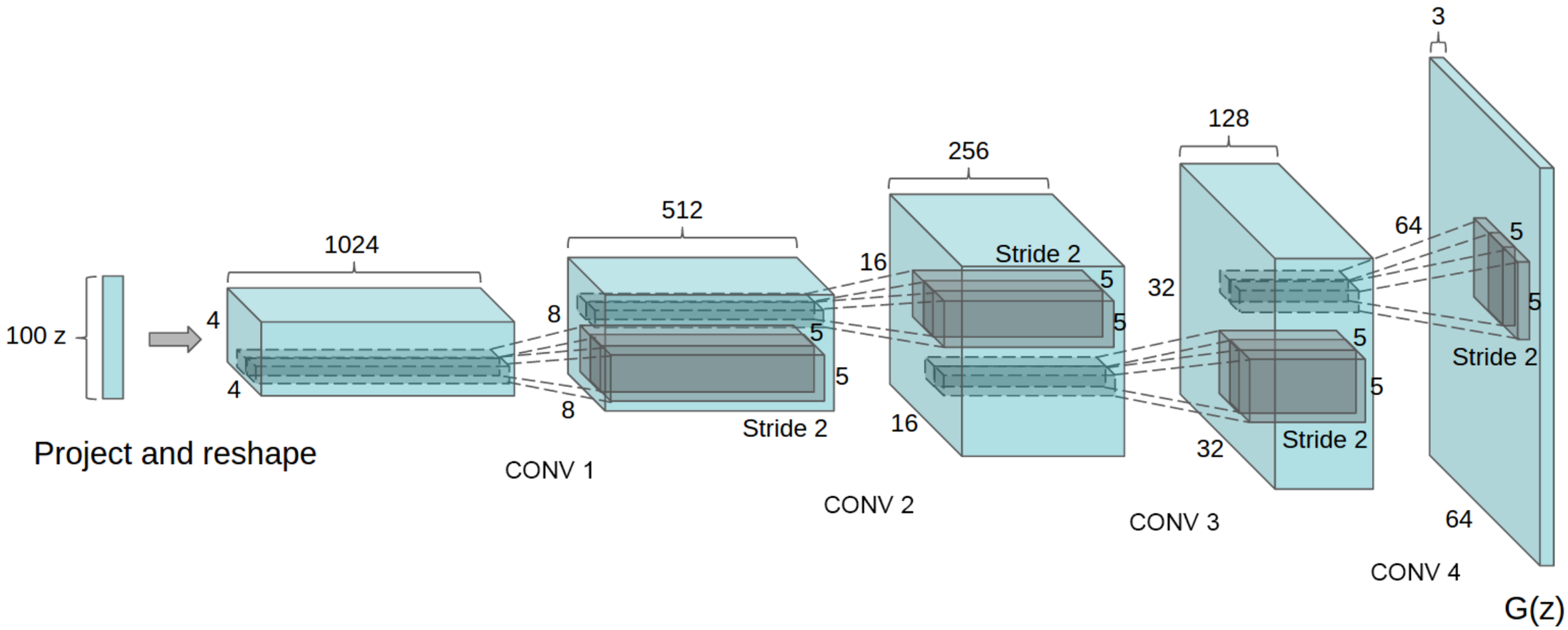
- Samples generated from the model:



# DC-GAN

*[Radford et al.'16]*

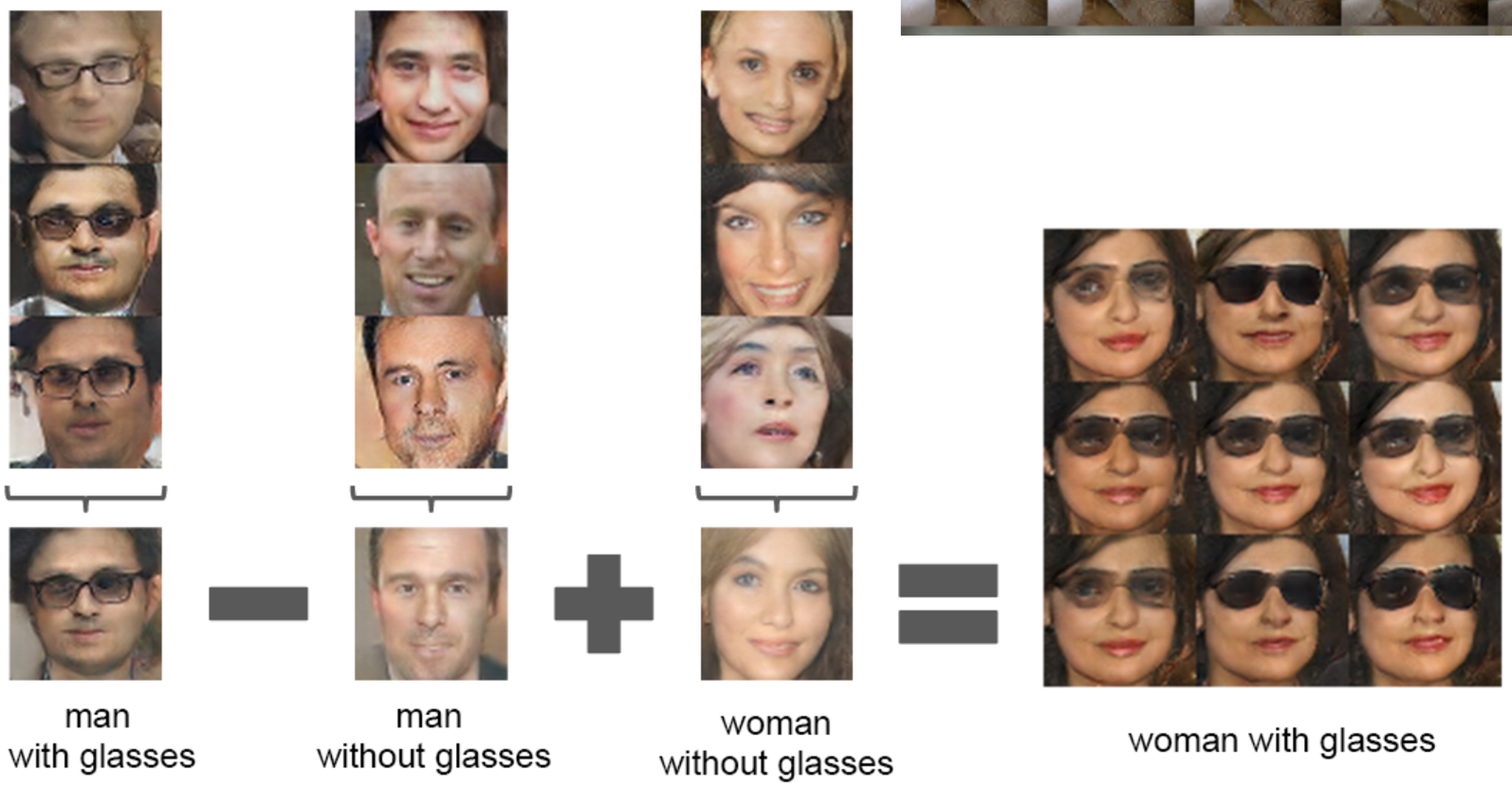
- Improved multi-scale architecture and Batch-Normalization:



# 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]

- A very *hot* topic within the Deep Learning community

# Some Recent Extensions

- *InfoGAN* [Chen & Rocky Duan et al., NIPS'16]

- *Image-to-Image Translation with Conditional Adversarial Networks* [Isola et al., '16]

- Also consider "Generative Models and Model Criticism via Optimized Maximum Mean Discrepancy", by Sutherland et al, submitted to ICLR'17.

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

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

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

- 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 ***x*** is a random vector of the form $z \sim F_0\ ,\ x \mid z = G(z, \varphi)$ , how to compute its entropy?
- If $z \mapsto G(z, \varphi)$ is injective, then

$$p_G(x) = F_0(G^{-1}(x)) \cdot |DG^{-1}(x)|\ ,\ \text{and}$$

$$\mathbb{E}_{z\sim F_0} f(G(z)) = \mathbb{E}_{x\sim p_G} f(x)\ \ \forall\ f \text{ measurable}\ .$$

- 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.
- 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

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

$$c \subset z$$

# InfoGAN

- Computing the mutual information requires access to the posterior

$$p(c \mid x)$$

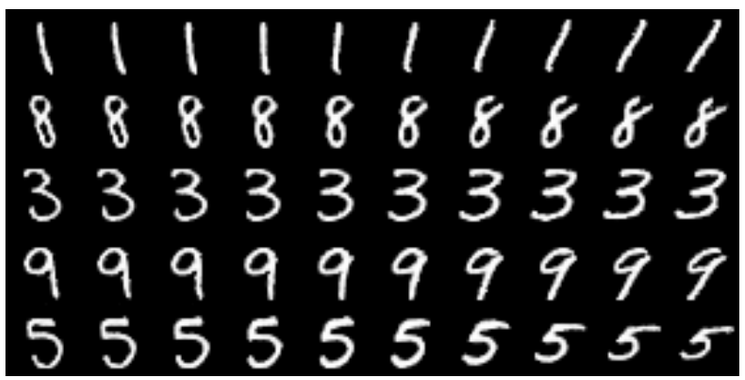
- Although in the GAN we don't have this term, we have something that can approximate it: the discriminator network!
- Variational Information Maximization:

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

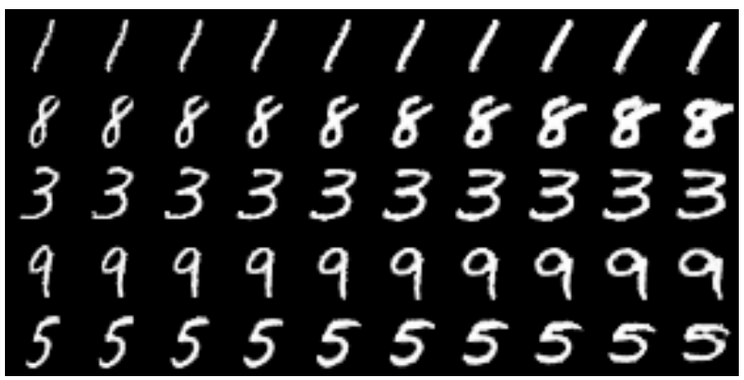
# InfoGAN

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

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



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



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

(a) Rotation

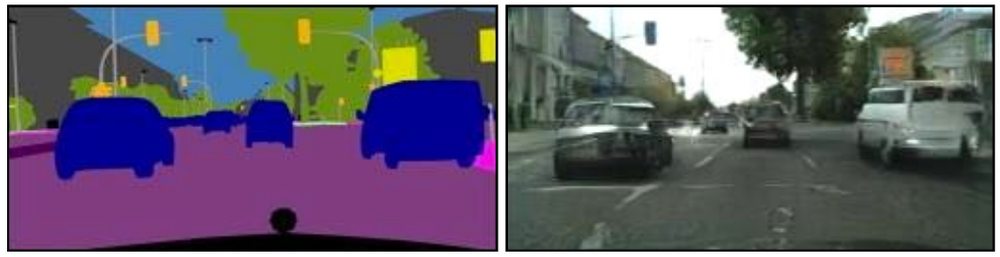
(b) Width

- Unsupervised "distentanglement" in low-dimensional data.
- It also alleviates the tendency of GAN to overfit the training data.
- WARNING: singular distributions have no well-defined entropy!

# Other Recent GAN work

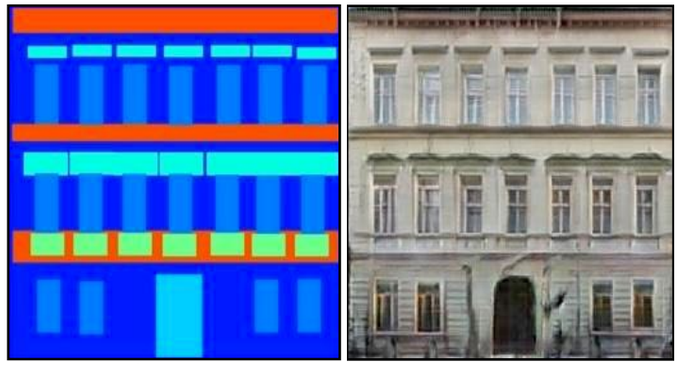
- Conditional GANS for computer vision tasks

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

# Generative Adversarial Networks

- Some open research directions:

1. **Optimization**:
    1. How to ensure a correct algorithm?
    2. Existence of a Lyapunov function?
2. **Statistics**:
    1. How to determine the discriminator power (eg VC-dimension) to obtain consistent estimators?
    2. Control of overfitting to the training distribution?
3. **Applications**:
    - Language Modeling
    - Reinforcement Learning
    - Algorithmic Tasks
    - Importance Sampling

# 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
  - Highly specific flows affect the flexibility of the $\Phi$ model.

- Indirect learning by the Discriminative Adversarial Training is implicit
  - No cheap way to evaluate the density
  - Also, no cheap way to do inference, e.g. $p(x)$
- How to regularize the density estimation? $p(z|x)$

# 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:

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

$$\textbf{Empirical class-conditional moments:} \quad \mu_k = \mathbb{E}_{(x,y)\sim\hat{p}}(\Phi(x)|y = k) \quad k = 1 \dots K$$

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

- Necessary condition:

Empirical class-conditional moments:

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

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

$$\max_p \; H(p)$$

$$s.t.\ \mathbb{E}_{x\sim p}\Phi(x) = \mu \ .\in \mathbb{R}^d$$

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

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

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

# Gibbs Models and Maximum Entropy

- Thus we have

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

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

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

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

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

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

$$k \sim \mathrm{cat}\{1, K\}$$
$$x \sim p_k(x) = \exp(\langle \lambda_k, \Phi(x) \rangle - A(\lambda_k)) \ , \quad \mathbb{E}_{x \sim p_k} \Phi(x) = \mu_k \ .$$

- 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:

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

  - 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

$$p(x|\theta) = \exp\left(\langle \theta, \Phi(x) \rangle - A(\theta)\right)$$
$$U_x = \Phi(x)$$
$$K(x, x') = \langle \tilde{\Phi}(x), \tilde{\Phi}(x') \rangle \ , \ (\tilde{\Phi} : \text{ whitened features }) \ .$$

# Gibbs Model and Fisher Kernels

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

$$U_x : \text{Fisher Vector} = \nabla_\theta \log p(x|\theta) \ .$$
$$I : \text{Fisher Information} = \mathbb{E}\{U_x U_x^T\} \ .$$

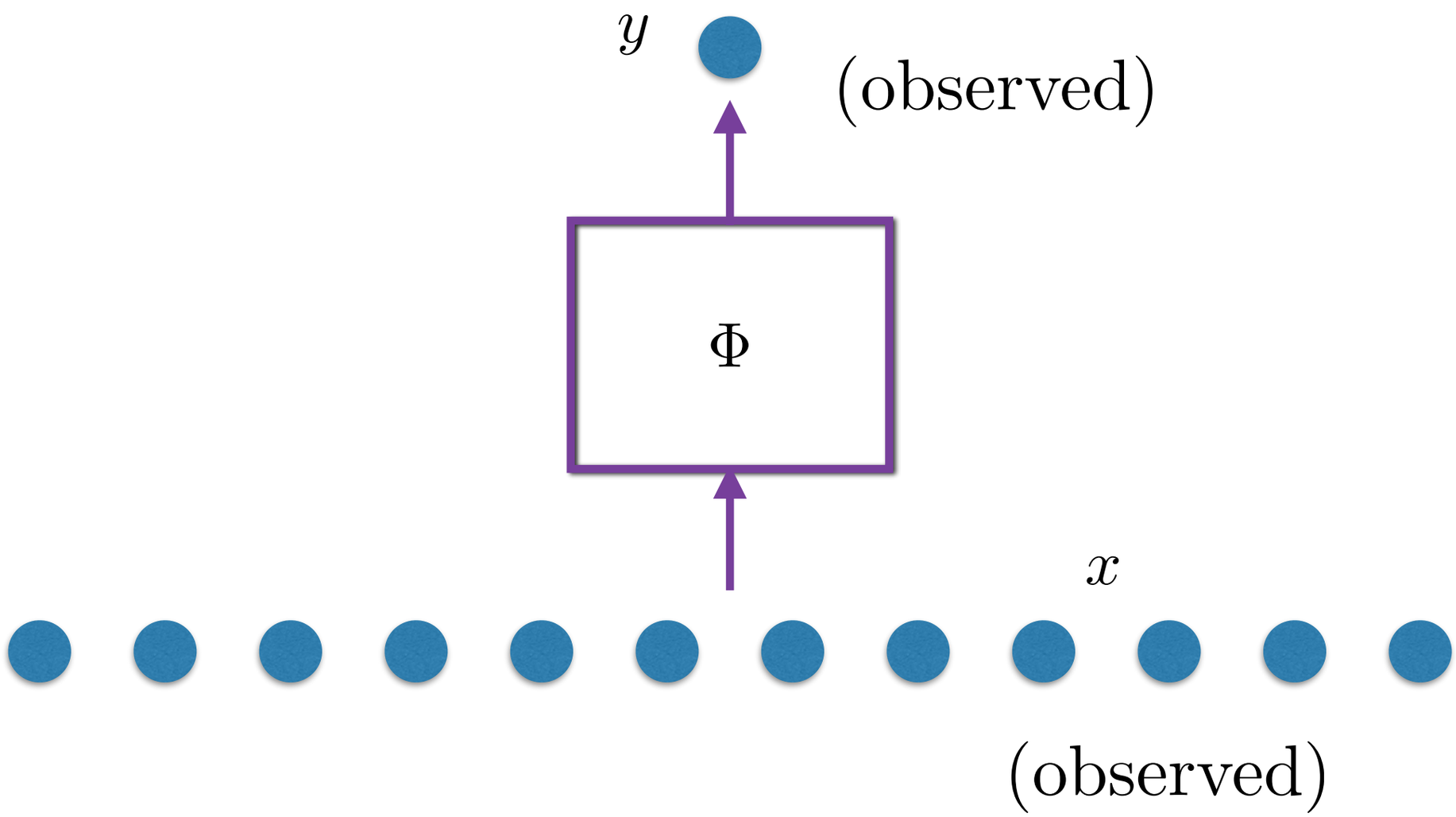
- When $K(x, x') = \langle U_x, I^{-1} U_{x'} \rangle$ ,, the Fisher vector is

$$p(x|\theta) = \exp\left(\langle \theta, \Phi(x) \rangle - A(\theta)\right)$$
$$U_x = \Phi(x) \qquad K(x, x') = \langle \tilde{\Phi}(x), \tilde{\Phi}(x') \rangle \ , \ (\tilde{\Phi} : \text{ whitened featur}$$

- Thus the maximum entropy model is the "canonical" generative model associated with linearization features $\Phi$

# Unsupervised Gibbs Models

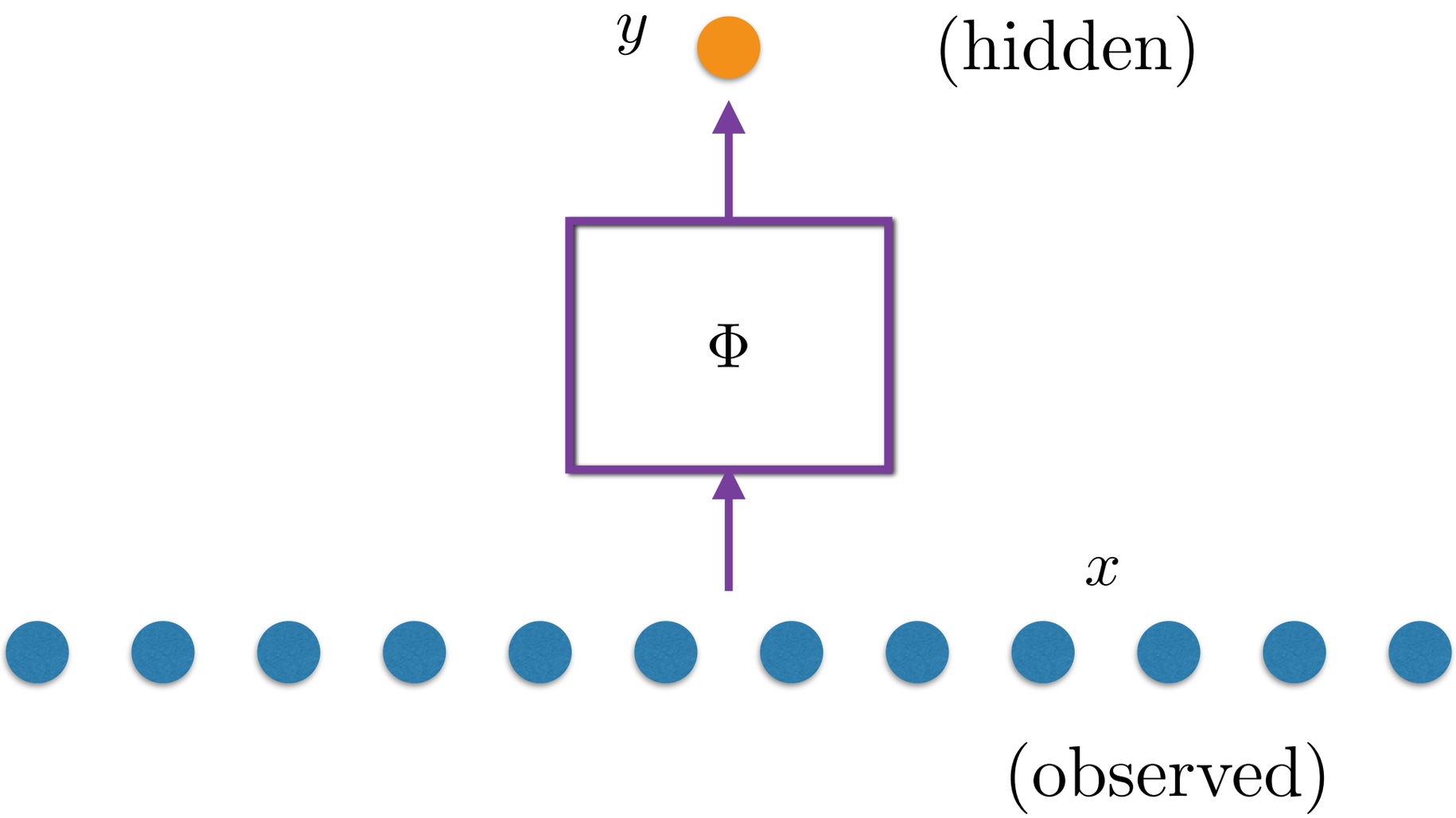
- We have derived a fully probabilistic model in a supervised setting:




$$y \sim \text{cat}\{1, K\}$$
$$p(x|y) = \exp\left(\langle \theta_y, \Phi(x) \rangle - A(\theta_y)\right)$$
$$\Rightarrow \quad p(y|x) = \text{softmax}(\{\langle \theta_y, \Phi(x) \rangle\}_y) \ .$$

# Unsupervised Gibbs Models

- How about when no labels are observed?




$$y \sim \text{mult}(\pi)$$
$$p(x|y) = \exp\left(\langle \theta_y, \Phi(x) \rangle - A(\theta_y)\right)$$

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

# 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

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

- We have

$$\begin{aligned}
\nabla_\theta A(\theta) &= \nabla_\theta \log \int \exp(\langle \theta, \Phi(x) \rangle) dx \\
&= \frac{\int \nabla_\theta \exp(\langle \theta, \Phi(x) \rangle) dx}{\int \exp(\langle \theta, \Phi(x) \rangle) dx} \\
&= \frac{\int \Phi(x) \exp(\langle \theta, \Phi(x) \rangle) dx}{\int \exp(\langle \theta, \Phi(x) \rangle) dx} \\
&= \mathbb{E}_{x \sim p(x|\theta)} \Phi(x)
\end{aligned}$$

- 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.

# Autoregressive Models

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

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

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

# Autoregressive Models

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

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

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

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

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

# Autoregressive Models

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

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

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

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

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

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

# Autoregressive Models

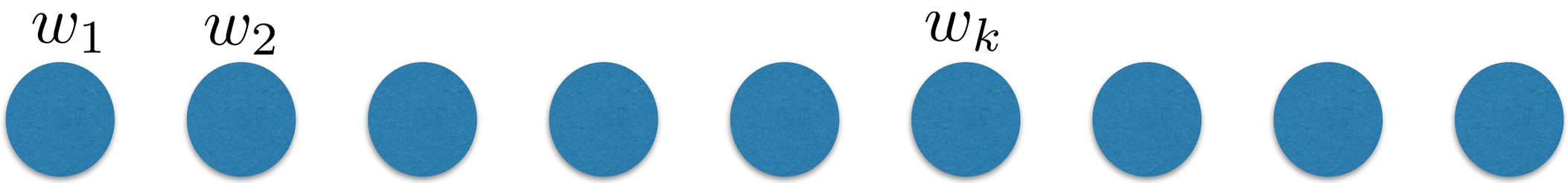
- Temporally ordered data
  - Speech, Music
  - Video
  - Language
  - Other time series (Weather, Finance, …)
- Spatially ordered data, Multi-Resolution data
  - Images

- Learning is thus reduced to the problem of conditional prediction.

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

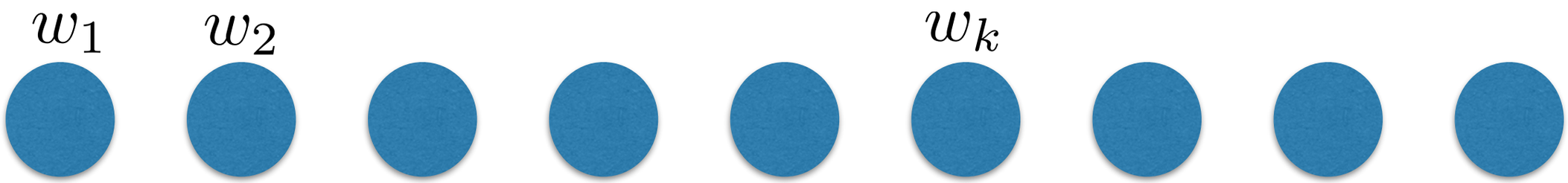
# Word2vec [Mikolov et al.'13].

- Unsupervised learning "success story".



# Word2vec [Mikolov et al.'13].

- Unsupervised learning "success story".

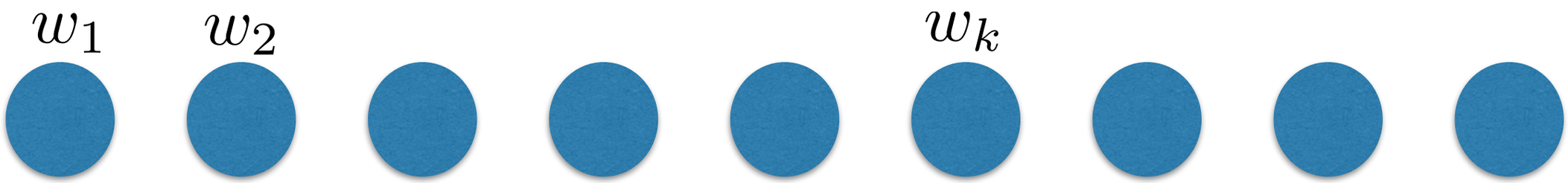



- Language creates a notion of similarity between words:

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

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

# 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

$$p(D = 1|c, w) = \sigma(\langle v_w, v_c \rangle) \ , \ v_w, v_c \in \mathbb{R}^d$$

.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

# 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

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

- Training with Maximum Likelihood:

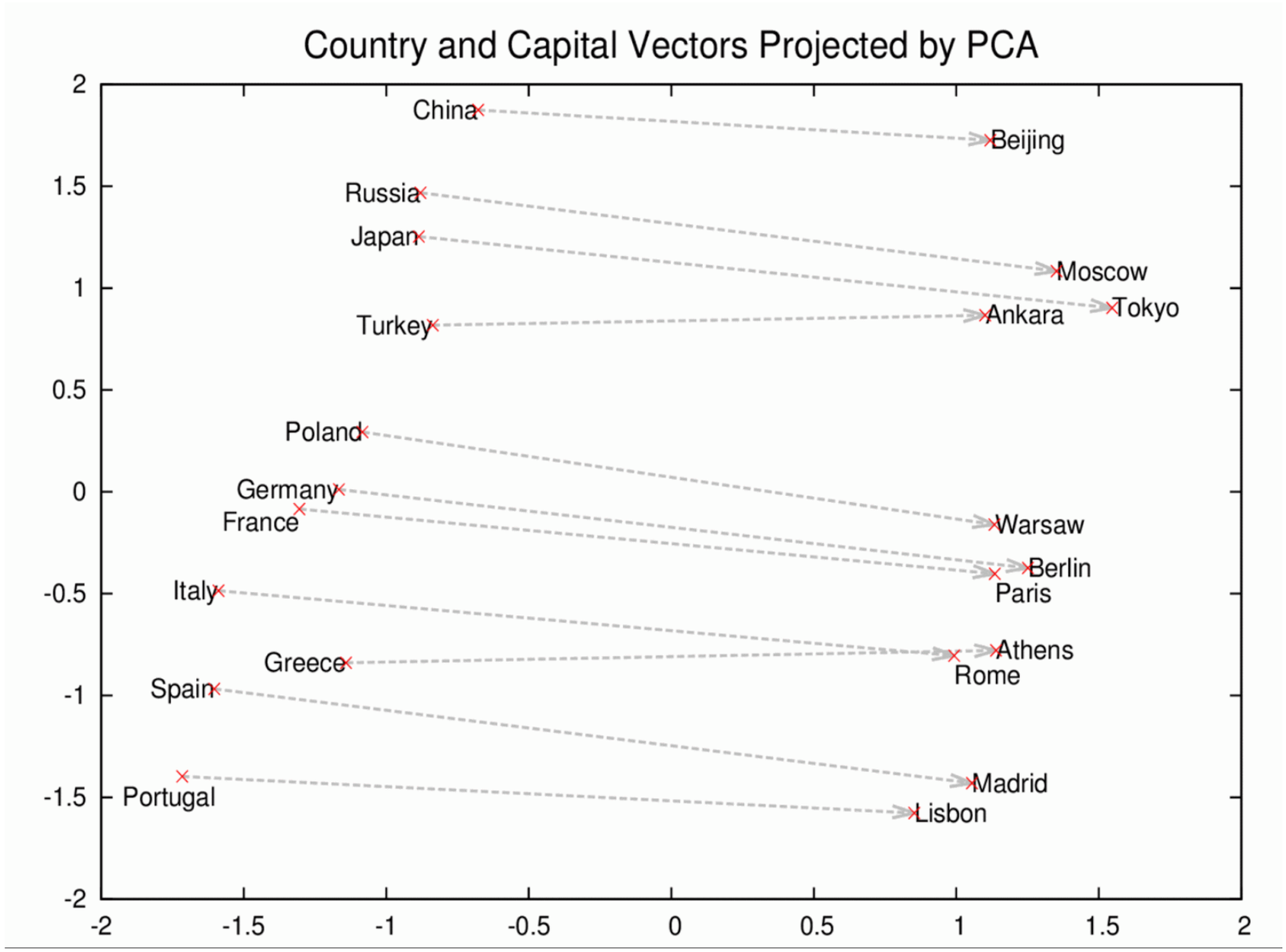
$$\arg\max_\theta \prod_{(w,c)\sim\mathcal{D}} p(D = 1|c, w, \theta) \prod_{(w,c)\sim\mathcal{D}'} p(D = 0|c, w, \theta)$$

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

$\mathcal{D}$ : positive contexts $\quad$ $\mathcal{D}'$ : 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.




*[google]*

# Video Prediction

- Rather than modeling the density of natural images

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

we may be also interested in modeling the conditional distributions

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

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

# Video Prediction

- Rather than modeling the density of natural images

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

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

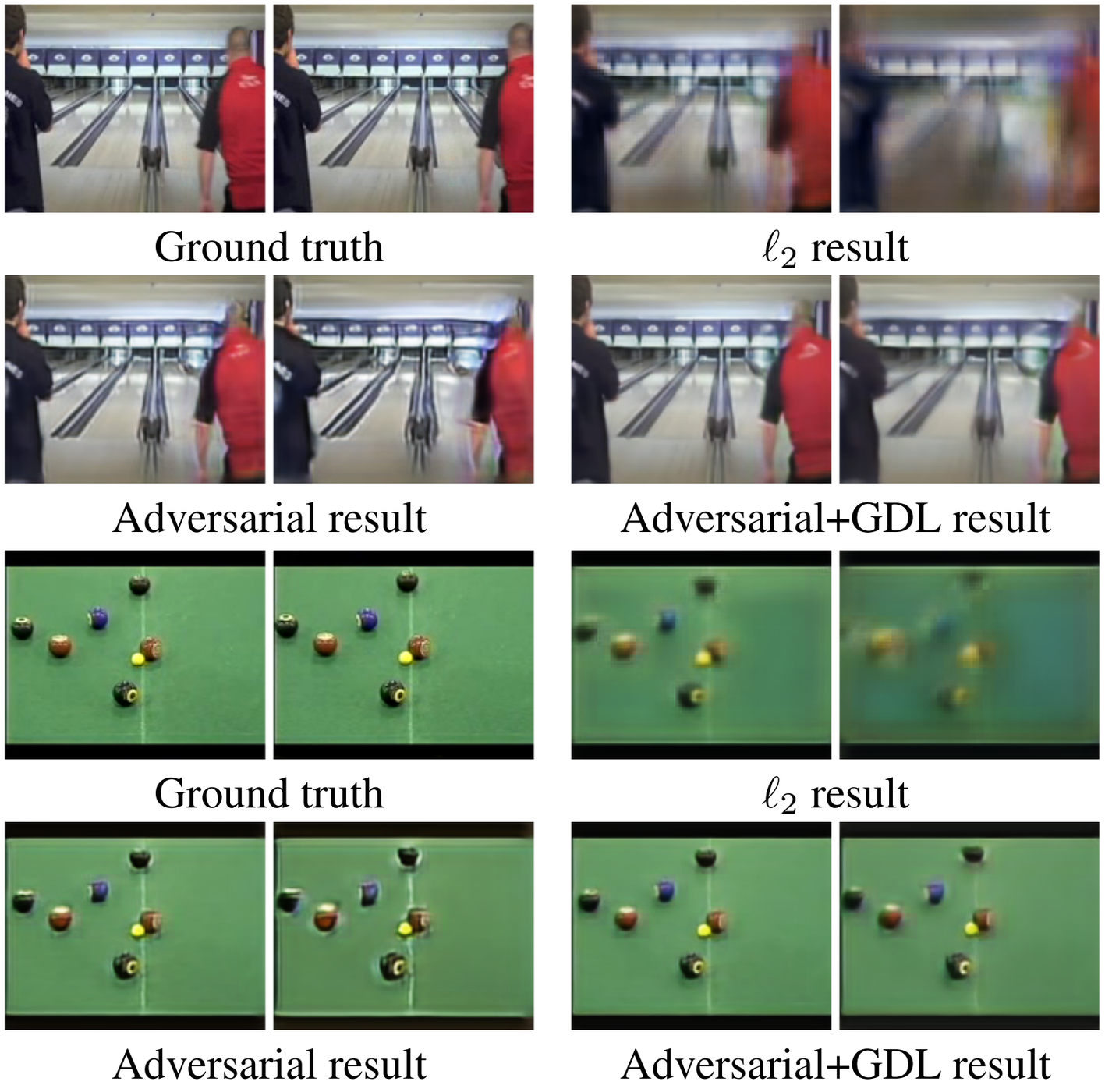
- Similarly, can we find a signal representation $\Phi(x_t)$ that is consistent with the "video language" metric? i.e.

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

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

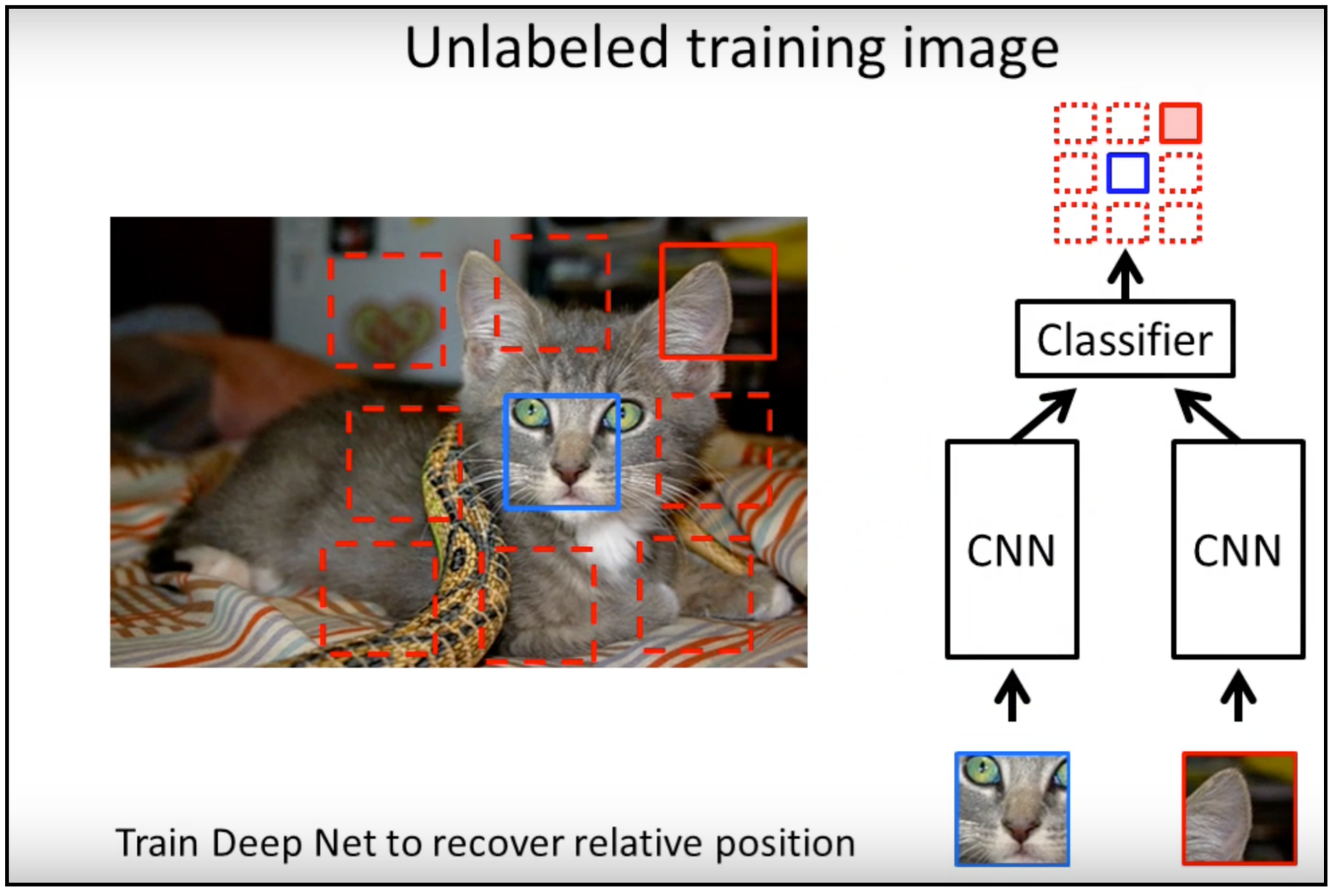
# Video Prediction

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



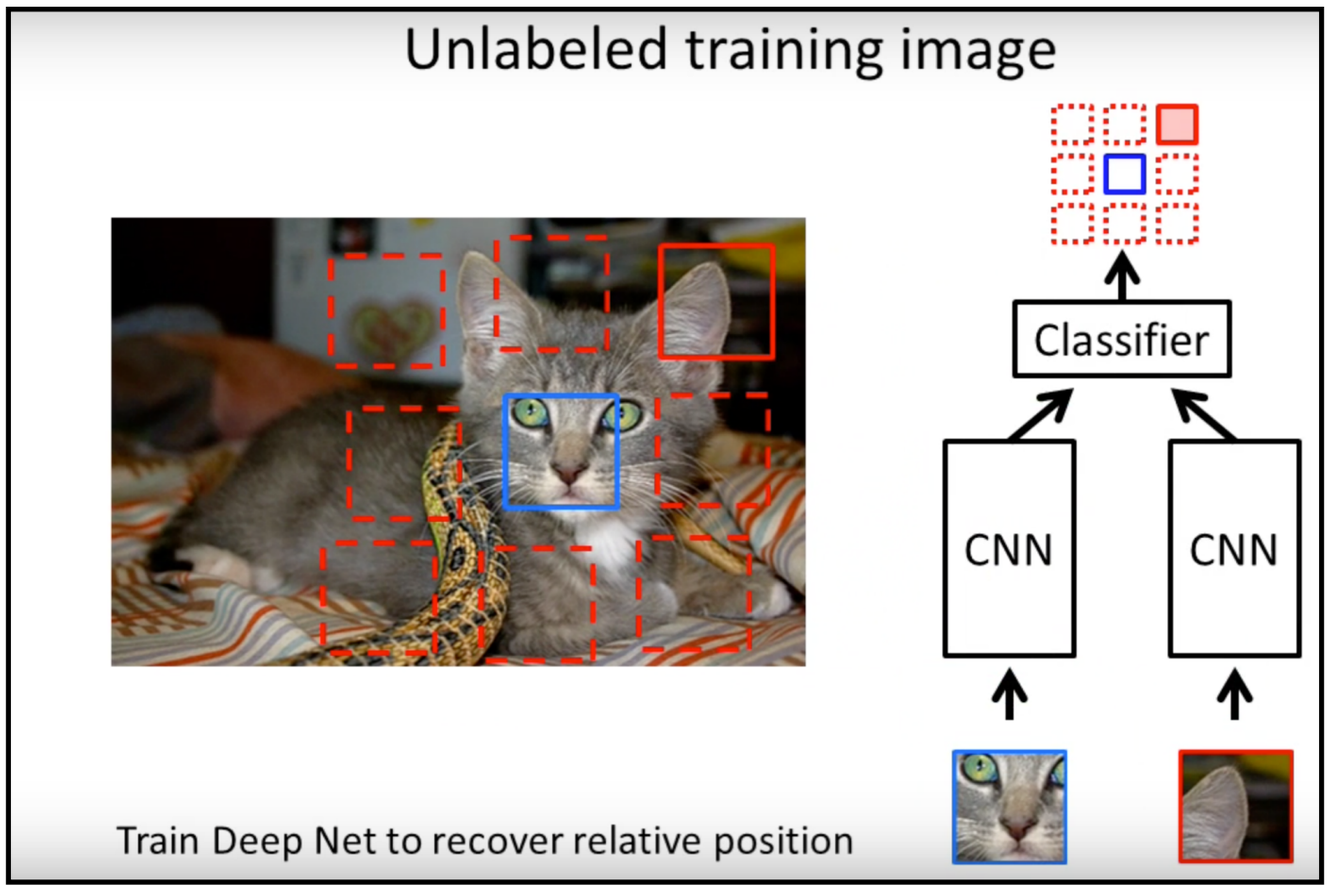
# Patch Relative Configuration [Doerch et al.'15]

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



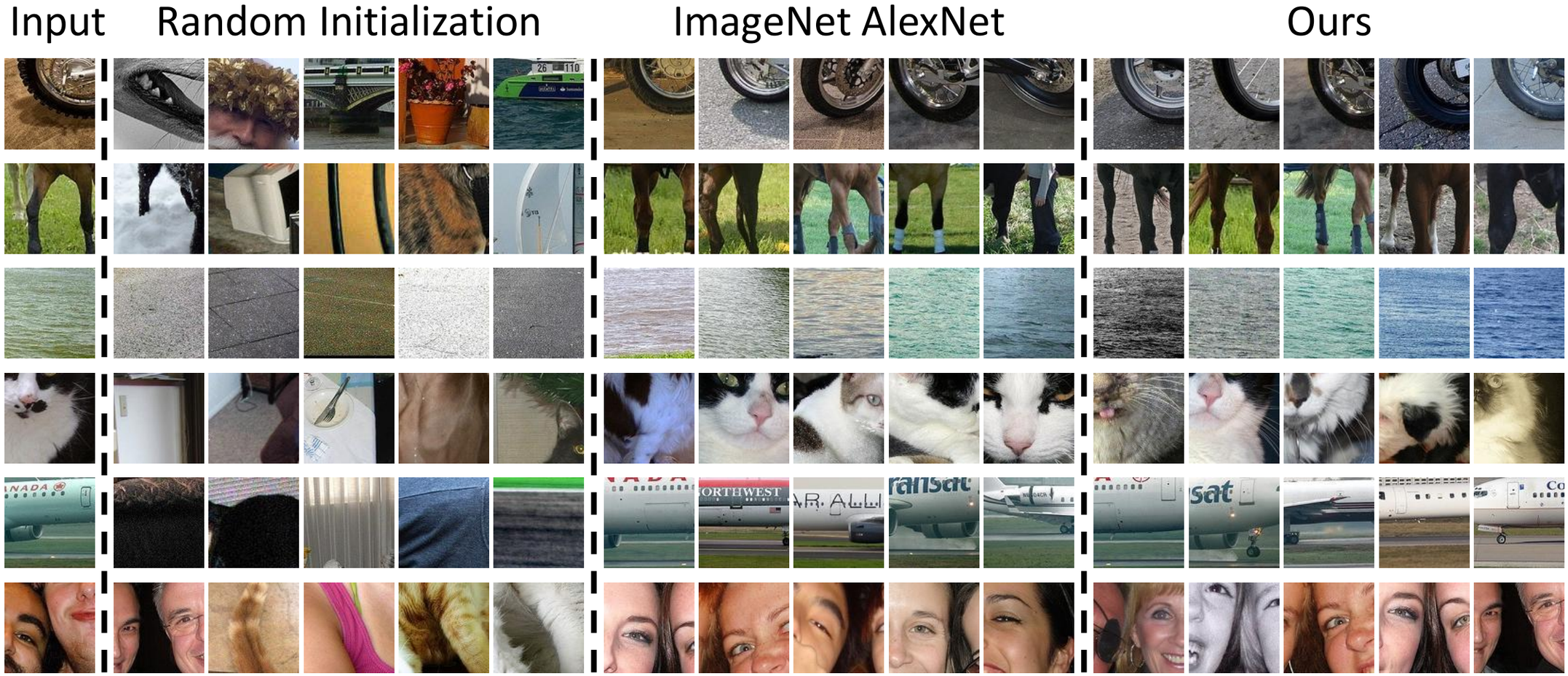
# Patch Relative Configuration [Doerch et al.'15]

- 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 $\hat{x_{t+1}} = F(x_1, \ldots, x_t)$ require a loss or an associated likelihood

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

- 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 $\hat{x_{t+1}} = F(x_1, \ldots, x_t)$ require a loss or an associated likelihood

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

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

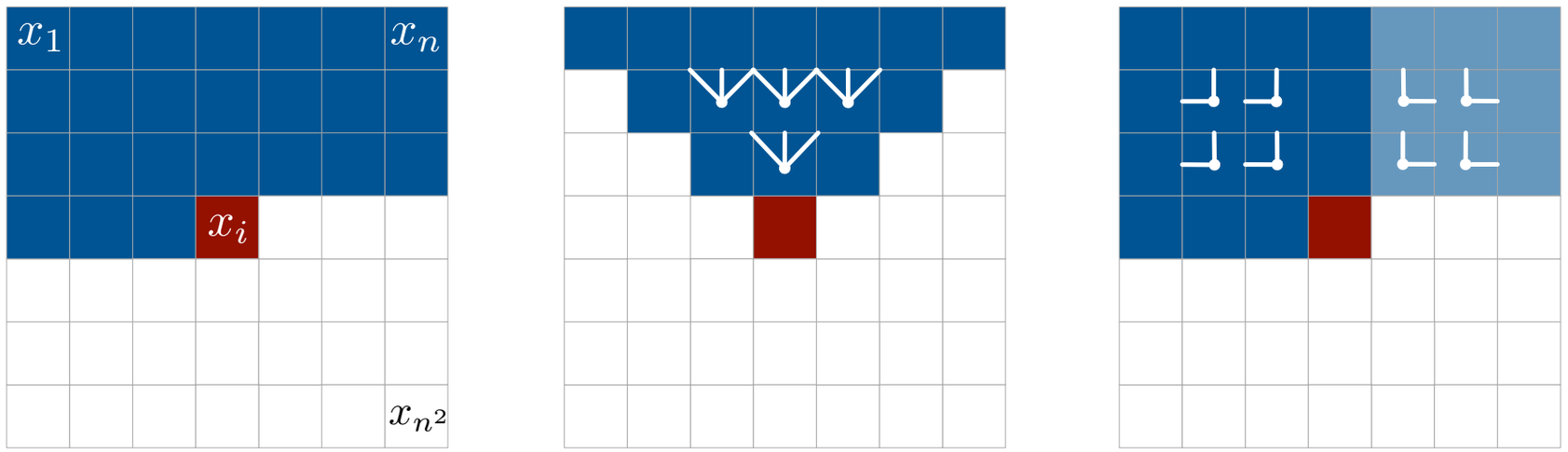
- How about images?
  - We can treat them as discrete two-dimensional grids

    $x(u) \in \{0, 255\}$
  - Model each pixel from its "past" context:

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

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

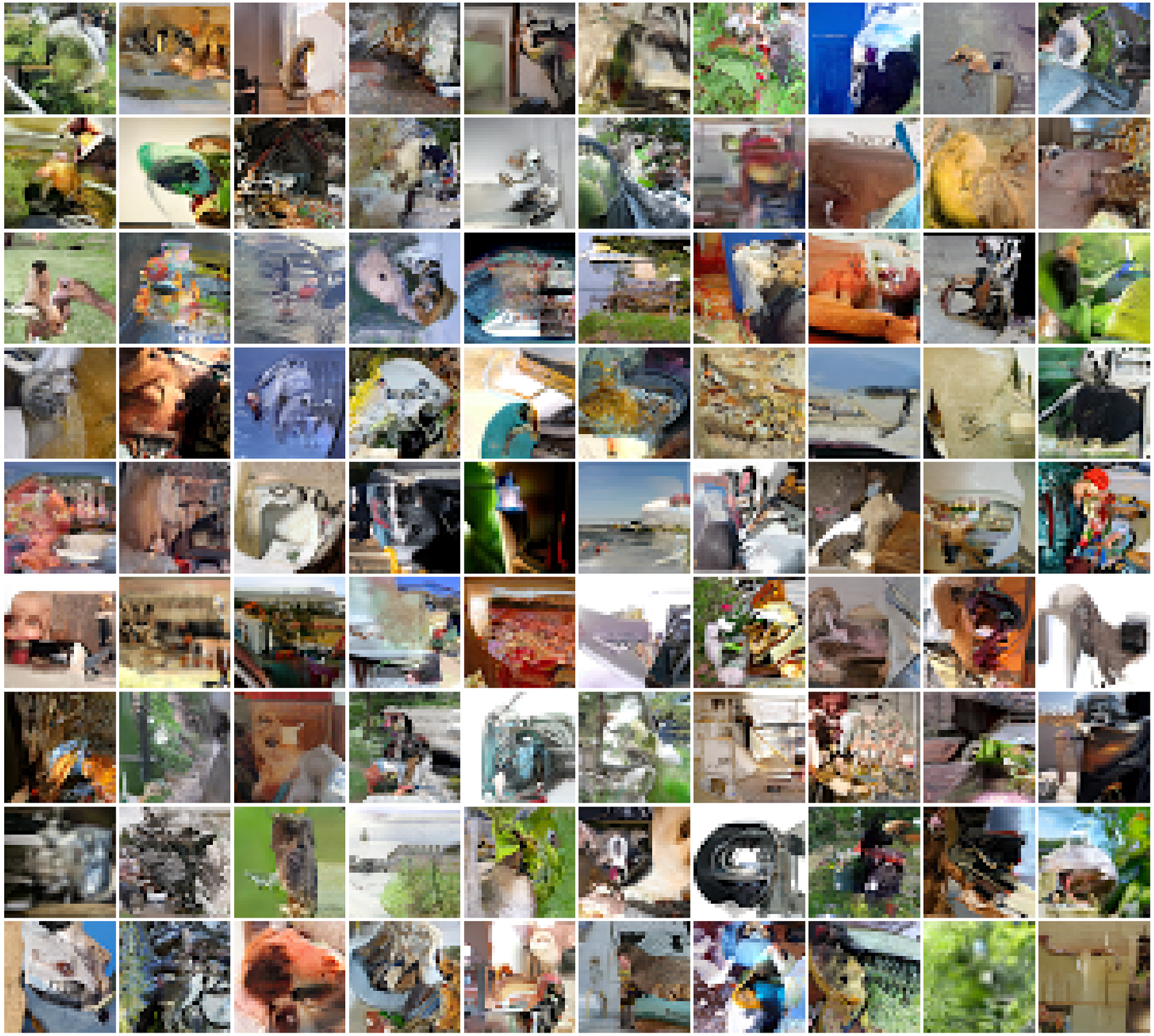
- Contexts are modeled using "diagonal BiLSTMS".




- 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, $h = 32$): | **80.75** |
| Diagonal BiLSTM (7 layers, $h = 16$): | **79.20** |

*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: | **3.00** (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 for details).