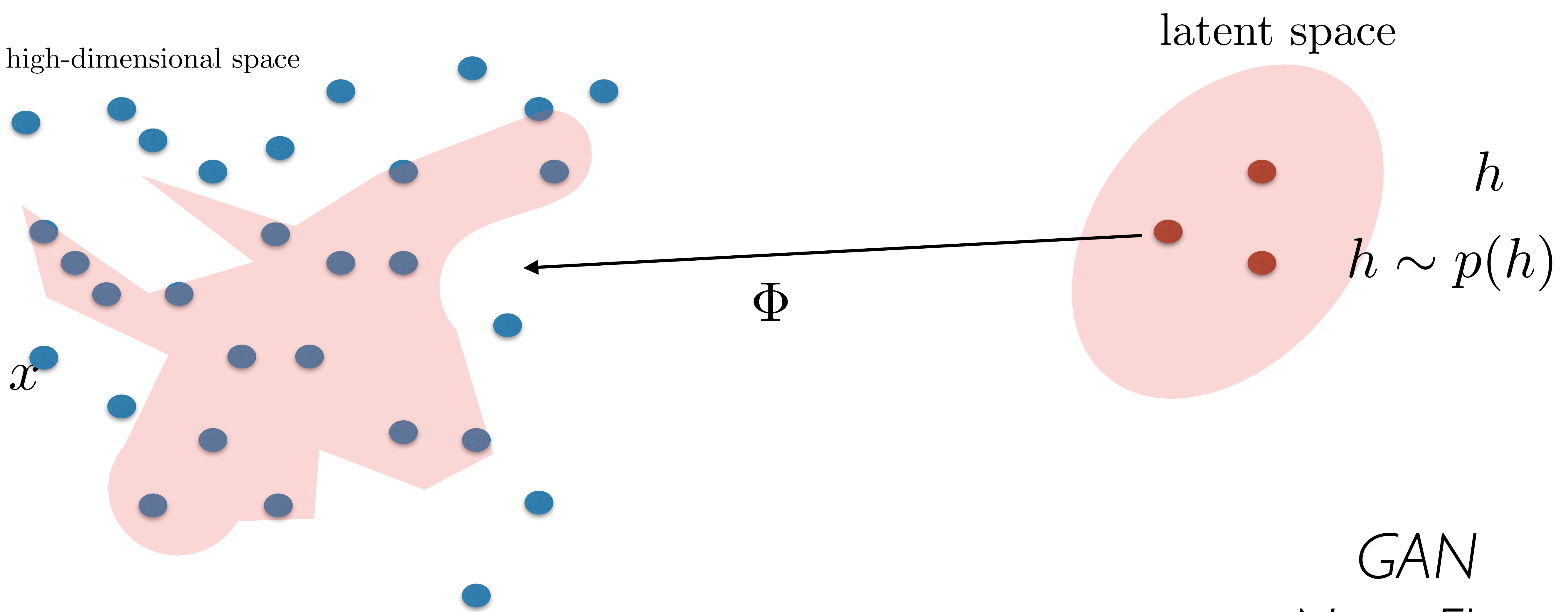
Stat 212b: Topics in Deep Learning
Lecture 18

Joan Bruna
UC Berkeley

Berkeley
UNIVERSITY OF CALIFORNIA

Review: Generative Models of Complex data

- Flows or Transports of Measure



GAN
NormFlow
$\ldots$

$p(x)$ defined implicitly with

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

Review: Normalizing Flows

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

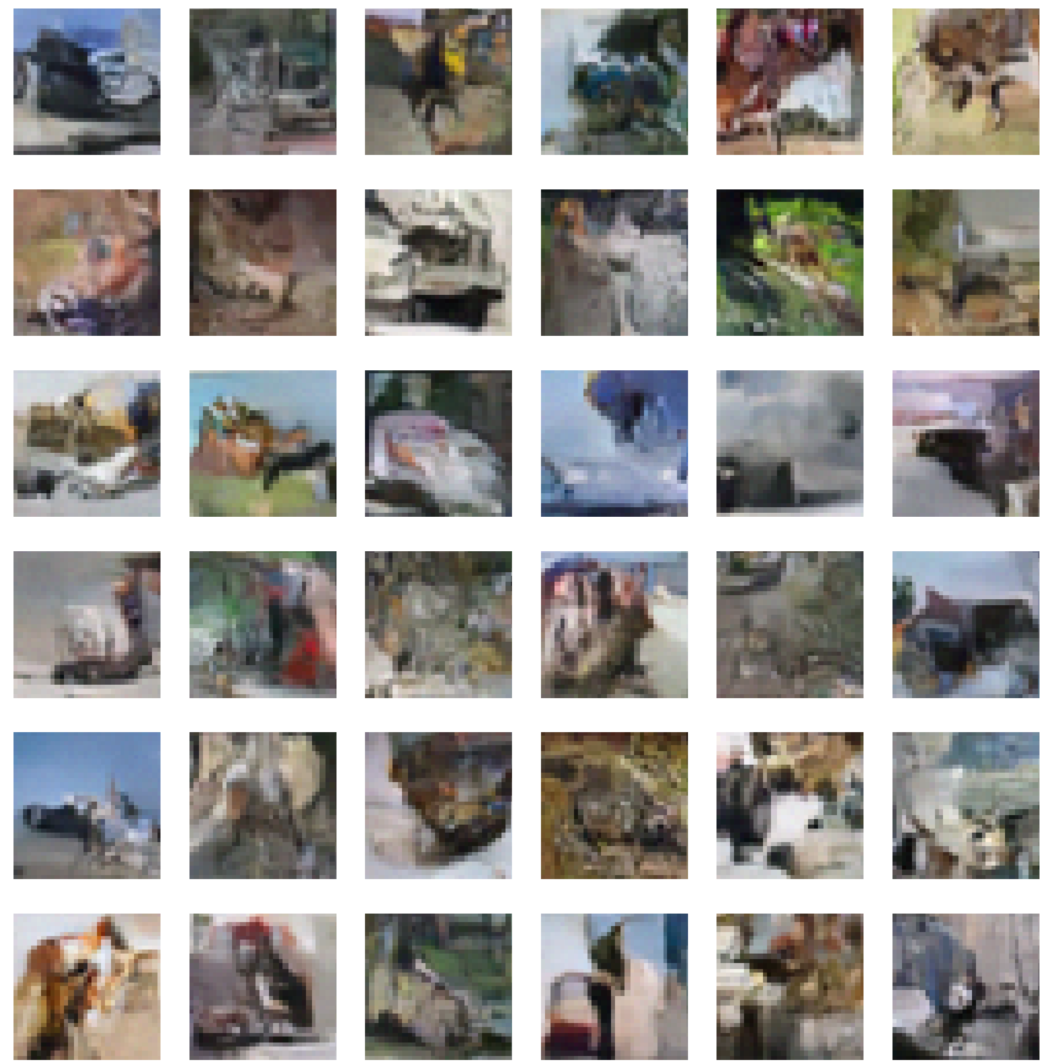
$$z_K = \Phi_K \circ \ldots \Phi_1(z_0) \ , \ \text{with } z_0 \sim q_0(z)$$

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

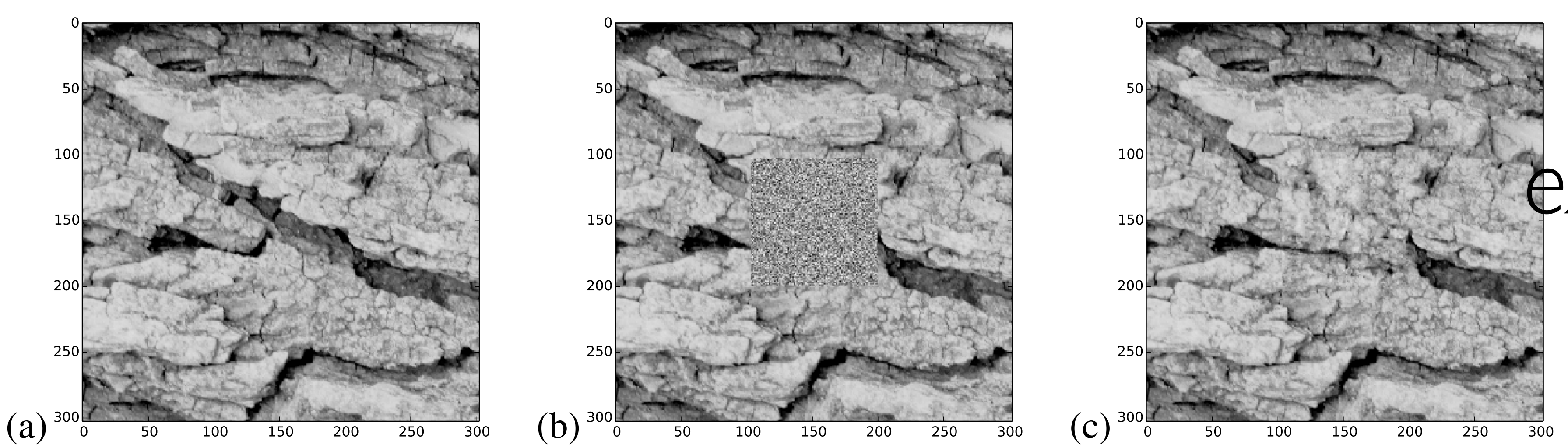
- One can parametrize invertible flows and use them within the variational inference to improve the variational approximation. [Rezende et al.'15]
- Also considered in ["NICE", Dinh et al'15].

Review: Diffusion and Non-equilibrium Thermodynamics

[Sohl-Dickstein et al.'15]



samples from the model trained on CIFAR-10



inpainting experiments

Review: Generative Adversarial Networks

[Goodfellow et al., '14]

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

trainable β $\longrightarrow X \sim p_\beta(x)$

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

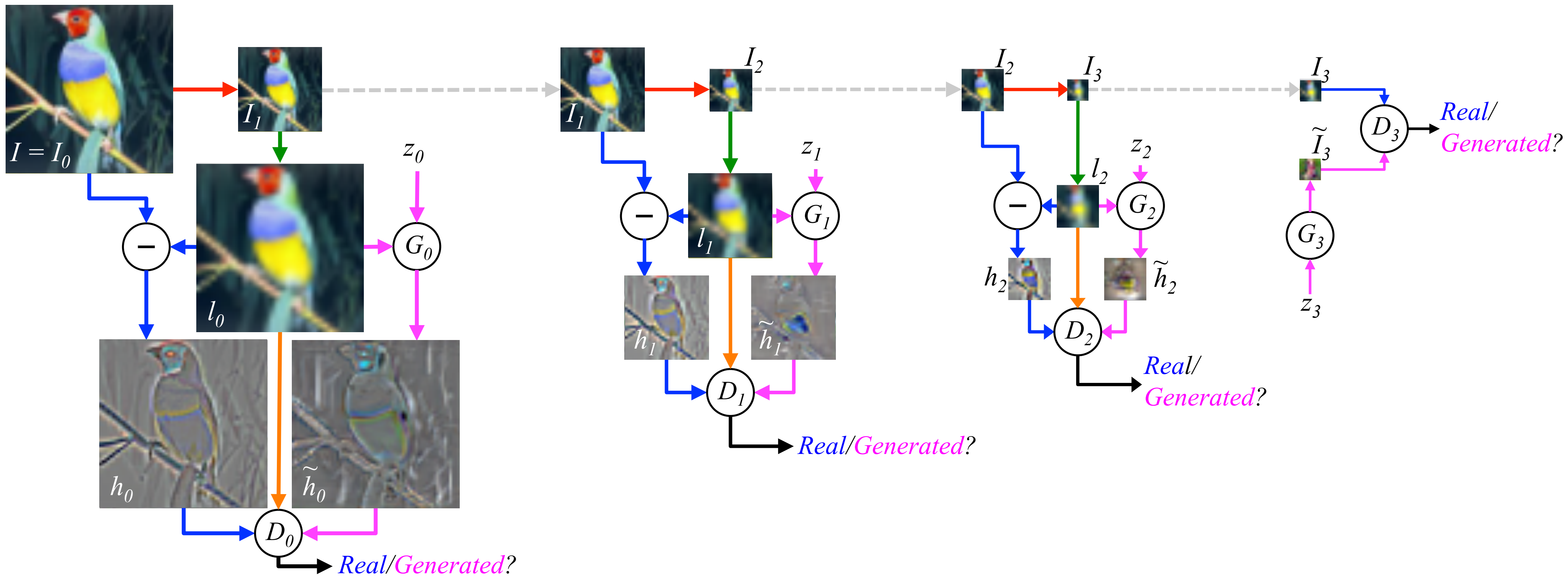
training data $\longrightarrow \{X_i\}_i\,;\; X_i \sim p(x) \longrightarrow$ Discriminative classifier θ

trainable β $\longrightarrow \{X_j\}_j\,;\; X_j \sim p_\beta(x) \longrightarrow$ Discriminative classifier θ

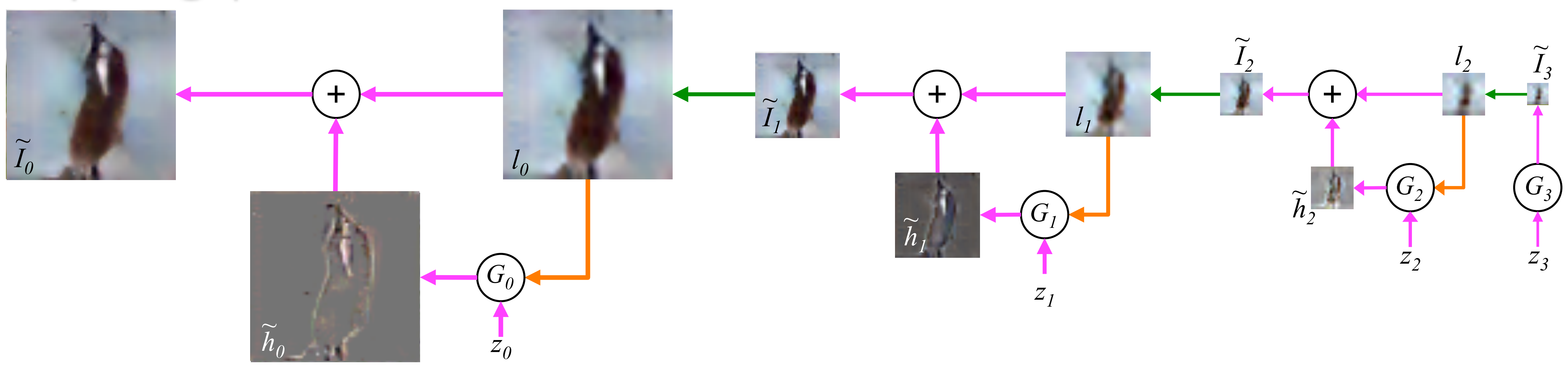
- The generator should make the classification task *as hard as possible for any discriminator.*

Review: LAPGAN

- Training procedure:



- Sampling procedure:



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

Objectives

- Maximum Entropy Distributions and Energy-Based Models
  - MCMC
  - Examples

- Self-Supervised Learning
  - Word2Vec
  - Slow Feature Analysis
  - Prediction

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 Φ
  - 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:

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

- 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:

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

- 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.

Gibbs Models

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

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

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.

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 $\forall\ k\ ,\ \mathbb{E}_{x\sim p_k}\Phi(x) = \mu_k$, those with maximum uncertainty (maximum smoothness).

Gibbs Models and Maximum Entropy

- We are thus interested in the problem

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

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

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

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

Examples of Maximum Entropy

- Maximum entropy given known energy

- Maximum entropy given known mean

Gibbs Model

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

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

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: 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:

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

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

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

- When $p(x|\theta) = \exp\left(\langle \theta, \Phi(x) \rangle - A(\theta)\right)$, the Fisher vector is

$$U_x = \Phi(x)$$

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

Gibbs Model and Fisher Kernels

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

$$U_x : \text{Fisher Vector} \ = \nabla_\theta \log p(x|\theta) \ .$$
$$I : \text{Fisher Information} \ = \mathbb{E}\{U_x U_x^T\} \ .$$
$$K(x, x') = \langle U_x, I^{-1} U_{x'} \rangle \ ,$$

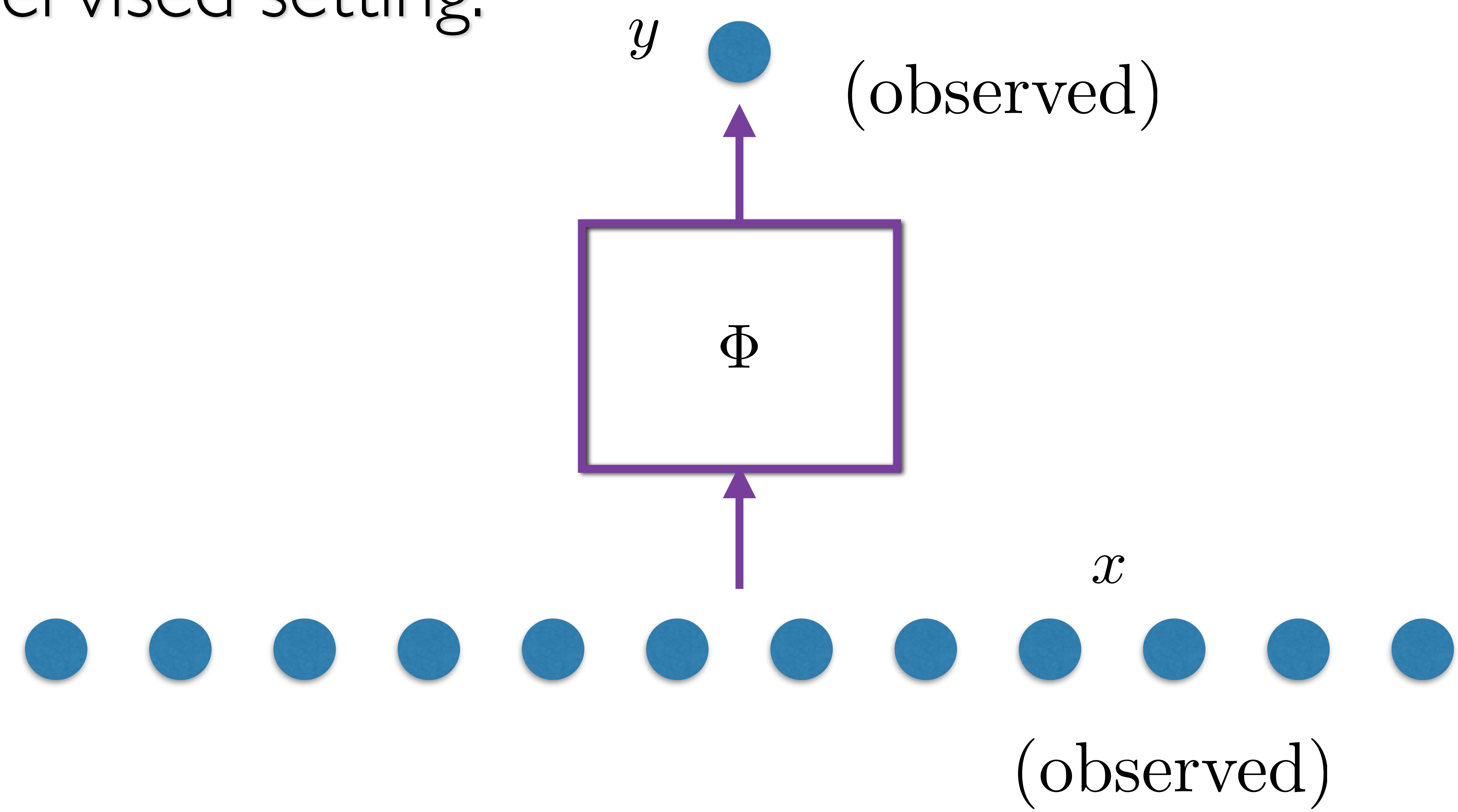
- When $p(x|\theta) = \exp\left(\langle \theta, \Phi(x) \rangle - A(\theta)\right)$, the Fisher vector is

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

- Thus the maximum entropy model is the "canonical" generative model associated with linearization features Φ

Unsupervised Gibbs Models

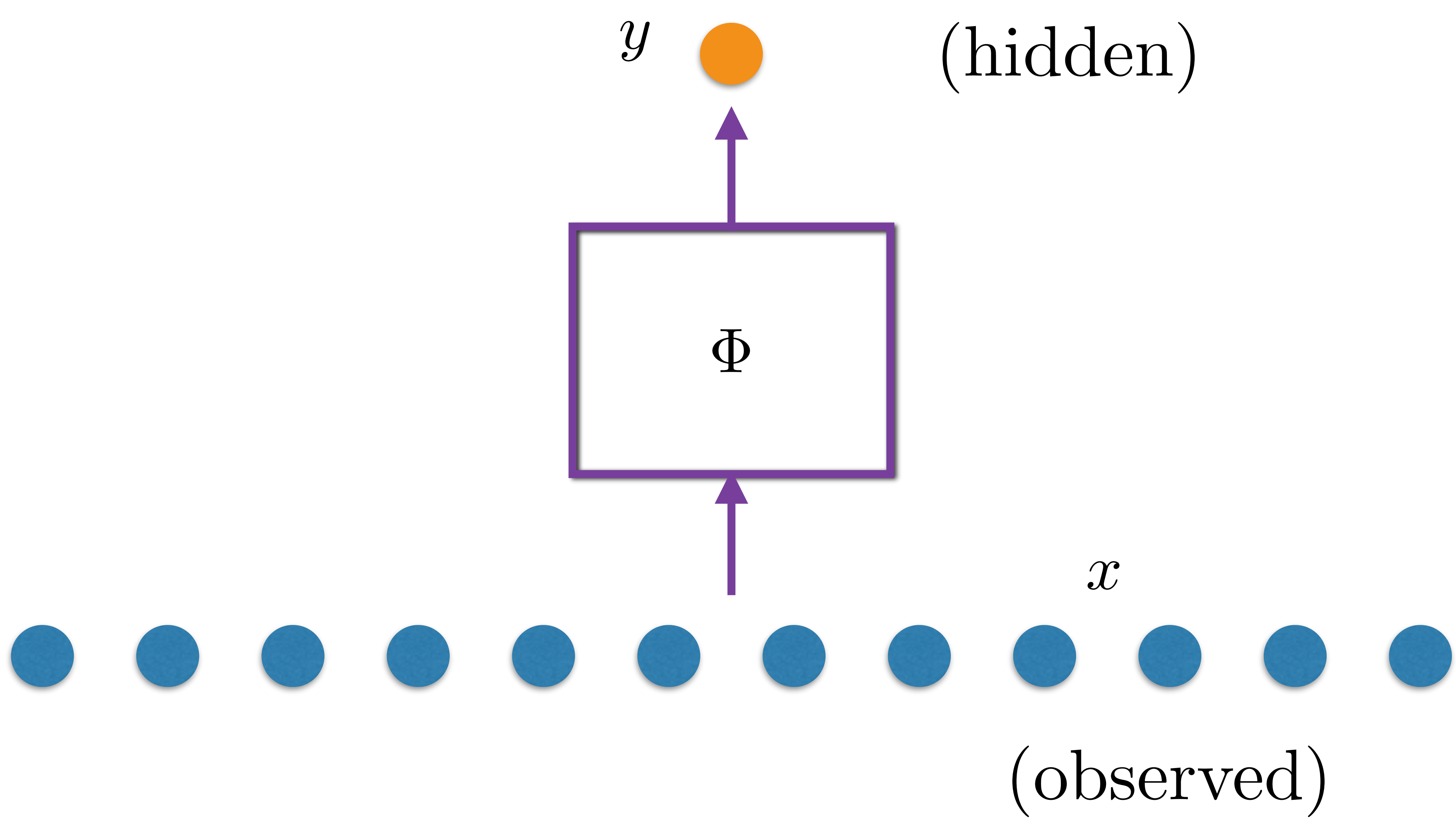
- We have derived a fully probabilistic model in a supervised setting:



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

Unsupervised Gibbs Models

- How about when no labels are observed?



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

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

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 θ 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 θ 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

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

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.

Markov Chain Monte-Carlo

- Basic principle: construct a Markov chain whose equilibrium distribution is precisely $p(x)$ and such that the transition distributions are easy to sample from:

$$x_0 \sim q_0(x) \qquad\qquad x_t \sim q(x|x_{t-1})$$

$$x_0 \to x_1 \to x_2 \to \dots$$

- Two very important algorithms (there are many more)
  - Metropolis-Hastings: very generic
  - Gibbs Sampler: for models with small factors.

Markov Chain Monte-Carlo

- Metropolis-Hastings [1953]: assumes one can easily compute a function $f(x)$ proportional to $p(x)$.
- Let $q(x|y)$ be a proposal transition kernel that is *irreducible*, i.e., it can move to any point in the state space.

- Given $X^{(t)} = x^{(t)}$

generate $Y_t \sim q(y|x^{(t)})$. and define

$$X^{(t+1)} = \begin{cases} Y_t & \text{with probability } \rho(x^{(t)}, Y_t) \ , \\ x^{(t)} & \text{with probability } 1 - \rho(x^{(t)}, Y_t) \ , \end{cases}$$

$$\text{with } \rho(x, y) = \min\left(1, \frac{f(y)q(x|y)}{f(x)q(y|x)}\right)$$

Markov Chain Monte-Carlo

Proposition [M-H,'53]: If $q(y|x)$ is irreducible ($q(y|x) > 0$ for all x, y), the chain converges to the stationary distribution $p(x)$.

Markov Chain Monte-Carlo

Proposition [M-H,'53]: If $q(y|x)$ is irreducible ($q(y|x) > 0$ for all x, y), the chain converges to the stationary distribution $p(x)$.

- This result does not inform about how fast we reach this stationary distribution (mixing time).
- Proposal distribution too wide: we might reject too often
- Proposal distribution too narrow: long mixing time.

Markov Chain Monte-Carlo

Proposition [M-H,'53]: If $q(y|x)$ is irreducible ($q(y|x) > 0$ for all x, y), the chain converges to the stationary distribution $p(x)$.

- This result does not inform about how fast we reach this stationary distribution (mixing time).
- Proposal distribution too wide: we might reject too often
- Proposal distribution too narrow: long mixing time.
- Several extensions
  - Langevin Dynamics: attempt to climb in the direction of $\nabla \log p(x)$, eg

$$Y_{n+1}|x_n \sim \mathcal{N}(x_n + \gamma \nabla \log p(x), \gamma \Sigma)$$

Gibbs sampler [Geman & Geman,'84]

- A special case of M-H algorithm when it is easy to sample from the conditional distributions

$$p(x_{(k)}|x_{(1)},\ldots,x_{(k-1)},x_{(k+1)},\ldots,x_{(n)})$$

Gibbs sampler [Geman & Geman,'84]

- A special case of M-H algorithm when it is easy to sample from the conditional distributions

$$p(x_{(k)}|x_{(1)},\ldots,x_{(k-1)},x_{(k+1)},\ldots,x_{(n)})$$

- In that case, the proposal distributions are of the form

$$q(x|y) = p(x_{(k)}|y_{(1)},\ldots,y_{(k-1)},y_{(k+1)},\ldots,y_{(n)})$$

- It refines a reversible Markov chain with stationary distribution $p(x)$
- Used extensively in Markov Random Fields models.

MCMC Pros and Cons

- Generic, provably correct sampling methods.
- Can scale to high-dimensional density models.

MCMC Pros and Cons

- Generic, provably correct sampling methods.
- Can scale to high-dimensional density models.
- Computationally expensive: in order to have good estimators of the gradient we require many iterations (samples are not statistically independent in general)
- The method does not inform about how to optimally select the proposal distributions.

MCMC Pros and Cons

- Generic, provably correct sampling methods.
- Can scale to high-dimensional density models.
- Computationally expensive: in order to have good estimators of the gradient we require many iterations (samples are not statistically independent in general)
- The method does not inform about how to optimally select the proposal distributions.

- Alternative/Extensions:
  - Annealed Importance Sampling [Neal'98]
  - Hybrid Monte-Carlo (e.g. Langevin)
  - Variational Inference

From unsupervised to *self-supervised* learning

From unsupervised to *self-supervised* learning

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

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

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

From unsupervised to *self-supervised* learning

- 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

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

From unsupervised to *self-supervised* learning

- 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?

From unsupervised to *self-supervised* learning

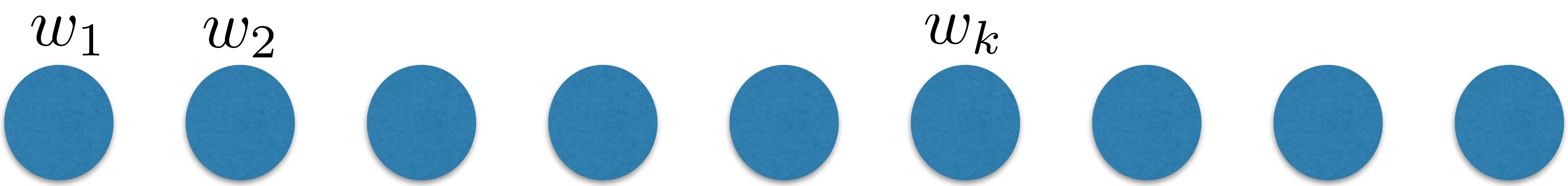
- Temporally ordered data
  - Speech, Music
  - Video
  - Language
  - Other time series (Weather, Finance, …)
- Spatially ordered data, Multi-Resolution data
  - Images

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

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

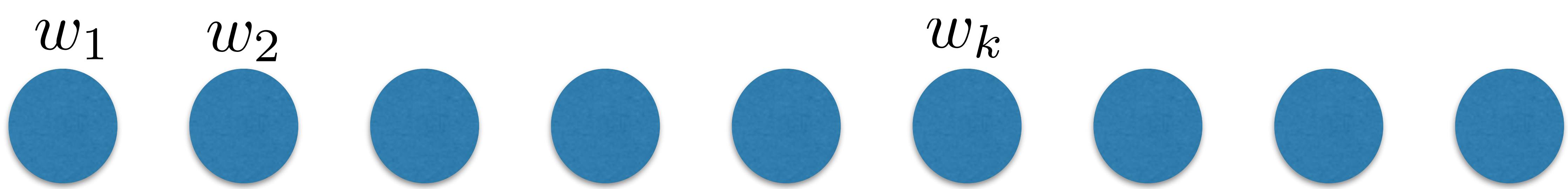
Word2vec [Mikolov et al.'13].

- Unsupervised learning "success story".



Word2vec [Mikolov et al.'13].

- Unsupervised learning "success story".

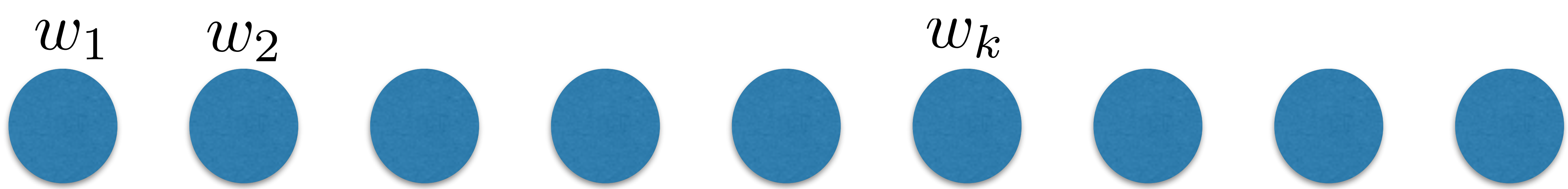


- Language creates a notion of similarity between words:

  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

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

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

- Training with Maximum Likelihood:

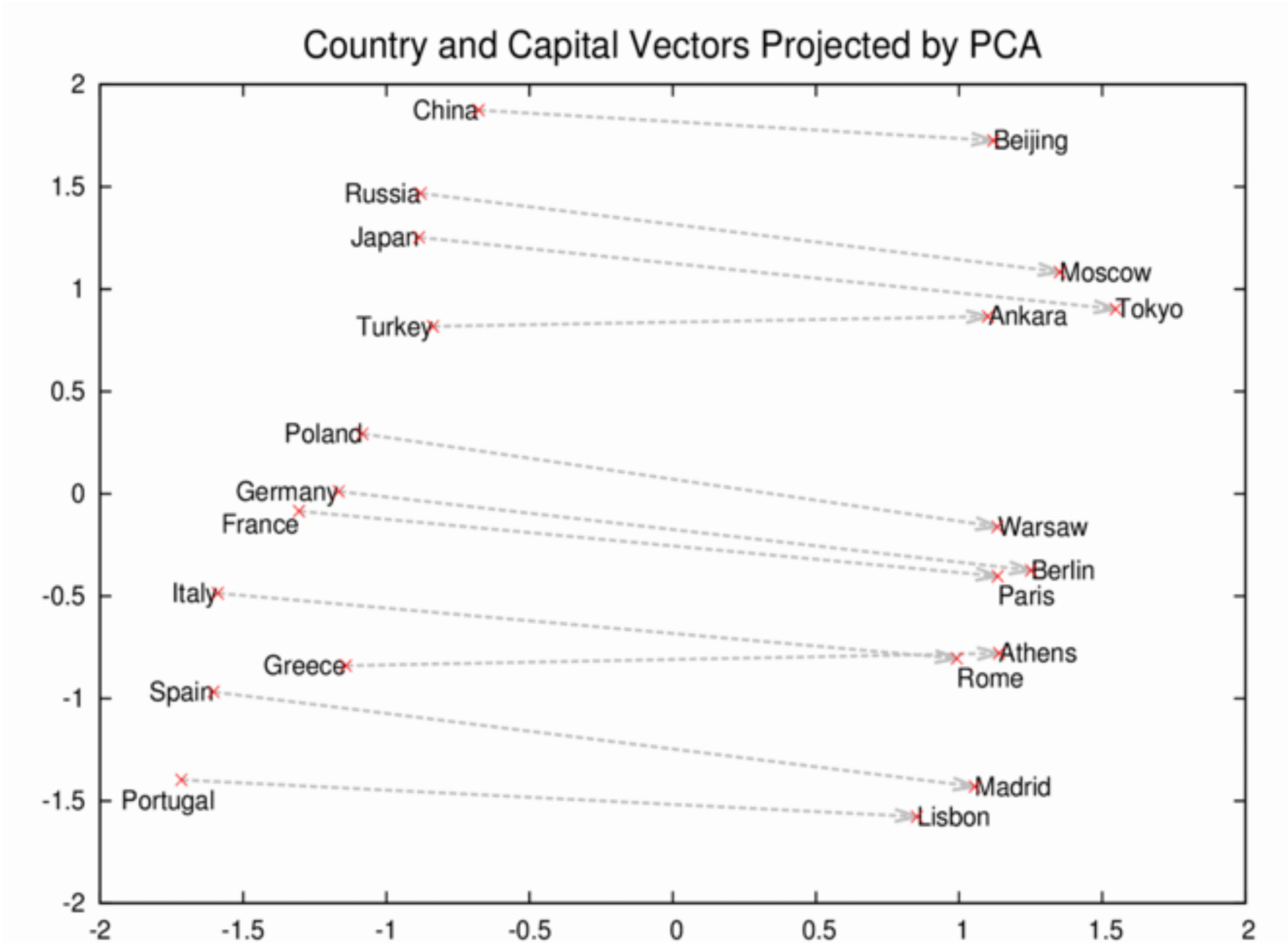
$$\arg\max_\theta \prod_{(w,c)\sim\mathcal{D}} p(D = 1 | c, w, \theta) \prod_{(w,c)\sim\mathcal{D}'} p(D = 0 | c, w, \theta)$$

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

$\mathcal{D}$: 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 $p(x_{t+1}|x_1, \dots, x_t)$ where $(x_t)_t$ is temporally ordered data.

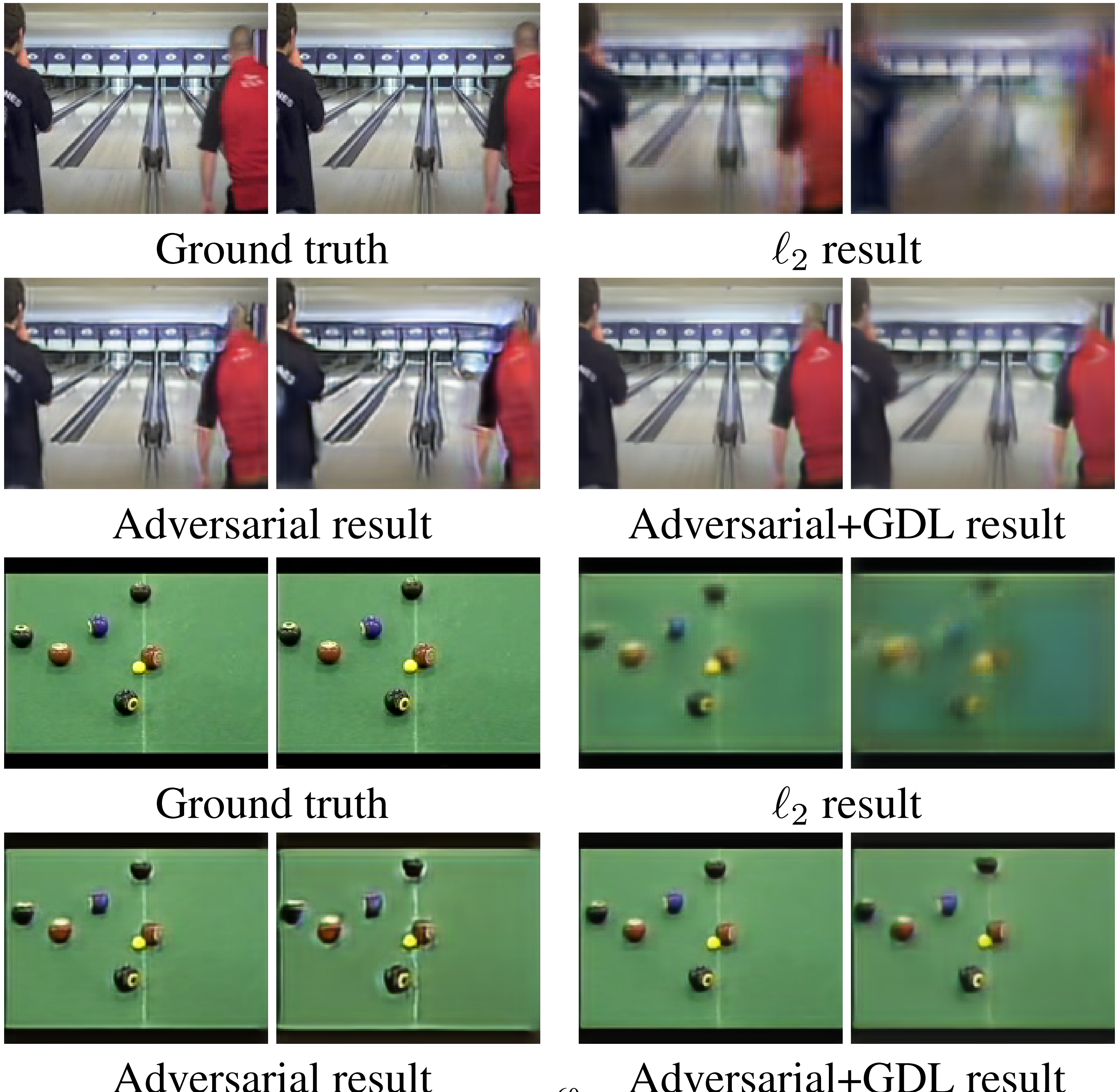
- Similarly, can we find a signal representation $\Phi(x_t)$ that is consistent with the "video language" metric? i.e.

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

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

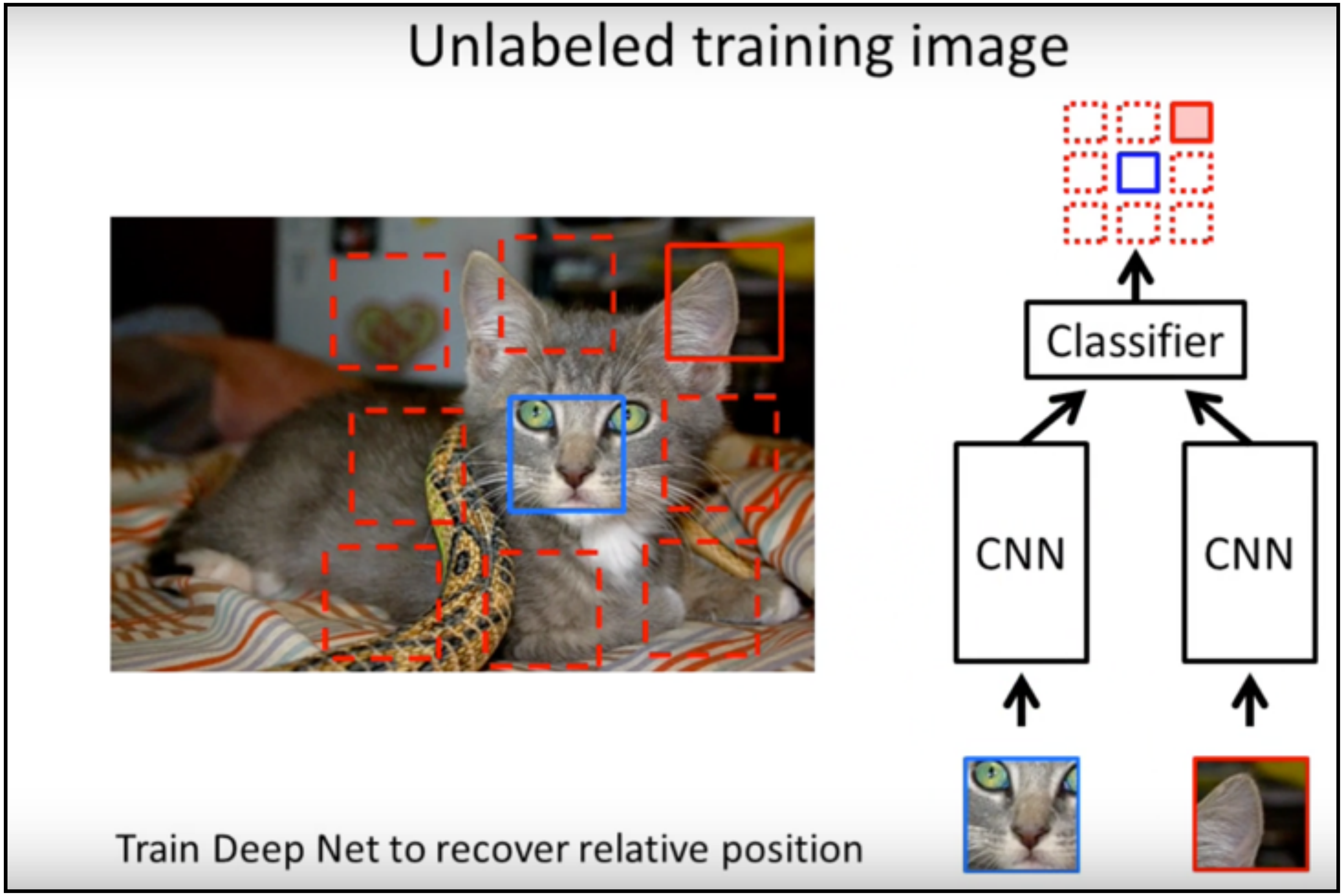
Video Prediction

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

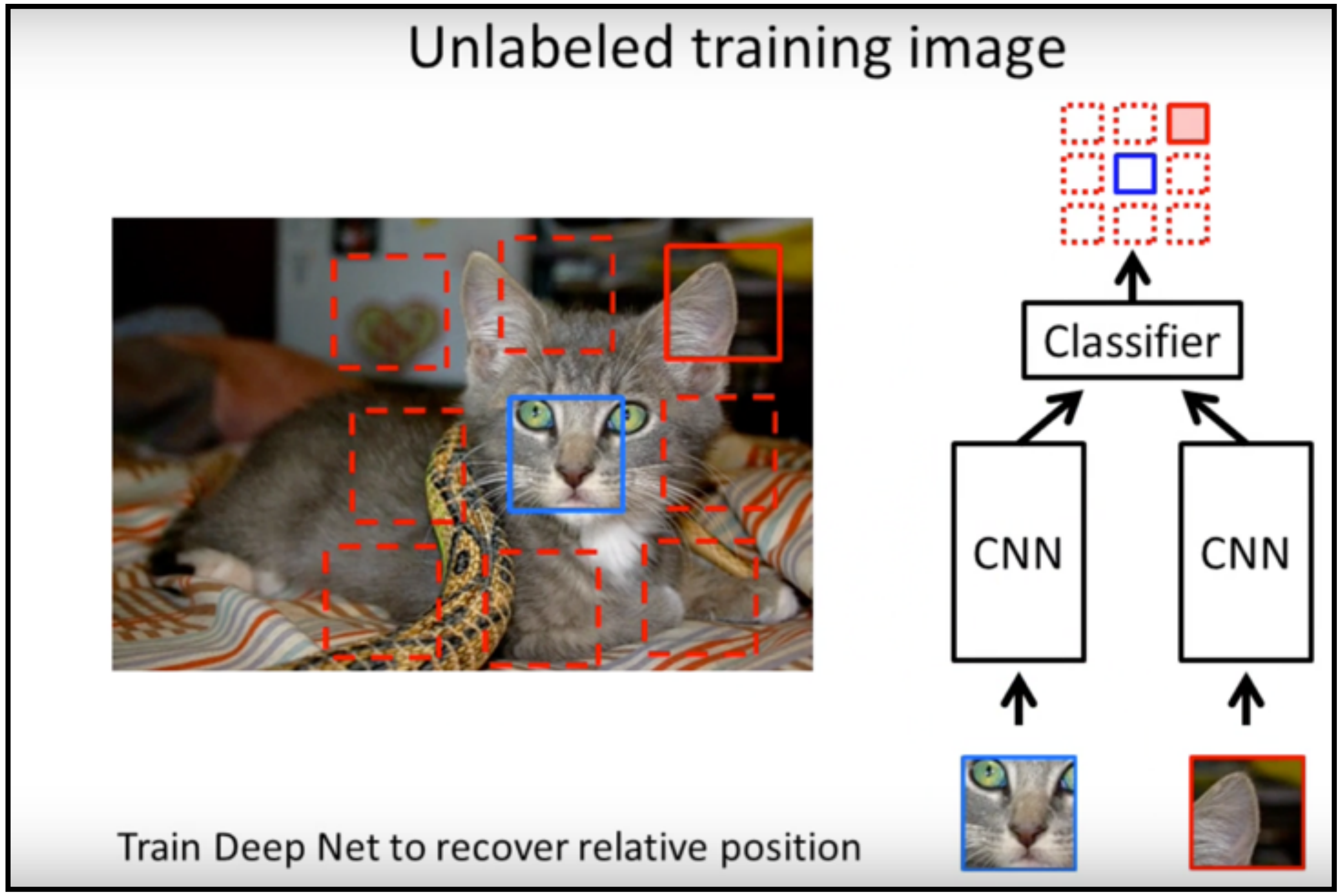


Patch Relative Configuration [Doerch et al.'15]

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



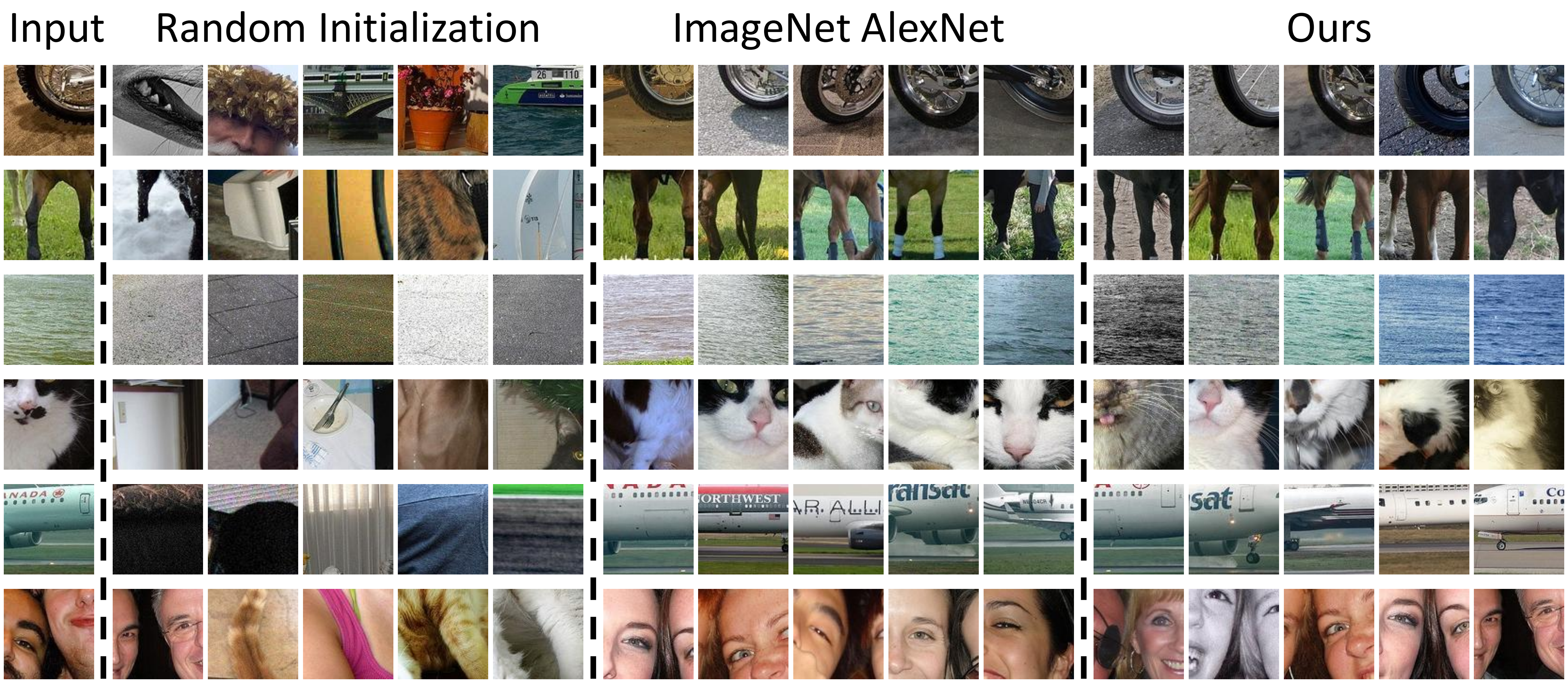
Patch Relative Configuration [Doerch et al.'15]



- 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

- 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

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

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

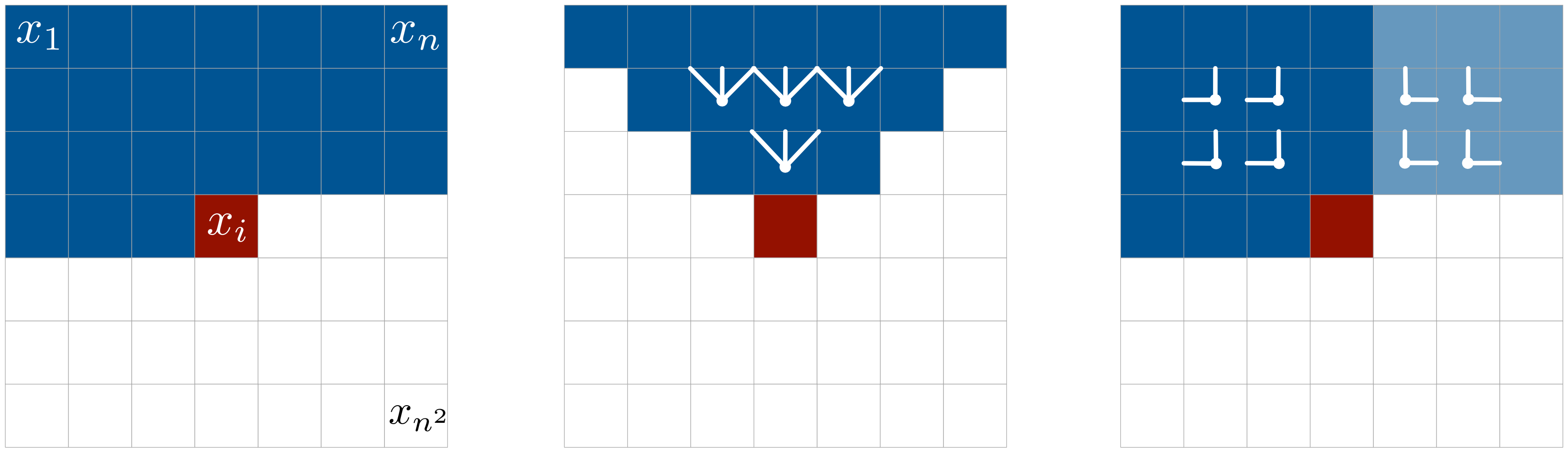
- In discrete domains we simply use a multinomial loss, in continuous domains there is no principled choice.

- How about images?
  - We can treat them as discrete two-dimensional grids $x(u) \in \{0, 255\}$
  - Model each pixel from its "past" context:

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

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

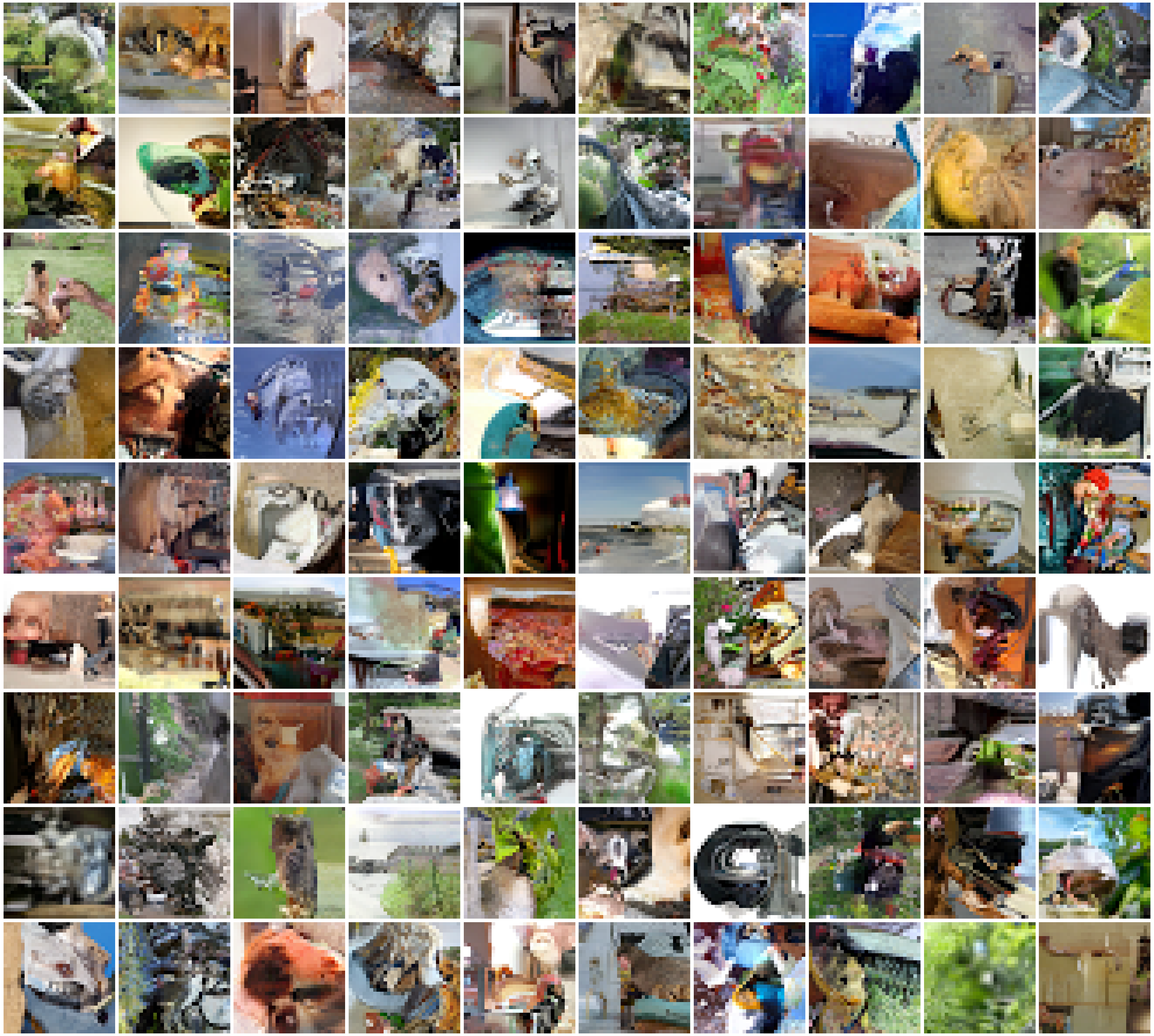
- Contexts are modeled using "diagonal BiLSTMS".



- 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]

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

- MNIST and Cifar-10 log-likelihoods

Model	**NLL Test**
DBM 2hl [1]:	≈ 84.62
DBN 2hl [2]:	≈ 84.55
NADE [3]:	88.33
EoNADE 2hl (128 orderings) [3]:	85.10
EoNADE-5 2hl (128 orderings) [4]:	84.68
DLGM [5]:	≈ 86.60
DLGM 8 leapfrog steps [6]:	≈ 85.51
DARN 1hl [7]:	≈ 84.13
MADE 2hl (32 masks) [8]:	86.64
DRAW [9]:	≤ 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).