Learning Deep Latent Variable Models by Short-Run MCMC Inference with Optimal Transport Correction

Jianwen Xie

Cognitive Computing Lab

@Baidu, Apr 26, 2021.

Related Paper

This talk is based on the following paper

Learning Deep Latent Variable Models by Short-Run MCMC Inference with Optimal Transport Correction

-- Dongsheng An, Jianwen Xie, Ping Li

-- *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2021*

Outline

1. Background of Deep Latent Variable Models

2. Short-Run MCMC Inference with Optimal Transport Correction

3. Experiments and Results

1. Background of Deep Latent Variable Models

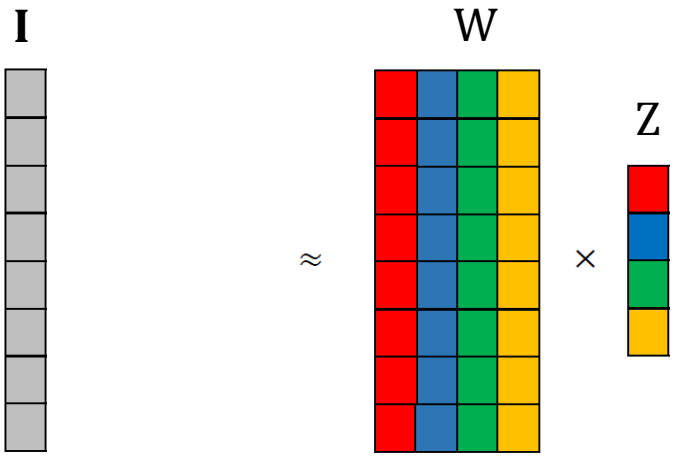
Factor Analysis and Beyond

Factor Analysis Model

Let $\mathbf{I}$ be a D-dimensional signal (e.g., image), and Z be a d-dimensional latent vector, i.e., $Z = (z_i, i = 1, \ldots, d)$. The traditional factor analysis model is of the form

$$\mathbf{I} \approx WZ + \epsilon,$$

where ϵ is a small Gaussian reconstruction error (observation residual), we also assume $Z \sim N(0, I)$, and D is a $D \times d$ matrix.



Factor Analysis and Beyond

Factor Analysis Model

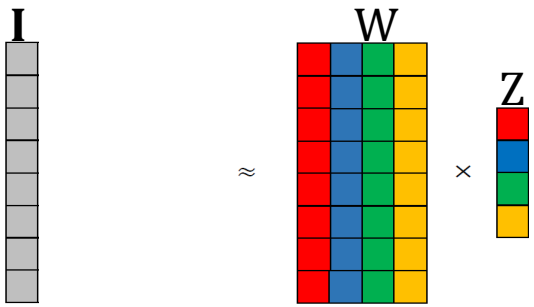
$$\mathbf{I} \approx WZ + \epsilon,$$

There are three perspectives to view W:

(1) Basis vectors. Write $W = (W_1, \ldots, W_d)$, where each W_i is a D-dimensional column vector. Then

$$\mathbf{I} \approx \sum_{i=1}^{d} W_i z_i + \epsilon,$$

i.e., $\{W_i\}$ are the basis vectors and $\{z_i\}$ are the coefficients.



Factor Analysis and Beyond

Factor Analysis Model

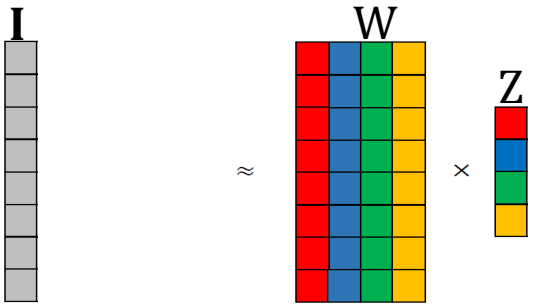
$$\mathbf{I} \approx WZ + \epsilon,$$

There are three perspectives to view W:

(2) Loading matrix. Write $W = (w_1, \ldots, w_D)^T$, where each w_j is row vector of W. Then

$$\mathbf{I_j} = < w_j, Z > + \epsilon_j$$

Each $\mathbf{I_j}$ is a loading of the d factors where w_j is a vector of loading weights, indicating which factors are important for determining $\mathbf{I_j}$. W is called the loading matrix.



Factor Analysis and Beyond

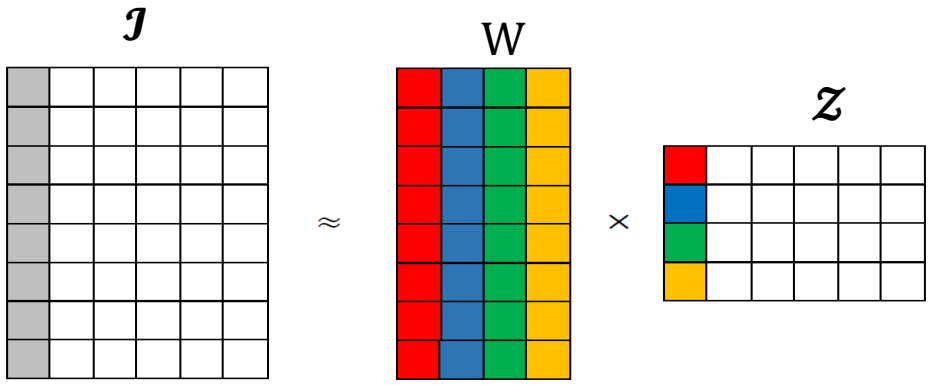
Factor Analysis Model

$$\mathbf{I} \approx WZ + \epsilon,$$

There are three perspectives to view W:

(2) Matrix Factorization. Suppose we observe $\mathcal{I} = (\mathbf{I}_1, \ldots, \mathbf{I}_n)$, whose factors are $\mathcal{Z} = (Z_1, \ldots, Z_n)$, then

$$\mathcal{I} \approx W\mathcal{Z}$$



Factor Analysis and Beyond

The factor analysis model can be learned by the Rubin-Thayer EM algorithm, which involves alternating regressions of Z on $\mathbf{I}$ in the E-step and of W on $\mathbf{I}$ in the M-step, with both steps powered by the sweep operator (Rubin and Thayer, 1982; Liu, Rubin, and Wu, 1998)

[1] Rubin, D. B., and Thayer, D. T. 1982. EM algorithms for ML factor analysis. Psychometrika 47(1):69–76.

[2] Liu, C.; Rubin, D. B.; and Wu, Y. N. 1998. Parameter expansion to accelerate em: The px-em algorithm. Biometrika 85(4):755–770.

Factor Analysis and Beyond

Factor analysis is related to principal component analysis, where W is obtained by the first *d* eigenvectors of the covariance matrix Cov($\mathbf{I}$).

The factor analysis model can be generalized to **independent component analysis** (Hyvärinen et al. 2004), **sparse coding** (Olshausen & Field 1997), **nonnegative matrix factorization** (Lee & Seung 2001), **recommender systems** (Koren et al. 2009) and so on, by modifying the prior distribution or prior assumption on Z.

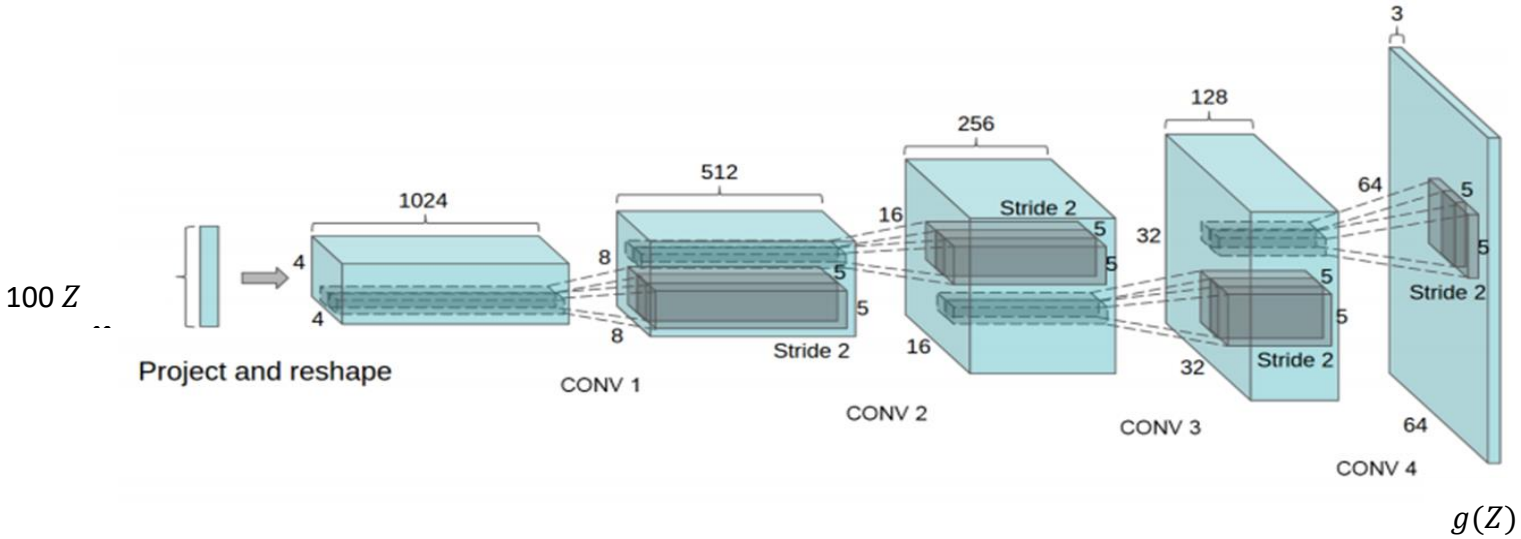
If we generalize the linear mapping from Z to $\mathbf{I}$ to a nonlinear mapping parameterized by a deep network, then the resulting model is commonly called **generator network** (Goodfellow et al. 2014, Kingma & Welling 2014).

Factor analysis is an example of generative representation, where the hidden vector Z generates the observed vector $\mathbf{I}$. We can also call it **latent variable model**, which is a generative model.

Deep Latent Variable Model

nonlinear mapping by neural network (generator network)

$$\mathbf{I} = g(Z; W_G)$$



$g(Z)$

Observed data (e.g.,images): $\{\mathbf{I}_i, i = 1, \ldots, n\}$

Corresponding latent vectors: $\{Z_i, i = 1, \ldots, n\}$

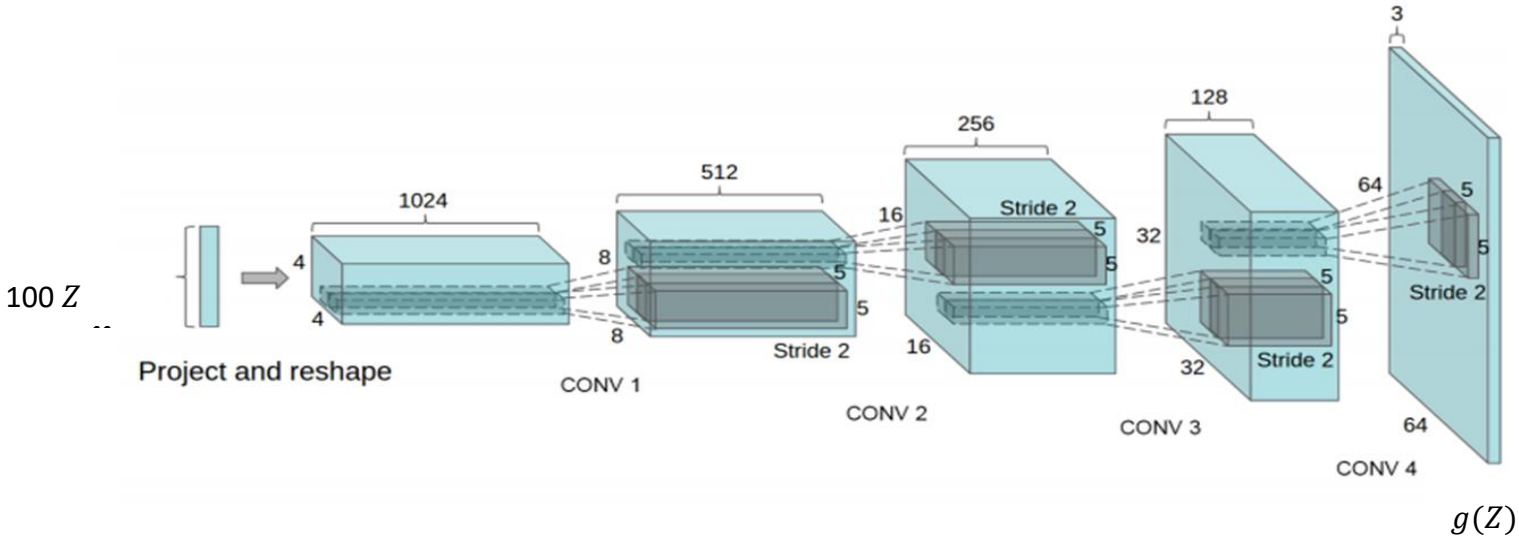
all $\mathbf{I}_i$ share the same ConvNet W_G

We also assume Z follows Gaussian distribution. (prior distribution)

Deep Latent Variable Model

nonlinear mapping by neural network (generator network)

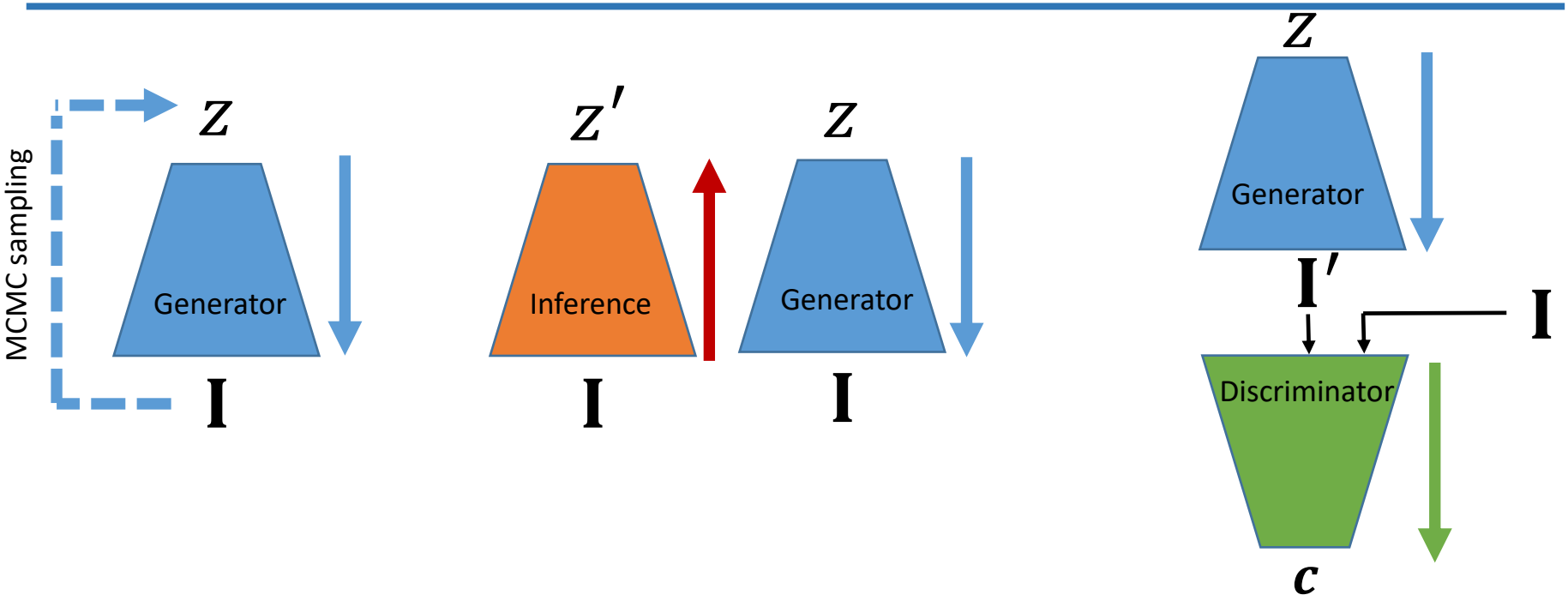
$$\mathbf{I} = g(Z; W_G)$$



(1) This type of model is very important because of the form is simple, natural, and also corresponds to a lot of classical problem.

(2) However, learning such a model is challenging due to non-linear parameterization of g. VAE and GAN are currently popular way to train it. These two models train the generator by recruiting an extra model for assisting in the training, and will disregard it in testing.

Different Ways of Training



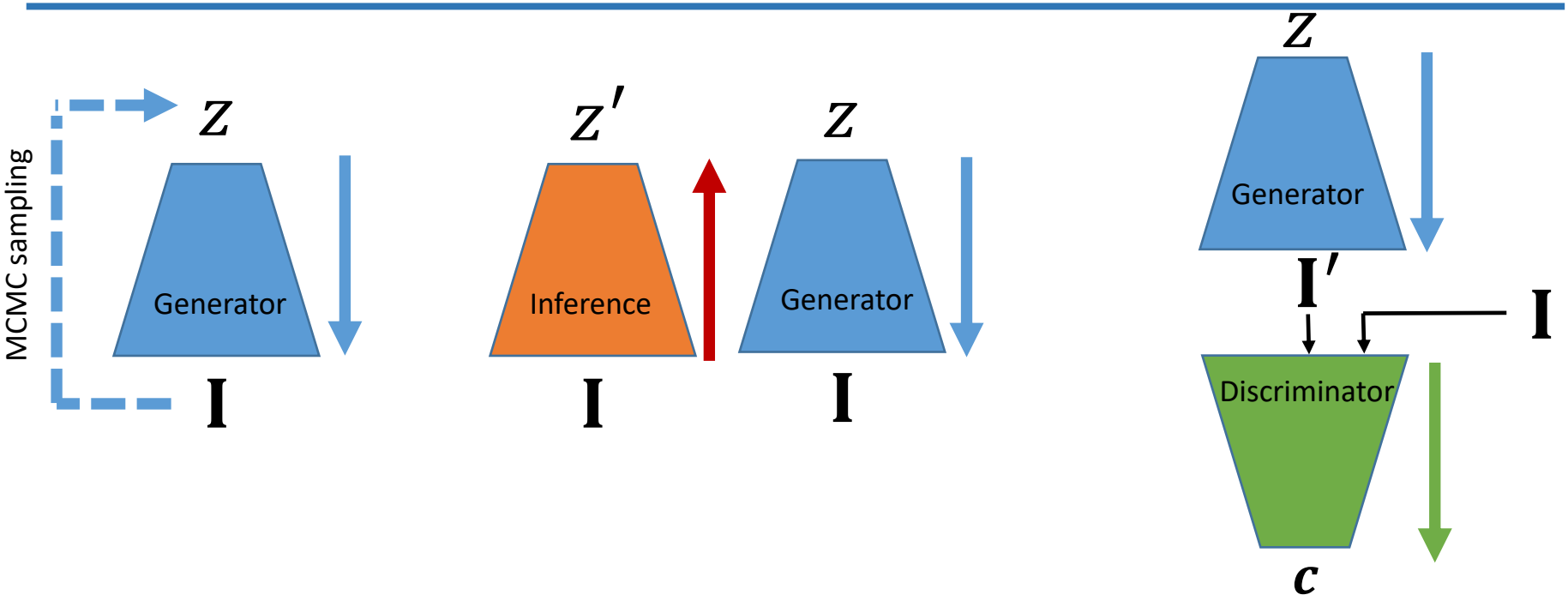
MCMC-based MLE [1] **Variational auto-encoder (VAE) [2]** **Generative Adversarial network (GAN) [3]**

[1] Tian Han, Yang Lu, Song-Chun Zhu, Ying Nian Wu. **Alternating Back-Propagation for Generator Network**. AAAI 2017.

[2] Diederik P. Kingma, Max Welling. **Auto-Encoding Variational Bayes**. ICLR 2014.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. **Generative Adversarial Nets,** NIPS 2014.

Different Ways of Training



MCMC-based MLE

Cons:
- MCMC is hard to converge

Pros:
- Simple (less parameters)
- No model collapse (v.s GAN)
- No design difficulty for inference

Variational auto-encoder (VAE)

Cons:
- Inaccurate inference.
- Two sets of parameters.
- Hard to design inference model

Pros:
- Fast inference

Generative Adversarial network (GAN)

Cons:
- Two sets of parameters
- No inference
- Mode collapse in training
- Hard to design inference model

Pros:
- Avoid inference in training

Examples of design difficulty of inference net

An extra effort is required to made in designing the inference model of VAE, especially for the generators that have complicated dependency structures with the latent variables, e.g.,

(1) Latent vectors are included in each layers of the generators

(2) Dynamic (recurrent) generator with latent vectors at each time step

It is not a simple task to design inference models that infer latent variables for models mentioned above. An arbitrary design of the inference model cannot guarantee the performance.

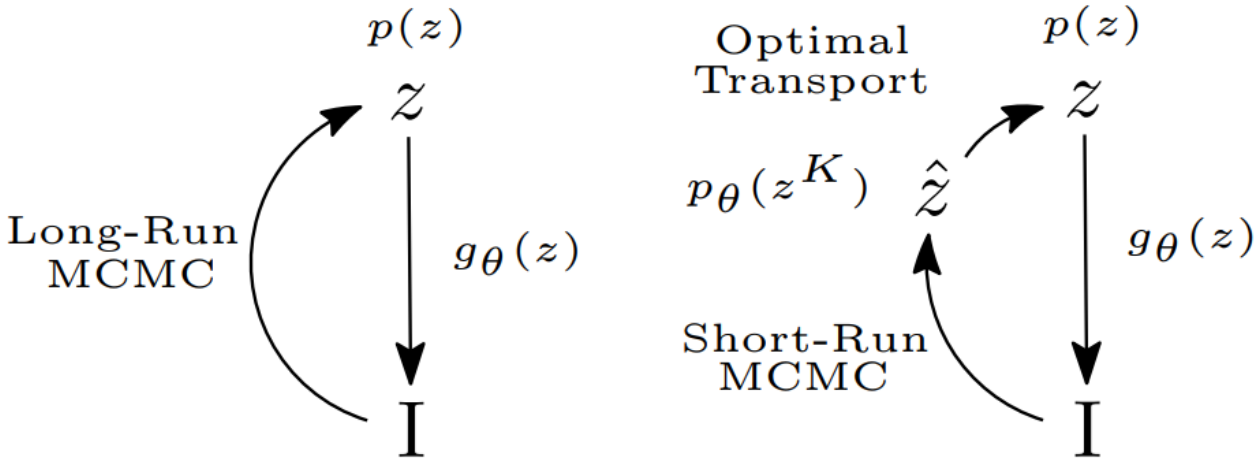
What we study in our paper?

We will totally abandon the idea of reparameterizing the inference process as in VAE, and will study the MCMC-based inference for training deep latent variable models.

To be specific, we use a short-run MCMC, such as a short-run Langevin dynamics, to perform the inference of the latent vectors during training.

However, the convergence of finite-step Langevin dynamics in each iteration might be questionable, so we accept the bias existing in such a short-run MCMC and propose to use the optimal transport (OT) method to correct the bias.

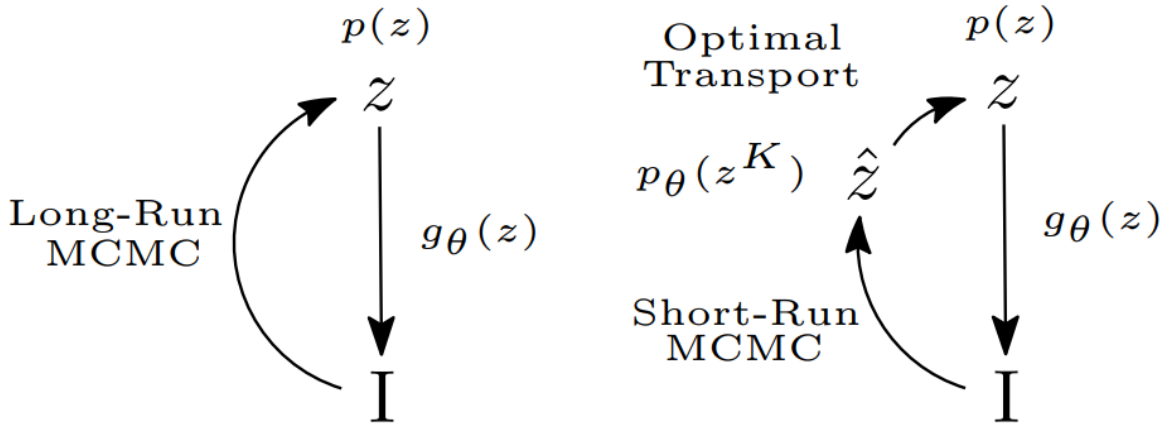
Long-run v.s. short-run MCMC inference



Diagrams of two learning strategies for latent variable models:

(left) the long-run MCMC inference framework.
(right) the proposed framework using a short-run MCMC with OT correction

Long-run v.s. short-run MCMC inference



(i) **Efficiency**: The learning and inference of the model are efficient with a short-run MCMC

(ii) **Convenience**: The approximate inference model represented by the short-run MCMC is automatic in the sense that there is nothing to worry about the design and training of a separate inference model. Both bottom-up inference and top-down generation are governed by the same set of parameters.

(iii) **Accuracy**: the optimal transport corrects the errors of the nonconvergent short-run MCMC inference, thus improves the accuracy of the model parameter estimation.

2. Short-Run MCMC Inference with Optimal Transport Correction

MLE Learning algorithm

Given a set of training examples $\{\mathbf{I}_i, i = 1, \ldots, n\} \sim p_{\text{data}}(\mathbf{I})$, where $p_{\text{data}}(\mathbf{I})$ is the unknown data distribution. We can train p_θ by maximizing the log-likelihood of the training samples

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(\mathbf{I}_i), \tag{2}$$

which is equivalent to the minimization of $\text{KL}(p_{\text{data}} || p_\theta)$

MLE Learning algorithm

The maximization of the log-likelihood function presented in Eq. (2) can be accomplished by gradient ascent algorithm that iterates

$$\theta_{t+1} = \theta_t + \gamma_t \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log p_\theta(\mathbf{I}_i), \tag{3}$$

where γ_t is the learning rate depending on time t and the gradient of the log probability is given by

$$\begin{aligned} \nabla_\theta \log p_\theta(\mathbf{I}) &= \frac{1}{p_\theta(\mathbf{I})} \nabla_\theta p_\theta(\mathbf{I}) \\ &= \int [\nabla_\theta \log p_\theta(\mathbf{I}, z)] \frac{p_\theta(\mathbf{I}, z)}{p_\theta(\mathbf{I})} dz \\ &= \mathrm{E}_{p_\theta(z|\mathbf{I})}[\nabla_\theta \log p_\theta(\mathbf{I}, z)]. \end{aligned} \tag{4}$$

MLE Learning algorithm

$$\mathrm{E}_{p_\theta(z|\mathbf{I})}[\nabla_\theta \log p_\theta(\mathbf{I}, z)]$$

$$p_\theta(\mathbf{I}, z) = p_\theta(\mathbf{I}|z)p(z),$$

where we assume the prior distribution $p(z) = \mathcal{N}(0, I_d)$ and $p(\mathbf{I}|z) = \mathcal{N}(g_\theta(z), \sigma^2 I_D)$.

$$\log p_\theta(\mathbf{I}, z) = -\frac{1}{2\sigma^2}\|\mathbf{I} - g_\theta(z)\|^2 - \frac{1}{2}\|z\|^2 + \text{const},$$

$\nabla_\theta \log p_\theta(\mathbf{I}, z) = \frac{1}{\sigma^2}(\mathbf{I} - g_\theta(z))\nabla_\theta g_\theta(z)$, where $\nabla_\theta g_\theta(z)$ can be efficiently computed by back-propagation.

Long-run MCMC inference

Given a step size $s > 0$, and an initial value z^0, Langevin dynamics [19, 45], which is a gradient-based MCMC method, can produce samples from the posterior density $p_\theta(z|\mathbf{I})$ by recursively computing

$$z^{k+1} = z^k + \frac{s^2}{2}\nabla_z \log p_\theta(z|\mathbf{I}) + s\xi_k, \tag{6}$$

where k indexes the time step of Langevin dynamics, $\xi_k \sim \mathcal{N}(0, I_d)$ is a random noise diffusion. Also, $\nabla_z \log p_\theta(z|\mathbf{I}) = \frac{1}{\sigma^2}(\mathbf{I} - g_\theta(z))\nabla_z g_\theta(z) - z$, where $\nabla_z g_\theta(z)$ can be efficiently computed by back-propagation.

Let us use K to denote the number of Langevin steps. When $s \to 0$ and $K \to \infty$, no matter what the initial distribution of z^0 is, z^K will converge to the posterior distribution $p_\theta(z|\mathbf{I})$ and become a fair sample from $p_\theta(z|\mathbf{I})$.

Long-run MCMC inference

Iterative learning algorithm:

Inference Step:

$$z^{k+1} = z^k + \frac{s^2}{2}\nabla_z \log p_\theta(z|\mathbf{I}) + s\xi_k,$$

Learning Step:

$$\theta_{t+1} = \theta_t + \gamma_t \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log p_\theta(\mathbf{I}_i)$$

[1] Tian Han, Yang Lu, Song-Chun Zhu, Ying Nian Wu. **Alternating Back-Propagation for Generator Network**. AAAI 2017.

Long-run MCMC inference

Iterative learning algorithm:

Inference Step:

$$z^{k+1} = z^k + \frac{s^2}{2}\nabla_z \log p_\theta(z|\mathbf{I}) + s\xi_k,$$

It is not sensible or realistic to use a long-run MCMC to train the model. Within each iteration, running a finite number of Langevin steps for inference toward $p_\theta(z|\mathbf{I})$ appears to be practical.

Learning Step:

$$\theta_{t+1} = \theta_t + \gamma_t \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log p_\theta(\mathbf{I}_i)$$

[1] Tian Han, Yang Lu, Song-Chun Zhu, Ying Nian Wu. **Alternating Back-Propagation for Generator Network**. AAAI 2017.

Short-run MCMC inference

K-step short run Langevin dynamics

$$z^0 \sim p_0(z),$$

$$z^{k+1} = z^k + \frac{s^2}{2}\nabla_z \log p_\theta(z|\mathbf{I}) + s\xi_k, k = 1, .., K.$$

Such a dynamics can be treated as a conditional generator that transforms a random noise z_0 to the target distribution under the condition $\mathbf{I}$. And the transformation itself can also be treated as a K-layer residual network, where each layer shares the same parameters θ and has a noise injection.

The conditional distribution of z^K given $\mathbf{I}$ is

$$q_\theta(z^K|\mathbf{I}) = \int p_0(z^0)\kappa_\theta(z^K|z^0, \mathbf{I})dz^0$$

The corresponding marginal distribution of z^K is

$$q_\theta(z^K) = \int q_\theta(z^K|\mathbf{I})p_{\text{data}}(\mathbf{I})d\mathbf{I}.$$

$q_\theta(z^K)$ should be close to the prior $p(z)$, otherwise there is a gap between them.

Short-run MCMC inference

We revise the original Iterative learning algorithm by adding a correction step by OT:

(1) Inference Step:

$$z^{k+1} = z^k + \frac{s^2}{2}\nabla_z \log p_\theta(z|\mathbf{I}) + s\xi_k,$$

We are not sampling from $\hat{z} \sim p_\theta(z|\mathbf{I})$, but $\hat{z} \sim p_\theta(z^K|\mathbf{I})$.

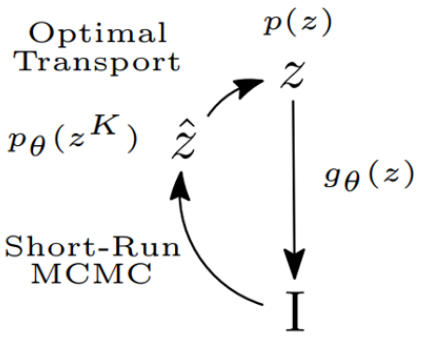
(2) Correction Step:

Compute the approximate Optimal Transport (OT) map $\hat{\mathrm{T}}$ from $\{\hat{z}_i\}$ to $\{z_j\}$

$$\hat{z}_i \leftarrow \alpha\hat{T}(\hat{z}_i) + (1-\alpha)\hat{z}_i$$

(3) Learning Step:

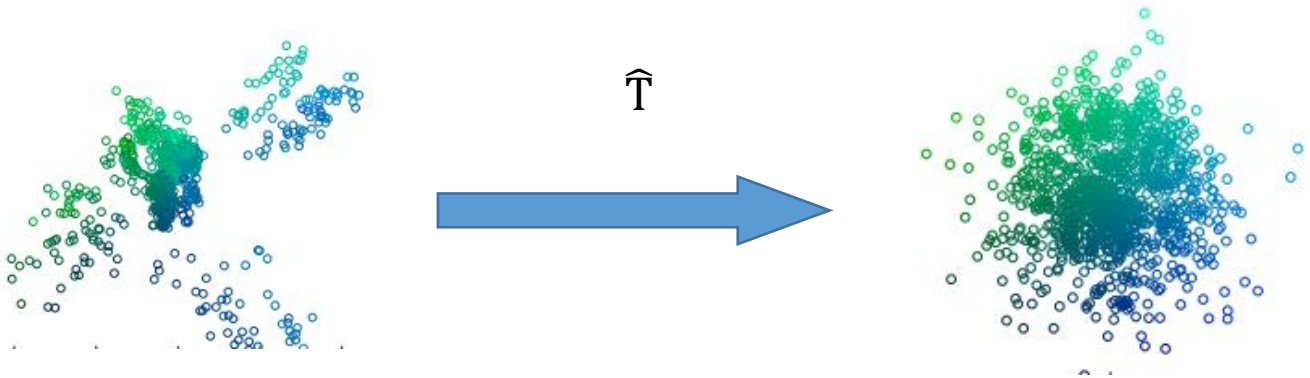
$$\theta_{t+1} = \theta_t + \gamma_t \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log p_\theta(\mathbf{I}_i)$$



Optimal Transport for Correction

Latent Space Z

Optimal Transport (OT) map $\hat{\mathrm{T}}$ from $\{\hat{z}_i\}$ to $\{z_j\}$



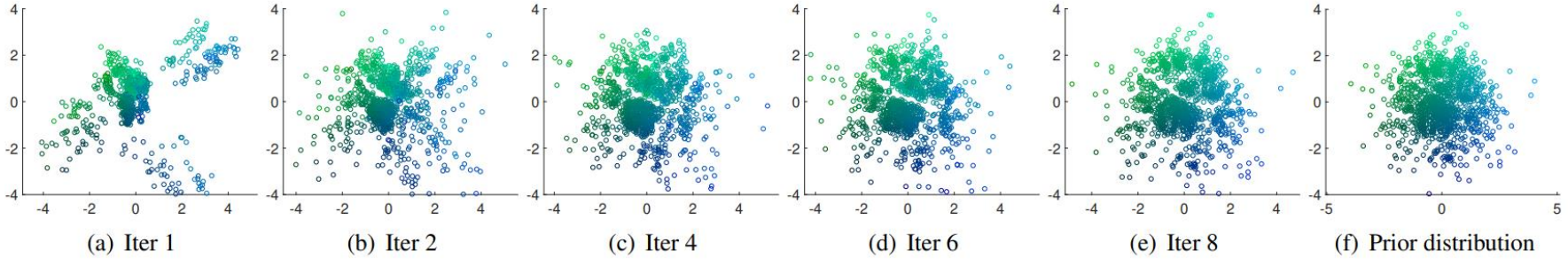
Output Population by short-run MCMC $\{\hat{z}_i\}$

$\{z_j\} \sim N(0,1)$

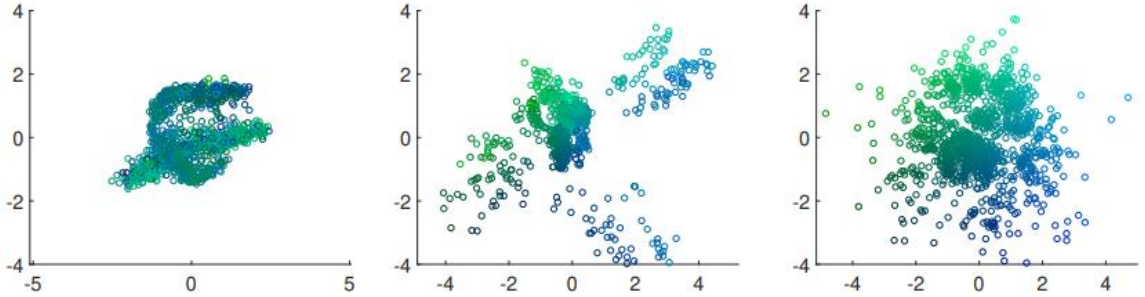
We reshape the population to the target distribution by OT. (assignment problem)

OT will move the source population to the target distribution with a minimum cost (i.e., moving distance).

Visualization of the latent codes



Visualization of the latent codes sampled from the marginal distribution $q_\theta(z^K)$ at different iterations and the prior distribution



(a) VAE (b) short-run only [1] (c) short-run with OT (ours)

[1] Tian Han, Yang Lu, Song-Chun Zhu, Ying Nian Wu. **Alternating Back-Propagation for Generator Network**. AAAI 2017.

2. Experiments and Results

Synthesis and Reconstruction

The reconstructed (the first row) and the generated images (the second row) of MNIST (the first column), SVHN (the second column) and CelebA (the third column) datasets.

Synthesis and Reconstruction

Models		VAE	2sVAE	RAE	ABP	SRI	SRI (L=5)	LEBM	Ours
MNIST	MSE	0.023	0.026	0.015	-	0.019	0.015	-	**0.0008**
	FID	19.21	18.81	23.92	-	-	-	-	**14.28**
SVHN	MSE	0.019	0.019	0.014	-	0.018	0.011	0.008	**0.002**
	FID	46.78	42.81	40.02	49.71	44.86	35.23	29.44	**19.48**
CelebA	MSE	0.021	0.021	0.018	-	0.020	0.015	0.013	**0.010**
	FID	65.75	49.70	40.95	51.50	61.03	47.95	37.87	**29.75**

The comparison results on different datasets. The MSE and FID (smaller is better) are used to test the quality of the reconstructed

Anomaly Detection

- Likelihood-based anomaly detection is another task that can help evaluate the proposed model.
- With a well-learned model from the normal data, we can detect the anomalous data by firstly sampling the latent code of the given testing image from the posterior distribution by the short-run Langevin dynamics, and then computing the logarithm of the joint probability.
- The joint probability $p(z, \mathbf{I})$ should be high for the normal images and low for the anomalous ones.

Anomaly Detection

We treat each class in the MNIST dataset as an anomalous class and leave the others as normal. We train the model only with the normal data.

Then the model is tested with both the normal and anomalous data. To evaluate the performance, we use $\log p_\theta(\mathbf{I}, z)$ as our decision function to compute the area under the precision-recall curve (AUPRC).

Heldout Digit	1	4	5	7	9
VAE	0.063	0.337	0.325	0.148	0.104
MEG	0.281± 0.035	0.401± 0.061	0.402± 0.062	0.290± 0.040	0.342± 0.034
Bigan-σ	0.287± 0.023	0.443± 0.029	0.514± 0.029	0.347± 0.017	0.307± 0.028
LEBM	0.336± 0.008	0.630± 0.017	0.619± 0.013	0.463± 0.009	0.413± 0.010
ABP	0.095± 0.028	0.138± 0.037	0.147±0.026	0.138± 0.021	0.102±0.033
Ours	**0.353±0.021**	**0.770± 0.024**	**0.726±0.030**	**0.550±0.013**	**0.555±0.023**

AUPRC scores (larger is better) for unsupervised anomaly detection on the MNIST dataset.

Conclusion

In this paper, we propose to use the OT theory to correct the bias of the short-run MCMC-based inference in training the deep latent variable models. Specifically, we correct the marginal distribution of the latent variables of the short-run Langevin dynamics through the OT map between this distribution and the prior distribution step by step.

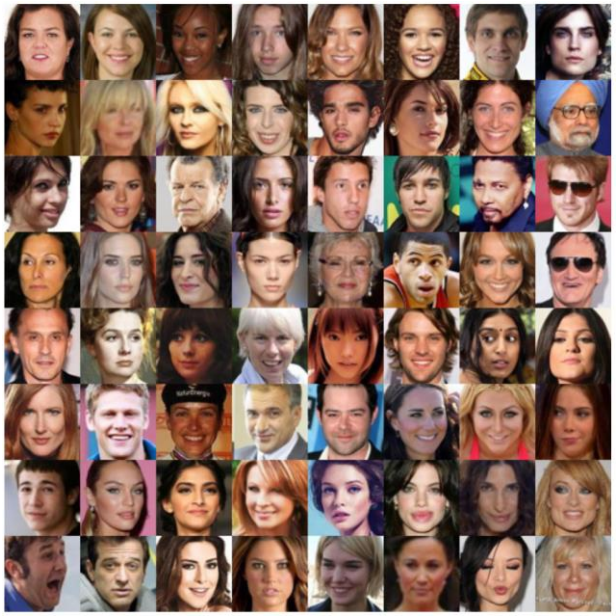
In such a way, the distribution of the inferred latent vectors will finally converge to the prior distribution, thus improving the accuracy of the subsequent parameter learning.

Experimental results show that the proposed training method performs better than the ABP (short-run only) and VAE models on the tasks like image reconstruction, image generation and anomaly detection.

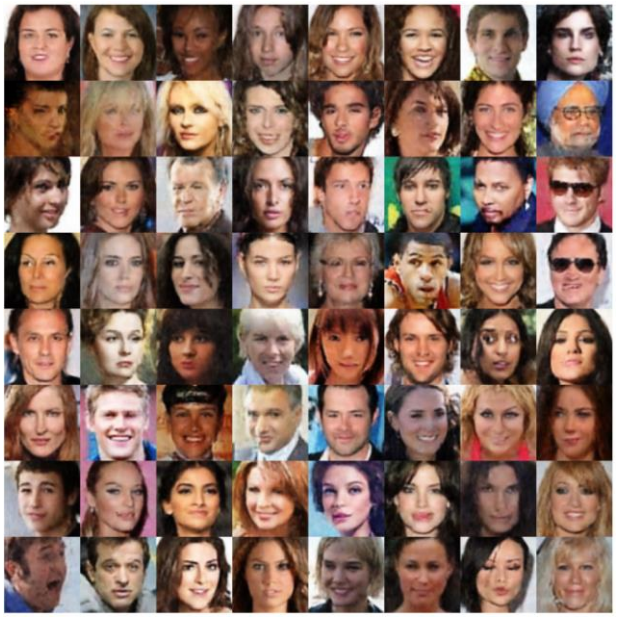
New results: Learning from incomplete data

50% occlusion

Incomplete training data



Original data



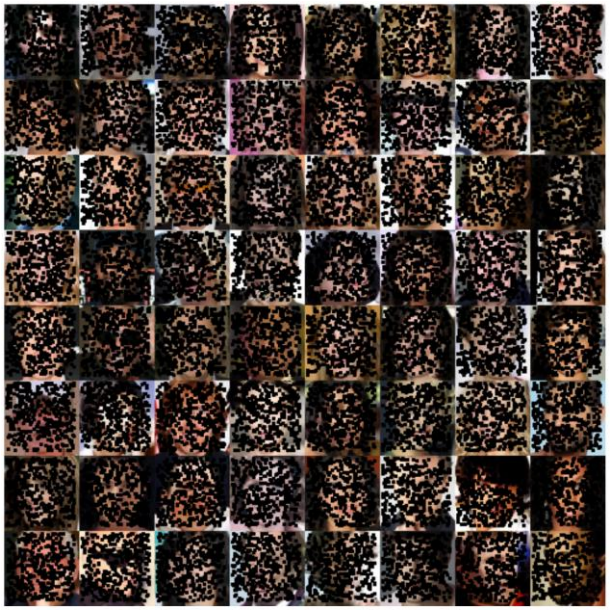
Reconstructed images

Experiment setting: randomly masked each training image, the locations of missing pixels are known (unsupervised inpainting).

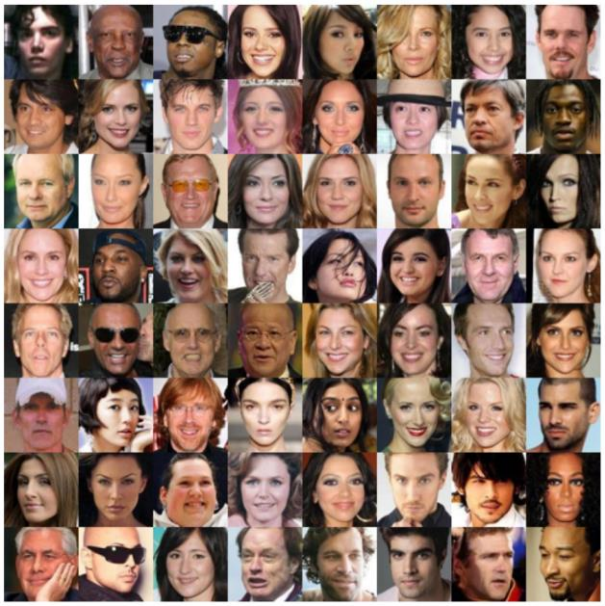
Solution: learn by maximizing the likelihood of the visible pixels.

New results: Learning from incomplete data

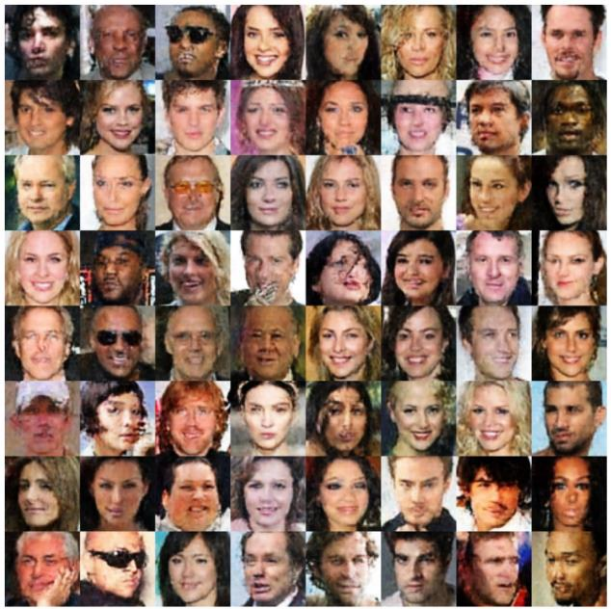
90% occlusion



Incomplete training data



Original data



Reconstructed images

Thank you !!