# A Tale of Two Latent Flows: Learning Latent Space Normalizing Flow with Short-run Langevin Flow for Approximate Inference

Jianwen Xie, Yaxuan Zhu, Yifei Xu, Dingcheng Li, Ping Li

Cognitive Computing Lab, Baidu Research, Bellevue, USA

## Introduction

1. We propose a simple but novel deep generative model where a latent space normalizing flow model, which serves as a prior, stands on a single top-down generator network.
2. We propose a principled maximum likelihood learning algorithm to jointly train the normalizing flow prior and the top-down network with short-run Langevin flow as an approximate inference.
3. We provide theoretical understanding of the proposed learning framework.
4. We provide extensive and strong experimental results on different aspects, including image synthesis, inference, reconstruction, inpainting and recovery, to validate the effectiveness of the proposed models and learning algorithms.

Table 1: A comparison of latent variable models with different priors and inference process. (● is good, ◐ is OK, and ○ is bad.)

| Methods | easy design | informative prior | fast sample | fast inference | less parameters | fast training | practical |
|---|---|---|---|---|---|---|---|
| NF + short-run MCMC | ● | ● | ● | ◐ | ● | ◐ | ● |
| NF + long-run MCMC | ● | ● | ● | ○ | ● | ○ | ○ |
| NF + inference net | ○ | ● | ● | ● | ○ | ● | ● |
| EBM + short-run MCMC | ● | ● | ○ | ◐ | ● | ◐ | ● |
| EBM + long-run MCMC | ● | ● | ○ | ○ | ● | ○ | ○ |
| EBM + inference net | ○ | ● | ○ | ● | ○ | ● | ● |
| Gaussian + short-run MCMC | ● | ○ | ● | ◐ | ● | ◐ | ● |
| Gaussian + long-run MCMC | ● | ○ | ● | ○ | ● | ○ | ○ |
| Gaussian + inference net | ○ | ○ | ● | ● | ○ | ● | ● |

## Latent Space Normalizing Flow Model

**(1) Latent variable model**:
Let $x \in \mathbb{R}^D$ denote an observed signal, $z \in \mathbb{R}^d$ denote the latent variables of $x$. The joint distribution of $(x, z)$ is given by

$$p_\theta(x, z) = p_\alpha(z)p_\beta(x|z), \tag{1}$$

where $p_\alpha(z)$ is the prior model parameterized by $\alpha$, and $p_\beta(x|z)$, is the top-down generation model parameterized by $\beta$. For notational convenience, let $\theta = (\alpha, \beta)$.

**(2) Top-down generation model** $p_\beta(x|z)$
The top-down generaton model is a non-linear transformation of the latent variables $z$ to generate the signal $x$, in which the transformation is parameterized by a neural network $g_\beta : \mathbb{R}^d \to \mathbb{R}^D$,

$$x = g_\beta(z) + \epsilon \tag{2}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2 I_D)$ is an observation residual. Thus,

$$p_\beta(x|z) = \mathcal{N}(g_\beta(z), \sigma^2 I_D), \tag{3}$$

where the standard deviation $\sigma$ is a hyper-parameter and assumed to be given.

**(3) Flow-based prior model** $p_\alpha(z)$
We formulate the prior $p_\alpha(z)$ as a flow-based model which is of the form

$$z_0 \sim q_0(z_0),\ z = f_\alpha(z_0), \tag{4}$$

where $q_0(z_0) = \mathcal{N}(0, I_d)$. $f_\alpha : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible or bijective function, which is a composition of a sequence of invertible transformations, i.e., $f_\alpha(z_0) = f_{\alpha_L} \circ \cdots \circ f_{\alpha_2} \circ f_{\alpha_1}(z_0)$, whose inverse and logarithm of the determinants of the Jacobians can be explicitly obtained in closed form.
According to the change-of-variable law of probabilities, $q_0(z_0)dz_0 = p_\alpha(z)dz$, the density of the flow-based prior model can be written as

$$p_\alpha(z) \quad = q_0(z_0)\frac{dz_0}{dz} = q_0(f_\alpha^{-1}(z))\left|\det\left(\frac{\partial f_\alpha^{-1}(z)}{\partial z}\right)\right| = q_0(f_\alpha^{-1}(z))\prod_{l=1}^{L}\left|\det\left(\frac{\partial z_{l-1}}{\partial z_l}\right)\right|, \tag{5}$$

where $f_\alpha^{-1}(z) = f_{\alpha_1}^{-1} \circ \cdots \circ f_{\alpha_{L-1}}^{-1} \circ f_{\alpha_L}^{-1}(z)$, and the determinant of the Jacobian matrix $(\partial z_{l-1}/\partial z_l)$ can be easy to compute with well-designed transformation functions in the flow-based models.

## Learning Latant Space Normalizing Flow with Short-run Langevin Flow

**(1) Maximum likelihood learning**
For the training examples $\{x_i, i = 1, ..., N\}$, the log-likelihood function is given by

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\log p_\theta(x_i), \tag{6}$$

where the marginal distribution is $p_\theta(x) = \int p_\theta(x, z)dz = \int p_\alpha(z)p_\beta(x|z)dz$.
The gradient of $\mathcal{L}(\theta)$ can be computed according to

$$\nabla_\theta \log p_\theta(x) = \mathbb{E}_{p_\theta(z|x)}\left[\nabla_\theta \log p_\theta(x, z)\right] = \mathbb{E}_{p_\theta(z|x)}[\nabla_\theta(\log p_\alpha(z) + \log p_\beta(x|z))],$$

where the posterior distribution of $z$ is given by $p_\theta(z|x) = p_\theta(x, z)/p_\theta(x) \propto p_\alpha(z)p_\beta(x|z)$. $p_\theta(z|x)$ is dependent on both the prior model $\alpha$ and the generation model $\beta$.

**(2) Learning** $\alpha$
The learning gradient of $\alpha$ for a datapoint $x$ is

$$\nabla_\alpha \log p_\theta(x) = \mathbb{E}_{p_\theta(z|x)}[\nabla_\alpha \log p_\alpha(z)] = \mathbb{E}_{p_\theta(z|x)}[\nabla_\alpha l_\alpha(z)]. \tag{7}$$

**(3) Learning** $\beta$
The learning gradient of $\beta$ for a datapoint $x$ is

$$\nabla_\beta \log p_\theta(x) = \mathbb{E}_{p_\theta(z|x)}[\nabla_\beta \log p_\beta(x|z)]. \tag{8}$$

Since $p_\beta(x|z)$ is in the form of a Gaussian distribution with a mean of $g_\beta(z)$ and a standard deviation of $\sigma$, $\nabla_\beta \log p_\beta(x|z) = \nabla_\beta(-\frac{1}{2\sigma^2}||x - g_\beta(z)||^2 + \text{const}) = \frac{1}{\sigma^2}(x - g_\beta(z))\nabla_\beta g_\beta(z)$.

**(4) Inference**
Sampling from $p_\theta(z|x)$ can be achieved by Langevin dynamics that iterates

$$z_{(k+1)} = z_{(k)} + \xi\nabla_z \log p_\theta(z_{(k)}|x) + \sqrt{2\xi}\epsilon_{(k)};\quad z_{(0)} \sim q_0(z), \epsilon_{(k)} \sim \mathcal{N}(0, I_d), \tag{9}$$

where $\nabla_z \log p_\theta(z|x) = \nabla_z \log p_\alpha(z) + \nabla_z \log p_\beta(x|z) = \nabla_z l_\alpha(z) + \frac{1}{\sigma^2}(x - g_\beta(z))\nabla_z g_\beta(z)$, and $\xi$ is the Langevin step size.

## Experiment 1: Image generation

**Synthesis**: The model can generate examples by first sampling latent vectors from the learned flow-based prior distribution and then transforming the vectors to image space.

**Reconstruction**: The models can reconstruct images by first inferring the latent vectors from the images, and then mapping the inferred latent vectors back to data space. The inference of latent variables can be achieved by the MCMC.

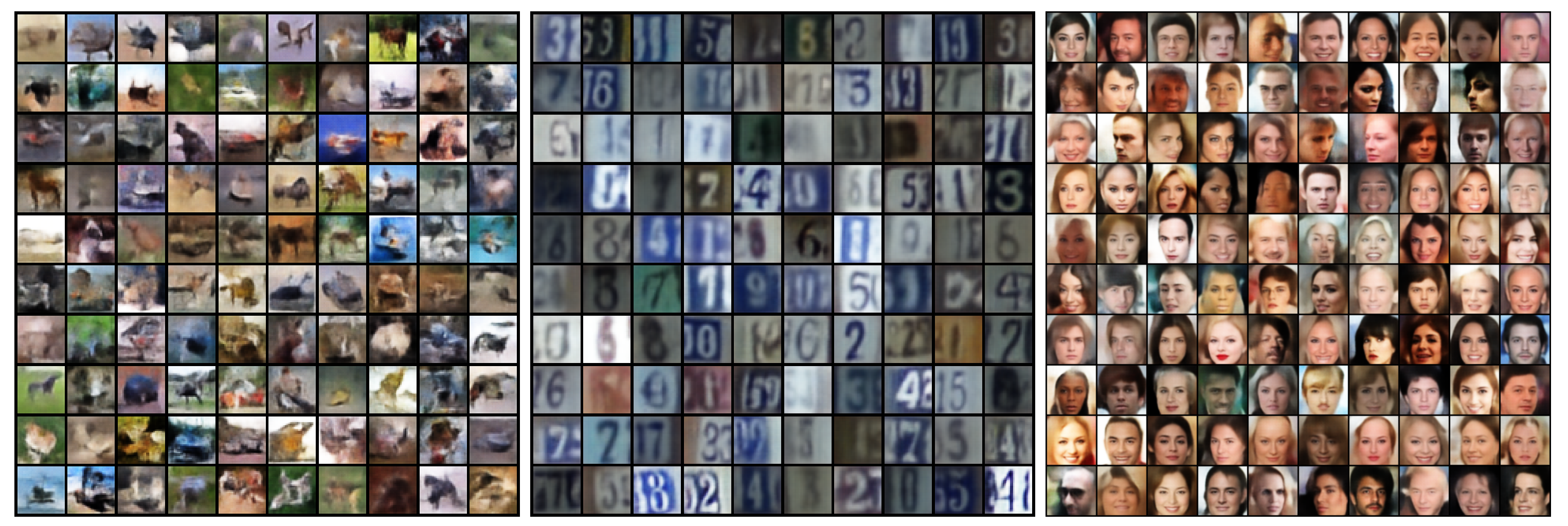


Figure 1: Generated samples from CIFAR10, SVHN, and CelebA datasets.

Table 2: Quantitative results of image reconstruction and generation.

| Models | | VAE | 2sVAE | RAE | SRI | SRI (L=5) | ABP | LEBM | LFBM VAE | LFBM MCMC |
|---|---|---|---|---|---|---|---|---|---|---|
| SVHN | MSE | 0.019 | 0.019 | 0.014 | 0.018 | 0.011 | - | 0.008 | 0.005 | **0.005** |
| | FID | 46.78 | 42.81 | 40.02 | 44.86 | 35.23 | 49.71 | 29.44 | 24.96 | **23.64** |
| Cifar10 | MSE | 0.057 | 0.056 | 0.027 | - | - | 0.018 | 0.020 | 0.020 | **0.016** |
| | FID | 106.37 | 72.90 | 74.16 | - | - | 90.30 | 70.15 | 69.70 | **66.41** |
| CelebA | MSE | 0.021 | 0.021 | 0.018 | 0.020 | 0.015 | - | 0.013 | 0.014 | **0.011** |
| | FID | 65.75 | 44.40 | 40.95 | 61.03 | 47.95 | 51.50 | 37.87 | 33.64 | **33.64** |

## Experiment 2: Supervised Image Inpainting

We can train an LFBM from fully-observed training images, and then use the learned model to complete the missing pixels of testing images.

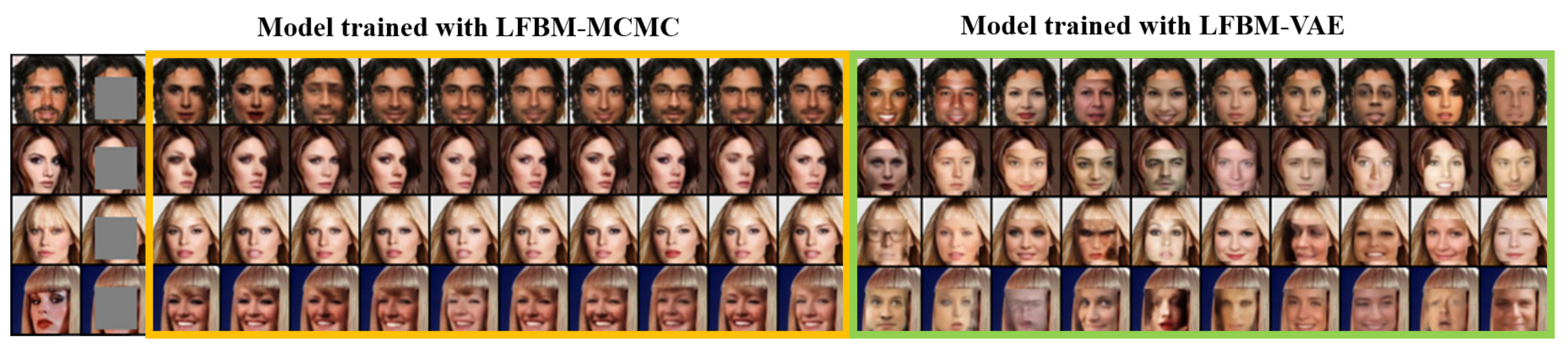


Figure 2: Supervised image inpainting results on the CelebA dataset.

## Experiment 3: Anomaly Detection

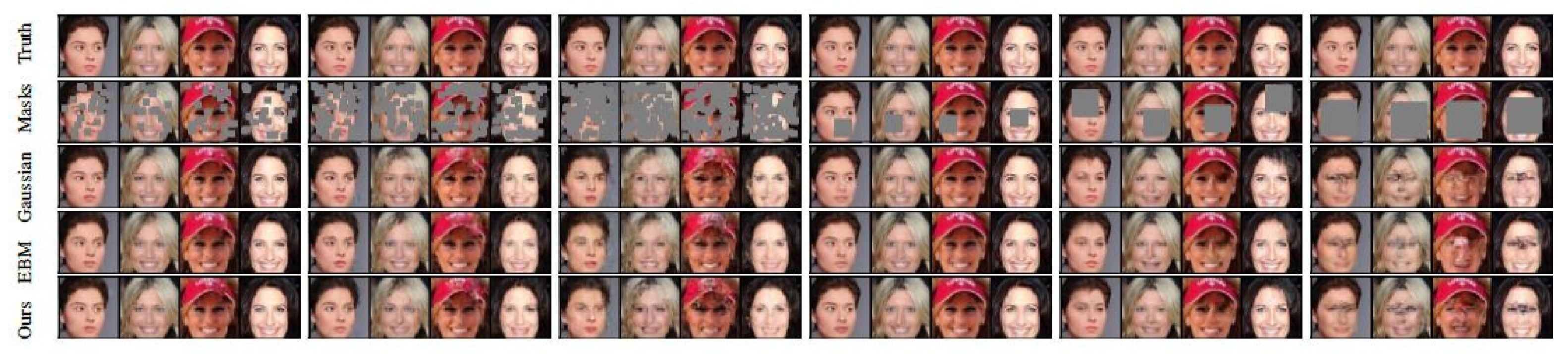
We can perform anomaly detection on a testing image $x$ by firstly inferring its latent variables $z$ and then computing the logarithm of the joint probability $\log p_\theta(x, z) = \log p_\alpha(z) + \log p_\beta(x|z) = l_\alpha(z) - \frac{1}{2\sigma^2}||x - g_\beta(z)||^2 - \log\sigma\sqrt{2\pi}$ as a decision score. The score should be high for a normal example and low for an anomalous one.

Table 3: AUPRC scores for unsupervised anomaly detection on MNIST dataset

| Heldout Digit | 1 | 4 | 5 | 7 | 9 |
|---|---|---|---|---|---|
| VAE | 0.063 | 0.337 | 0.325 | 0.148 | 0.104 |
| MEG | $0.281 \pm 0.035$ | $0.401 \pm 0.061$ | $0.402 \pm 0.062$ | $0.290 \pm 0.040$ | $0.342 \pm 0.034$ |
| BiGAN-$\sigma$ | $0.287 \pm 0.023$ | $0.443 \pm 0.029$ | $0.514 \pm 0.029$ | $0.347 \pm 0.017$ | $0.307 \pm 0.028$ |
| EBM-VAE | $0.297 \pm 0.033$ | $0.723 \pm 0.042$ | $0.676 \pm 0.041$ | $0.490 \pm 0.041$ | $0.383 \pm 0.025$ |
| LEBM | $0.336 \pm 0.008$ | $0.630 \pm 0.017$ | $0.619 \pm 0.013$ | $0.463 \pm 0.009$ | $0.413 \pm 0.010$ |
| ABP | $0.095 \pm 0.028$ | $0.138 \pm 0.037$ | $0.147 \pm 0.026$ | $0.138 \pm 0.021$ | $0.102 \pm 0.033$ |
| LFBM (ours) | $\mathbf{0.349 \pm 0.002}$ | $\mathbf{0.812 \pm 0.007}$ | $\mathbf{0.823 \pm 0.009}$ | $\mathbf{0.682 \pm 0.004}$ | $\mathbf{0.514 \pm 0.008}$ |

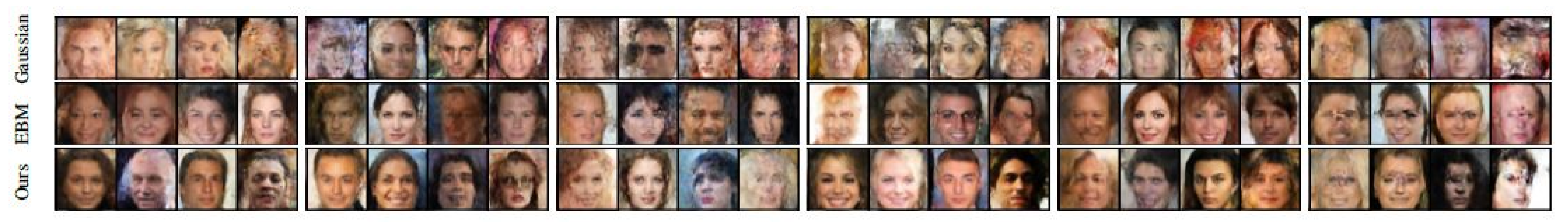
## Experiment 4: Unsupervised Image Recovery

The LFBM can be learned from incomplete training data, e.g., images with occluded pixels. The learning algorithm updates the model parameters by maximizing the likelihood of the visible pixels in training images.



(a) salt-pepper 30% (b) salt-pepper 50% (c) salt-pepper 70% (d) mask size $20\times20$ (e) mask size $30\times30$ (f) mask size $40\times40$

Figure 3: A comparison of unsupervised image recovery results by different methods on training images with different levels of occlusions.



(a) salt-pepper 30% (b) salt-pepper 50% (c) salt-pepper 70% (d) mask size $20\times20$ (e) mask size $30\times30$ (f) mask size $40\times40$

Figure 4: Image synthesis by models learned from incomplete images.

Table 4: MSEs with different priors

| Salt and pepper mask | | | |
|---|---|---|---|
| Occ % | 30% | 50% | 70% |
| flow (ours) | **0.0244** | **0.0317** | **0.0464** |
| EBM | 0.0256 | 0.0319 | 0.0465 |
| Gaussian | 0.0259 | 0.0326 | 0.0472 |
| Single region mask | | | |
| mask size | $20 \times 20$ | $30 \times 30$ | $40 \times 40$ |
| flow (ours) | 0.0420 | 0.0587 | **0.0864** |
| EBM | 0.0429 | 0.0684 | 0.0957 |
| Gaussian | **0.0404** | **0.0572** | 0.0918 |

Table 5: FIDs with different priors

| Salt and pepper mask | | | |
|---|---|---|---|
| Occ % | 30% | 50% | 70% |
| flow (ours) | **46.2** | **59.14** | **86.77** |
| EBM | 52.78 | 61.91 | 88.27 |
| Gaussian | 153.01 | 156.71 | 172.77 |
| Single region mask | | | |
| mask size | $20 \times 20$ | $30 \times 30$ | $40 \times 40$ |
| flow (ours) | **42.39** | **47.52** | **72.47** |
| EBM | 49.16 | 51.59 | 77.39 |
| Gaussian | 150.95 | 146.41 | 184.53 |