Training Energy-Based Normalizing Flow with Score-Matching Objectives

Chen-Hao Chao[1], Wei-Fang Sun[1] [2], Yen-Chang Hsu[3], Zsolt Kira[4], and Chun-Yi Lee[1]

[1] Elsa Lab, National Tsing Hua University, Hsinchu City, Taiwan *[2] NVIDIA AI Technology Center, NVIDIA Corporation, Santa Clara, CA, USA*
[3] Samsung Research America , Mountain View, CA, USA *[4] Georgia Institute of Technology , Atlanta, GA, USA*

Abstract

In this work, we establish a connection between the parameterization of flow-based and energy-based generative models, and present a new flow-based modeling approach called **energy-based normalizing flow (EBFlow)**. We demonstrate that by optimizing EBFlow with score-matching objectives, the computation of Jacobian determinants for linear transformations can be entirely bypassed. This feature enables the use of **arbitrary linear layers** in the construction of flow-based models without increasing the computational time complexity of each training iteration from $\mathcal{O}(D^2L)$ **to** $\mathcal{O}(D^3L)$ for an L-layered model that accepts D-dimensional inputs. The experimental results demonstrate that our approach achieves a significant speedup compared to maximum likelihood training.

Background

Flow-Based Models

Flow-based models parameterize probability density functions (pdf) $p(\cdot\,;\theta)$ using a prior distribution $p_{\mathbf{u}}$ of a variable $\mathbf{u}$ and an invertible mapping $g = g_L \circ \cdots \circ g_1$, where $g_i(\cdot\,;\theta)\colon \mathbb{R}^D \to \mathbb{R}^D, i \in \{1, \dots, L\}$. Let $\boldsymbol{x}_0 = \boldsymbol{x}$ be an input vector, and $\boldsymbol{x}_i = g_i \circ \cdots \circ g_1(\boldsymbol{x}_0)$ be a transformed vector. Based on the change of variable theorem, the pdf $p(\cdot\,;\theta)$ can be expressed as:

$$p(\boldsymbol{x};\theta) = p_{\mathbf{u}}(g(\boldsymbol{x};\theta)) \prod_{i=1}^{L} |\det(\boldsymbol{\mathcal{J}}_{g_i}(\boldsymbol{x}_{i-1};\theta))|, \quad (1)$$

where and $\boldsymbol{\mathcal{J}}_{g_i}$ represents the Jacobian of g_i. A conventional approach to optimize θ is maximum likelihood (**ML**) training, which involves minimizing the **Kullback-Leibler (KL) divergence** $\mathbb{D}_{KL}[p_{\mathrm{x}}(\boldsymbol{x})||p(\boldsymbol{x};\theta)]$ between the true pdf p_{x} and $p(\boldsymbol{x};\theta)$. The ML loss is written as:

$$\mathcal{L}_{ML}(\theta) = \mathbb{E}_{p_{\mathrm{x}}(\boldsymbol{x})}[-\log p(\boldsymbol{x};\theta)]. \quad (2)$$

Energy-Based Models

Energy-based models are formulated based on a Boltzmann distribution, which is expressed using a scalar-valued energy function $E(\cdot\,;\theta)\colon \mathbb{R}^D \to \mathbb{R}$ and a normalizing constant $Z(\theta) = \int \exp(-E(\boldsymbol{x};\theta))\, d\boldsymbol{x}$ as the following equation:

$$p(\boldsymbol{x};\theta) = \exp(-E(\boldsymbol{x};\theta))\, Z^{-1}(\theta). \quad (3)$$

Optimizing θ in Eq. (3) through directly evaluating $\mathcal{L}_{ML}(\theta)$ in Eq. (2) is computationally infeasible due to the integral in $Z(\theta)$. To address this, a widely-used technique is to reformulate $\nabla_\theta \mathcal{L}_{ML}(\theta)$ as its sampling-based variant $\nabla_\theta \mathcal{L}_{SML}(\theta)$, which is written as follows:

$$\mathcal{L}_{SML}(\theta) = \mathbb{E}_{p_{\mathrm{x}}(\boldsymbol{x})}[E(\boldsymbol{x};\theta)] - \mathbb{E}_{sg(p(\boldsymbol{x};\theta))}[E(\boldsymbol{x};\theta)], \quad (4)$$

where $sg(\cdot)$ indicates the stop-gradient operator.

Another line of studies suggests optimizing θ by minimizing the **Fisher divergence** $\mathbb{D}_F[p_{\mathrm{x}}(\boldsymbol{x})||p(\boldsymbol{x};\theta)]$ through **score matching** (SM). Several SM techniques, including sliced score matching (**SSM**) [1], finite difference sliced score matching (**FDSSM**) [2], and denoising score matching (**DSM**) [3], have been proposed. These losses are written as:

$$\mathcal{L}_{SSM}(\theta) = \mathbb{E}_{p_{\mathrm{x}}(\boldsymbol{x})p_{\mathbf{v}}(\boldsymbol{v})}[\|\nabla_{\boldsymbol{x}} E(\boldsymbol{x};\theta)\|^2 - \boldsymbol{v}^T \nabla_{\boldsymbol{x}} E(\boldsymbol{x};\theta)\boldsymbol{v}], \quad (5)$$

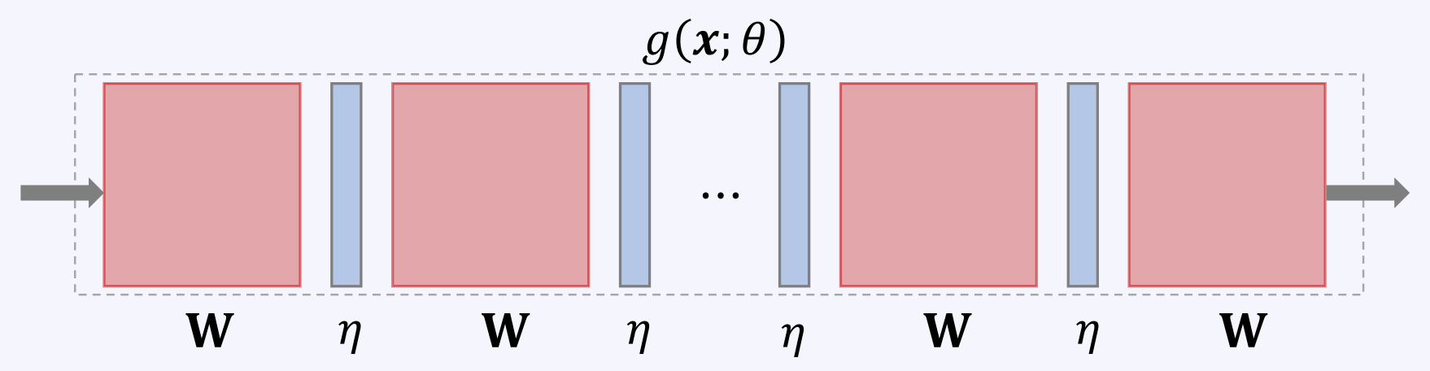
$$\mathcal{L}_{FDSSM}(\theta) = \mathbb{E}_{p_{\mathrm{x}}(\boldsymbol{x})p_{\xi}(\boldsymbol{\varepsilon})}[2E(\boldsymbol{x};\theta) - E(\boldsymbol{x}+\boldsymbol{\varepsilon};\theta) - E(\boldsymbol{x}-\boldsymbol{\varepsilon};\theta)] + \mathbb{E}_{p_{\mathrm{x}}(\boldsymbol{x})p_{\xi}(\boldsymbol{\varepsilon})}\left[\left(E(\boldsymbol{x}+\boldsymbol{\varepsilon};\theta) - E(\boldsymbol{x}-\boldsymbol{\varepsilon};\theta)\right)^2/8\right], \quad (6)$$

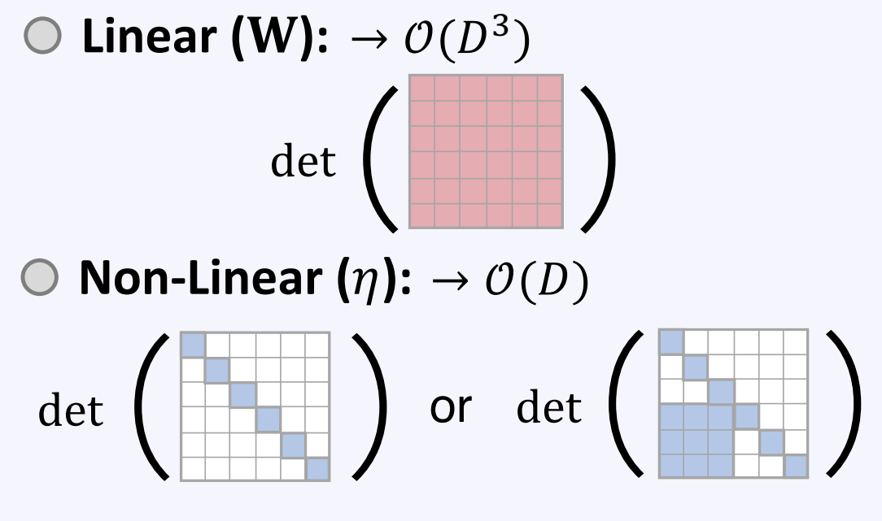
$$\mathcal{L}_{DSM}(\theta) = \mathbb{E}_{p_{\mathrm{x}}(\boldsymbol{x})p_{\sigma}(\tilde{\boldsymbol{x}}|\boldsymbol{x})}[\nabla_{\boldsymbol{x}} E(\boldsymbol{x};\theta) + (\boldsymbol{x}-\tilde{\boldsymbol{x}})/\sigma^2], \quad (7)$$

where $p_{\mathbf{v}}$ is a Rademacher distribution, p_σ is a Gaussian with standard deviation σ, and p_ξ is a uniform distribution with $\|\boldsymbol{\varepsilon}\| = \xi$.

Related Works

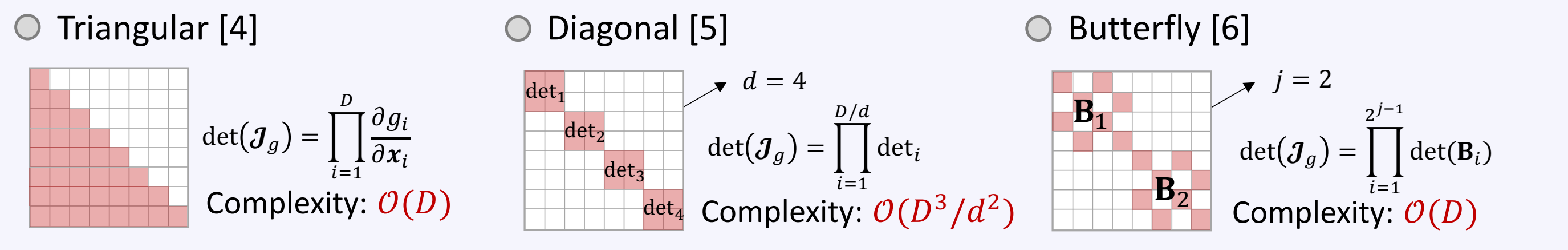
Accelerating Maximum Likelihood Training of Flow-Based Models





- **Specially Designed Linear Transformations**

Examples [4-6]:



- (✓) Complexity can be reduced significantly. • (✗) Impose architectural constraints on the model.

- **Specially Designed Optimization Methods**

Example [7]:

$$\frac{\Delta f(\mathbf{W})}{\Delta \mathbf{W}} \begin{matrix} \longrightarrow f((\mathbf{I}+\boldsymbol{\epsilon})\mathbf{W}) - f(\mathbf{W}) = \langle \underset{\text{Relative Gradient}}{\boxed{\nabla_{\mathbf{W}} f(\mathbf{W})\mathbf{W}^{\mathrm{T}}}}, \boldsymbol{\epsilon}\rangle + \underset{\text{Error}}{\boxed{o(\mathbf{W})}} \\ \longrightarrow (\mathbf{I}+\boldsymbol{\epsilon})\mathbf{W} - \mathbf{W} \end{matrix}$$

$\because \nabla_{\mathbf{W}} \log|\det \mathbf{W}|\mathbf{W}^{\mathrm{T}} = (\mathbf{W}^{\mathrm{T}})^{-1}\mathbf{W}^{\mathrm{T}} = \mathbf{I}.$

∴ The determinant calculation is bypassed.

- (✓) Complexity of each update is $\mathcal{O}(D^2L)$. • (✗) An error term proportional to the weight matrix $\mathbf{W}$.

Methodology

Energy-Based Normalizing Flow (EBFlow)

Let $\mathcal{S}_n$ and $\mathcal{S}_l$ be the sets of non-linear and linear transformations in $g(\cdot\,;\theta)$. Our key observation is that the parametric density function $p(\cdot\,;\theta)$ can be explicitly factorized into an **unnormalized density** and a corresponding **normalizing constant** as follows:

$$\begin{aligned} p(\boldsymbol{x};\theta) &= p_{\mathbf{u}}(g(\boldsymbol{x};\theta)) \prod_{i=1}^{L} |\det(\boldsymbol{\mathcal{J}}_{g_i}(\boldsymbol{x}_{i-1};\theta))| \\ &= p_{\mathbf{u}}(g(\boldsymbol{x};\theta)) \prod_{g_i \in \mathcal{S}_n} |\det(\boldsymbol{\mathcal{J}}_{g_i}(\boldsymbol{x}_{i-1};\theta))| \prod_{g_i \in \mathcal{S}_l} |\det(\boldsymbol{\mathcal{J}}_{g_i}(\theta))| \quad (8) \\ &\triangleq \underbrace{\exp(-E(\boldsymbol{x};\theta))}_{\text{Unnorm. density}} \underbrace{Z^{-1}(\theta)}_{\text{Norm. Const.}}, \end{aligned}$$

where the energy function $E(\cdot\,;\theta)$ and the normalizing constant $Z(\theta)$ are selected as follows:

$$E(\boldsymbol{x};\theta) = -\log p_{\mathbf{u}}(g(\boldsymbol{x};\theta)) \prod_{g_i \in \mathcal{S}_n} |\det(\boldsymbol{\mathcal{J}}_{g_i}(\boldsymbol{x}_{i-1};\theta))|, Z^{-1}(\theta) = \prod_{g_i \in \mathcal{S}_l} |\det(\boldsymbol{\mathcal{J}}_{g_i}(\theta))|. \quad (9)$$

By isolating the computationally expensive terms in $p(\cdot\,;\theta)$, the parametric pdf becomes suitable for the training methods of energy-based models.

Techniques for Enhancing the Training of EBFlow

Training flow-based models with SM objectives is challenging as the training process is **numerically unstable** and usually exhibits **significant variances** [1,2]. To address these issues, we propose to adopt the following two techniques:

- **Match after Preprocessing (MaP):**

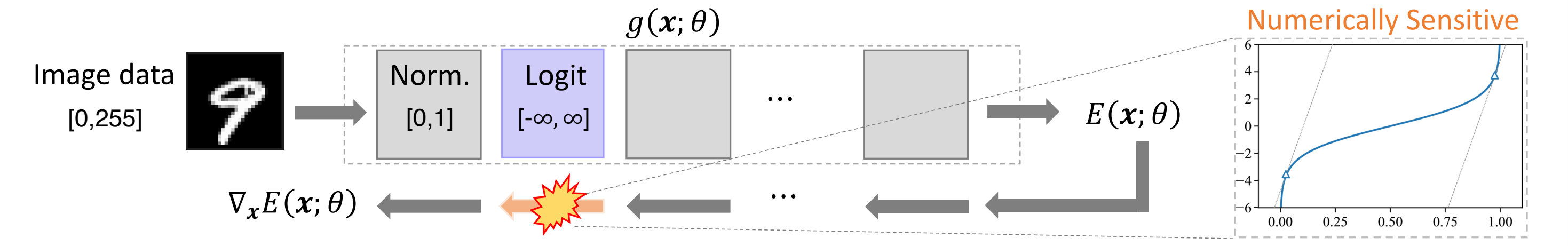


Figure 1. An illustrative example of training EBFlow with a numerically sensitive layer. In this example, the logit preprocessing layer exhibits extremely large derivatives when its input values are near zeros or ones.

Proposition. Let p_j be a pdf modeled as $p_{\mathbf{u}}(g_L \circ \cdots \circ g_j(\cdot))\Pi_{\mathrm{i=j+1}}^{\mathrm{L}}|\det(\boldsymbol{\mathcal{J}}_{g_i})|$, where $j \in \{0, \dots, L-1\}$. It follows that:

$$\mathbb{D}_F[p_{\mathrm{x_j}}||p_j] = 0 \Leftrightarrow \mathbb{D}_F[p_{\mathbf{x}}||p_0] = 0. \quad (10)$$

- **Exponential Moving Average (EMA)** [8]:

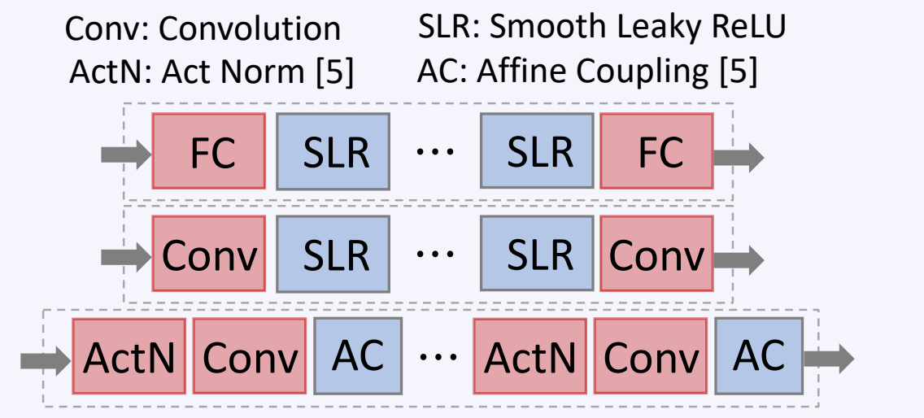
$$\tilde{\theta} \leftarrow m\tilde{\theta} + (1-m)\theta_i, \quad (11)$$

where $\tilde{\theta}$ is a set of shadow parameters, θ_i is the model's parameters at the i–th iteration, and m is the momentum parameter.

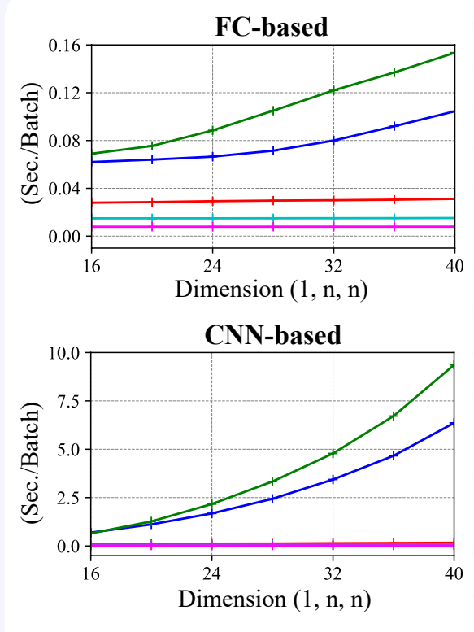
Experiments

Density Modeling

- **Model Architecture:**
  - Fully-Connected (FC) based:
  - Convolutional Neural Network (CNN) based:
  - Generative Flow (Glow) [5]:



- **Training Methods:**
  - Baseline (ML)
  - EBFlow (SML, SSM, DSM, FDSSM)



MNIST ($D = 784$)										
Model	FC-based					CNN-based				
Num. Param.	1.230 M					0.027 M				
Method	Baseline (ML)	EBFlow (SML)	EBFlow (SSM)	EBFlow (DSM)	EBFlow (FDSSM)	Baseline (ML)	EBFlow (SML)	EBFlow (SSM)	EBFlow (DSM)	EBFlow (FDSSM)
NLL (↓)	1092.4 ± 0.1	**1092.3 ± 0.6**	1092.8 ± 0.3	1099.2 ± 0.2	1104.1 ± 0.5	1101.3 ± 1.3	**1098.3 ± 6.6**	1107.5 ± 1.4	1109.5 ± 2.4	1122.1 ± 3.1
Bits/Dim (↓)	**2.01 ± 0.00**	**2.01 ± 0.00**	**2.01 ± 0.00**	2.02 ± 0.00	2.03 ± 0.00	2.03 ± 0.00	**2.02 ± 0.01**	2.03 ± 0.00	2.04 ± 0.00	2.06 ± 0.01
Batch/Sec. (↑)	8.00	12.27	33.11	66.67	**130.21**	0.21	0.29	7.09	18.32	**38.76**
CIFAR-10 ($D = 3,072$)										
Model	FC-based					CNN-based				
Num. Param.	18.881 M					0.241 M				
Method	Baseline (ML)	EBFlow (SML)	EBFlow (SSM)	EBFlow (DSM)	EBFlow (FDSSM)	Baseline (ML)	EBFlow (SML)	EBFlow (SSM)	EBFlow (DSM)	EBFlow (FDSSM)
NLL (↓)	**11912.9 ± 10.5**	11915.6 ± 5.6	11917.7 ± 15.5	11940.0 ± 6.6	12347.8 ± 6.8	**11408.7 ± 26.7**	11553.6± 151.7	11435.5 ± 12.0	11462.3 ± 7.9	11766.0 ± 36.8
Bits/Dim (↓)	**5.59 ± 0.00**	5.60 ± 0.00	5.60 ± 0.01	5.61 ± 0.00	5.80 ± 0.00	**5.36 ± 0.01**	5.41 ± 0.07	5.37 ± 0.00	5.38 ± 0.00	5.54 ± 0.02
Batch/Sec. (↑)	5.05	7.35	29.85	57.14	**62.50**	0.02	0.03	7.35	18.41	**39.84**

Figure 2 (Left). Runtime comparison of different objective functions used in EBFlow and the baseline method for different input sizes. **Table 1 (Right).** The performance (i.e., NLL and Bits/Dim) and throughput (i.e., Batch/Sec.) of the FC-based and CNN-based models trained with the baseline and the proposed method on MNIST and CIFAR-10. Each result is reported in terms of the mean and confidence interval of three independent runs.

FC-based				
EMA	MaP	EBFlow(SSM)	EBFlow(DSM)	EBFlow(FDSSM)
		1757.5 ± 28.0	4660.3 ± 19.8	3267.0 ± 99.2
✓		1720.5 ± 0.8	4455.0 ± 1.6	3166.3 ± 17.3
✓	✓	**1092.8 ± 0.3**	**1099.2 ± 0.2**	**1104.1 ± 0.5**
CNN-based				
EMA	MaP	EBFlow(SSM)	EBFlow(DSM)	EBFlow(FDSSM)
		3518.0 ± 33.9	3170.0 ± 7.2	3593.3 ± 12.5
✓		3504.5 ± 2.4	3180.0 ± 2.9	3560.3 ± 1.7
✓	✓	**1107.5 ± 1.4**	**1109.5 ± 2.6**	**1122.1 ± 3.1**

Table 2. The results in terms of Negative Log Likelihood (NLL) of the FC-based and CNN-based models trained using SSM, DSM, and FDSSM losses on the MNIST dataset.

Figure 3. The norm of $\nabla_\theta \mathcal{L}_{SSM}(\theta)$ of an FC-based model trained on the MNIST dataset. The curves and shaded area depict the mean and 95% confidence interval of three independent runs.

Data Generation

- **MCMC Generation**
  - Complexity: $\mathcal{O}(D^2LT)$
  - Sampler: $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \nabla_{\mathbf{x}_t} E(\mathbf{x}_t;\theta) + \sqrt{2\alpha}\mathbf{z}, t \in \{1, \dots, T\}$
    - t: the index of an iteration
    - α: the step size
    - T: the total number of iterations
    - $\mathbf{z}$: noises sampled from a Gaussian
- **Inverse Generation**
  - Complexity: $\mathcal{O}(D^3L)$
  - Sampler: $\mathbf{x} = g^{-1}(\mathbf{u};\theta)$, where $\mathbf{u} \sim p_{\mathbf{u}}$.

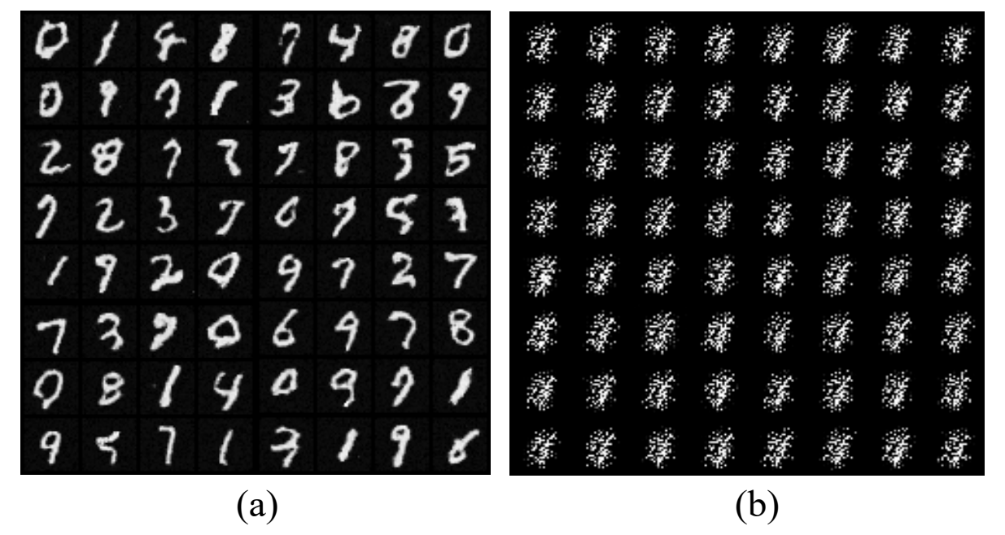

(a) (b)

Figure 4. A comparison between (a) Glow trained with our method (NLL=728) and (b) the model in [2] (NLL=1,637) on the inverse generation task.

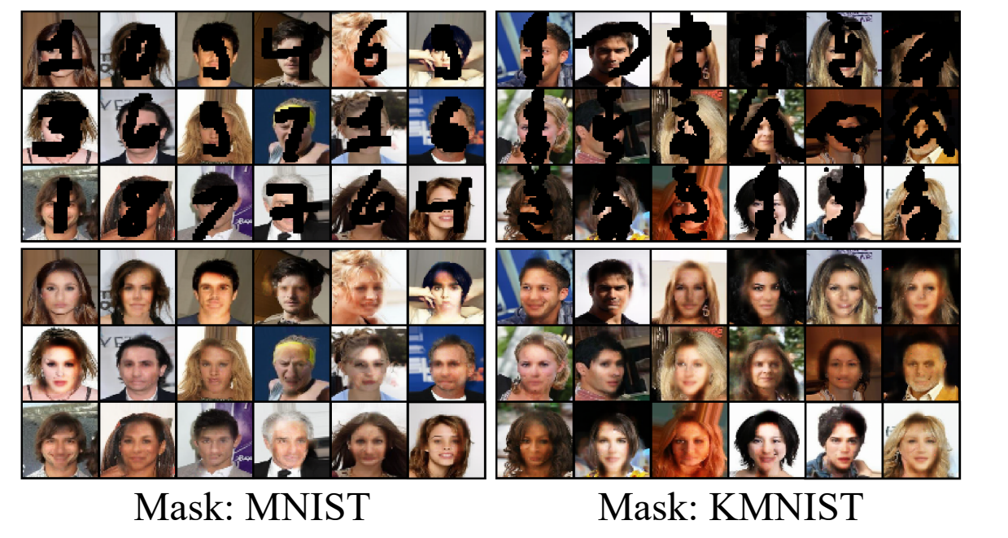

Mask: MNIST Mask: KMNIST

Figure 5. A qualitative demonstration of the FC-based models trained using the DSM objective on the imputation task.

References

[1] Song *et al*. A Scalable Approach to Density and Score Estimation, *UAI*, 2019.
[2] Pang *et al*. Efficient Learning of Generative Models via Finite-Difference Score Matching, *NeurIPS,* 2020.
[3] P. Vincent. A Connection between Score Matching and Denoising Autoencoders, *Neural Computation*, 2011.
[4] Song *et al*. MintNet: Building Invertible Neural Networks with Masked Convolutions, *NeurIPS*, 2019.
[5] Kingma *et al*. Glow: Generative Flow with Invertible 1x1 Convolutions, *NeurIPS*, 2018.
[6] Meng *et al*. ButterflyFlow: Building Invertible Layers with Butterfly Matrices*, ICML*, 2022.
[7] Gresele *et al*. Relative Gradient Optimization of the Jacobian Term in Unsupervised Deep Learning, *NeurIPS*, 2020.
[8] Song *et al*. Improved Techniques for Training Score-Based Generative Models, *NeurIPS*, 2020.

Acknowledgement

TWCC NVIDIA NSTC

Questions?

lance_chao@gap.nthu.edu.tw
cylee@cs.nthu.edu.tw

Github