Residual Flows for Invertible Generative Modeling

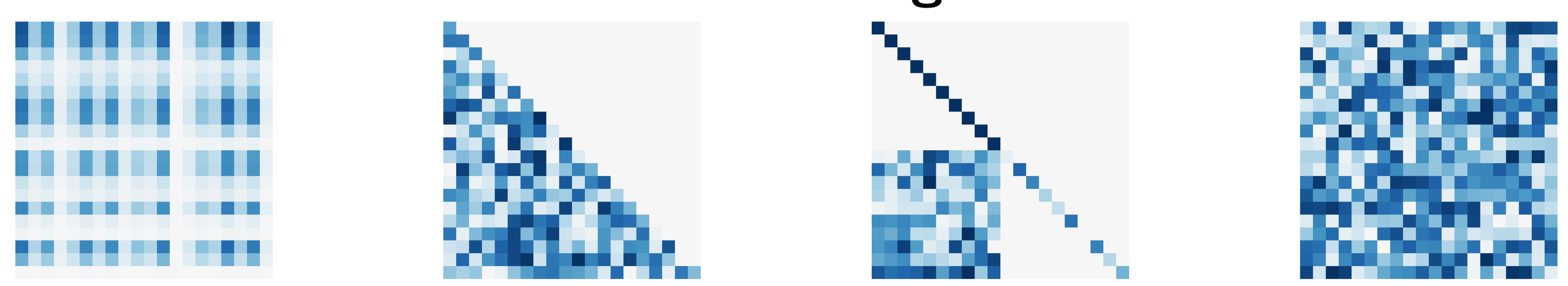
UNIVERSITY OF TORONTO — VECTOR INSTITUTE — Universität Bremen

Ricky T. Q. Chen[1,3], Jens Behrmann[2], David Duvenaud[1,3], Jörn-Henrik Jacobsen[1,3]

University of Toronto[1], University of Bremen[2], Vector Institute[3]

Residual Flows are ...

Highly scalable invertible generative model that allows **free-form Jacobian** and make use of **unbiased log-likelihood**.



(a) Det. Identities (Low Rank) (b) Autoregressive (Lower Triangular) (c) Coupling (Structured Sparsity) (d) **Unbiased Est.** (Free-form)

Background: Invertible Generative Models

Maximum likelihood estimation. To perform maximum likelihood with stochastic gradient descent, we require estimating

$$\nabla_\theta \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log p_\theta(x)] = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\nabla_\theta \log p_\theta(x)] \tag{1}$$

Change of Variables. With an invertible transformation f, we can build a generative model

$$z \sim p(z), \; x = f^{-1}(z). \tag{2}$$

Then the log-density of x is given by

$$\log p(x) = \log p(f(x)) + \log\left|\det \frac{df(x)}{dx}\right|. \tag{3}$$

Flow-based generative models can be

1. sampled, if (2) can be computed with arbitrary precision.
2. trained using maximum likelihood, if (3) can be unbiasedly estimated.

Background: Invertible Residual Networks

Residual networks are composed of simple transformations

$$y = f(x) = x + g(x) \tag{4}$$

Behrmann *et al.* (2019) proved if g has *Lipschitz strictly less than one*, then the residual transformation (4) is invertible.

Sampling. The inverse f^{-1} can be computed by a fixed-point iteration

$$x^{(i+1)} = y - g(x^{(i)}) \tag{5}$$

which converges *superlinearly* by the Banach fixed-point theorem.

Log-likelihood. The change of the variables can be applied.

$$\log p(x) = \log p(f(x)) + \operatorname{tr}\left(\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}[J_g(x)]^k\right) \tag{6}$$

The infinite series is intractable to exactly compute.
Fixed truncation creates a biased training objective.

Unbiased Log-likelihood via "Russian Roulette"

Russian roulette estimator. Used for estimating infinite series.

$$\sum_{k=1}^{\infty} \Delta_k = \mathbb{E}_{n\sim p(N)}\left[\sum_{k=1}^{n} \frac{\Delta_k}{\mathbb{P}(N\geq k)}\right] \tag{7}$$

Residual Flows. Unbiased estimation of the log-likelihood leads to our training objective. Easily trains with maximum likelihood.

$$\log p(x) = \log p(f(x)) + \mathbb{E}_{n,v}\left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} \frac{v^T[J_g(x)^k]v}{\mathbb{P}(N\geq k)}\right], \tag{8}$$

where $n \sim p(N)$ and $v \sim \mathcal{N}(0, I)$. We use a shifted geometric distribution for $p(N)$ with an expected compute of 4 terms.

Compared to fixed truncation, this

- Allows making use of big networks and high Lipschitz constants.
- Allows training with higher dimensions (from 32×32 to 256×256).

Memory-efficient Gradient Estimation

Neumann gradient series. For estimating (1), we can either

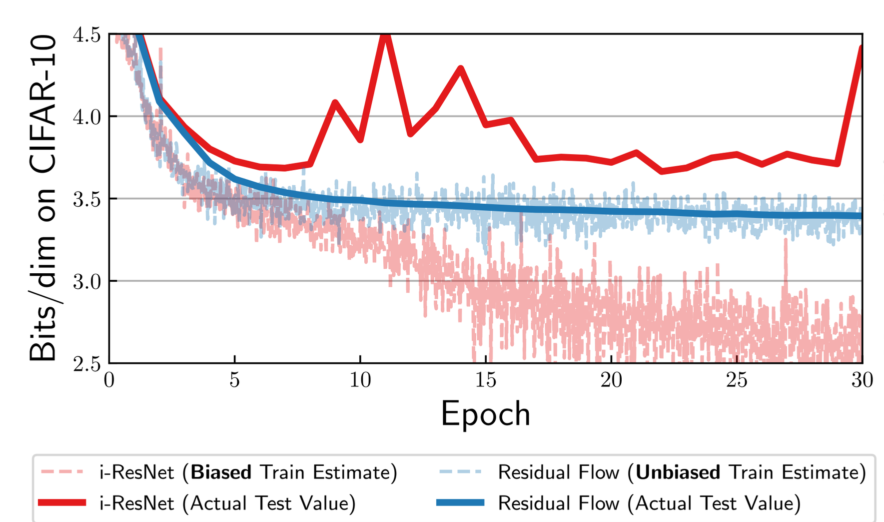
(i) Estimate $\log p(x)$, then take gradient.

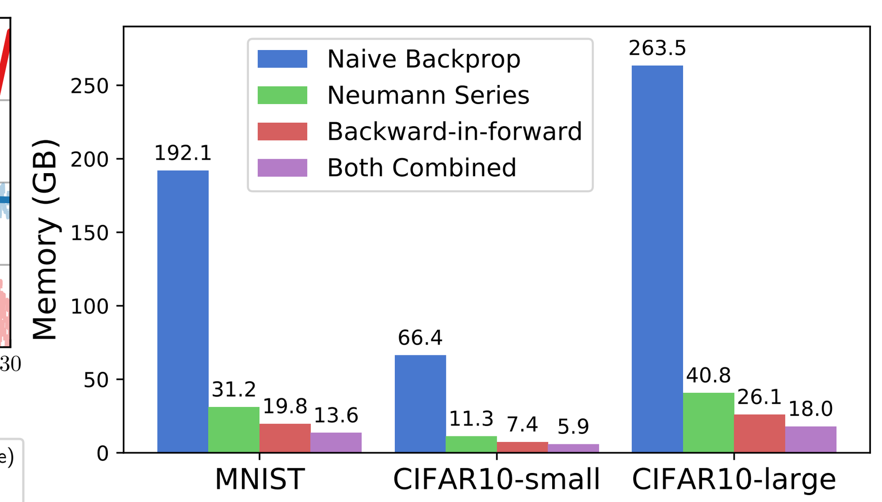
(ii) Analytically compute the gradient power series, then estimate it.

The first option uses variable amount of memory as n is stochastic. The second option, by using a Neumann series we obtain constant memory cost:

$$\frac{\partial}{\partial\theta}\log\left|\det\frac{df(x)}{dx}\right| = \mathbb{E}_{n,v}\left[\left(\sum_{k=0}^{n}\frac{(-1)^k}{\mathbb{P}(N\geq k)} v^T J(x,\theta)^k\right)\frac{\partial(J_g(x,\theta))}{\partial\theta}v\right]$$

Now **tractable to train with large networks**.





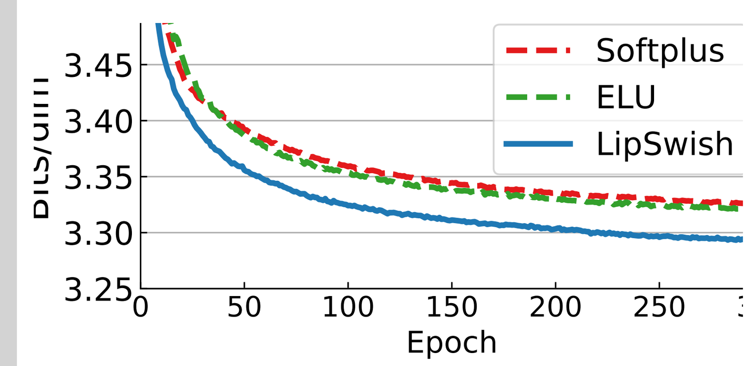
Density Modeling Benchmarks

Table: Results [bits/dim] on standard benchmark datasets for density estimation.

Model	MNIST	CIFAR-10	ImageNet 32	ImageNet 64	CelebA-HQ 256
Real NVP (Dinh *et al.*, 2017)	1.06	3.49	4.28	3.98	—
Glow (Kingma & Dhariwal, 2018)	1.05	3.35	4.09	3.81	1.03
FFJORD (Grathwohl *et al.*, 2019)	0.99	3.40	—	—	—
Flow++ (Ho *et al.*, 2019)	—	3.29 (3.09)	— (3.86)	— (3.69)	—
i-ResNet (Behrmann *et al.*, 2019)	1.05	3.45	—	—	—
Residual Flow (Ours)	**0.970**	**3.280**	**4.010**	**3.757**	**0.992**

We also show **residual blocks** > **coupling blocks** for joint classification and generative modeling, ie. hybrid modeling.

Ablation Experiments



Training Setting	MNIST	CIFAR-10	CIFAR-10†
i-ResNet + ELU	1.05	3.45	3.66∼4.78
Residual Flow + ELU	1.00	3.40	3.32
Residual Flow + LipSwish	**0.97**	**3.39**	**3.28**

Table: Ablation results. †Larger network.

Qualitative Samples

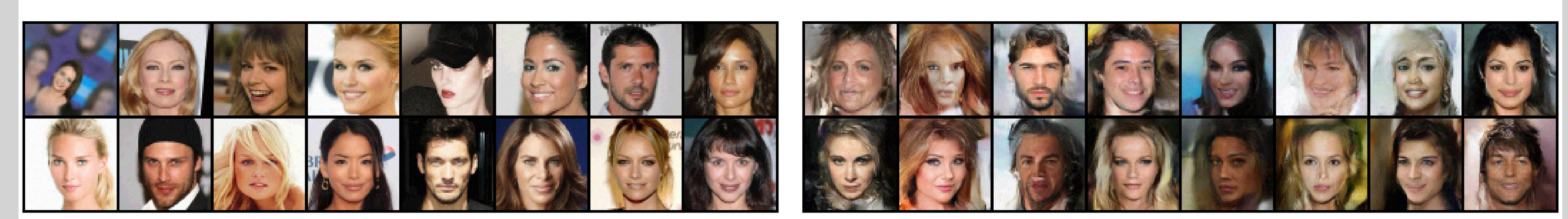
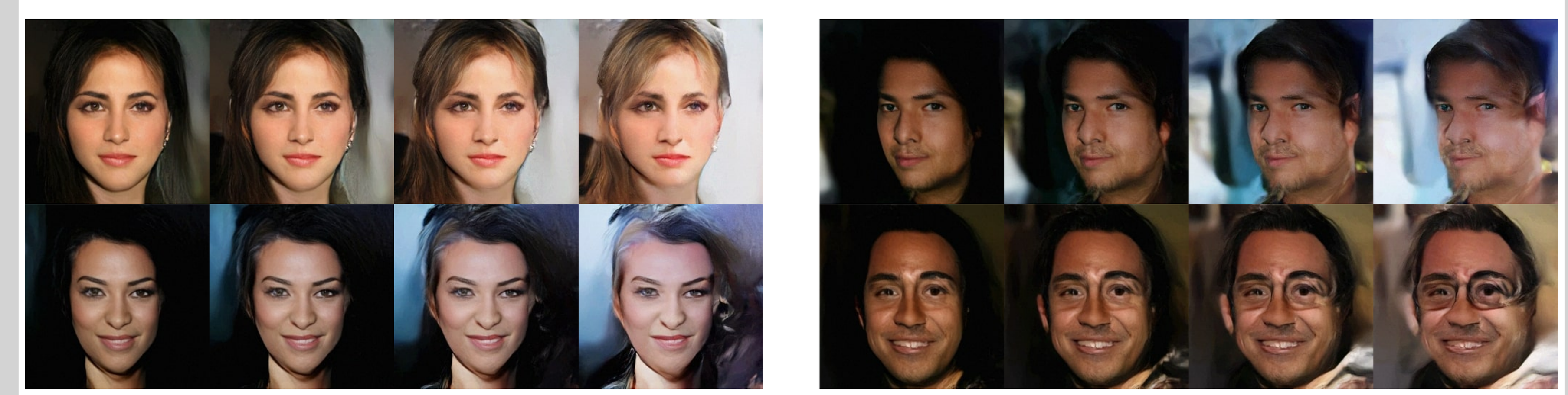


Figure: Real (left) and random samples (right) trained on 5bit 64×64 CelebA.



T=0.7 T=0.8 T=0.9 T=1.0 T=0.7 T=0.8 T=0.9 T=1.0

References

Behrmann *et al.*. "Invertible residual networks." (2019)

Kahn. "Use of different monte carlo sampling techniques." (1955)

Beatson & Adams. "Efficient Optimization of Loops and Limits..." (2019)

Nalisnick *et al.*. "Hybrid Models with Deep and Invertible Features." (2019)