Benefits of Invertible Flow-based Models

Maximum Likelihood Training:

Log-Likelihood

$$\log p_\theta(x) = \log p(f(x)) + \log \left| \det \frac{df_\theta(x)}{dx} \right|$$
Benefits of Invertible Flow-based Models

Maximum Likelihood Training:

\[
\log p_\theta(x) = \log p(f(x)) + \log \left| \det \frac{df_\theta(x)}{dx} \right|
\]

However, requires restricted architectures for invertibility & tractable log det.
Pathways to Designing a *Scalable* Normalizing Flow

1. Det Identities

- Planar NF
- Sylvester NF
- ...

Jacobian

(Low rank)
Pathways to Designing a *Scalable* Normalizing Flow

1. Det Identities
   - Planar NF
   - Sylvester NF
   - ...
   - Jacobian
     - (Low rank)

2. Coupling Blocks
   - NICE
   - Real NVP
   - Glow
   - ...
   - Jacobian
     - (Lower triangular + structured)
Pathways to Designing a *Scalable* Normalizing Flow

1. Det Identities
   - Planar NF
   - Sylvester NF
   - ...
   - Jacobian
     - (Low rank)

2. Coupling Blocks
   - NICE
   - Real NVP
   - Glow
   - ...
   - (Lower triangular + structured)

3. Autoregressive
   - Inverse AF
   - Neural AF
   - Masked AF
   - ...
   - (Lower triangular)
Pathways to Designing a **Scalable** Normalizing Flow

1. Det Identities
   - Planar NF
   - Sylvester NF
   - ...

2. Coupling Blocks
   - NICE
   - Real NVP
   - Glow
   - ...

3. Autoregressive
   - Inverse AF
   - Neural AF
   - Masked AF
   - ...

4. Unbiased Estimation
   - FFJORD
   - **Residual Flows**

Jacobian

- (Low rank)
- (Lower triangular + structured)
- (Lower triangular)
- (Arbitrary)
Invertible Residual Networks (i-ResNet)

It can be shown that residual blocks

\[ y = f(x) = x + g(x) \]

can be inverted by fixed-point iteration

\[ x^{(i)} = y - g(x^{(i-1)}) \]

and has a unique inverse (i.e. invertible) if

\[ |g(x) - g(y)| < |x - y| \]

(i.e. Lipschitz. Enforced with spectral normalization.)

(Behrmann et al. 2019)
Applying Change of Variables to i-ResNets

If

\[ y = f(x) = x + g(x) \]

Then

\[
\log p(x) = \log p(f(x)) + \log \left| \det \frac{df(x)}{dx} \right|
\]

\[
\log p(x) = \log p(f(x)) + \sum_{i=1}^{\infty} \frac{(-1)^{k+1}}{k} \text{tr}( [J_g(x)]^k )
\]

(Behrmann et al. 2019)
Unbiased Estimation of Log Probability Density

Enter the “Russian roulette” estimator (Kahn, 1955). Suppose we want to estimate

\[ \sum_{k=1}^{\infty} \Delta_k \]

(Require \( \sum_{k=1}^{\infty} |\Delta_k| < \infty \))
Unbiased Estimation of Log Probability Density

Enter the “Russian roulette” estimator (Kahn, 1955). Suppose we want to estimate

$$\sum_{k=1}^{\infty} \Delta_k$$

(Require $\sum_{k=1}^{\infty} |\Delta_k| < \infty$)

Flip a coin $b$ with probability $q$.

$$\mathbb{E} \left[ \Delta_1 + \right]$$
Unbiased Estimation of Log Probability Density

Enter the “Russian roulette” estimator (Kahn, 1955). Suppose we want to estimate

\[
\sum_{k=1}^{\infty} \Delta_k
\]

(Require \( \sum_{k=1}^{\infty} |\Delta_k| < \infty \))

Flip a coin \( b \) with probability \( q \).

\[
\mathbb{E} \left[ \Delta_1 + \begin{bmatrix} \mathbb{1}_{b=0} + [ \ ] \mathbb{1}_{b=1} \end{bmatrix} \right]
\]
Unbiased Estimation of Log Probability Density

Enter the “Russian roulette” estimator (Kahn, 1955). Suppose we want to estimate

\[ \sum_{k=1}^{\infty} \Delta_k \]

\( \text{(Require } \sum_{k=1}^{\infty} |\Delta_k| < \infty \text{)} \)

Flip a coin \( b \) with probability \( q \).

\[
\mathbb{E} \left[ \Delta_1 + \left[ \frac{1}{1-q} \sum_{k=2}^{\infty} \Delta_k \right] 1_{b=0} + [0] 1_{b=1} \right]
\]
Unbiased Estimation of Log Probability Density

Enter the “Russian roulette” estimator (Kahn, 1955). Suppose we want to estimate

\[ \sum_{k=1}^{\infty} \Delta_k \]

(Require \( \sum_{k=1}^{\infty} |\Delta_k| < \infty \))

Flip a coin \( b \) with probability \( q \).

\[
\mathbb{E} \left[ \Delta_1 + \left[ \frac{1}{1-q} \sum_{k=2}^{\infty} \Delta_k \right] \mathbb{1}_{b=0} + [0] \mathbb{1}_{b=1} \right]
\]

\[
= \Delta_1 + \left[ \frac{1}{1-q} \sum_{k=2}^{\infty} \Delta_k \right] (1 - q)
\]

\[
= \sum_{k=1}^{\infty} \Delta_k
\]

Has probability \( q \) of being evaluated in \textbf{finite} time.
Unbiased Estimation of Log Probability Density

If we repeatedly apply the same procedure \textit{infinitely many times}, we obtain an unbiased estimator of the infinite series.

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

\textbf{Computed in finite time with prob. 1!!}
Unbiased Estimation of Log Probability Density

If we repeatedly apply the same procedure infinitely many times, we obtain an unbiased estimator of the infinite series.

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

Computed in finite time with prob. 1!!

Directly sample the first successful coin toss.
Unbiased Estimation of Log Probability Density

If we repeatedly apply the same procedure \textit{infinitely many times}, we obtain an unbiased estimator of the infinite series.

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

Directly sample the first successful coin toss.

k-th term is weighted by prob. of seeing \( \geq k \) tosses.

Computed in finite time with prob. 1!!
Unbiased Estimation of Log Probability Density

If we repeatedly apply the same procedure infinitely many times, we obtain an unbiased estimator of the infinite series.

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

Directly sample the first successful coin toss. The k-th term is weighted by prob. of seeing \( \geq k \) tosses.

Residual Flow:

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

Computed in finite time with prob. 1!!
Decoupled Training Objective & Estimation Bias
Decoupled Training Objective & Estimation Bias

Unbiased but... variable compute and memory!
Constant-Memory Backpropagation

Naive gradient computation:

\[
E_{n, v} \left[ \sum_{k=1}^{n} \alpha_k \frac{\partial v^T [J_g(x)]^k v}{\partial \theta} \right]
\]

Alternative (Neumann series) gradient formulation:

\[
E_{n, v} \left[ (\sum_{k=1}^{n} \alpha_k v^T [J_g(x)]^k ) \frac{\partial J_g(x)v}{\partial \theta} \right]
\]

Don’t need to store random number of terms in memory!!
Density Estimation Experiments

Contribution Summary:
- Unbiased estimator of log-likelihood.
- Memory-efficient computation of log-likelihood.
- LipSwish activation function [not discussed in talk].

<table>
<thead>
<tr>
<th>Model</th>
<th>MNIST</th>
<th>CIFAR-10</th>
<th>ImageNet 32</th>
<th>ImageNet 64</th>
<th>CelebA-HQ 256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real NVP</td>
<td>1.06</td>
<td>3.49</td>
<td>4.28</td>
<td>3.98</td>
<td>—</td>
</tr>
<tr>
<td>Glow</td>
<td>1.05</td>
<td>3.35</td>
<td>4.09</td>
<td>3.81</td>
<td>1.03</td>
</tr>
<tr>
<td>FFJORD</td>
<td>0.99</td>
<td>3.40</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Flow++</td>
<td>—</td>
<td>3.29 (3.09)</td>
<td>— (3.86)</td>
<td>— (3.69)</td>
<td>—</td>
</tr>
<tr>
<td>i-ResNet</td>
<td>1.05</td>
<td>3.45</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Residual Flow</td>
<td><strong>0.970</strong></td>
<td><strong>3.280</strong></td>
<td><strong>4.010</strong></td>
<td><strong>3.757</strong></td>
<td><strong>0.992</strong></td>
</tr>
</tbody>
</table>

(LipSwish)
Density Estimation Experiments

Contribution Summary:
- Unbiased estimator of log-likelihood.
- Memory-efficient computation of log-likelihood.
- LipSwish activation function [not discussed in talk].

<table>
<thead>
<tr>
<th>Training Setting</th>
<th>MNIST</th>
<th>CIFAR-10†</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-ResNet + ELU</td>
<td>1.05</td>
<td>3.45</td>
<td>3.66~4.78</td>
</tr>
<tr>
<td>Residual Flow + ELU</td>
<td>1.00</td>
<td>3.40</td>
<td>3.32</td>
</tr>
<tr>
<td>Residual Flow + LipSwish</td>
<td>0.97</td>
<td>3.39</td>
<td>3.28</td>
</tr>
</tbody>
</table>
Qualitative Samples

CelebA:

CIFAR10:

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR10 FID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PixelCNN*</td>
<td>65.93</td>
</tr>
<tr>
<td>PixelIQN*</td>
<td>49.46</td>
</tr>
<tr>
<td>i-ResNet</td>
<td>65.01</td>
</tr>
<tr>
<td>Residual Flow</td>
<td>46.37</td>
</tr>
<tr>
<td>DCGAN*</td>
<td>37.11</td>
</tr>
<tr>
<td>WGAN-GP*</td>
<td>36.40</td>
</tr>
</tbody>
</table>
Qualitative Samples

CelebA: Data

CelebA-HQ 256x256:

Residual Flow

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR10 FID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PixelCNN*</td>
<td>65.93</td>
</tr>
<tr>
<td>PixelIQN*</td>
<td>49.46</td>
</tr>
<tr>
<td>i-ResNet</td>
<td>65.01</td>
</tr>
<tr>
<td>Residual Flow</td>
<td>46.37</td>
</tr>
<tr>
<td>DCGAN*</td>
<td>37.11</td>
</tr>
<tr>
<td>WGAN-GP*</td>
<td>36.40</td>
</tr>
</tbody>
</table>
Thanks for Listening!

Code online: https://github.com/rtqichen/residual-flows

Co-authors:

Jens Behrmann  
David Duvenaud  
Jörn-Henrik Jacobsen