Introduction to Flow-based Generative Models

Pascal Mignon

pmignon@student.ethz.ch

13/04/2022

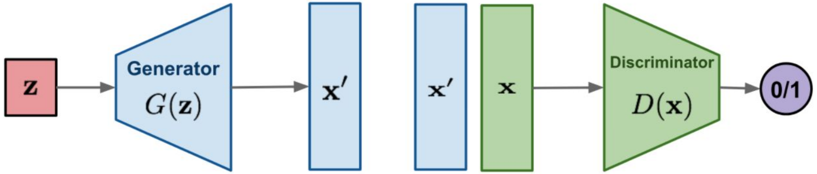
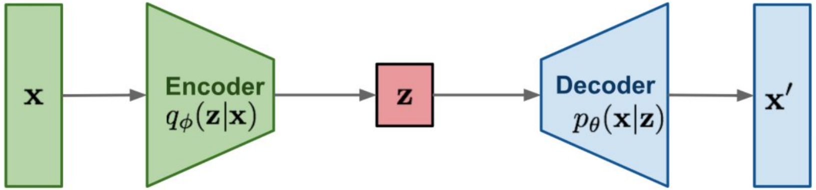
Agenda

- Recap: Generative Models
- Flow-based Generative Models
- RealNVP & Glow: illustrations of Flow-based models
- Related works
- Q&A

Recap: Generative Models

Recap: Generative models

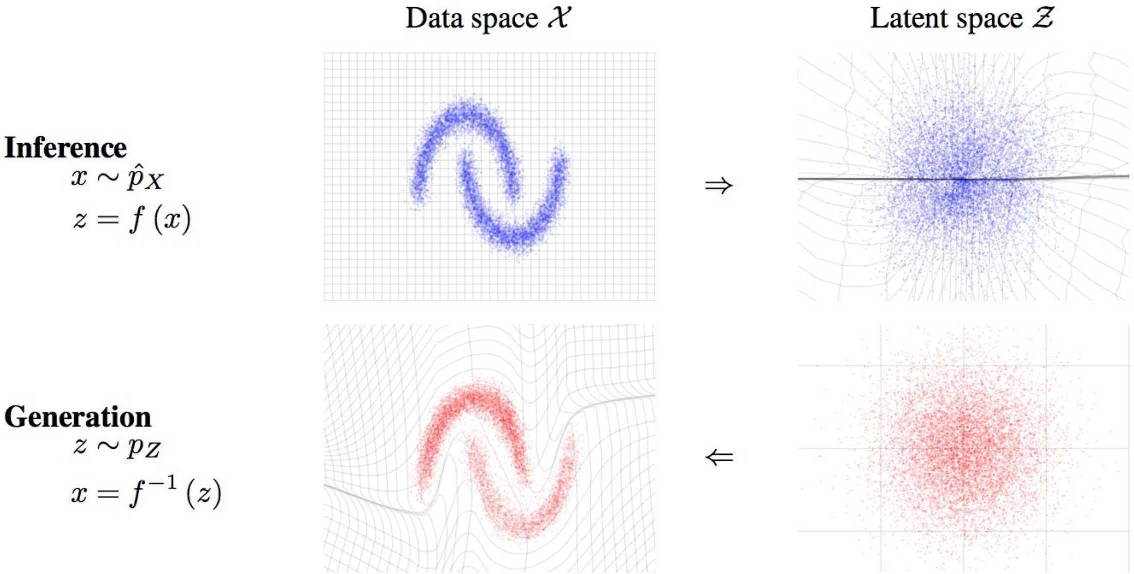
Types of Generative Models

	Description	Architecture
Generative adversarial networks	Minimize the discriminator error loss	
Variational autoencoders	Maximize the Evidence Lower Bound (ELBO)	
Flow-based generative models	?	?

Flow-based Generative Models

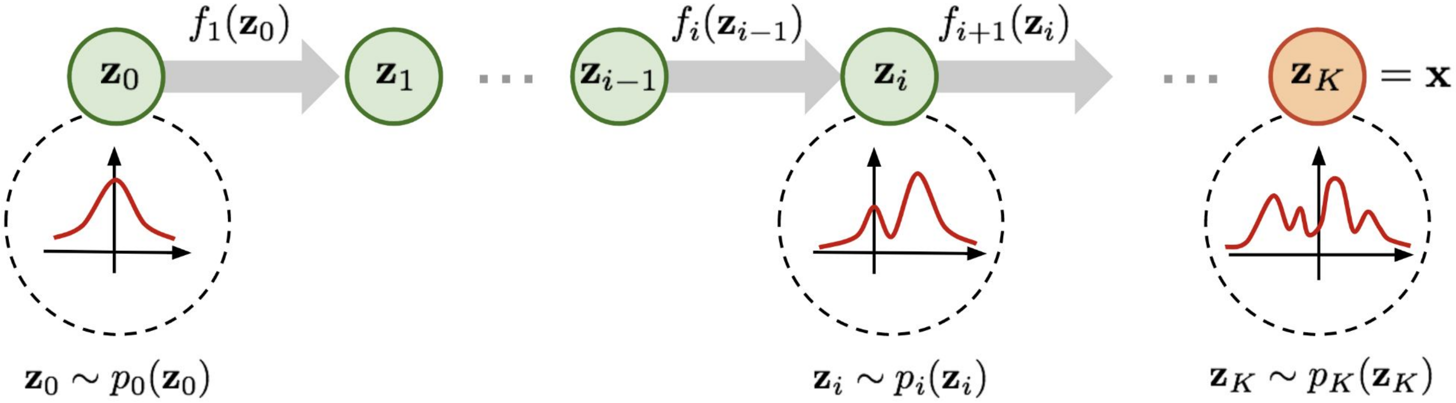
Flow-based Generative Models

Normalizing flows - overall idea



Flow-based Generative Models

Normalizing flows - overall idea



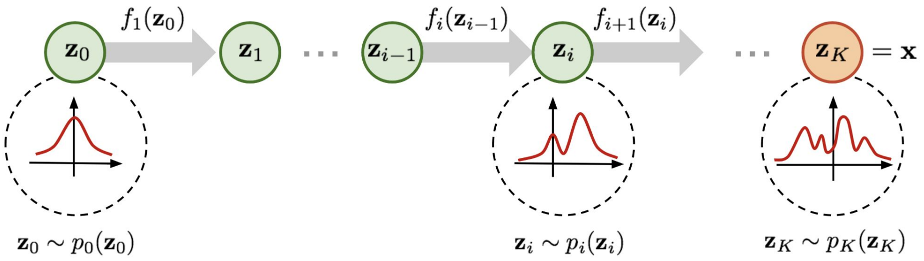
Flow-based Generative Models

Normalizing flows - requirements

- ❏ **f must be reversible, ie. bijective**
- ❏ **f jacobian must be easily computable**

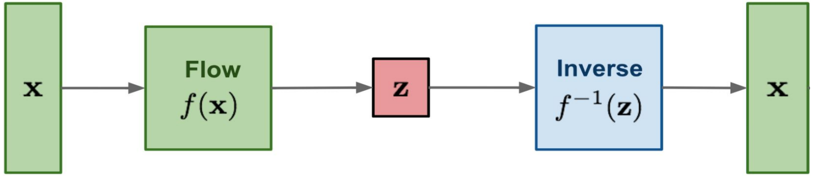
$$\mathbf{z}_K = f_K \circ \ldots \circ f_2 \circ f_1(\mathbf{z}_0)$$

$$\ln p_K(\mathbf{z}_K) = \ln p_0(\mathbf{z}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|$$



Flow-based Generative Models

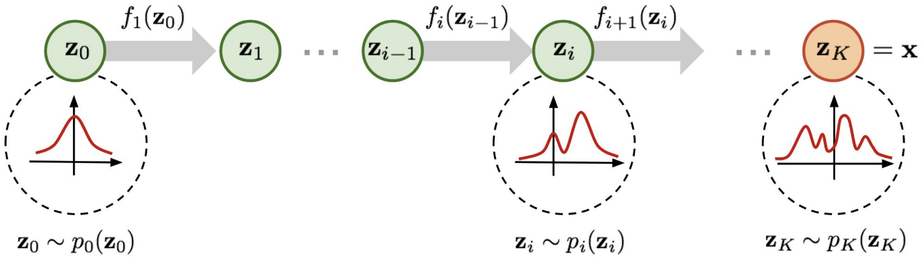
Normalizing flows - summary

	Description	Architecture
Flow-based generative models	Reversible - Using explicitly log-likelihood to train the generative model	

$$\mathbf{z}_K = f_K \circ \ldots \circ f_2 \circ f_1(\mathbf{z}_0)$$

$$\ln p_K(\mathbf{z}_K) = \ln p_0(\mathbf{z}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|$$

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x})$$



RealNVP & Glow: illustrations of Flow-based models

Illustration of Flow-based models

RealNVP, 2016

DENSITY ESTIMATION USING REAL NVP

Laurent Dinh*
Montreal Institute for Learning Algorithms
University of Montreal
Montreal, QC H3T1J4

Jascha Sohl-Dickstein
Google Brain

Samy Bengio
Google Brain

Illustration of Flow-based models

RealNVP, 2016, objective

Objective: create a reversible flow from input space x to latent space z

Constraints: intermediate functions must be reversible and the jacobian easily computable

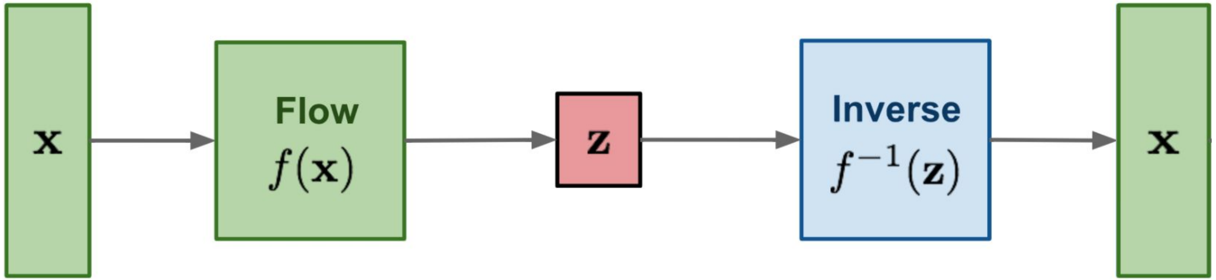
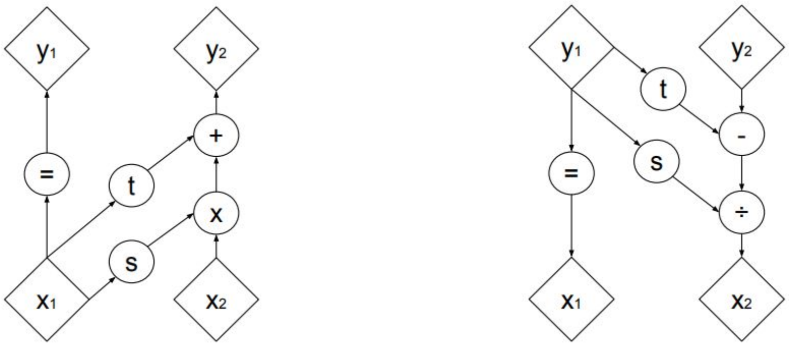


Illustration of Flow-based models

RealNVP, 2016, proposed solution: coupling layers



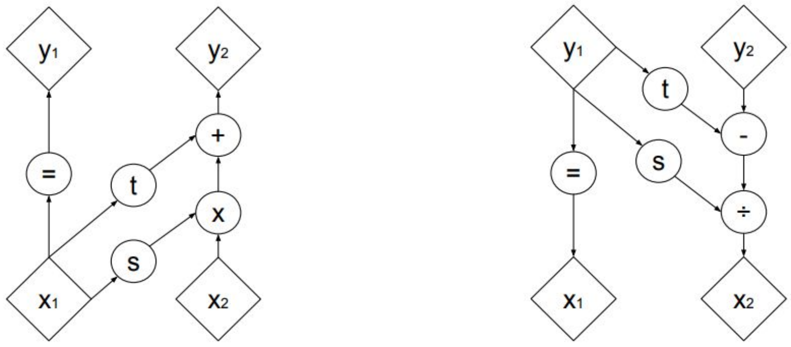
(a) Forward propagation (b) Inverse propagation

$$\begin{cases} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot \exp\big(s(x_{1:d})\big) + t(x_{1:d}) \end{cases}$$

$$\Leftrightarrow \begin{cases} x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= \big(y_{d+1:D} - t(y_{1:d})\big) \odot \exp\big(-s(y_{1:d})\big), \end{cases}$$

Illustration of Flow-based models

RealNVP, 2016, proposed solution: coupling layers



(a) Forward propagation (b) Inverse propagation

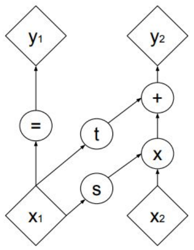
$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \operatorname{diag}\left(\exp\left[s\left(x_{1:d}\right)\right]\right) \end{bmatrix}$$

$$\begin{cases} y_{1:d} & = x_{1:d} \\ y_{d+1:D} & = x_{d+1:D} \odot \exp\left(s(x_{1:d})\right) + t(x_{1:d}) \end{cases}$$

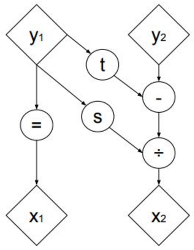
$$\Leftrightarrow \begin{cases} x_{1:d} & = y_{1:d} \\ x_{d+1:D} & = \left(y_{d+1:D} - t(y_{1:d})\right) \odot \exp\left(-s(y_{1:d})\right), \end{cases}$$

Illustration of Flow-based models

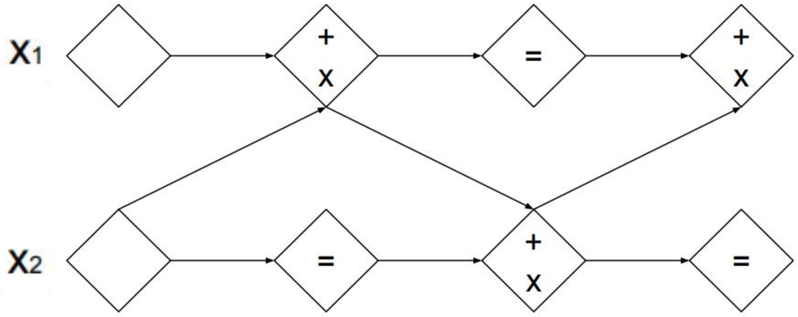
RealNVP, 2016, proposed solution: coupling layers with permutation



(a) Forward propagation



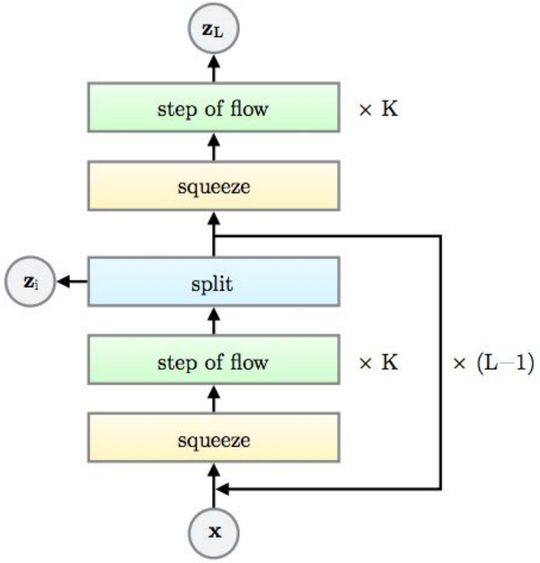
(b) Inverse propagation



(a) In this alternating pattern, units which remain identical in one transformation are modified in the next.

Illustration of Flow-based models

RealNVP, 2016, proposed solution: multi-scale arhitecture



(b) Multi-scale architecture (Dinh et al., 2016).

Illustration of Flow-based models

Glow, 2018

Glow: Generative Flow with Invertible 1×1 Convolutions

Diederik P. Kingma*, Prafulla Dhariwal*
OpenAI, San Francisco

Illustration of Flow-based models

Glow, 2018, objective

Objective: create a reversible flow from input space x to latent space z

Constraints: intermediate functions must be reversible and the jacobian easily computable

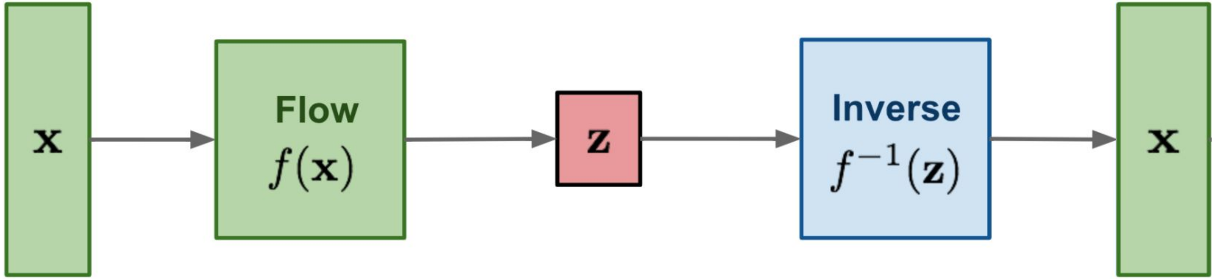


Illustration of Flow-based models

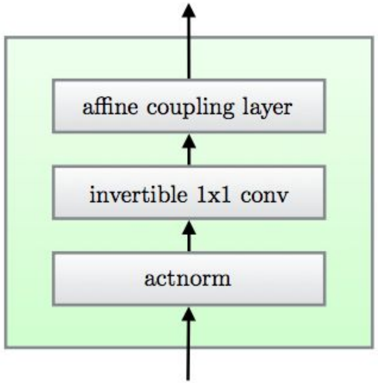
Glow, 2018, Architecture

$$\mathbf{x} \xleftrightarrow{\mathbf{f}_1} \mathbf{h}_1 \xleftrightarrow{\mathbf{f}_2} \mathbf{h}_2 \cdots \xleftrightarrow{\mathbf{f}_K} \mathbf{z}$$

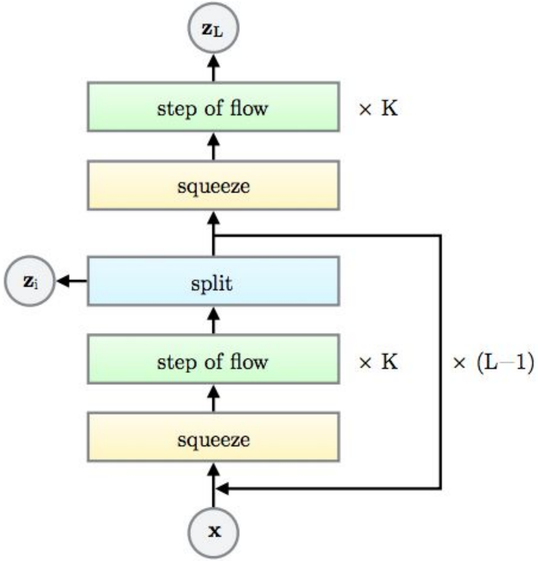
$$\mathbf{z} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x})$$

$$\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z})$$

$$\mathcal{L}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} -\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$$



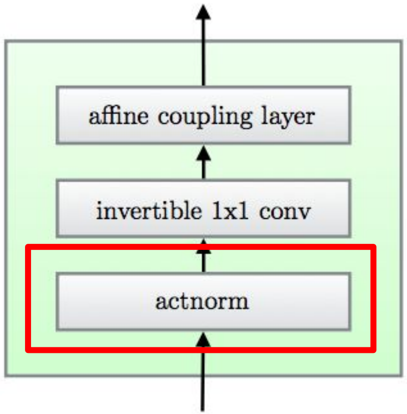
(a) One step of our flow.



(b) Multi-scale architecture (Dinh et al., 2016).

Illustration of Flow-based models

Glow, 2018, Actnorm (activation normalization) layer

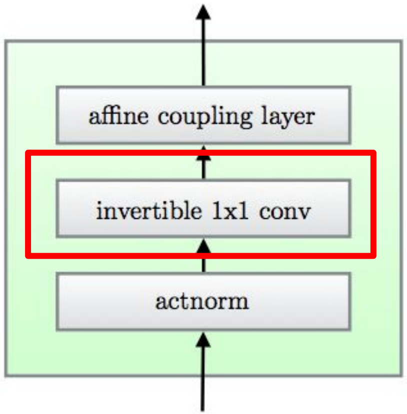


Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$

- Affine transformation for batch normalization
- Parameters are chosen to have as output 0 mean and 1 standard deviation

Illustration of Flow-based models

Glow, 2018, 1x1 convolutional layer

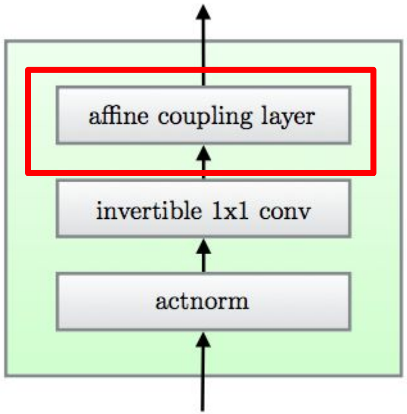


Description	Function	Reverse Function	Log-determinant
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2.	$\forall i, j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i, j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert \det(\mathbf{W}) \rvert$ or $h \cdot w \cdot \mathtt{sum}(\log \lvert \mathbf{s} \rvert)$ (see eq. (10))

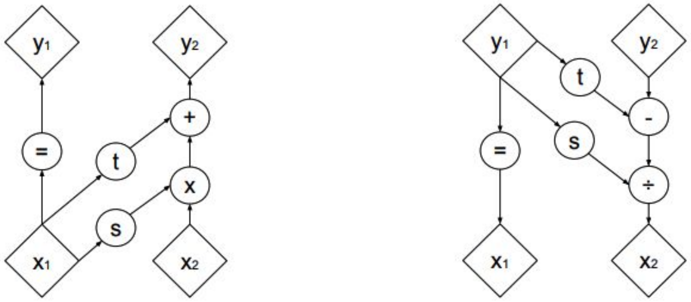
- 1x1 convolution for permutation operation
- Weights are initialized as a rotation matrix
- Computation time can be optimized by choosing weights in its LU matrix decomposition

Illustration of Flow-based models

Glow, 2018, Affine coupling layer



Description	Function	Reverse Function	Log-determinant
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \texttt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \texttt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \texttt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \texttt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \texttt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t}) / \mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \texttt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\texttt{sum}(\log(\lvert\mathbf{s}\rvert))$



(a) Forward propagation (b) Inverse propagation

Illustration of Flow-based models

RealNVP, 2016 vs. Glow, 2018, Comparative results

Table 2: Best results in bits per dimension of our model compared to RealNVP.

Model	CIFAR-10	ImageNet 32x32	ImageNet 64x64	LSUN (bedroom)	LSUN (tower)	LSUN (church outdoor)
RealNVP	3.49	4.28	3.98	2.72	2.81	3.08
Glow	**3.35**	**4.09**	**3.81**	**2.38**	**2.46**	**2.67**

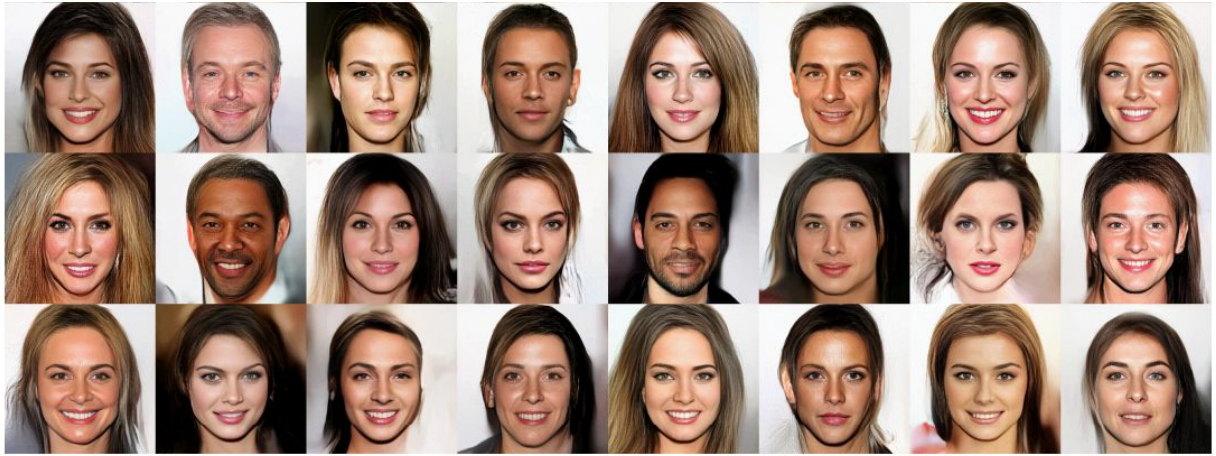
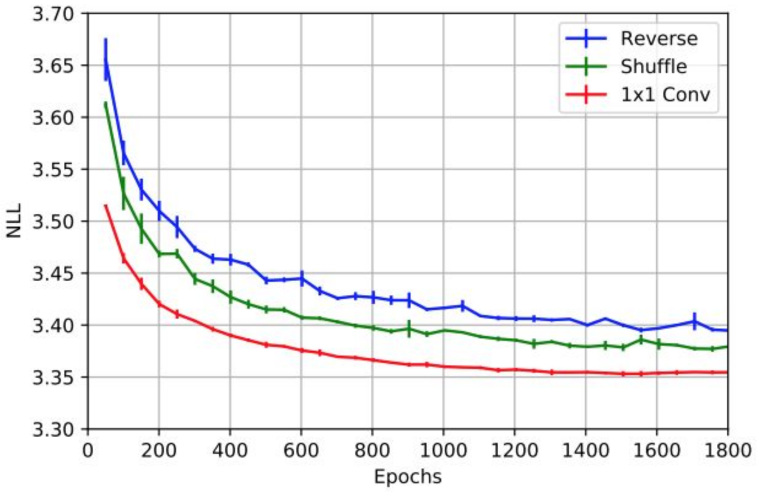


Illustration of Flow-based models

RealNVP, 2016 vs. Glow, 2018, Comparative results

Table 2: Best results in bits per dimension of our model compared to RealNVP.

Model	CIFAR-10	ImageNet 32x32	ImageNet 64x64	LSUN (bedroom)	LSUN (tower)	LSUN (church outdoor)
RealNVP	3.49	4.28	3.98	2.72	2.81	3.08
Glow	**3.35**	**4.09**	**3.81**	**2.38**	**2.46**	**2.67**



(b) Affine coupling.

Conclusion

Illustration of Flow-based models

Strengths and weaknesses of flow-based models

Strengths

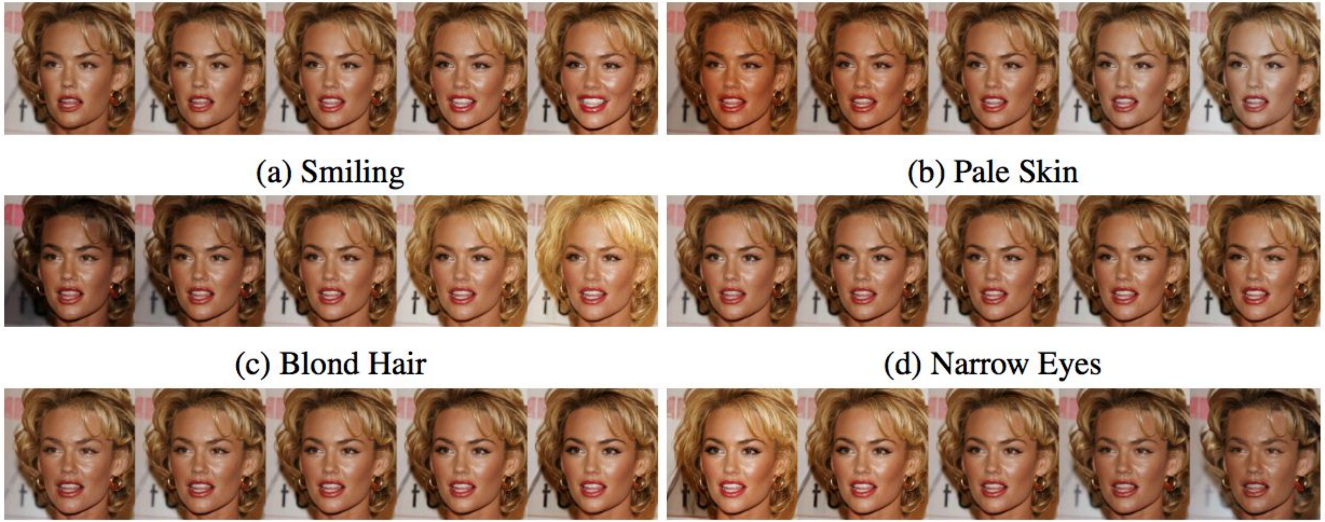
- **Reversible**, ie. direct transformation between input space and latent space
- Direct use of **log-likelihood**
- Easily **parallelizable**

Weaknesses

- Important **computation** time
- Not easy to **compute** log-likelihood

Illustration of Flow-based models

Glow, 2018 - Interesting experiments

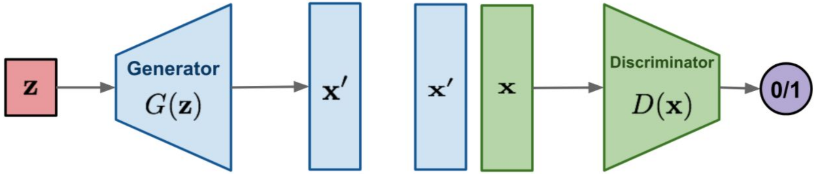
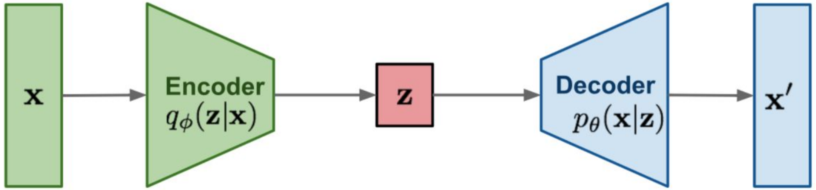
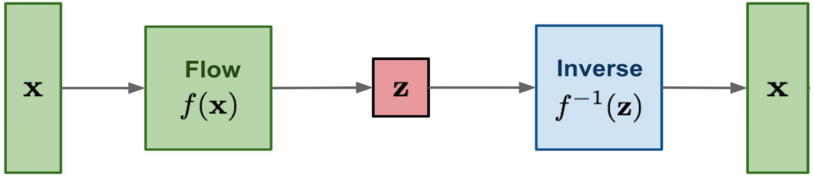


(a) Smiling (b) Pale Skin (c) Blond Hair (d) Narrow Eyes (e) Young (f) Male

Figure 6: Manipulation of attributes of a face. Each row is made by interpolating the latent code of an image along a vector corresponding to the attribute, with the middle image being the original image

Recap: Generative models

Types of Generative Models

	Description	Architecture
Generative adversarial networks	Minimize the discriminator error loss	
Variational autoencoders	Maximize the Evidence Lower Bound (ELBO)	
Flow-based generative models	Reversible - Using explicitly log-likelihood to train the generative model	

Related Works

Related works

Other related or relevant topics

- Other models with Normalizing Flows
  - NICE 2014
- Models based on Autoregressive Flows
  - PixelRNN 2016
- Divergence-based Generative Models
  - Moser Flow: Divergence based generative models on manifolds 2021

Q&A

Additional content

Change of variable formula

Change of Variables: Z and X be random variables which are related by a mapping $f : \mathbb{R}^n \to \mathbb{R}^n$ such that $X = f(Z)$ and $Z = f^{-1}(X)$. Then

$$p_X(\mathbb{x}) = p_Z(f^{-1}(\mathbb{x})) \left| \det \left(\frac{\partial f^{-1}(\mathbb{x})}{\partial \mathbb{x}} \right) \right|$$

Jacobian Matrix

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \qquad \mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}$$

Illustration of Flow-based models

Glow, 2018 - Interesting experiments

Figure 9: Samples from shallow model on left vs deep model on right. Shallow model has $L = 4$ levels, while deep model has $L = 6$ levels

Illustration of Flow-based models

Glow, 2018 - Interesting experiments

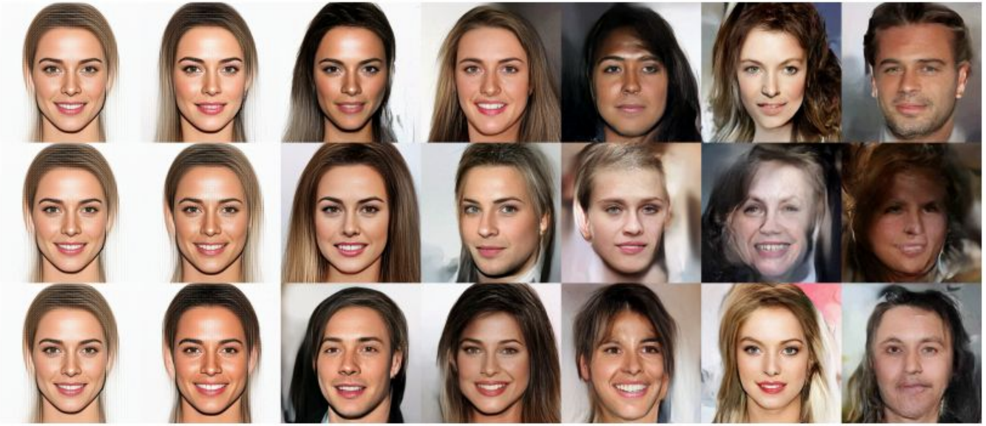
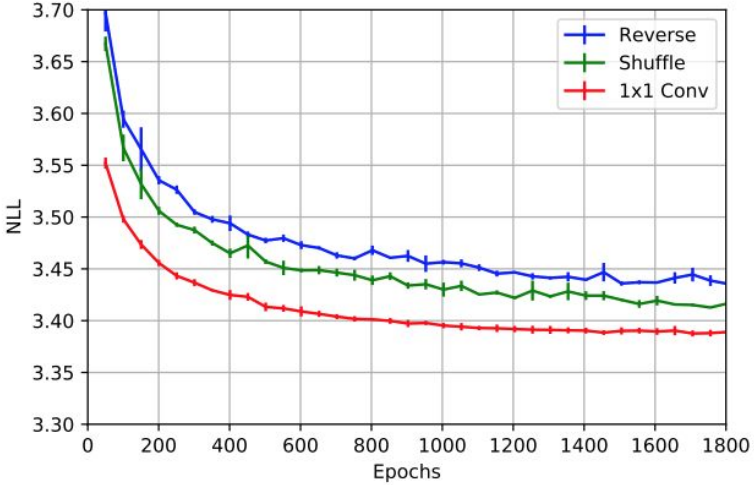


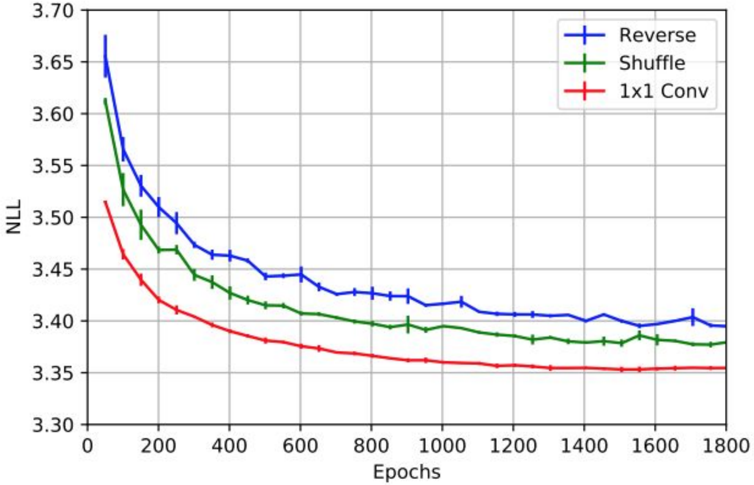
Figure 8: Effect of change of temperature. From left to right, samples obtained at temperatures $0, 0.25, 0.6, 0.7, 0.8, 0.9, 1.0$

Illustration of Flow-based models

[RealNVP, 2016](#) vs. [Glow, 2018](#), Improvements of 1x1 convolution



(a) Additive coupling.



(b) Affine coupling.