Glow : Generative Flow with Invertible 1x1 Convolutions

NeurlPS 2019

Presenter : Dajin Han

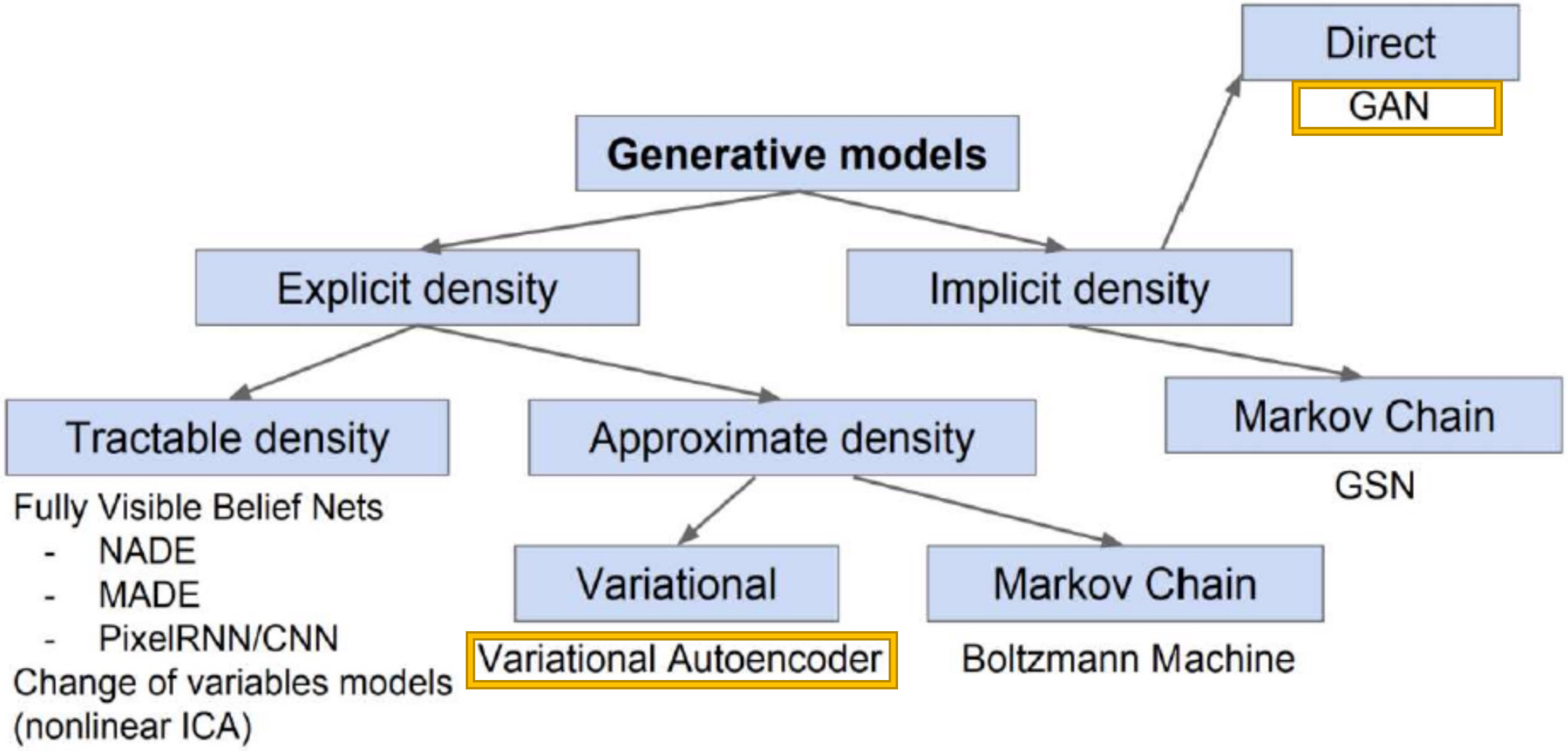
What is Generative Model?

Training samples $\sim p_{data}(x)$

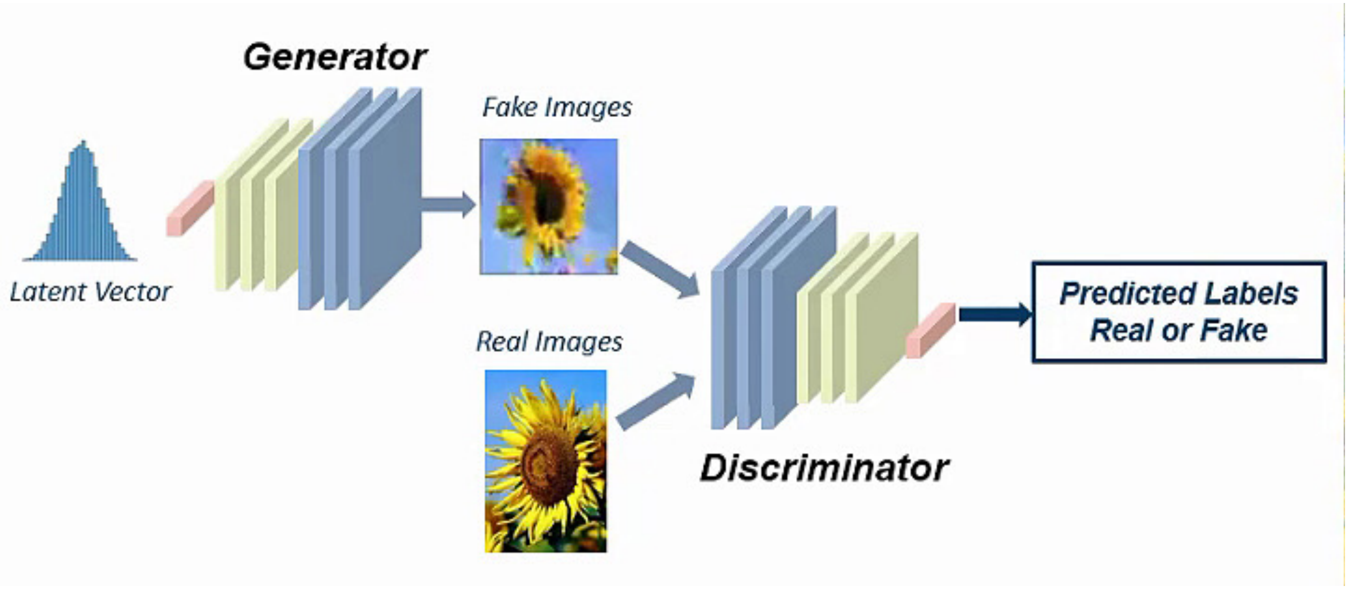
Generated samples $\sim p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Generative models



GAN



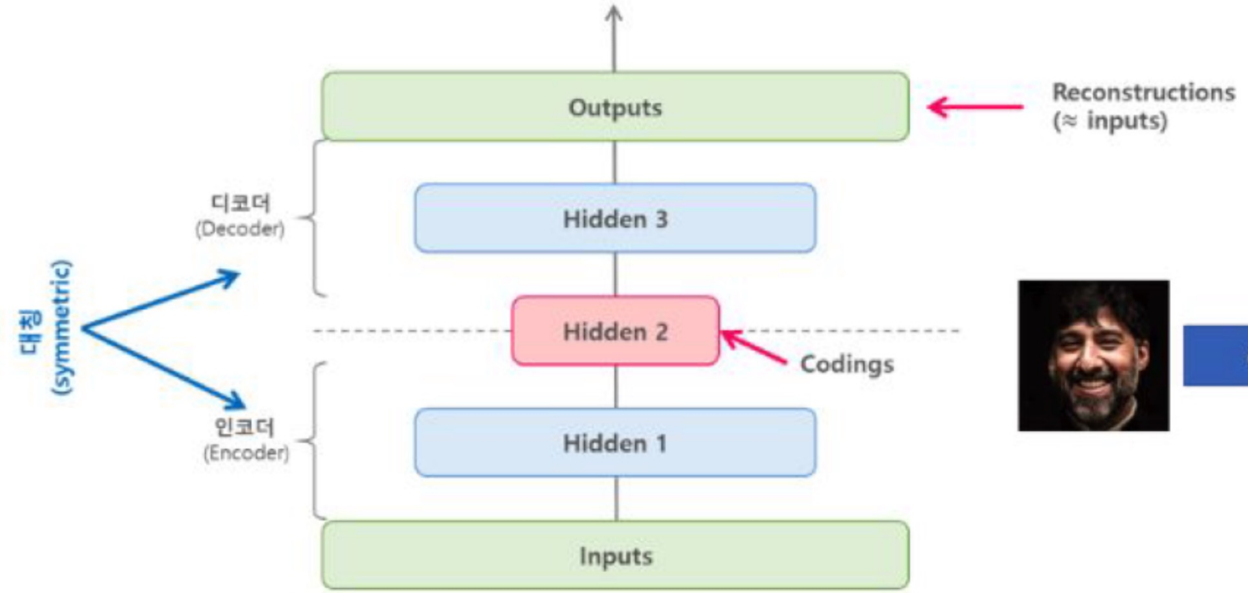
Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log D_{\theta_d}(x)+\mathbb{E}_{z\sim p(z)}\log\left(1-D_{\theta_d}(G_{\theta_g}(z))\right)\right]$$

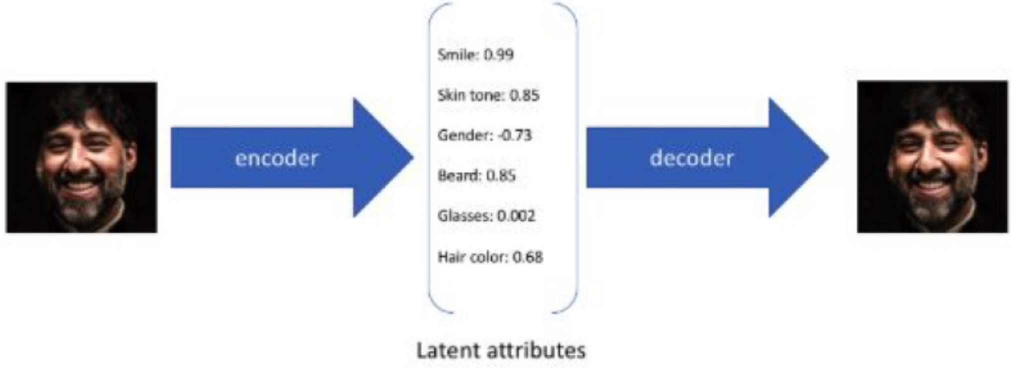
Discriminator output for real data x

Discriminator output for generated fake data G(z)

AE

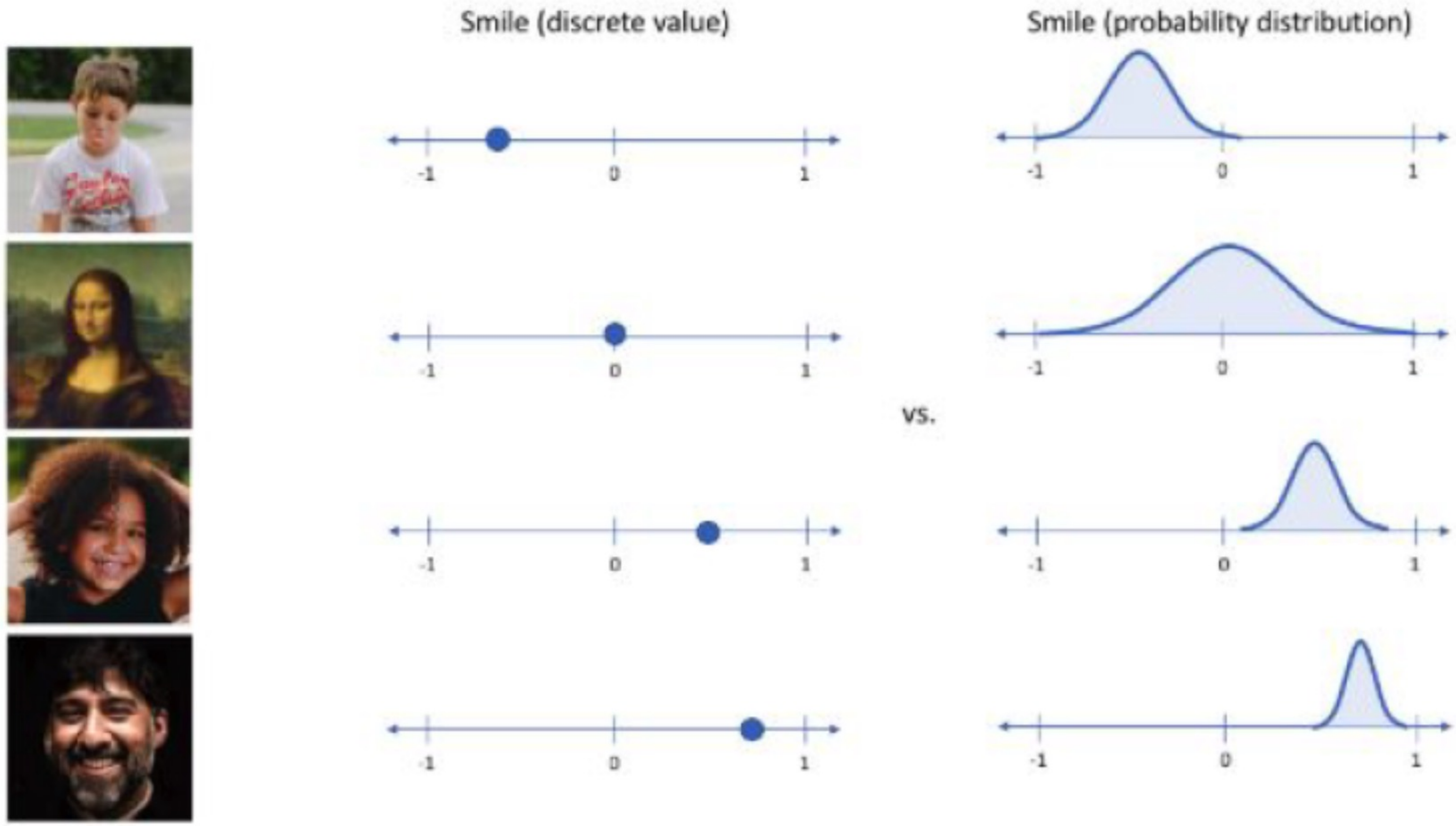


Outputs
Reconstructions
(≈ inputs)
디코더
(Decoder)
Hidden 3
대칭
(symmetric)
Hidden 2
Codings
인코더
(Encoder)
Hidden 1
Inputs



encoder
Smile: 0.99
Skin tone: 0.85
Gender: -0.73
Beard: 0.85
Glasses: 0.002
Hair color: 0.68
decoder
Latent attributes

VAE

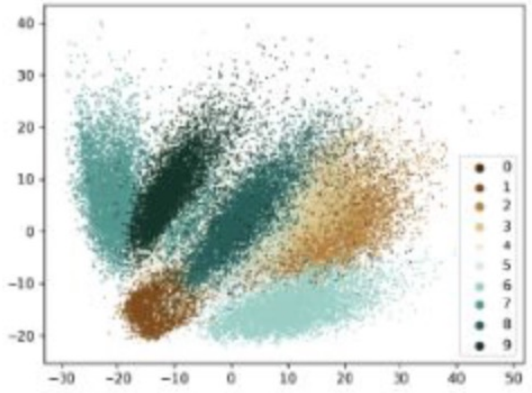


Smile (discrete value)
Smile (probability distribution)
-1
0
1
vs.

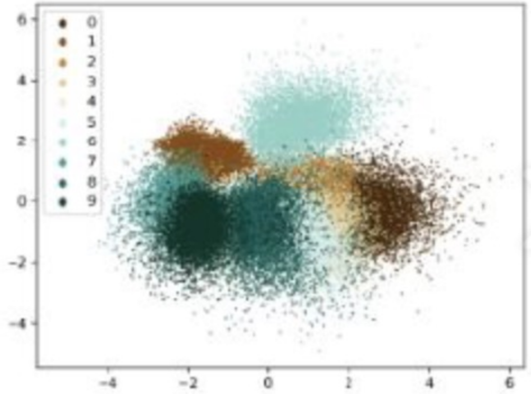
VAE

- Variational Inference
- KL Divergence
- ELBO

- NF : improve function q()

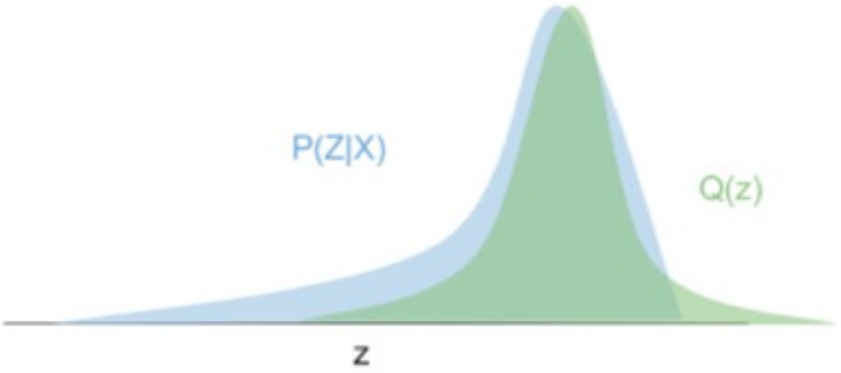


(a) *Latent Distribution by Label for AE*



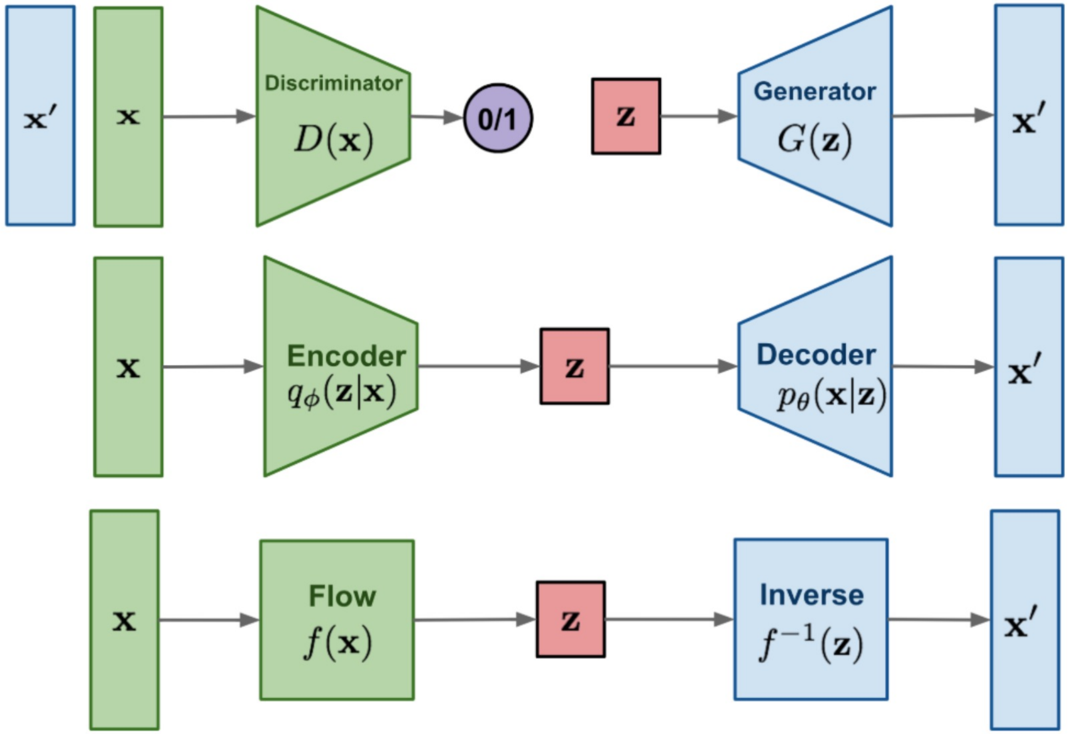
(b) *Latent Distribution by Label for VAE*

$$ELBO = E_{z\sim q(z|x)}\left[\log p(x|z)\right] - D_{KL}\left(q\left(z|x\right)||p\left(z|x\right)\right)$$



Generative models

- Generative Adversarial Networks
  - Generate fake image looks real
  - Minmax the classification error loss
- Variational AutoEncoder
  - Approximate data distribution
  - Maximize ELBO
- Flow-based generative models
  - Approximate data distribution
  - Minimize the negative log-likelihood



Jacobian Matrix

- Jacobian Matrix (MxN)
  - Input vector X (Nx1)
  - Output vector Y (Mx1)
  - Y = f(X)
  - Jacobian Matrix = f’

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Change of Variable Theorem

The multivariable version has a similar format:

$$\mathbf{z} \sim \pi(\mathbf{z}), \mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x})$$

$$p(\mathbf{x}) = \pi(\mathbf{z}) \left| \det \frac{d\mathbf{z}}{d\mathbf{x}} \right| = \pi(f^{-1}(\mathbf{x})) \left| \det \frac{df^{-1}}{d\mathbf{x}} \right|$$

where $\det \frac{\partial f}{\partial \mathbf{z}}$ is the Jacobian determinant of the function f. The full proof of the multivariate version is out of the scope of this post; ask Google if interested ;)

data space에서의 likelihood

Z space에서의 likelihood

transform에 의해 변환된 영역의 비율

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

$$\log(p_X(x)) = \log\Big(p_Z(f(x))\Big) + \log\left(\left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right| \right)$$

Normalizing Flow

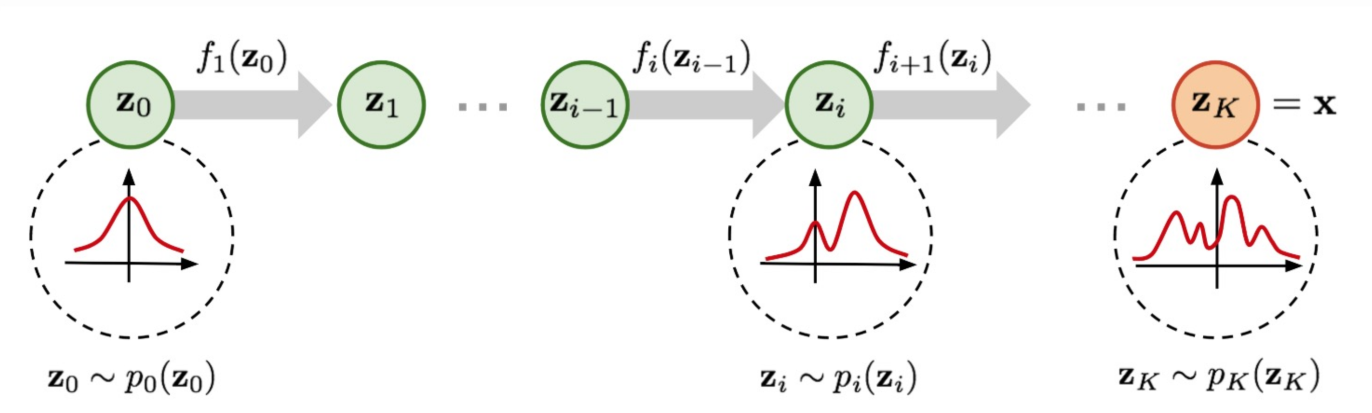
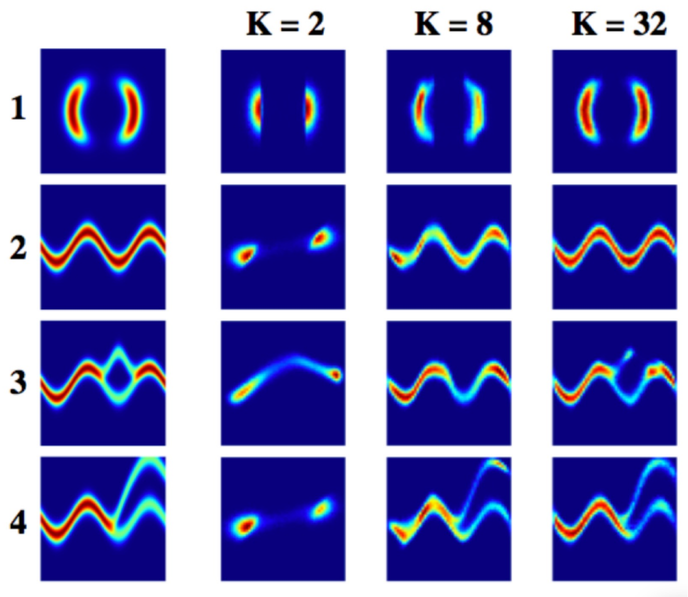
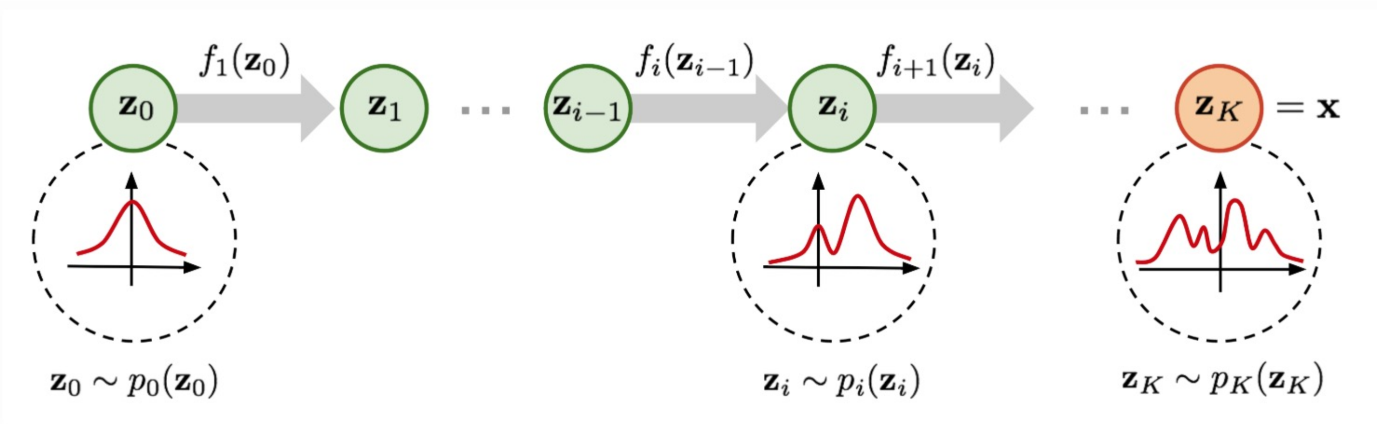


Fig. 2. Illustration of a normalizing flow model, transforming a simple distribution $p_0(\mathbf{z}_0)$ to a complex one $p_K(\mathbf{z}_K)$ step by step.



Normalizing Flow



$$\mathbf{z} \sim \pi(\mathbf{z}), \mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x})$$
$$p(\mathbf{x}) = \pi(\mathbf{z}) \left|\det \frac{d\mathbf{z}}{d\mathbf{x}}\right| = \pi(f^{-1}(\mathbf{x})) \left|\det \frac{df^{-1}}{d\mathbf{x}}\right|$$

$$\mathbf{z}_{i-1} \sim p_{i-1}(\mathbf{z}_{i-1})$$
$$\mathbf{z}_i = f_i(\mathbf{z}_{i-1}), \text{thus } \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i)$$
$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left|\det \frac{df_i^{-1}}{d\mathbf{z}_i}\right|$$

Normalizing Flow

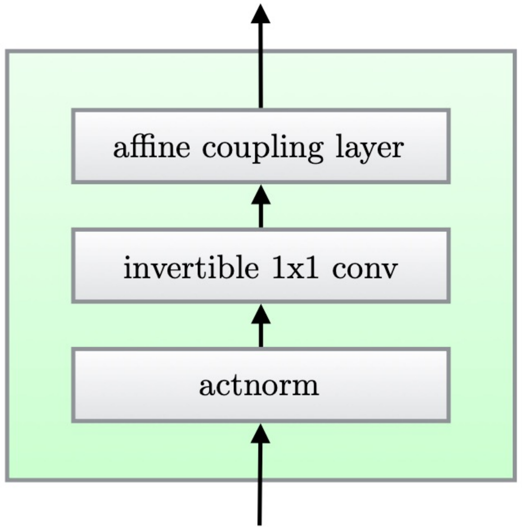
$$
\begin{aligned}
\mathbf{x} = \mathbf{z}_K &= f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0) \\
\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log\left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\
&= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log\left|\det \frac{df_{K-1}}{d\mathbf{z}_{K-2}}\right| - \log\left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\
&= \dots \\
&= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log\left|\det \frac{df_i}{d\mathbf{z}_{i-1}}\right|
\end{aligned}
$$

Function f should satisfy two properties:

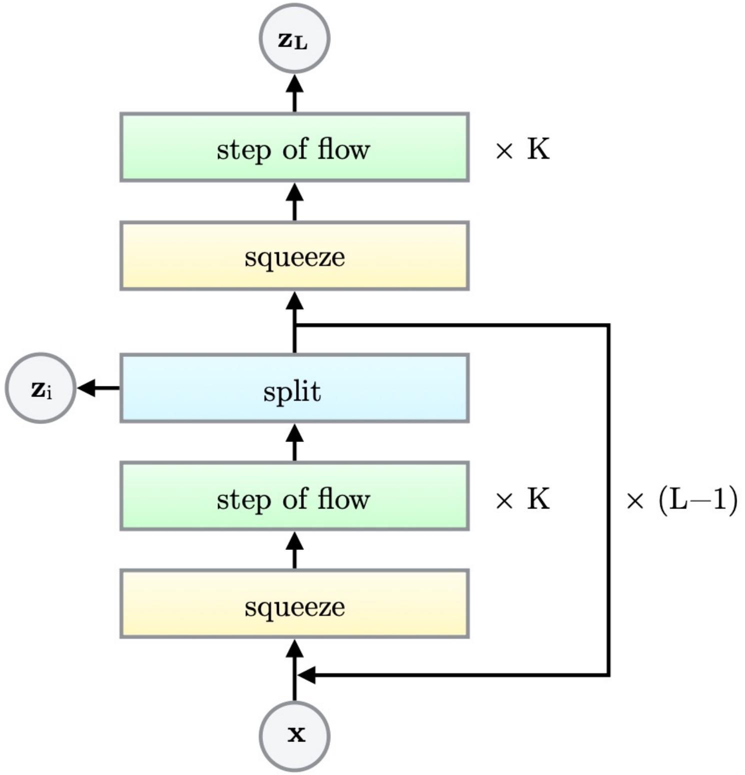
1. It is easily invertible
2. Its Jacobian Determinant is easy to compute

Architecture

1. Affine coupling layer
2. Invertible 1x1 conv
3. actnorm



(a) One step of our flow.



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

Architecture

- Activation normalization (actnorm)
  - Affine transformation using a scale and bias parameter per channel
  - Trainable parameters, but initialized
  - mean = 0, standard deviation = 1

$$\forall i, j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$$

Architecture

- Invertible 1x1 conv
  - Ordering of channels is reversed so that all the data dimensions have a change to be altered.

$$\log \left| \det \frac{\partial \texttt{conv2d}(\mathbf{h}; \mathbf{W})}{\partial \mathbf{h}} \right| = \log(|\det \mathbf{W}|^{h \cdot w}|) = h \cdot w \cdot \log |\det \mathbf{W}|$$

Architecture

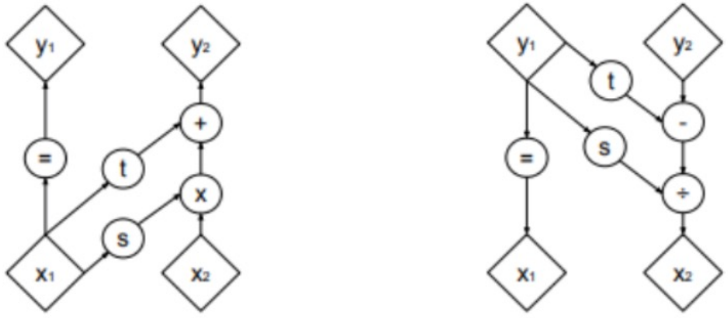
- Affine coupliung layer
  - Same as in RealNVP

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$
$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d})$$

Architecture

- The first d dimensions stay same;
- The second part, $d+1$ to D dimensions, undergo an affine transformation ("scale-and-shift") and both the scale and shift parameters are functions of the first d dimensions.

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$
$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d})$$



(a) Forward propagation (b) Inverse propagation

$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} & = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} & = (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d})) \odot \exp(-s(\mathbf{y}_{1:d})) \end{cases}$$

Architecture

- The first d dimensions stay same;
- The second part, $d+1$ to D dimensions, undergo an affine transformation ("scale-and-shift") and both the scale and shift parameters are functions of the first d dimensions.

$$\begin{aligned}
\mathbf{y}_{1:d} &= \mathbf{x}_{1:d} \\
\mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d})
\end{aligned}$$

Condition 2: "Its Jacobian determinant is easy to compute."

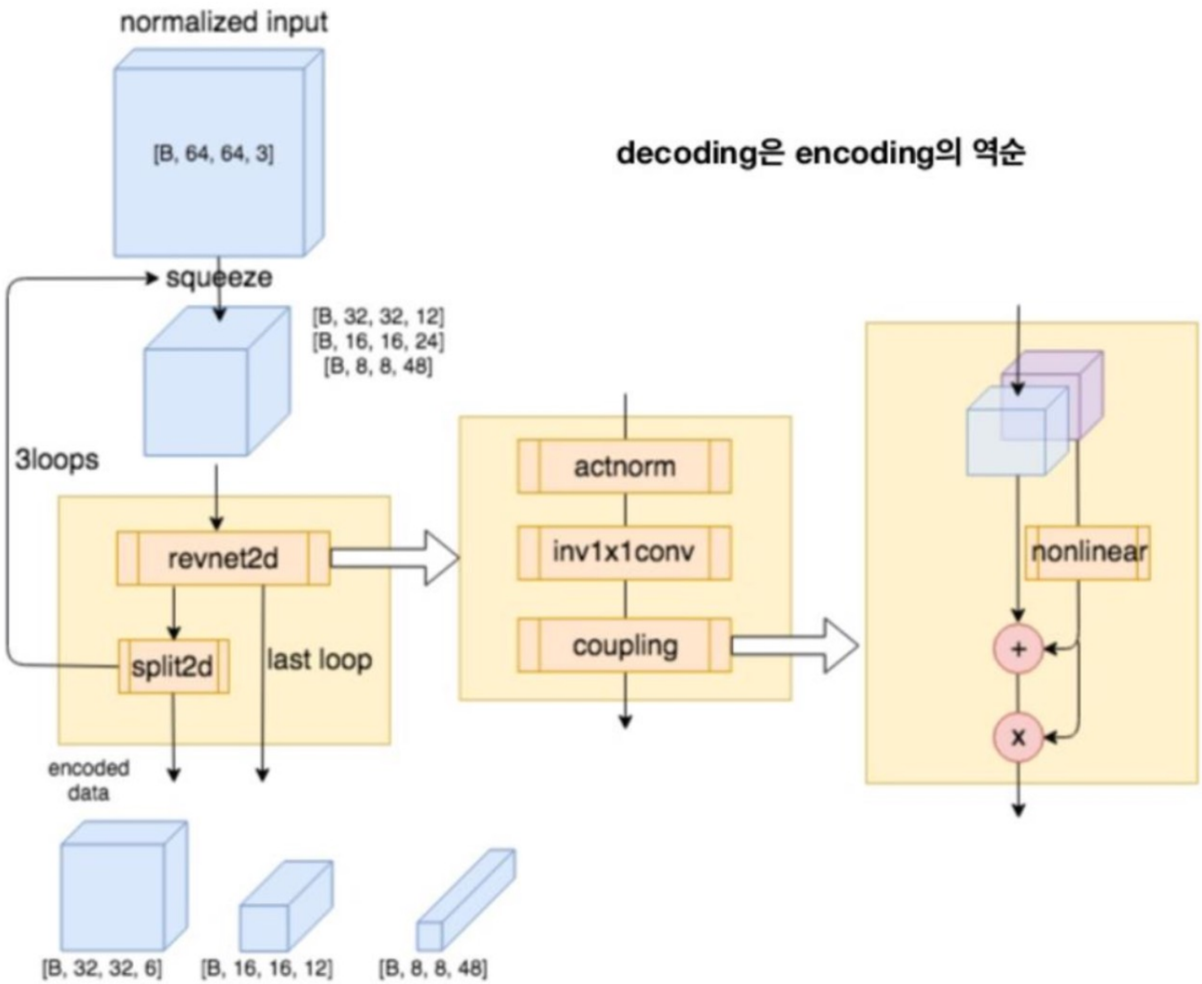
Yes. It is not hard to get the Jacobian matrix and determinant of this transformation. The Jacobian is a lower triangular matrix.

$$\mathbf{J} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0}_{d\times(D-d)} \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}} & \text{diag}(\exp(s(\mathbf{x}_{1:d}))) \end{bmatrix}$$

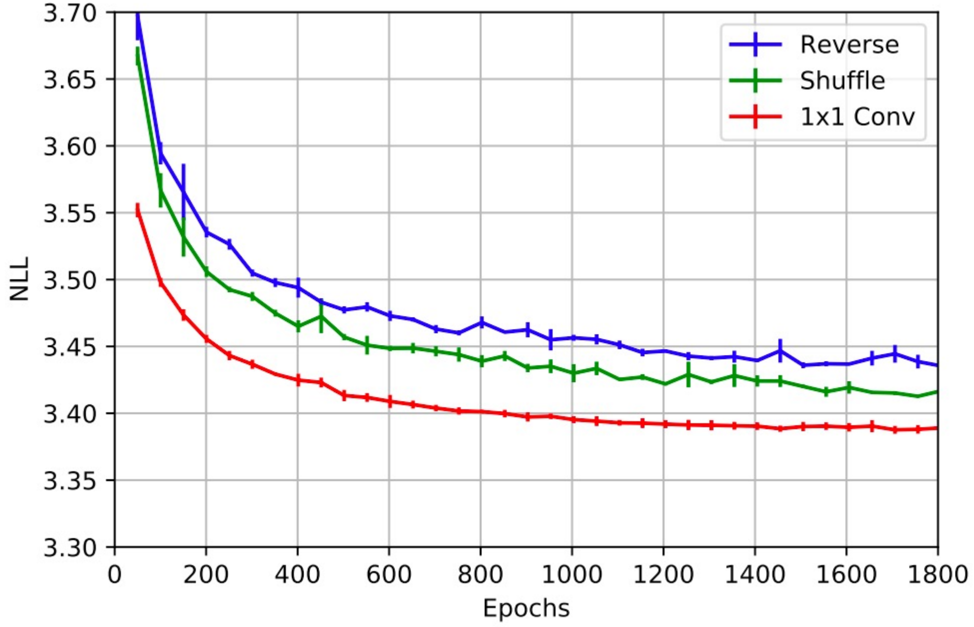
Hence the determinant is simply the product of terms on the diagonal.

$$\det(\mathbf{J}) = \prod_{j=1}^{D-d} \exp(s(\mathbf{x}_{1:d}))_j = \exp(\sum_{j=1}^{D-d} s(\mathbf{x}_{1:d})_j)$$

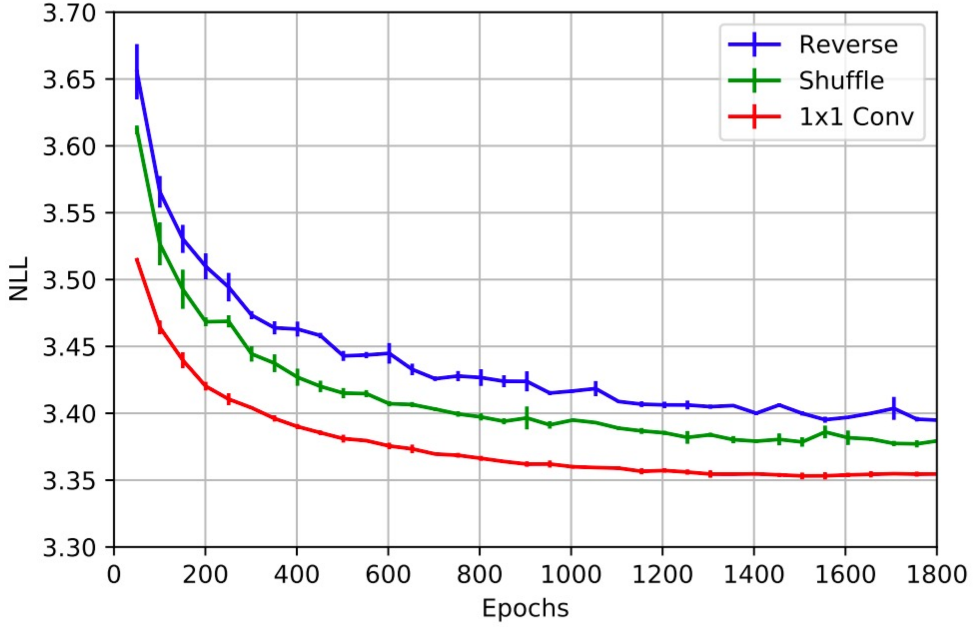
Architecture



Results



(a) Additive coupling.



(b) Affine coupling.

Results

Figure 5: Linear interpolation in latent space between real images

Results

(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

References

- https://minsuksung-ai.tistory.com/12
- https://ratsgo.github.io/generative%20model/2018/01/29/NF/