NF 1: Flow based Models

CS 598: Deep Generative and Dynamical Models

Instructor: Arindam Banerjee

September 30, 2021

Non-linear Functions of Independent Components

- Base distribution over h has independent components

$$p_H(\mathrm{h}) = \prod_{i=1}^{d} p_{H_d}(h_d)$$

- Observed distribution over x
  - Same dimensionality as h, can define invertable maps $x = g(h)$
  - Change of variables $h = f(x)$, so that $g(\cdot) = f^{-1}(\cdot)$

$$p_X(\mathrm{x}) = p_H(f(\mathrm{x})) \left| \det \frac{\partial f(\mathrm{x})}{\partial \mathrm{x}} \right|$$

- Sampling:

$$\mathrm{h} \sim p_H(\mathrm{h})$$
$$\mathrm{x} \sim f^{-1}(\mathrm{h})$$

Invertable Transformations, Change of Variables

- Two key requirements for $\mathrm{x} = f(\mathrm{h})$:
  - f should be easy to invert, i.e., $\mathrm{h} = g(\mathrm{x}) = f^{-1}(\mathrm{x})$
  - Jacobian $\frac{\partial f(\mathrm{x})}{\partial \mathrm{x}} \in \mathbb{R}^{d \times d}$ should be easy to compute
- Core idea: By dimension splitting, for some complex $m(\cdot)$

$$\begin{aligned} \mathrm{h}_1 &= \mathrm{x}_1 \\ \mathrm{h}_2 &= \mathrm{x}_2 + m(\mathrm{x}_1) \end{aligned}$$

- Inverse is easy, Jacobian is 1

$$\begin{aligned} \mathrm{x}_1 &= \mathrm{h}_1 \\ \mathrm{x}_2 &= \mathrm{h}_2 - m(\mathrm{h}_1) \end{aligned}$$

Recall "Normalizing Flows"

- Transforming r.v.s with smooth invertable functions $f : \mathbb{R}^d \mapsto \mathbb{R}^d$

$$\mathrm{z}' = f(\mathrm{z}) \qquad \Rightarrow \qquad \mathrm{z} = f^{-1}(\mathrm{z}') = g(\mathrm{z}')$$

- Density of the transformed variable

$$q(\mathrm{z}') = q(\mathrm{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathrm{z}'} \right| = q(\mathrm{z}) \left| \det \frac{\partial f}{\partial \mathrm{z}} \right|^{-1}$$

  - Change of variables, volume changes by the (abs) determinant
  - Jacobian $\frac{\partial f}{\partial \mathrm{z}}$ is a $d \times d$ matrix
  - Theorem 10.9, W. Rudin, Principles of Mathematical Analysis

- Apply multiple such transformations

$$\mathrm{z}_K = f_K(f_{K-1}(\cdots f_1(\mathrm{z}_0)))$$

$$\log q_K(\mathrm{z}_K) = \log q_0(\mathrm{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial \mathrm{z}_{k-1}} \right|$$

Non-linear Independent Component Estimation (NICE)

- Components of $\mathrm{h} = f(\mathrm{x})$ are independent

$$[h_1 \;\; h_2 \;\; \cdots h_D] = [f_1(\mathrm{x}) \;\; f_2(\mathrm{x}) \;\; \cdots f_D(\mathrm{x})]$$

- Likelihood of observed data

$$\log p_X(\mathrm{x}) = \log p_H(f(\mathrm{x})) + \log\left(\left|\det \frac{\partial f(\mathrm{x})}{\partial \mathrm{x}}\right|\right)$$

$$= \sum_{d=1}^{D} \log p_{H_d}(f_d(\mathrm{x})) + \log\left(\left|\det \frac{\partial f(\mathrm{x})}{\partial \mathrm{x}}\right|\right)$$

- Encoder-Decoder Perspective
  - f: encoder, f^{-1}: decoder, inverse of the encoder
  - Sampling is easy, using $f^{-1} : H \mapsto X$
  - Likelihood computation is exact
  - Dimensionality of h and x are the same

Coupling Layer II

- General coupling: Split dimensions $[I_1 \;\; I_2], |I_1| = d$

$$\mathrm{h}_{I_1} = \mathrm{x}_{I_1}$$
$$\mathrm{h}_{I_2} = g(\mathrm{x}_{I_2}; m(\mathrm{x}_{I_1}))$$

  - Coupling $g : \mathbb{R}^{D-d} \times m(\mathbb{R}^d) \mapsto \mathbb{R}^{D-d}$
  - g is invertible w.r.t. first argument given the second
- Jacobian

$$\frac{\partial \mathrm{h}}{\partial \mathrm{x}} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial \mathrm{h}_{I_2}}{\partial \mathrm{x}_{I_1}} & \frac{\partial \mathrm{h}_{I_2}}{\partial \mathrm{x}_{I_2}} \end{bmatrix}$$

  - We have $\det \frac{\partial \mathrm{h}}{\partial \mathrm{x}} = \det \frac{\partial \mathrm{h}_{I_2}}{\partial \mathrm{x}_{I_2}}$
- For simplicity, use:

$$\mathrm{h}_{I_2} = \mathrm{x}_{I_2} + m(\mathrm{x}_{I_1}) \quad \Rightarrow \quad \mathrm{x}_{I_2} = \mathrm{h}_{I_2} - m(h_{I_1})$$

Rescaling, Prior Distribution

- Rescaling
  - Determinant of Jacobian is 1 is the basic setup
  - Multiple coupling layers also have determinant 1
  - Consider diaginal scaling matrix S_{dd}, likelihood

$$\log p_X(\mathrm{x}) = \sum_{d=1}^{D} \left[\log p_{H_d}(f_d(\mathrm{x})) + \log(|S_{dd}|) \right]$$

- Prior distribution, component-wise independent $p_H(\mathrm{h}) = \prod_d p_{H_d}(h_d)$
  - Normal distribution, on $\mathbb{R}$

$$\log p_{H_d} = -\frac{1}{2}(h_d^2 + \log(2\pi))$$

  - Logistic distribution, on $\mathbb{R}$

$$\log p_{H_d} = -h_d - 2\log(1 + \exp(-h_d))$$

Results: Log-Likelihood

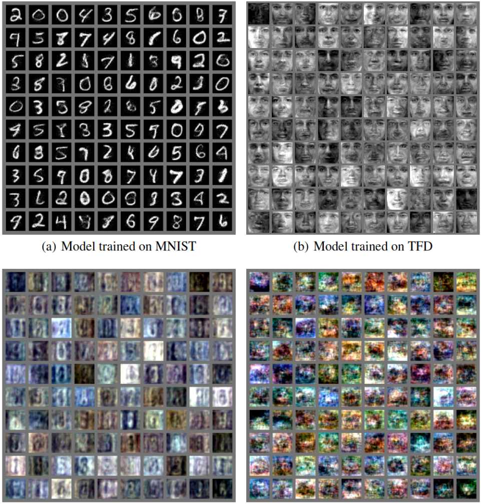
Dataset	MNIST	TFD	SVHN	CIFAR-10
# dimensions	784	2304	3072	3072
Preprocessing	None	Approx. whitening	ZCA	ZCA
# hidden layers	5	4	4	4
# hidden units	1000	5000	2000	2000
Prior	logistic	gaussian	logistic	logistic
Log-likelihood	1980.50	5514.71	11496.55	5371.78

Figure 3: Architecture and results. # hidden units refer to the number of units per hidden layer.

Model	TFD	CIFAR-10
NICE	5514.71	5371.78
Deep MFA	5250	3622
GRBM	2413	2365

Figure 4: Log-likelihood results on TFD and CIFAR-10. Note that the Deep MFA number correspond to the best results obtained from (Tang et al., 2012) but are actually variational lower bound.

Results: Samples

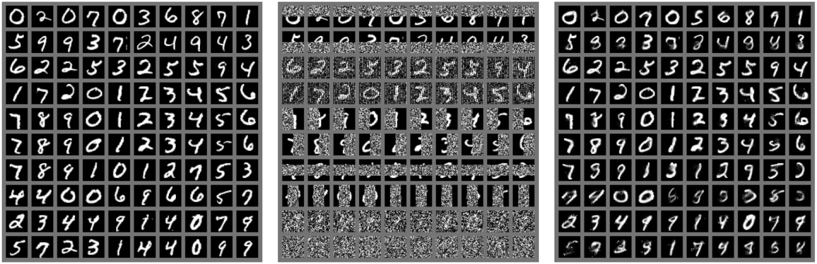


(a) Model trained on MNIST (b) Model trained on TFD

(c) Model trained on SVHN (d) Model trained on CIFAR-10

Figure 5: Unbiased samples from a trained NICE model. We sample $h \sim p_H(h)$ and we output $x = f^{-1}(h)$.

Results: In-Painting



(a) MNIST test examples (b) Initial state (c) MAP inference of the state

Figure 6: Inpainting on MNIST. We list below the type of the part of the image masked per line of the above middle figure, from top to bottom: top rows, bottom rows, odd pixels, even pixels, left side, right side, middle vertically, middle horizontally, 75% random, 90% random. We clamp the pixels that are not masked to their ground truth value and infer the state of the masked pixels by projected gradient ascent on the likelihood. Note that with middle masks, there is almost no information available about the digit.

Coupling Layers III: Affine Coupling Layers

- Affine coupling layer, with functions $s, t : \mathbb{R}^d \mapsto \mathbb{R}^{D-d}$

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

- Inverse function

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

- Jacobian

$$
\frac{\partial \mathrm{h}}{\partial \mathrm{x}} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial \mathrm{h}_{d+1:D}}{\partial \mathrm{x}_{1:d}} & \operatorname{diag}(\exp[s(\mathrm{x}_{1:d})]) \end{bmatrix}
$$

- Determinant is $\exp[\sum_{j=d+1}^{D} s(\mathrm{x}_{1:d})_j]$

Partitioning: Masked Convolution

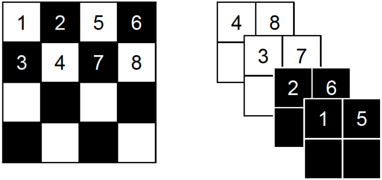


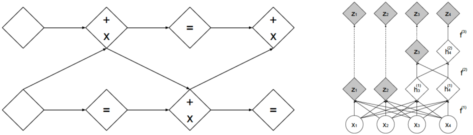
Figure 3: Masking schemes for affine coupling layers. On the left, a spatial checkerboard pattern mask. On the right, a channel-wise masking. The squeezing operation reduces the $4 \times 4 \times 1$ tensor (on the left) into a $2 \times 2 \times 4$ tensor (on the right). Before the squeezing operation, a checkerboard pattern is used for coupling layers while a channel-wise masking pattern is used afterward.

- Partition with binary mask b

$$\mathrm{h} = \mathrm{b} \odot \mathrm{x} + (1 - \mathrm{b}) \odot (\mathrm{x} \odot \exp(s(\mathrm{b} \odot \mathrm{x})) + t(b \odot x))$$

- Spatial checkerboard masks and channel-wise masks
- $s(\cdot), t(\cdot)$ are conv nets
- Combine coupling layers with alternating patterns

Multi-scale Architecture



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

(b) Factoring out variables. At each step, half the variables are directly modeled as Gaussians, while the other half undergo further transformation.

Figure 4: Composition schemes for affine coupling layers.

- Layers: 3 alternating checkerboard masks, squeezing $(a \times a \times c \mapsto a/2 \times a/2 \times 4c)$, 3 alternating channel-wise masks
- Factor out half the dimensions, $f^{(i)}$ is couple-squeeze-couple

$$
\begin{aligned}
\mathrm{h}^{(0)} &= \mathrm{x} \\
(\mathrm{z}^{(i+1)}, \mathrm{h}^{(i+1)}) &= f^{(i+1)}(\mathrm{h}^{(i)}) \\
\mathrm{z}^{(L)} &= f^{(L)}(\mathrm{h}^{(L-1)}) \\
\mathrm{z} &= (\mathrm{z}^{(1)}, \cdots, \mathrm{z}^{(L)})
\end{aligned}
$$

- Multi-scale factoring out to Gaussians

De-quantization, Bits/Dimension, etc.

- De-quantization: Images $\mathrm{y} \in \{0, \dots, 255\}^D$
  - Create noisy image: $\mathrm{z} = \mathrm{y} + \mathrm{u}$, $\mathrm{u} \in [0, 1[^D$, density $p(\mathrm{z})$ on $[0, 256]^D$
  - Fit models $q(\mathrm{z})$ based on z
  - Can bound $\mathbb{E}_{p(\mathrm{z})}[\log q(\mathrm{z})]$ with $\mathbb{E}_{P(\mathrm{y})}[\log Q(\mathrm{y})]$
- Model $x = \mathrm{logit}(\alpha + (1-\alpha) \odot \mathrm{z}/256)$, small α
  - Recall: $b = \mathrm{Logit}(a) = \log \frac{a}{1-a}$, inverse $a = \sigma(b) = \frac{1}{1+\exp(-b)}$
- Convert density back to image space (verify)

$$p(\mathrm{z}) = p(\mathrm{x}) \left(\frac{1-\alpha}{256} \right)^D \left(\prod_{i=1}^{D} \sigma(x_i)(1 - \sigma(x_i) \right)^{-1}$$

- Results based on bits/dimension

$$\begin{aligned} b(\mathrm{x}) &= -\frac{1}{D} \log_2 p(\mathrm{z}) \\ &= -\frac{\log p(\mathrm{x})}{D \log 2} + \frac{1}{D} \sum_i [\log_2 \sigma(x_i) + \log_2(1 - \sigma(x_i)] + c(\alpha) \end{aligned}$$

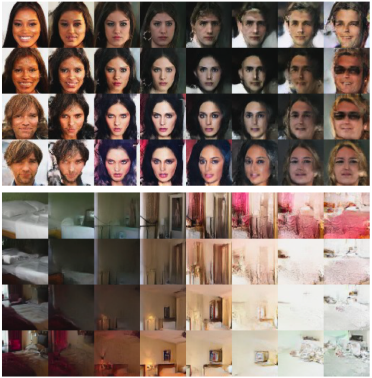
Results: Bits/Dimension

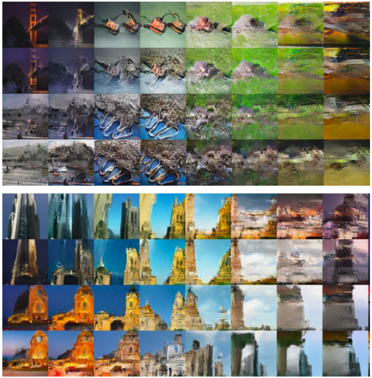
Dataset	PixelRNN [46]	Real NVP	Conv DRAW [22]	IAF-VAE [34]
CIFAR-10	3.00	3.49	< 3.59	< 3.28
Imagenet (32×32)	3.86 (3.83)	4.28 (4.26)	< 4.40 (4.35)	
Imagenet (64×64)	3.63 (3.57)	3.98 (3.75)	< 4.10 (4.04)	
LSUN (bedroom)		2.72 (2.70)		
LSUN (tower)		2.81 (2.78)		
LSUN (church outdoor)		3.08 (2.94)		
CelebA		3.02 (2.97)		

Table 1: Bits/dim results for CIFAR-10, Imagenet, LSUN datasets and CelebA. Test results for CIFAR-10 and validation results for Imagenet, LSUN and CelebA (with training results in parenthesis for reference).

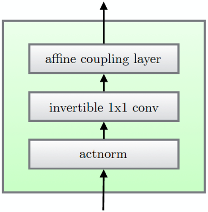
Results: Samples

Results: Latent Space Interpolation

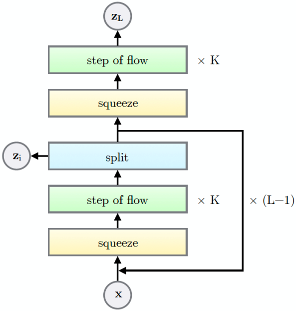




Glow: Generative Flow with 1×1 Convolution



(a) One step of our flow.



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

Glow Details

Table 1: The three main components of our proposed flow, their reverses, and their log-determinants. Here, $\mathbf{x}$ signifies the input of the layer, and $\mathbf{y}$ signifies its output. Both $\mathbf{x}$ and $\mathbf{y}$ are tensors of shape $[h \times w \times c]$ with spatial dimensions (h, w) and channel dimension c. With (i, j) we denote spatial indices into tensors $\mathbf{x}$ and $\mathbf{y}$. The function $\mathtt{NN}()$ is a nonlinear mapping, such as a (shallow) convolutional neural network like in ResNets (He et al., 2016) and RealNVP (Dinh et al., 2016).

Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i, j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i, j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$
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))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathtt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathtt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\mathtt{sum}(\log(\lvert\mathbf{s}\rvert))$

Invertible 1×1 Convolutions

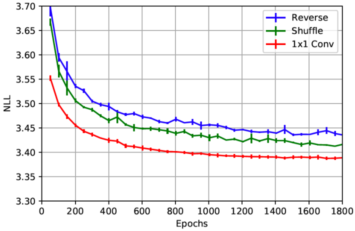
- Prior work: Fixed permutation over channels
- Generalization
  - Initialize with random rotation matrix ($\log \det = 0$)
  - Use $c \times c$ convolution W for each spatial location (i, j)
- Computation of $\det W$ is c^3, use LU decomposition

$$W = PL(U + \mathrm{diag}(s))$$

  - P is a permutation matrix, kept fixed
  - L is lower triangular with ones in the diagonal
  - U is upper triangular with zeros in the diagonal

$$\log |\det W| = \sum_{j=1}^{c} \log |s_j|$$

Results: Training, Bits/Dimension



(a) Additive coupling. (b) Affine coupling.

Figure 3: Comparison of the three variants - a reversing operation as described in the RealNVP, a fixed random permutation, and our proposed invertible 1×1 convolution, with additive (left) versus affine (right) coupling layers. We plot the mean and standard deviation across three runs with different random seeds.

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**

Results: Samples

Figure 4: Random samples from the model, with temperature 0.7

Results: Latent Space Interpolation

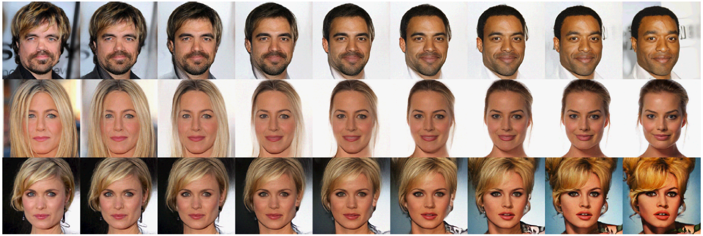
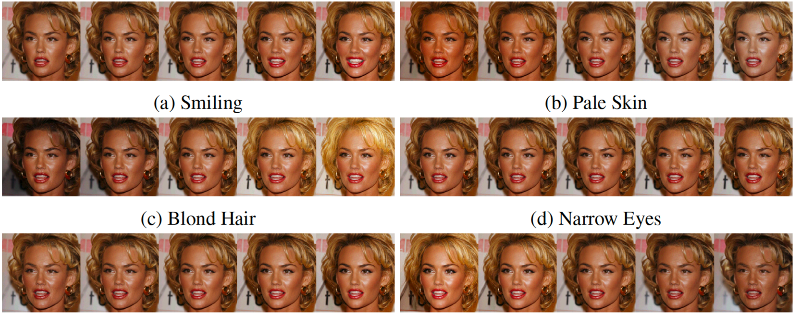


Figure 5: Linear interpolation in latent space between real images



(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

Results: Effect of Temperature

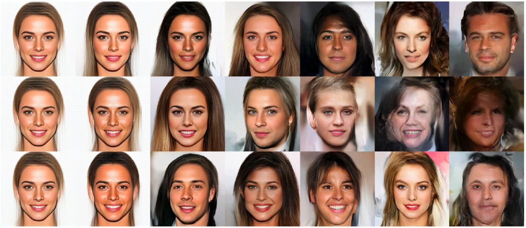


Figure 8: Effect of change of temperature. From left to right, samples obtained at temperatures $0, 0.25, 0.6, 0.7, 0.8, 0.9, 1.0$

- Additive coupling layers: Multiply standard deviation of $p_\theta(z)$ by T
- Distribution at temperature T: $p_{\theta,T}(x) \propto (p_\theta(x))^{1/T^2}$ [check!]

Results: Shallow vs. Deep

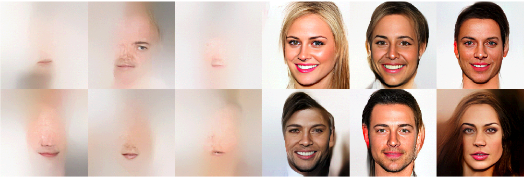


Figure 9: Samples from shallow model on left vs deep model on right. Shallow model has $L = 4$ levels, while deep model has $L = 6$ levels

References

- L. Dinh, D. Krueger, Y. Bengio. NICE: Non-linear Independent Components Estimation. ICLR Workshop, 2015.
- L. Dinh, J. Sohl-Dickstein, S. Bengio. Density Estimation using Real NVP. ICLR, 2017.
- D. P. Kingma, P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. NeurIPS, 2018.
- L. Theis, A. van den oord, M. Nethe, A note on the evaluation of generative models. ICLR, 2016.
- G. Papamakarios, T. Pavlakou, I. Murray, Masked Autoregressive Flow for Density Estimation. NIPS, 2017.