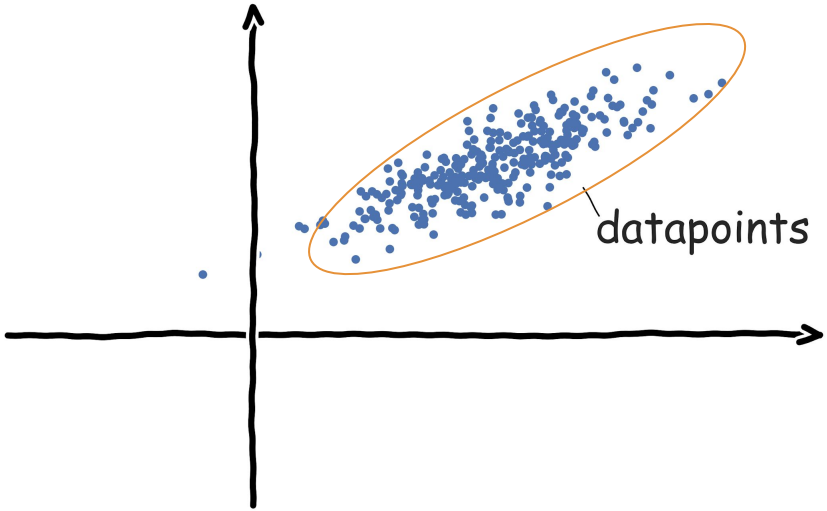
Normalizing Flows

Arsenii Ashukha
PhD Candidate at Samsung AI Center Moscow

Probabilistic View on Generative models

Generative models – aim to learn a data distribution.



Data points are vectors $x \in \mathbb{R}^2$

Goal is to learn a model so that:

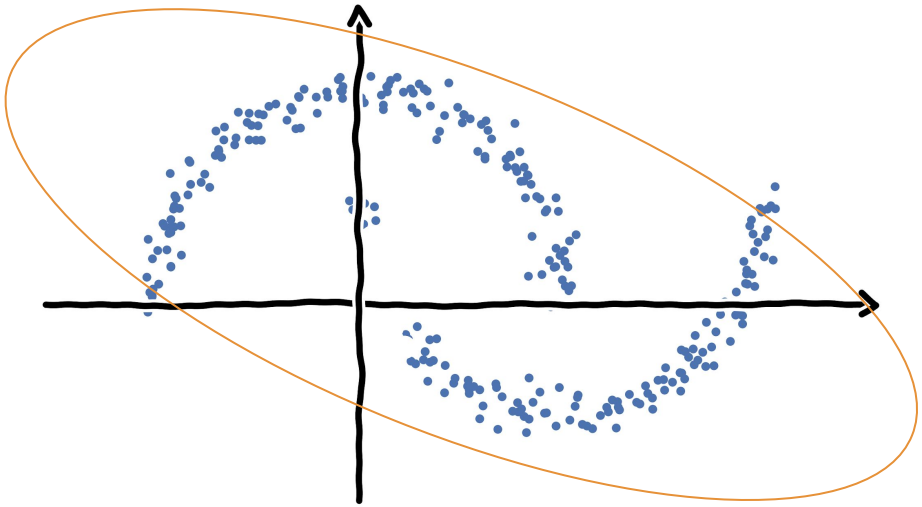
$$p_{model}(x) \approx p_{data}(x)$$

Reasonable model for the case is 2d-*Gaussian* distribution :

$$p_{model} = \mathcal{N}(x \,|\, \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$$

Probabilistic View on Generative models

Generative models – aim to learn a data distribution.



Data points are vectors $x \in \mathbb{R}^2$

Goal is to learn a model so that:

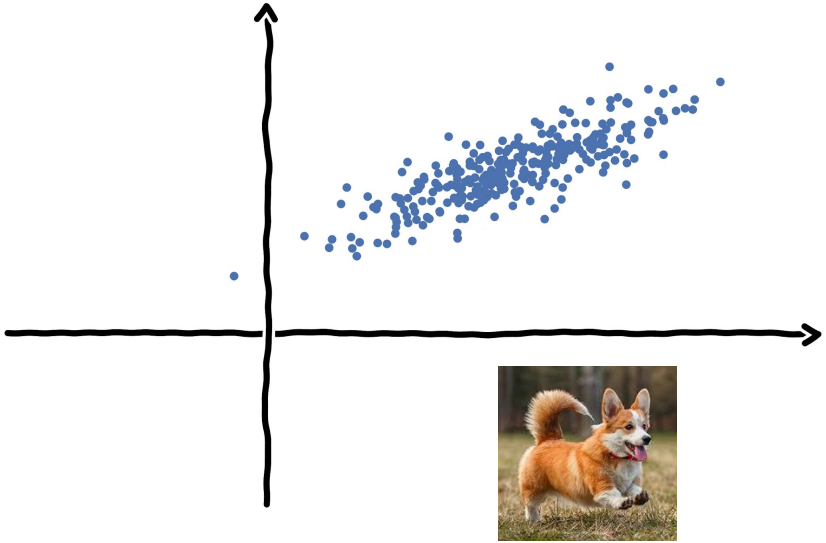
$$p_{model}(x) \approx p_{data}(x)$$

Reasonable model for the case is 2d-*Gaussian* distribution :

$$p_{model} = \mathcal{N}(x \,|\, \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$$

The model mismatches the data :(

Ok, but what do we need this for?



1. Generation of new objects:
    a. data augmentation
    b. generation of private data for learning
    c. specific problems: translation / text2speech, superresolution, ...
2. Probability estimate (out-of-distribution detection)
3. Representation learning (semi-supervised learning)
4. ....

Gradient-based training of probabilistic GMs

Get samples from target data distribution (dataset):

$$X = (x_1, \dots, x_N), \quad x_i \sim p_{\text{data}}(x)$$

Define a parametric model:

$$p_\theta(x)$$

- assume that a model is suitable for gradient optimization by θ
- a model gives a p.d.f. value for any point x (sum-ups to 1, nonnegative)
- desirable to have a tractable sampling procedure

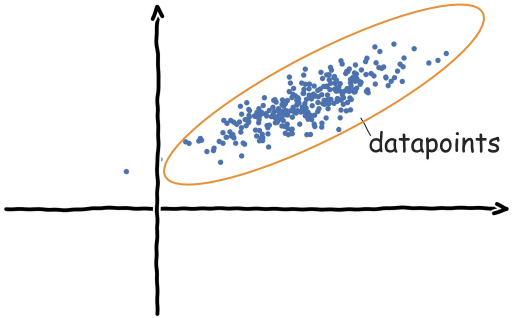
Train a model by Maximum likelihood:

$$\log p_\theta(X) = \sum_i \log p_\theta(x_i) \to \max_\theta$$

can be solved by SGA or Adam-like methods

Gradient-based training of probabilistic GMs

Get samples from target data distribution (dataset):



Define a parametric model:

$$p_\theta(x) = \mathcal{N}(x \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad \theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Train a model by Maximum likelihood:

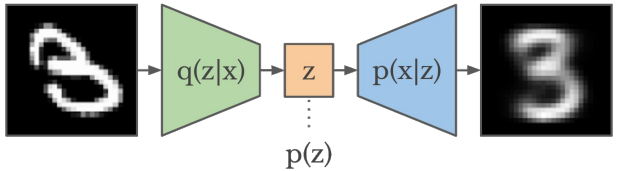
$$\sum_i \log \mathcal{N}(x_i \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \to \max_{\boldsymbol{\mu}, \boldsymbol{\Sigma}}$$

```
μ = [2 2]
Σ =
[[ 0.4  0.2]
 [ 0.2  0.2]]
```

Gradient based training of probabilistic GMs

How to choose a model for more complex data?

Variational autoencoders (semi-implicit models)



$$\log p_\theta(x) = \log \mathbb{E}_{p(z)}\ p_\theta(x|z)$$

$$\log p_\theta(X) \to \max_\theta$$

$$\log p_\theta(X) \geq \mathcal{L}(X, \theta) \to \max_\theta$$

- But can we define an expressive and tractable pdf based on DNNs?

- Yes!

$p_\theta(x)$ = Normalizing Flow :)

- Expressive parametric model
- Tractable computation/optimization of an exact $p_\theta(x)$
- Fast and tractable sampling

pic: https://danijar.com/building-variational-auto-encoders-in-tensorflow/

Change of variables formula

1d-case:

Given: $p(z) \quad z = f(x)$ What is $p(x)$?

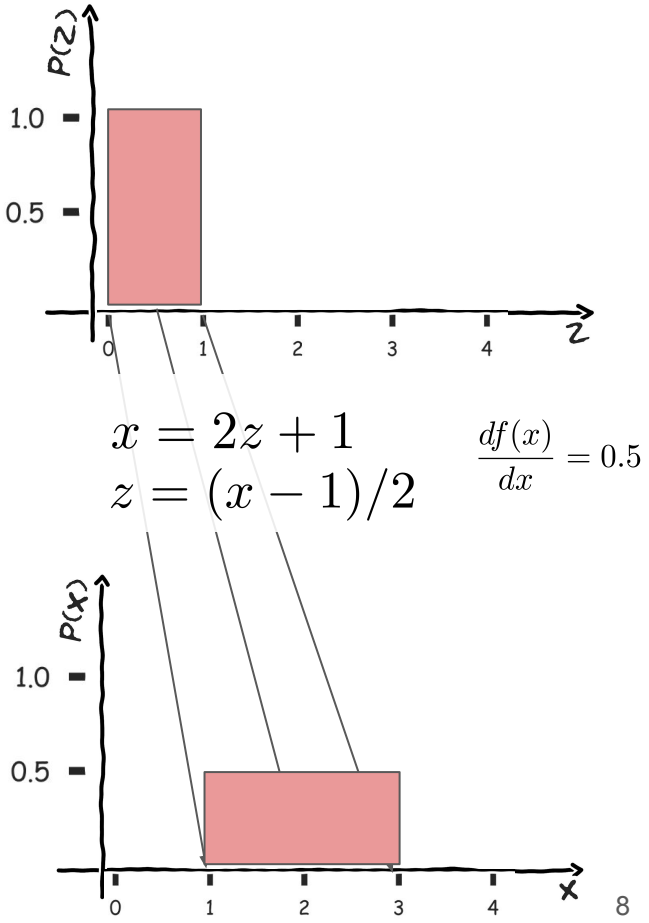
$$p(x)|dx| = p(z)|dz| \qquad p(x) = p(z)\left|\frac{dz}{dx}\right|$$

$$p(x) = p(f(x))\left|\frac{df(x)}{dx}\right|$$

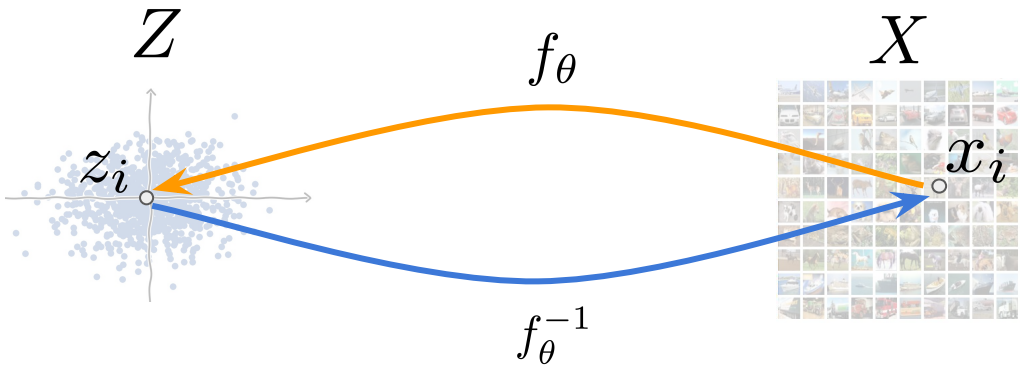
Nd-case:

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

$$\left(\frac{\partial f(x)}{\partial x^T}\right)_{ij} = \frac{\partial f(x)_i}{\partial x_j^T}$$



Normalizing flows concept

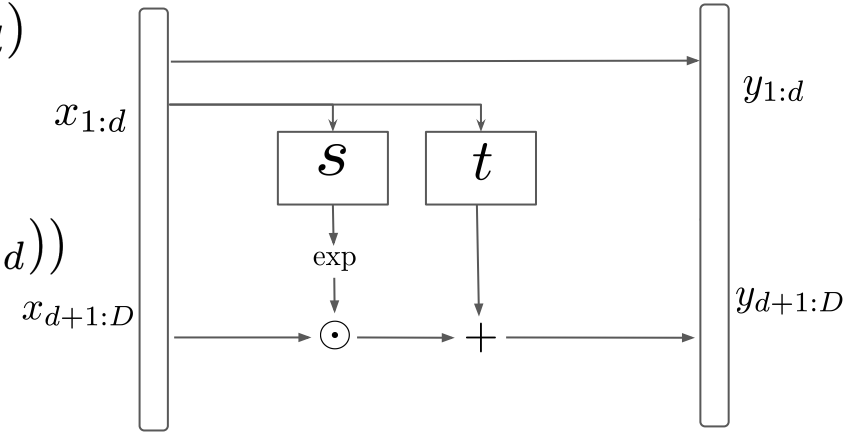


$$p(x) = p(f(x)) \left| \det \frac{\partial f(x)}{\partial x^T} \right|$$

Coupling layer

Forward:

$$y_{1:d} = x_{1:d}$$
$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d})$$



Inverse:

$$x_{1:d} = y_{1:d}$$
$$x_{d+1:D} = (y_{d+1:D} - t(y_{1:d})) \odot \exp(-s(y_{1:d}))$$

Jacobian:

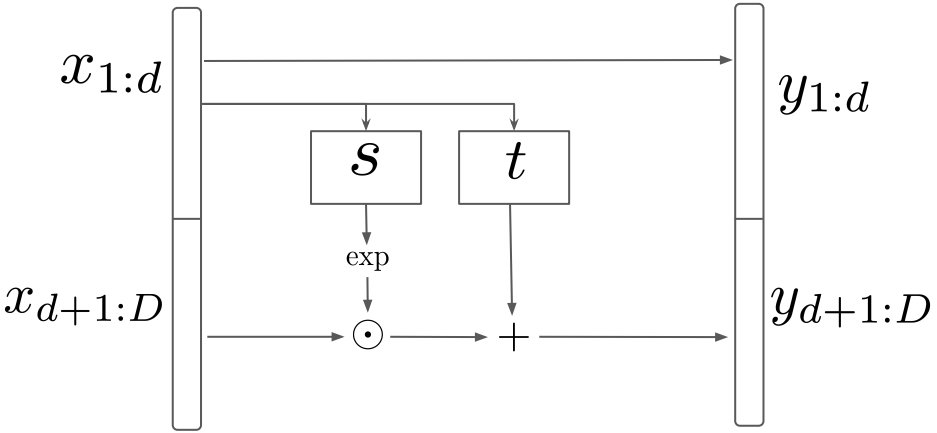
$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} & \\ & \end{bmatrix}$$

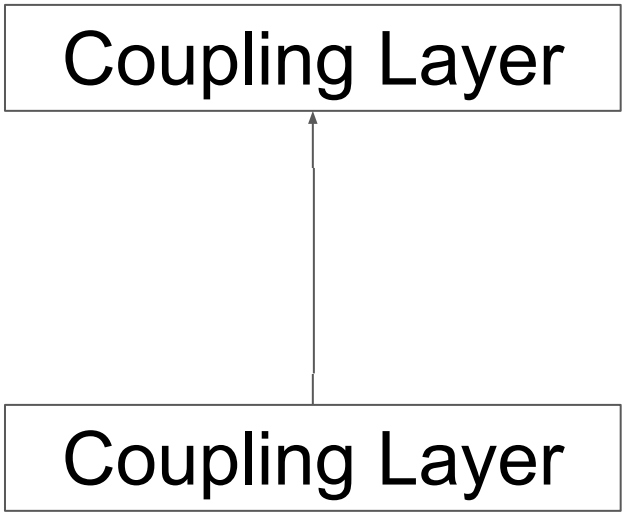
det J:

$$\det \left| \frac{\partial y}{\partial x^T} \right| = \exp \sum_j s(x_{1:d})_j$$

Density estimation using Real NVP, https://arxiv.org/abs/1605.08803

Combining coupling layers





Density estimation using Real NVP, https://arxiv.org/abs/1605.08803

The model

$$f_\theta(x) : \mathbb{R}^D \to \mathbb{R}^D \qquad f_\theta(x) = f^N \circ \cdots \circ f^1(x)$$

$$\log p_\theta(x) = \log p(f_\theta(x)) + \log \left| \det \frac{\partial f_\theta(x)}{\partial x^T} \right| = \log p(f(x)) + \sum_{i=1}^{N} \log \left| \det \frac{\partial f^i}{f^{i-1}} \right|$$

$$\log p_\theta(X) = \sum_i \log p_\theta(x_i) \to \max_\theta$$

Density estimation using Real NVP, https://arxiv.org/abs/1605.08803

Simple data

Inference

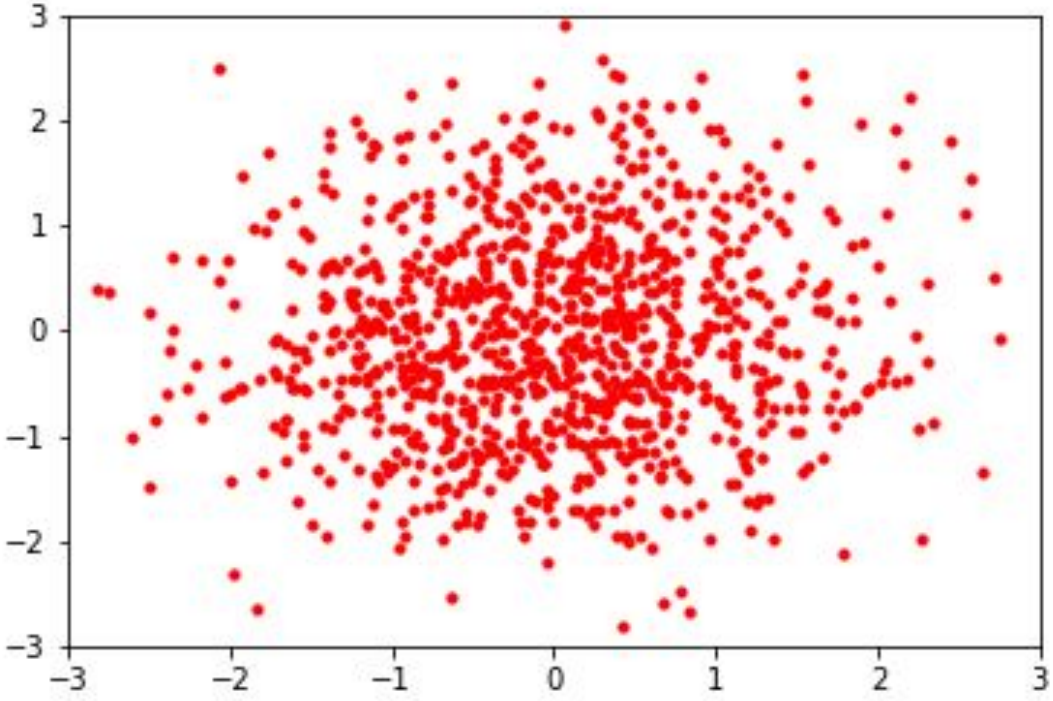
$$x \sim \hat{p}_X$$

$$z = f(x)$$

Density estimation using Real NVP, https://arxiv.org/abs/1605.08803

Simple data

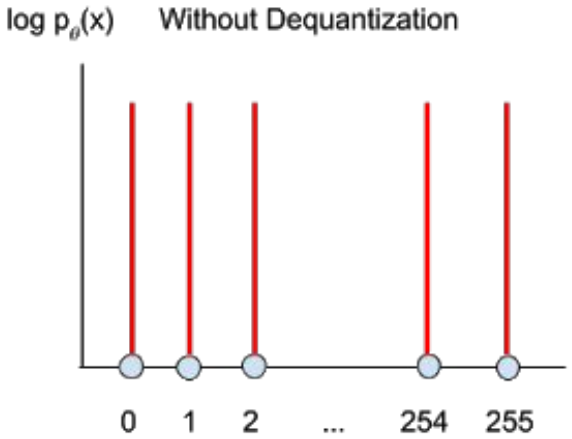
$$f^{-1}(z) = f_1^{-1} \circ \cdots \circ f_N^{-1}(z)$$



pic: https://blog.evjang.com/2019/07/likelihood-model-tips.html

Image generation: Data Dequantization

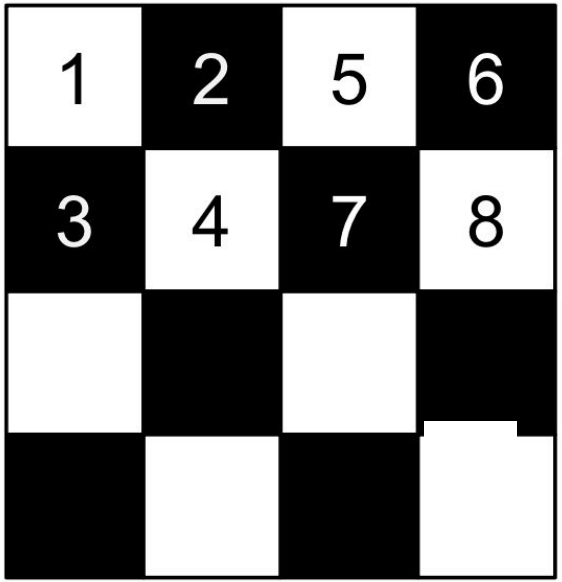
$$x_i \in \{0, \ldots, 255\}$$

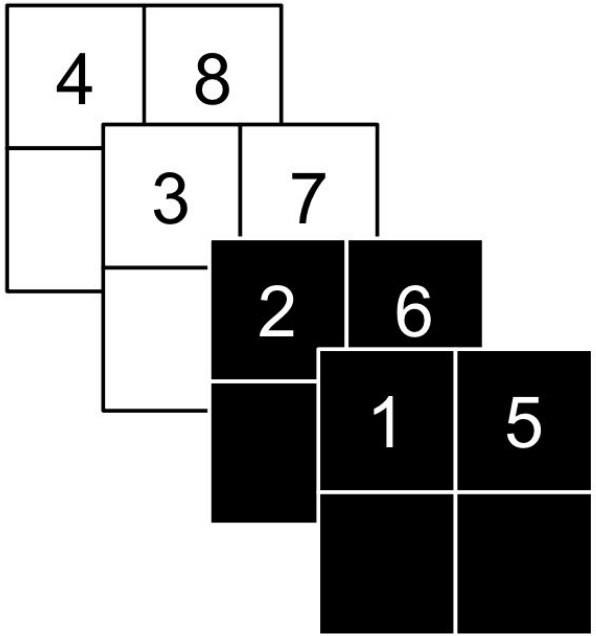


NICE: Non-linear Independent Components Estimation, https://arxiv.org/pdf/1410.8516.pdf pic: https://blog.evjang.com/2019/07/likelihood-model-tips.html

Image generation: Masking

$$y_{1:d} = x_{1:d}$$
$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d})$$





Density estimation using Real NVP, https://arxiv.org/abs/1605.08803

Image results: Real NVP

Image results: Glow

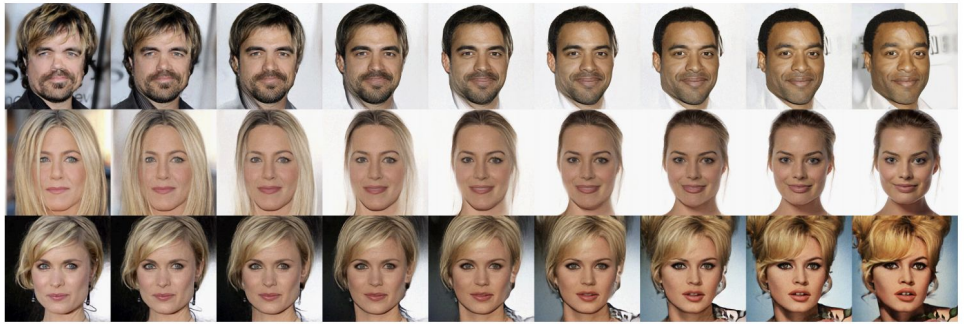
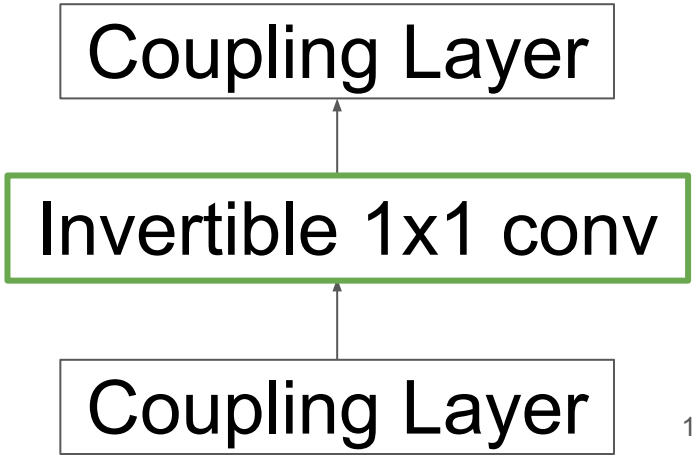


Figure 5: Linear interpolation in latent space between real images



Glow: Generative Flow with Invertible 1×1 Convolutions https://arxiv.org/pdf/1807.03039.pdf

Measuring Quality

Continuous data: average test log-likelihood

$$\frac{1}{|X_{\text{test}}|} \sum_{x \in X_{\text{test}}} \log p_\theta(x)$$

De-quantized data: an estimate of probability (so-called bits-per-dimension)

$$\text{Bits-per-dim}(x) = -\frac{1}{D} \log \hat{P}(x)$$

Measuring Quality: Results

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)		

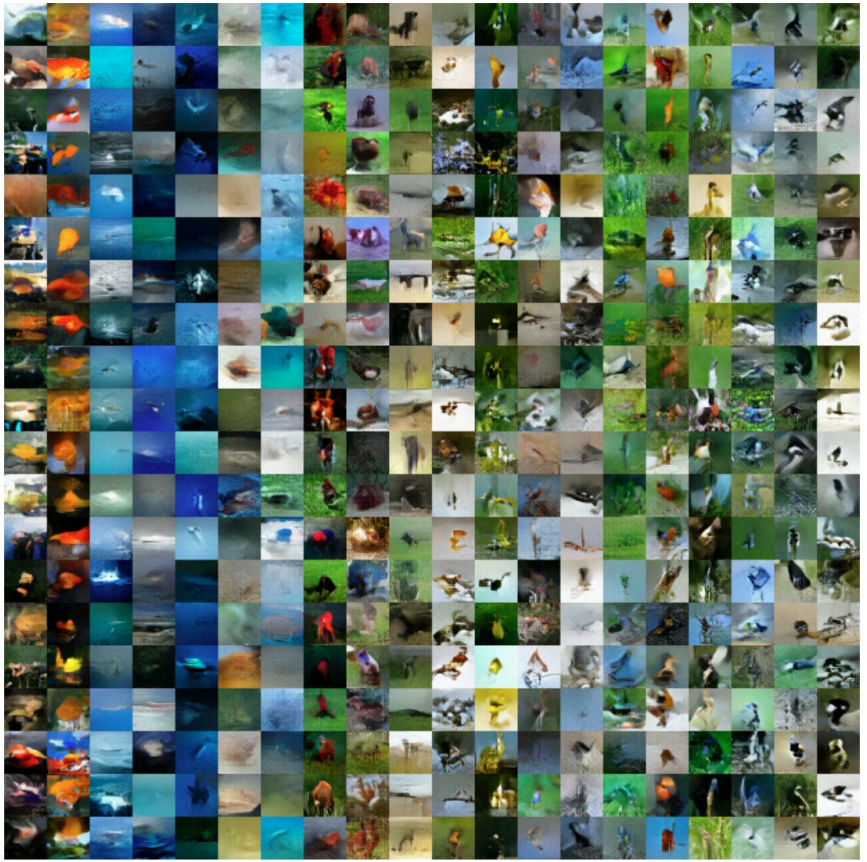
Model	CIFAR-10	ImageNet 32x32	ImageNet 64x64	LSUN (bedroom)	LSUN (tower)	LSUN (church outdoor)
RealNVP	3.49	4.28	3.98	2.72	2.81	3.08
Glow	**3.35**	**4.09**	**3.81**	**2.38**	**2.46**	**2.67**

Conditional Normalizing Flows

$$D = \{(x, y)\}_{i=0,\ldots,N} \qquad p(x, y) = p(x|y)p(y)$$

$$\log p(x \mid y) = \log p(f(x) \mid y) + \log \left| \det \frac{\partial f_y(x)}{\partial x^T} \right|$$

$$s(x_{1:d}), t(x_{1:d}) \rightarrow s(x_{1:d}, y), t(x_{1:d}, y)$$



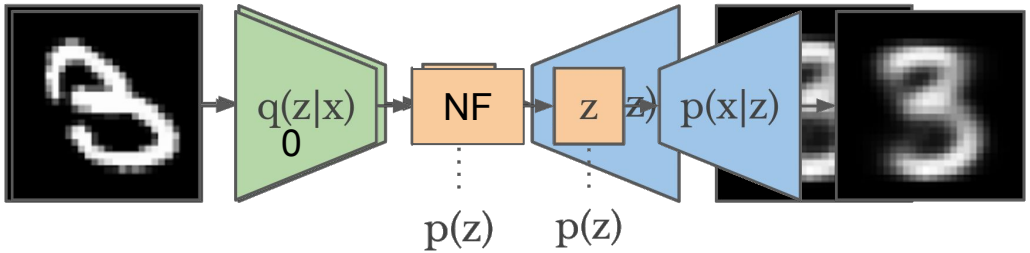
(b) Class conditional 32 × 32 ImageNet samples

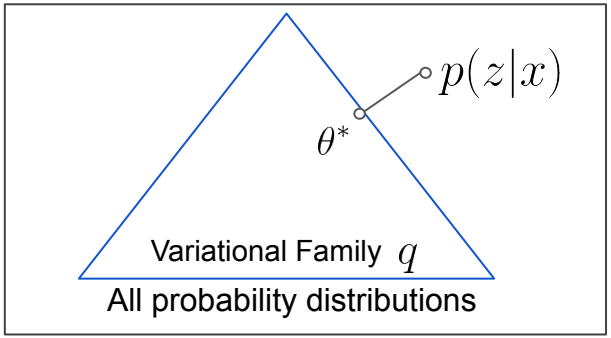
Glow: Generative Flow with Invertible 1×1 Convolutions https://arxiv.org/pdf/1807.03039.pdf

Variational Inference with Normalizing Flows

$$\log p(X) = \mathcal{L}(X, \theta) + KL(q_\theta(z \mid x) \,||\, p(z \mid x))$$

$$N(z \mid \mu(x), \mathrm{diag}(\sigma(x)))$$





Algorithm 1 Variational Inf. with Normalizing Flows

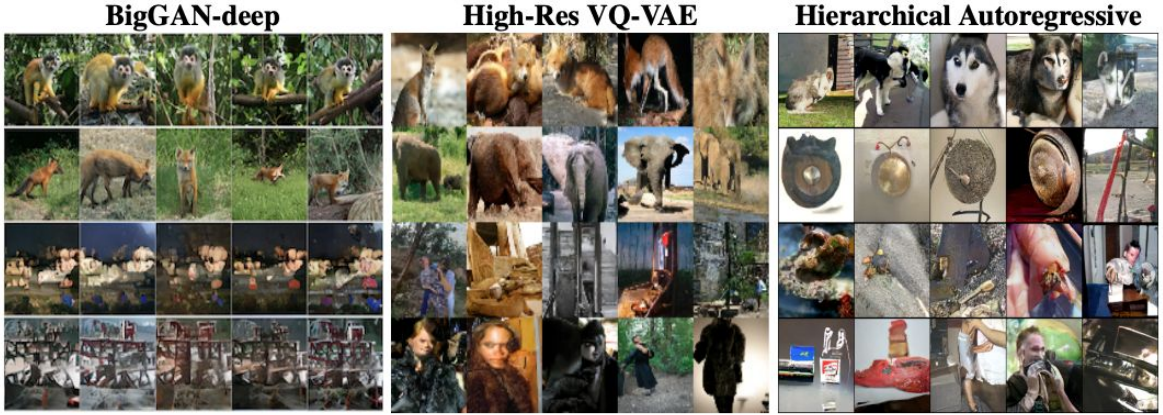
Parameters: ϕ variational, θ generative
while not converged **do**
  $\mathbf{x} \leftarrow$ {Get mini-batch}
  $\mathbf{z}_0 \sim q_0(\bullet|\mathbf{x})$
  $\mathbf{z}_K \leftarrow f_K \circ f_{K-1} \circ \ldots \circ f_1(\mathbf{z}_0)$
  $\mathcal{F}(\mathbf{x}) \approx \mathcal{F}(\mathbf{x}, \mathbf{z}_K)$
  $\Delta\boldsymbol{\theta} \propto -\nabla_\theta \mathcal{F}(\mathbf{x})$
  $\Delta\boldsymbol{\phi} \propto -\nabla_\phi \mathcal{F}(\mathbf{x})$
end while

Variational Inference with Normalizing Flows, https://arxiv.org/abs/1505.05770

Classification Accuracy Score for Conditional Generative Models

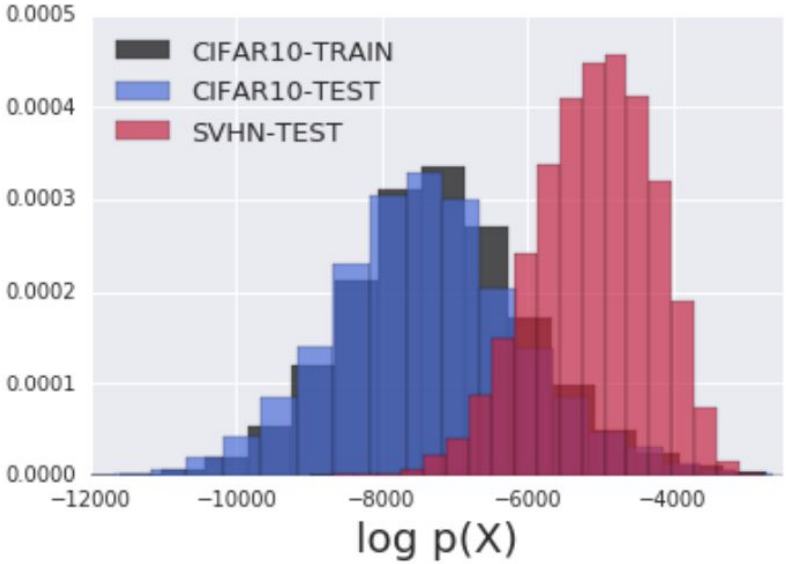
Table 1: CAS for different models at 128×128 and 256×256 resolutions. BigGAN-deep samples taken from best truncation parameter of 1.5.

Training Set	Resolution	Top-5 Accuracy	Top-1 Accuracy	Inception Score	FID-50K
Real	128×128	88.79%	68.82%	165.38 ± 2.84	1.61
BigGAN-deep	128×128	64.44%	40.64%	71.31 ± 1.57	4.22
Hierarchical Autoregressive	128×128	**77.33%**	**54.05%**	17.02 ± 0.79	46.05
Real	256×256	91.47%	73.09%	331.83 ± 5.00	2.47
BigGAN-deep	256×256	65.92%	42.65%	109.39 ± 1.56	11.78
High-Res VQ-VAE	256×256	**77.59%**	**54.83%**	43.44 ± 0.87	38.05



Classification Accuracy Score for Conditional Generative Models, https://arxiv.org/abs/1905.10887

Do Deep Generative Models Know What They Don't Know?



(b) Train on CIFAR-10, Test on SVHN

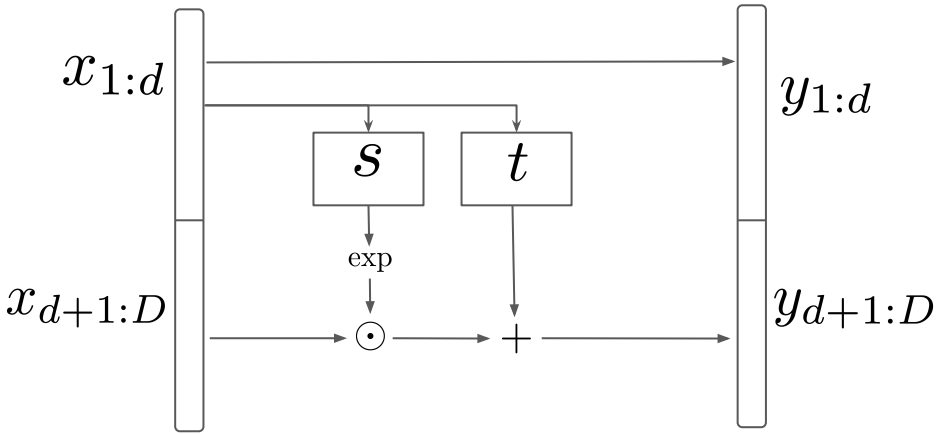
Data Set	Avg. Bits Per Dimension
Glow Trained on CIFAR-10	
CIFAR10-Train	3.386
CIFAR10-Test	3.464

NFs

$$f_\theta(x) = f^N \circ \cdots \circ f^1(x)$$

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

$$\log p_\theta(X) = \sum_i \log p_\theta(x_i) \to \max_\theta$$

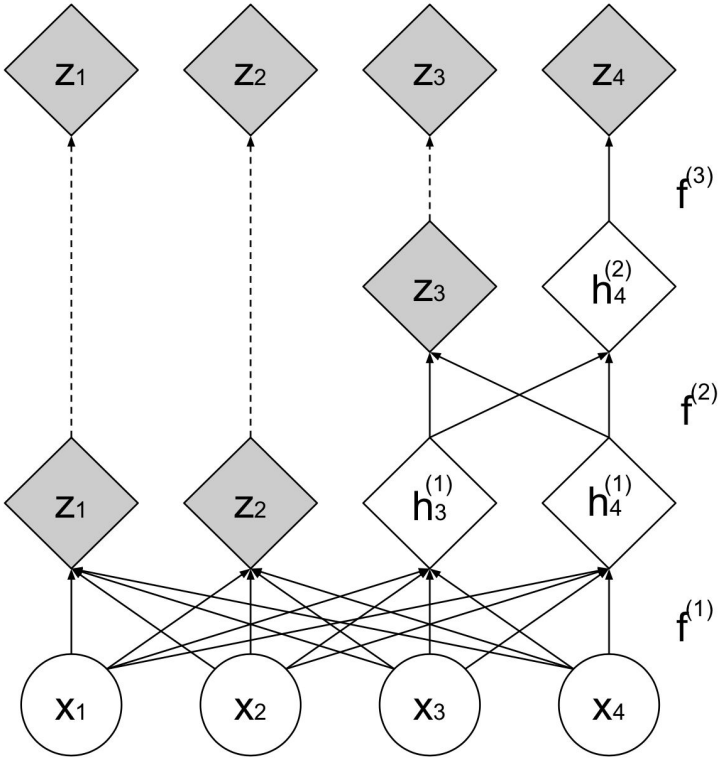


- Expressive parametric model
- Tractable computation/optimization of an exact pθ(x)
- Fast and tractable sampling

Reading List

- Tutorial on normalizing flows, slideslive.com/38917907/tutorial-on-normalizing-flows
- Tips for Training Likelihood Models, blog.evjang.com/2019/07/likelihood-model-tips.html
- FFJORD tutorial, https://youtu.be/_ALdCSSVYkw
- Must read papers:
  - Variational Inference with Normalizing Flows, https://arxiv.org/abs/1505.05770
  - Density estimation using Real NVP, https://arxiv.org/abs/1605.08803
  - Glow: Generative Flow with Invertible 1×1 Convolutions https://arxiv.org/abs/1807.03039
  - Sylvester Normalizing Flows for Variational Inference, https://arxiv.org/abs/1803.05649
  - FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models, https://arxiv.org/abs/1810.01367
  - Do Deep Generative Models Know What They Don't Know?, https://arxiv.org/abs/1810.09136
  - Classification Accuracy Score, https://arxiv.org/abs/1905.10887

Image generation: Multi-scale architecture



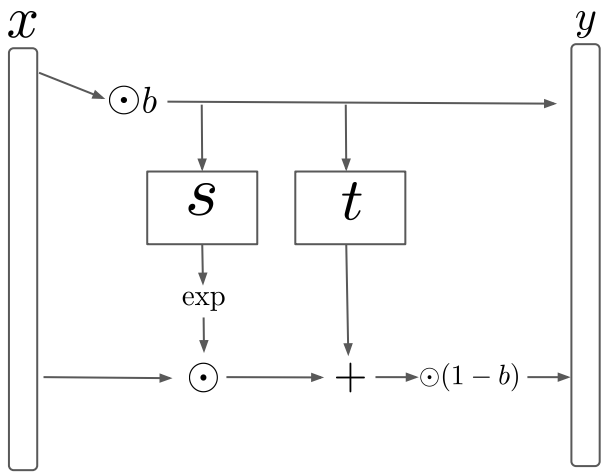
Density estimation using Real NVP, https://arxiv.org/abs/1605.08803

Masked Coupling Layers (MCL)

Coupling Layer

Permutation

$$y = b \odot x + (1 - b) \odot \left(x \odot \exp\left(s(b \odot x)\right) + t(b \odot x) \right)$$



- b is a layer specific binary mask
- b_i = 1: copy
- b_i = 0: transform

Density estimation using Real NVP, https://arxiv.org/abs/1605.08803