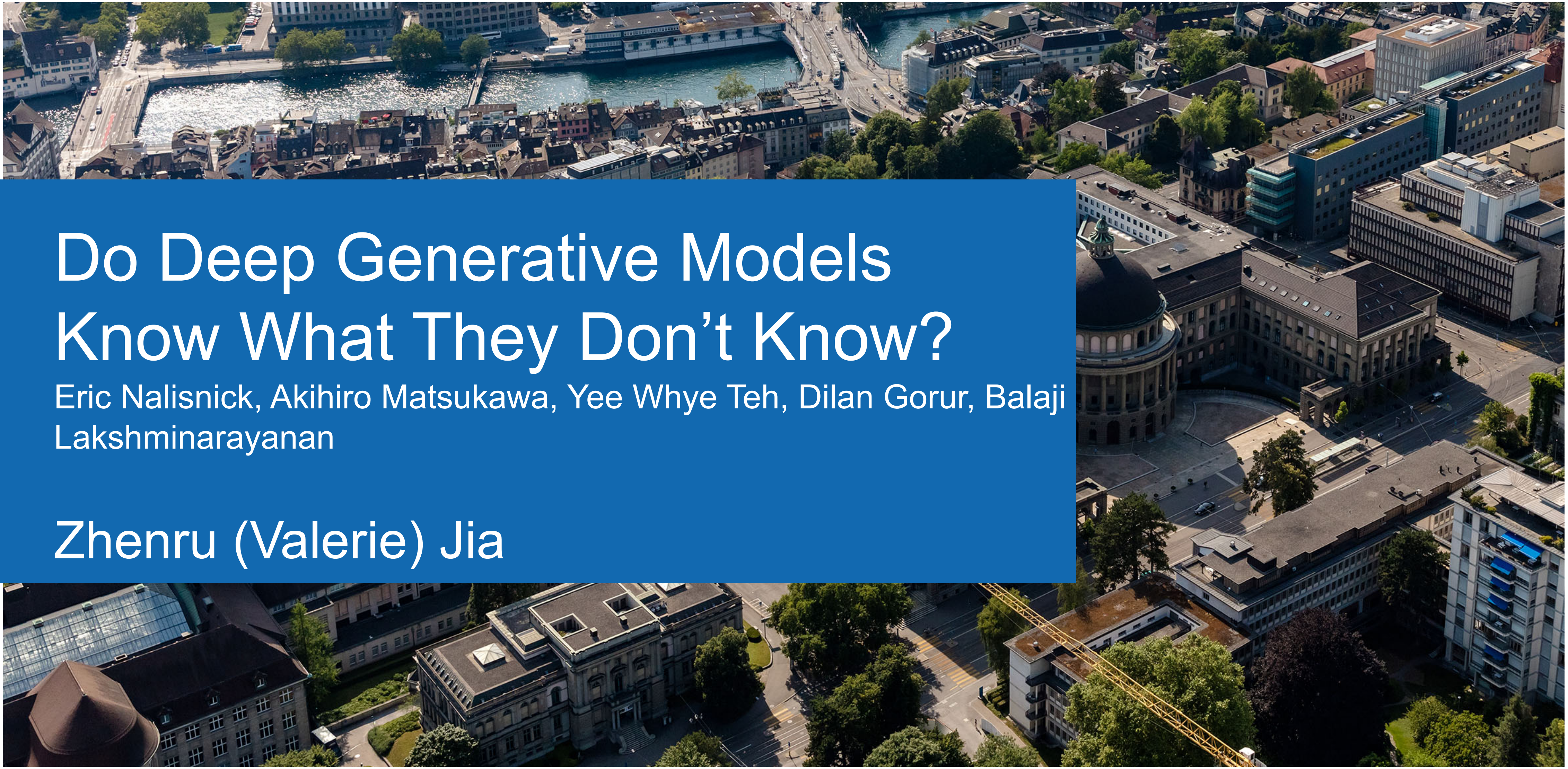
ETH zürich



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

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, Balaji Lakshminarayanan

Zhenru (Valerie) Jia

Agenda

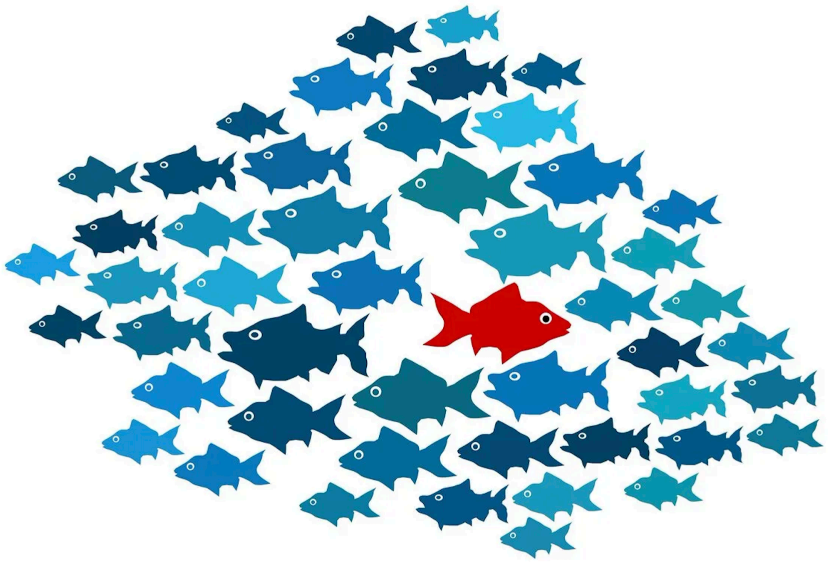
1. Motivating Problem
2. Background
3. Observations
4. Further into Flow-based Models
5. Conclusion

Motivating Problem

- Assumption: Generative models are robust to problems where the model is highly confident about a wrong result.

$$p(x, y) \text{ VS } p(y|x)$$

- Purpose: anomaly detection, active learning etc.
- The calibration w.r.t. out-of-distribution data is essential for applications such as safety

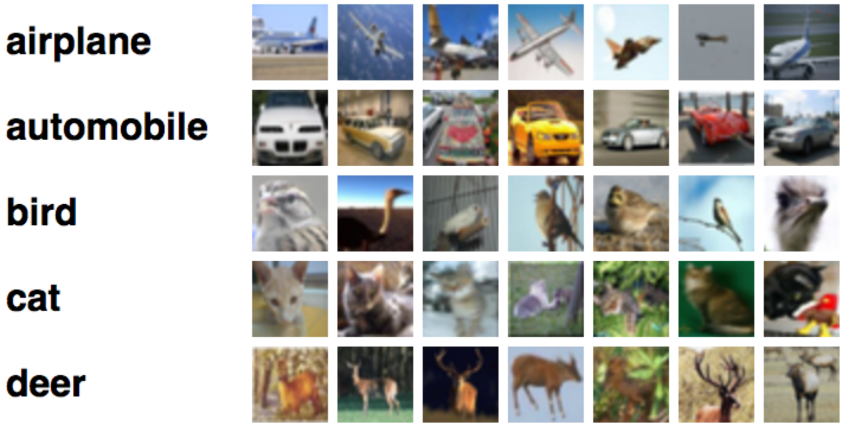


Background

1. Scope of the investigation
   - Implemented three types of generative models ($\mathrm{p}(\mathbf{X}\,;\theta) = \prod_{n=1}^{N} p(x_n;\theta)$) on pairs of commonly used image datasets.
   - In the pair of datasets, one of them is used in training and both of them will appear in the test set.
   - Investigate whether models will assign low confidence levels to the wrong predictions they give.

Background

1. Scope of the investigation
2. Datasets used
   - CIFAR-10

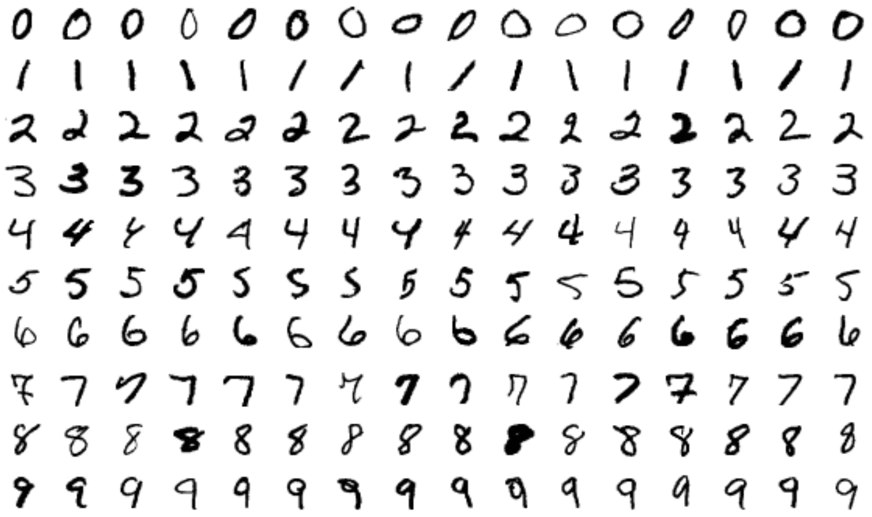


Background

1. Scope of the investigation
2. Datasets used
    - CIFAR-10
    - Street View House Numbers (SVHN)

Background

1. Scope of the investigation
2. Datasets used
   - CIFAR-10
   - Street View House Numbers (SVHN)
   - MNIST



Background

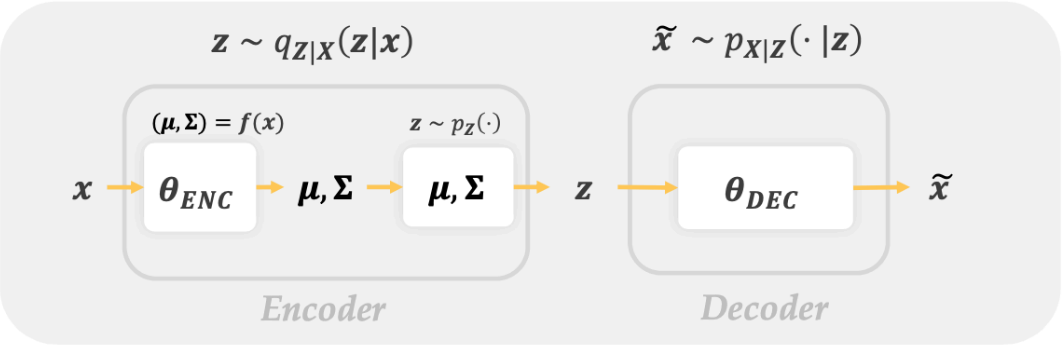
1. Scope of the investigation
2. Datasets used
    - CIFAR-10
    - Street View House Numbers (SVHN)
    - MNIST
    - FashionMNIST

Background

1. Scope of the investigation
2. Datasets used
3. Neural generative models
    - Latent Variable Models: Variational Autoencoders (VAE)
    - Autoregressive Models: PixelCNN
    - Invertible Flow-based Generative Models: Glow

Latent Variable Models: VAE

Structure:



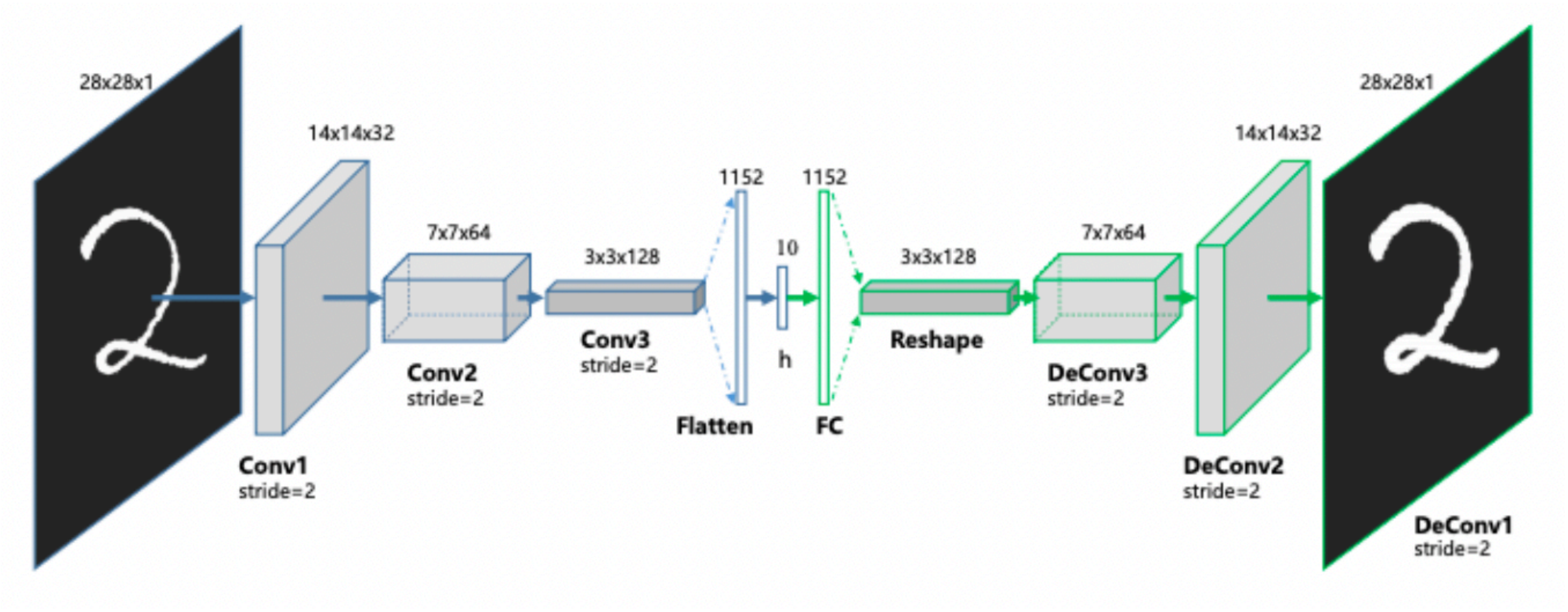
$$p_{Z|X}(z|x) = \frac{p_{X|Z}(x|z)p_Z(z)}{p_X(x)}$$

$$\Rightarrow p_X(x) \approx \frac{p_{X|Z}(x|z)p_Z(z)}{q_{Z|X}(z|x)}$$

Autoregressive Models: PixelCNN

1. Architecture: decompose the joint image distribution as a product of conditionals, where x_i is a single pixel: $p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$. Every pixel depends on all the pixels above and to the left of it.
2. Conditional PixelCNN: given a high-level image description represented as a latent vector ***h***, we seek to model the conditional distribution $p(x|\boldsymbol{h})$ of images suiting this description. Formally, $p(x|\boldsymbol{h}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1}, \boldsymbol{h})$
3. PixelCNN Auto-Encoders: consists of two parts:
   - An encoder that takes an input image x and maps it to a low-dimensional representation h
   - A decoder that tries to reconstruct the original image

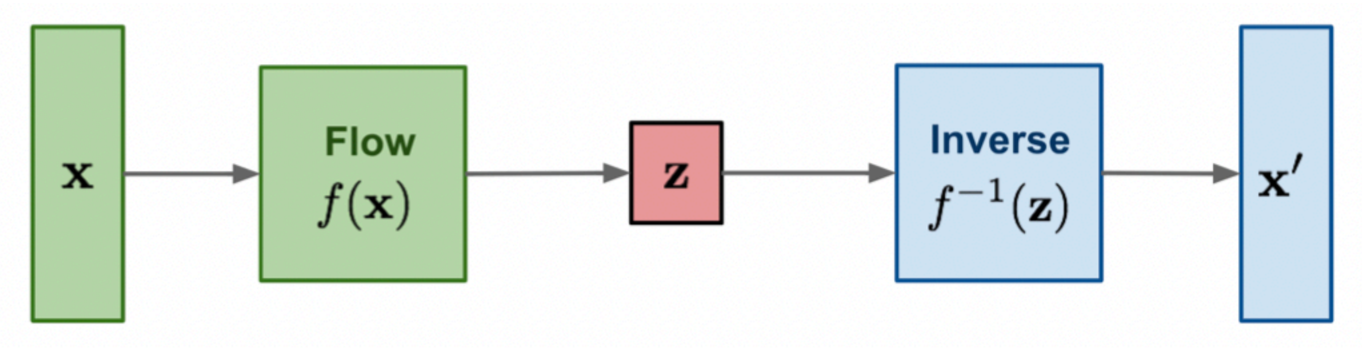
Conditional Image Generation with PixelCNN Decoders by van den Oord et al.



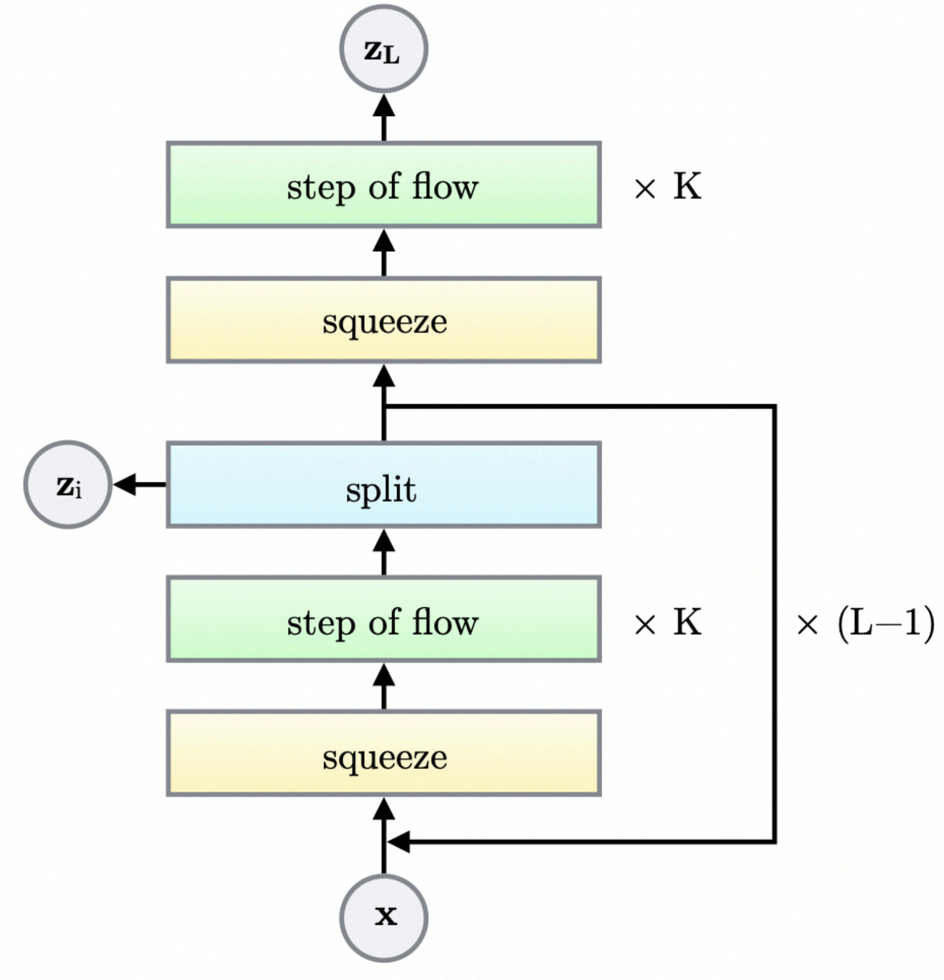
From: https://towardsdatascience.com/convolutional-autoencoders-for-image-noise-reduction-32fce9fc1763

Invertible Flow-based Generative Models: Glow

A flow-based generative model is constructed by a sequence of invertible transformations. The generative process is defined as $z \sim p(z)$ and $x = g(z)$, i.e. $z = f(x) = g^{-1}(x)$.



Glow: Generative Flow with Invertible 1×1 Convolutions by Kingma & Dhariwal



- This architecture has a depth of flow K, and number of levels L.
- An Affine Coupling Layer：A powerful reversible transformation where the forward function, the reverse function and the log- determinant are computationally efficient.

Glow: Generative Flow with Invertible 1×1 Convolutions by Kingma & Dhariwal

Background

1. Scope of the investigation
2. Datasets used
3. Neural generative models
4. Change of variables

Change of variables

Lemma 1. (Change of Variable) Let $\boldsymbol{X}$ and $\boldsymbol{Z}$ be random variables related by an *invertible* and *differentiable* mapping $f: \mathbb{R}^D \to \mathbb{R}^D$ such that $\boldsymbol{Z} = f(\boldsymbol{X})$, then the following equality holds:

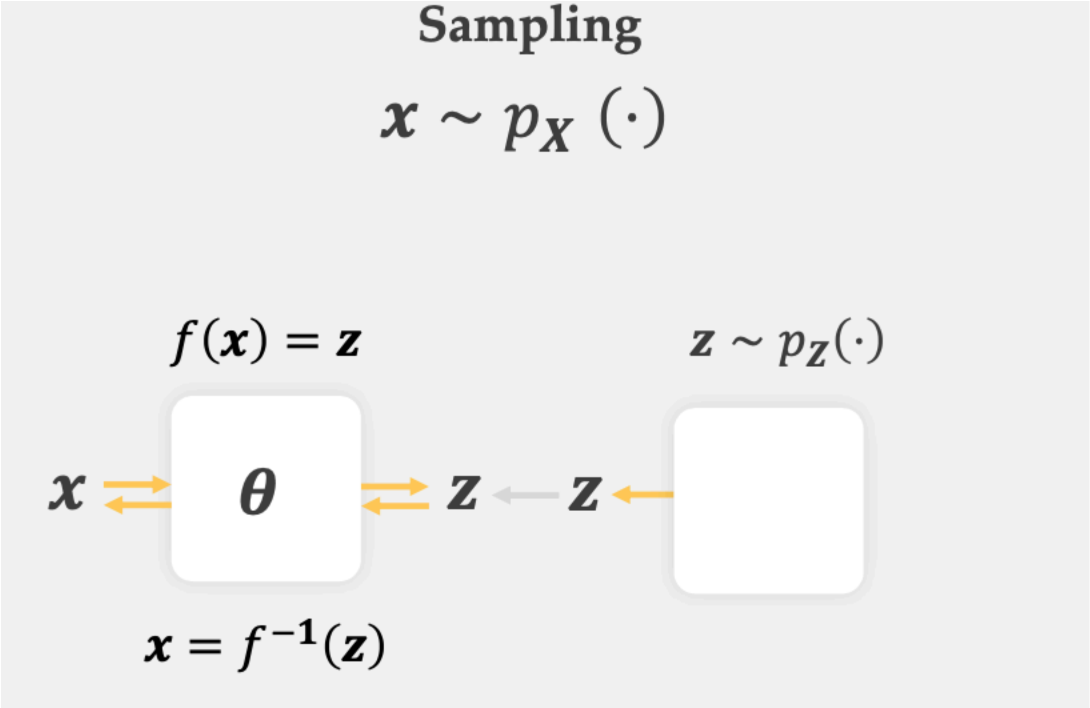
$$p_{\boldsymbol{X}}(\boldsymbol{x}) = p_{\boldsymbol{Z}}(f(\boldsymbol{x}))\left|\det(Df(\boldsymbol{x}))\right|$$

$$Df(\boldsymbol{x}) = \begin{bmatrix} \partial_{x_1} f_1(\boldsymbol{x}) & \cdots & \partial_{x_D} f_1(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ \partial_{x_1} f_D(\boldsymbol{x}) & \cdots & \partial_{x_D} f_D(\boldsymbol{x}) \end{bmatrix}$$

And inversely, $p_Z(z) = p_X\left(f^{-1}(z)\right)\left|\det\left(Df^{-1}(z)\right)\right|$

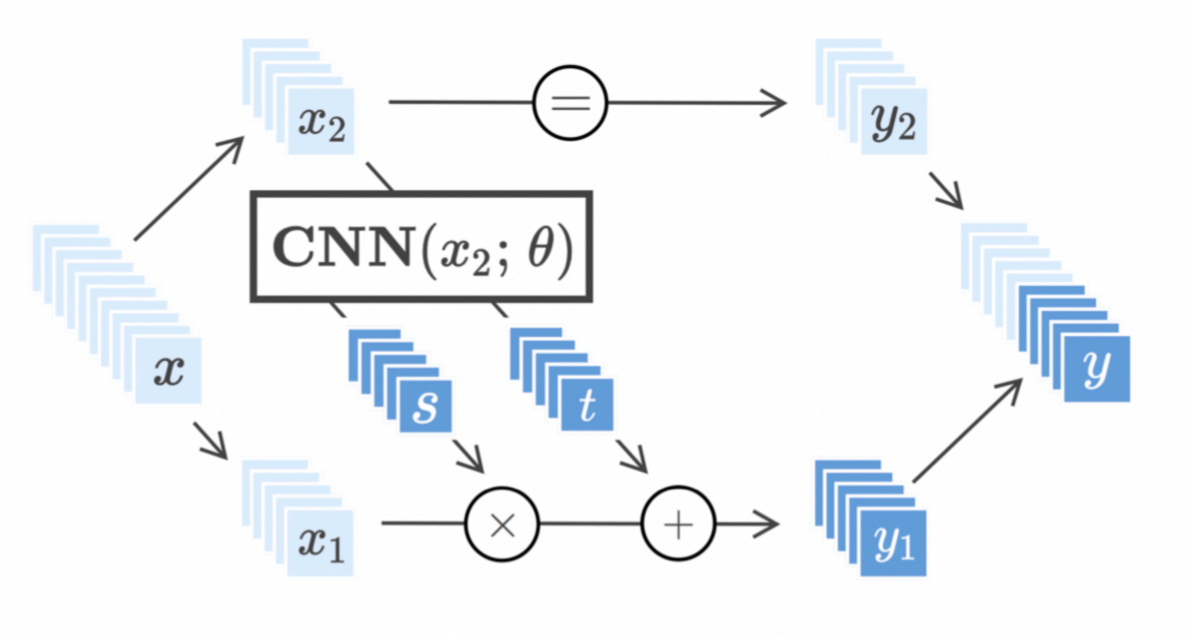
Where $\left|\det\left(Df(x)\right)\right|$ and $\left|\det\left(Df^{-1}(z)\right)\right|$ are known as the **volume elements** as they adjust the volume change under the alternate measure.

Change of variables



Change of variables

One particular form of f is the bijection from affine coupling layers (ACL), which transform x by way of translation and scaling operations.



Specifically, ACL takes the form:

$$f_{ACL}(x;\phi) = [\exp\{s(x_{d:};\phi_s)\} \odot x_{:d} + t(x_{d:};\phi_t), x_{d:}]$$

Invertible generative models for inverse problems: mitigating representation error and dataset bias by Muhammad Asim

Change of variables

$$Df(\boldsymbol{x}) = \begin{bmatrix} \partial_{x_1} f_1(\boldsymbol{x}) & \cdots & \partial_{x_D} f_1(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ \partial_{x_1} f_D(\boldsymbol{x}) & \cdots & \partial_{x_D} f_D(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} * & 0 & 0 \\ * & * & 0 \\ * & * & * \end{bmatrix}$$

So, the determinant equals to the multiplication of the diagonal inputs.

With ACL, we have $\log\left|\frac{\partial f_\phi}{\partial x}\right| = \sum_{j=d}^{D} s_j(x_{d:};\phi_s)$.

This class of transform is known as **non-volume preserving (NVP)** since the volume element can vary with each input x.

A transformation f can also be defined with just translation operations, i.e. $f_{ACL}(x;\phi) = [t(x_{d:};\phi_t), x_{d:}]$ and this transformation is **volume preserving (VP).**

Observations

Goal: test deep generative models' ability to quantify when an input comes from a different distribution than that of the training set.

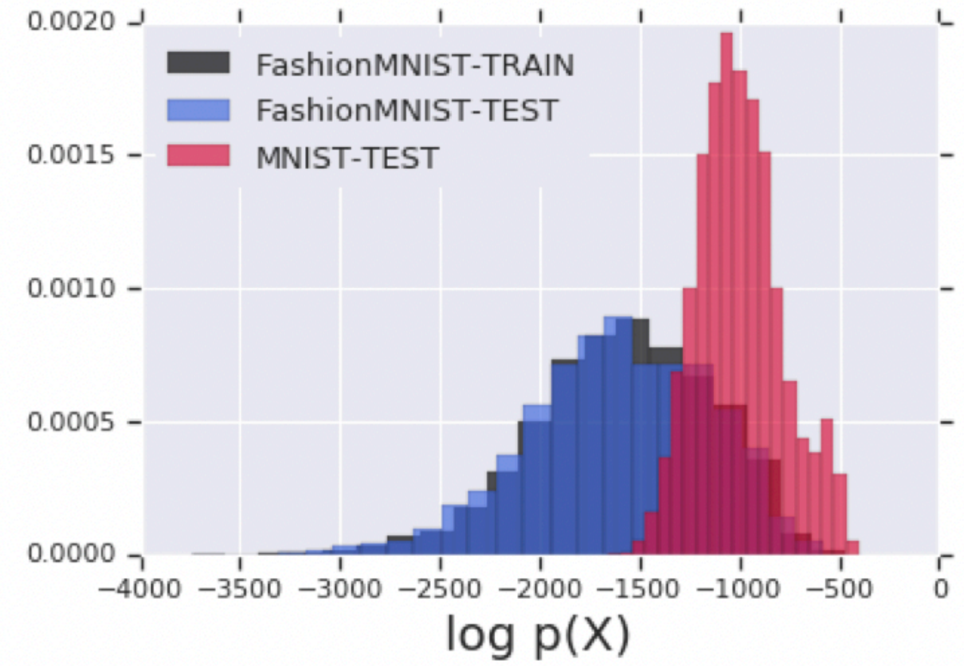
Criteria for comparison: bits-per-dimension (BPD, lower value the better) and log-likelihood (higher value the better)

$BPD(x) = -\frac{\log p(x)}{I \times J \times K \times \log 2}$ for an image of resolution I × J and K channels.

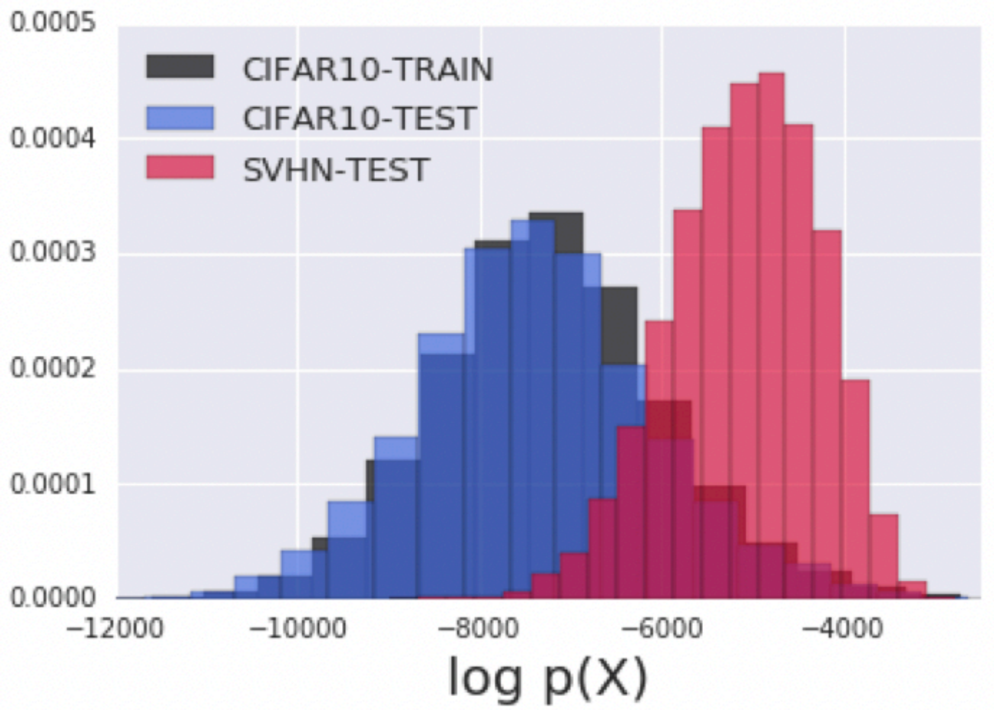
Expectation: models assign a lower probability to the out-of-distribution data because they are not trained on it.

For Glow

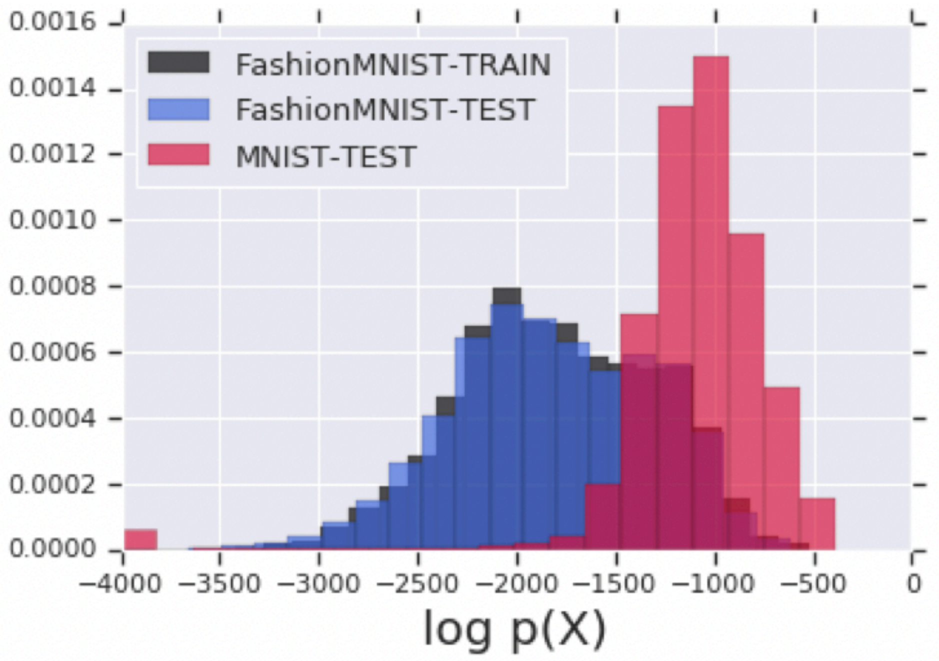
Data Set	Avg. Bits Per Dimension
Glow Trained on FashionMNIST	
FashionMNIST-Train	2.902
FashionMNIST-Test	2.958
MNIST-Test	**1.833**
Glow Trained on MNIST	
MNIST-Test	1.262



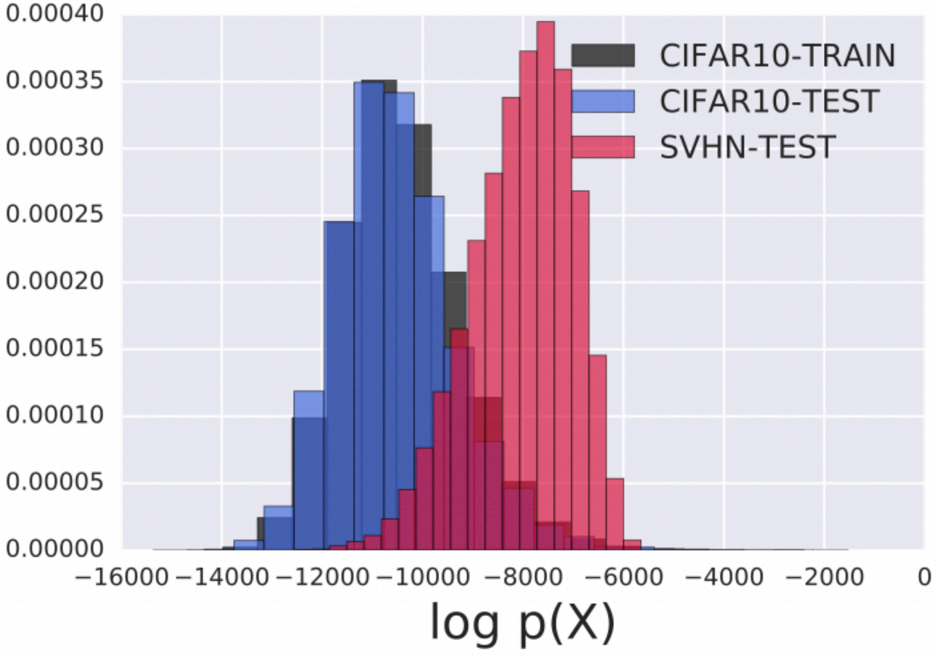
Data Set	Avg. Bits Per Dimension
Glow Trained on CIFAR-10	
CIFAR10-Train	3.386
CIFAR10-Test	3.464
SVHN-Test	**2.389**
Glow Trained on SVHN	
SVHN-Test	2.057



For VAE

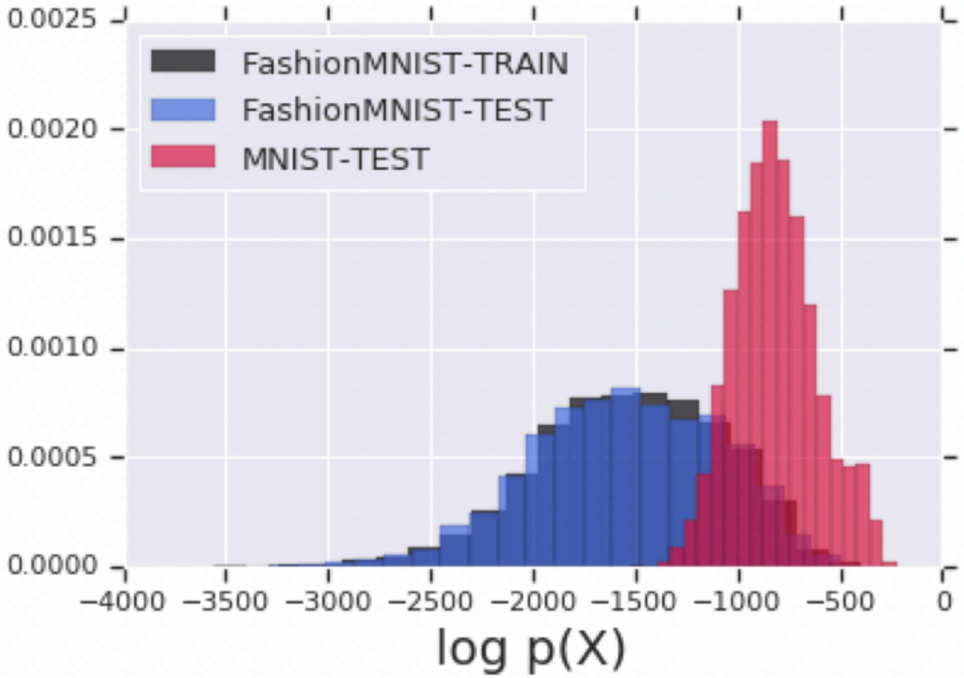


(b) **VAE**: FashionMNIST vs MNIST

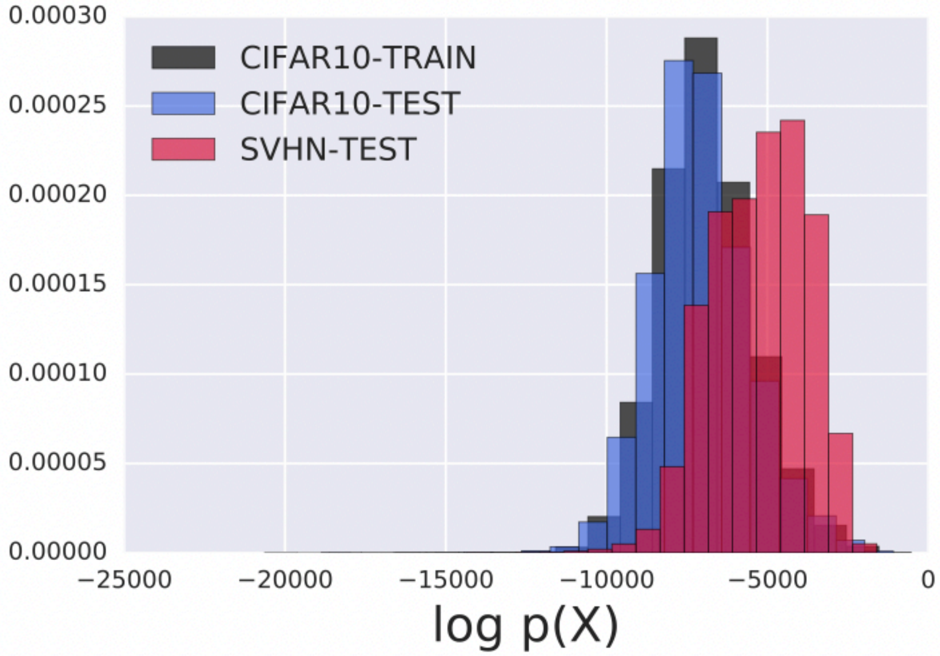


(d) **VAE**: CIFAR-10 vs SVHN

For PixelCNN



(a) **PixelCNN**: FashionMNIST vs MNIST



(c) **PixelCNN**: CIFAR-10 vs SVHN

Further into Invertible Flow-based Models

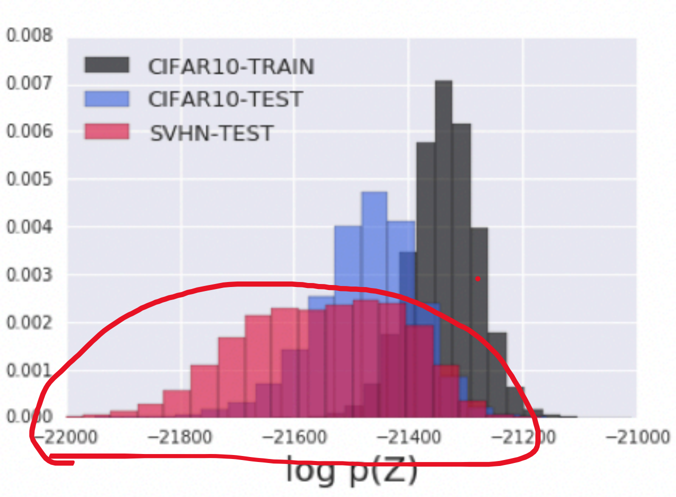
- Allow for better experimental control
  - Can compute exact marginal likelihoods
  - The transforms used in flow-based models have Jacobian constraints that simplify the analysis
- Flow of investigation
  - Separate the contributions to the likelihood of each term in the change-of-variable formula
  - Volume element is the primary cause?
  - Constant-volume flows?

Decomposing the change-of-variables objective

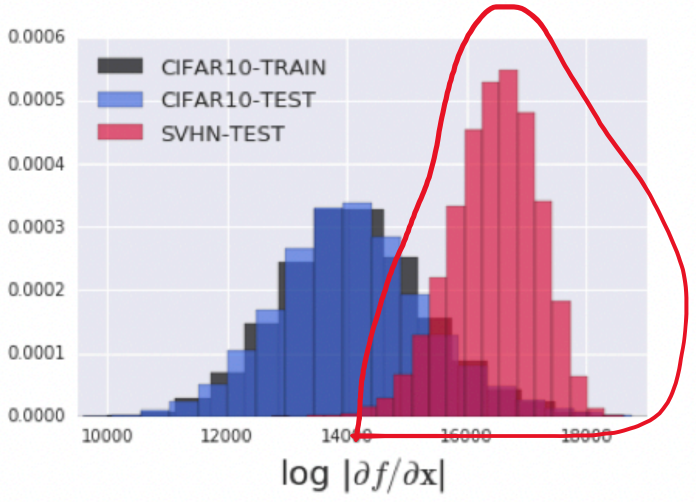
- Change-of-variable objective:

$$\theta^* = argmax_\theta \log L(\mathcal{D}|\theta)$$
$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p_Z(f(\boldsymbol{x}_i|\boldsymbol{\theta})) + \log\left|\det(Df(\boldsymbol{x}_i|\boldsymbol{\theta}))\right|$$

- Plot $\log \mathrm{p(z)}$ and $\log\left|\frac{\partial f}{\partial x}\right|$ terms for NVP-Glow



(a) CIFAR-10: $\log p(z)$

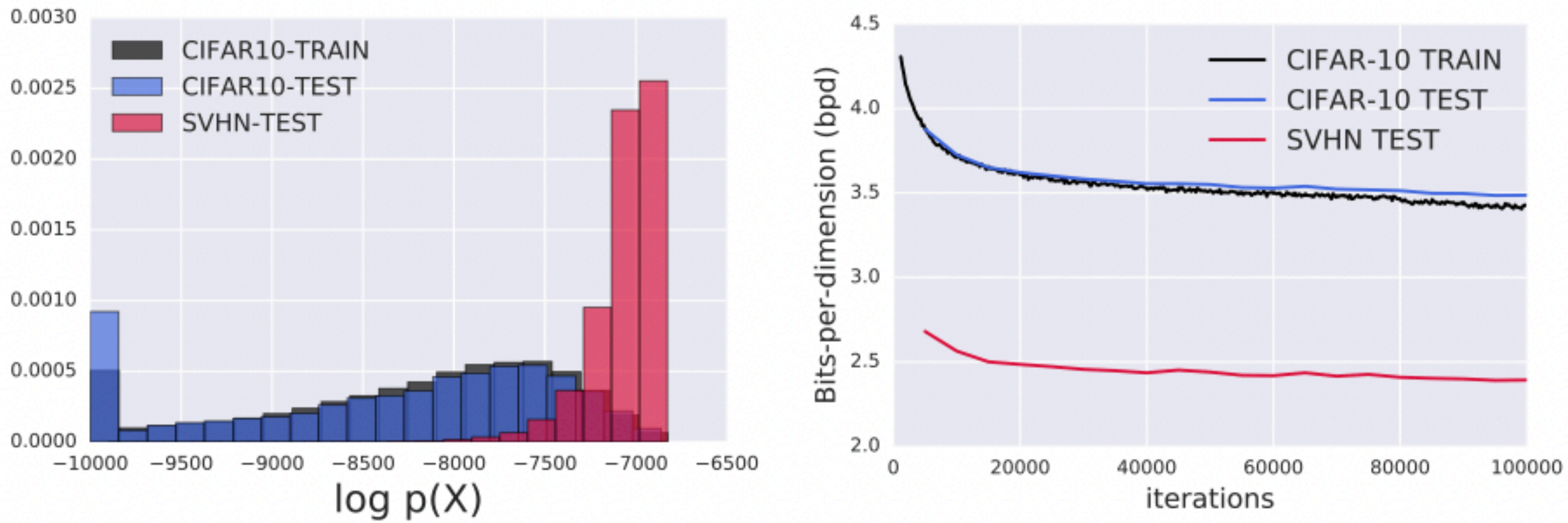


(b) CIFAR-10: Volume

ETH zürich

Is the volume the culprit?

- Rewarding the maximisation of the Jacobian determinant in the objective encourages the model to increase its sensitivity to perturbations in $\mathcal{X}$
- Contradicts a long history of derivative-based regularisation that rewards the model for decreasing its sensitivity to input directions (Stability)
- Then trained Glow with constant-volume transformations. Modify the affine layers to use only translation operations, but keep other components of the flow.



CIFAR10-TRAIN
CIFAR10-TEST
SVHN-TEST
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
−10000
−9500
−9000
−8500
−8000
−7500
−7000
−6500
log p(X)
CIFAR-10 TRAIN
CIFAR-10 TEST
SVHN TEST
Bits-per-dimension (bpd)
4.5
4.0
3.5
3.0
2.5
2.0
0
20000
40000
60000
80000
100000
iterations

ETH zürich

Second order analysis

- Analyse the phenomenon by way of linearizing the difference in expected log-likelihoods
- Two distributions: the training distribution $x \sim p^*$ and some dissimilar distribution $x \sim q$, both with support on $\mathcal{X}$.
- For a generative model $p(x;\theta)$, the problem can be formulated as:

$$\mathbb{E}_q[\log p(x;\theta)] - \mathbb{E}_{p^*}[\log p(x;\theta)] > 0$$

- Perform a second order expansion of the log-likelihood around an interior point x_0.

$$\log p(x;\theta) \approx \log p(x_0;\theta) + \nabla_{x_0} \log p(x_0;\theta)^T (x - x_0) + \frac{1}{2} Tr\{\nabla_{x_0}^2 \log p(x_0;\theta)(x - x_0)(x - x_0)^T\}$$

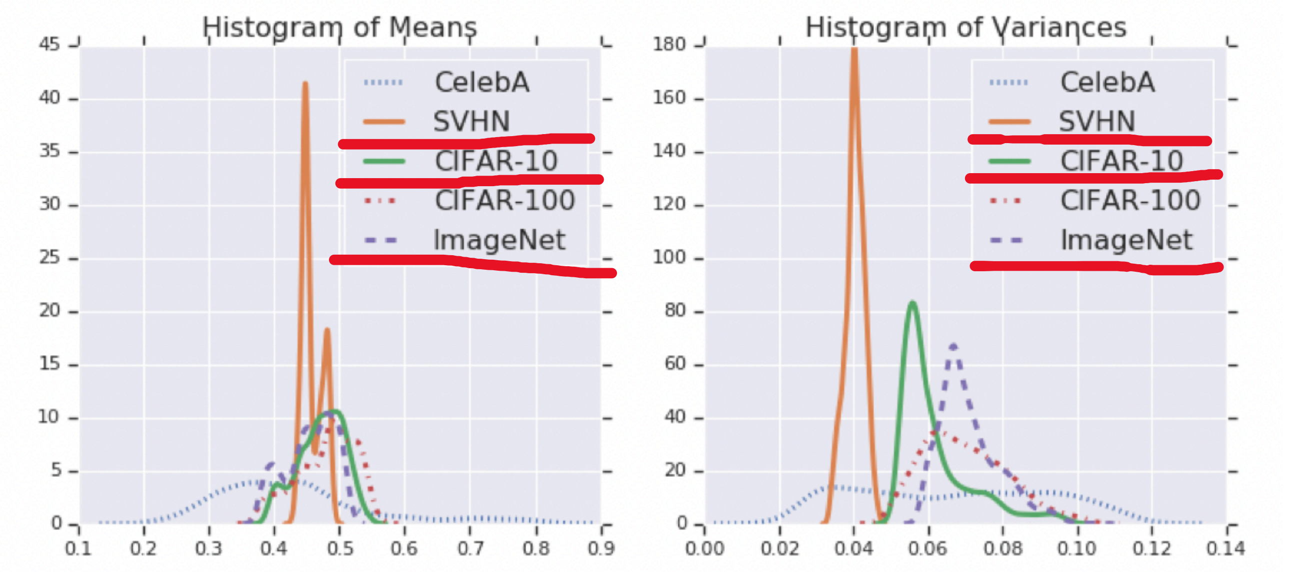
- Assumption:
  - $\mathbb{E}_q[\log p(x_0;\theta)] = \mathbb{E}_{p^*}[\log p(x_0;\theta)]$
  - $\mathbb{E}_q[x] = \mathbb{E}_{p^*}[x] = x_0$
  - The generative model is flow-based and volume-preserving
- $0 < \mathbb{E}_q[\log p(x;\theta)] - \mathbb{E}_{p^*}[\log p(x;\theta)]$

  $\approx \frac{1}{2}\mathrm{Tr}\,\{\nabla^2_{x_0}\log p(x_0;\theta)(\Sigma_q - \Sigma_{p^*})\} = \frac{1}{2}Tr\left\{\left[\nabla^2_{x_0}\log p_z\big(f(x_0;\phi)\big) + \nabla^2_{x_0}\log\left|\frac{\partial f_\phi}{\partial x_0}\right|\right](\Sigma_q - \Sigma_{p^*})\right\}$

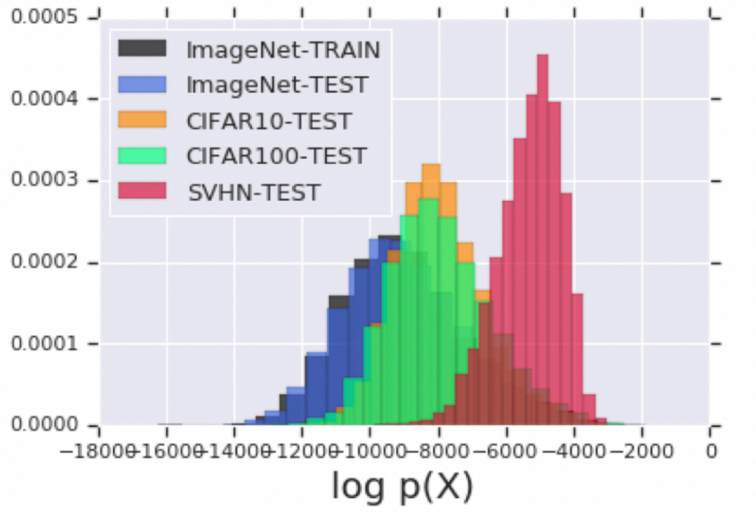
  $= \frac{1}{2}Tr\left\{\underbrace{\left[\nabla^2_{x_0}\log p_z\big(f(x_0;\phi)\big)\right]}\,(\Sigma_q - \Sigma_{p^*})\right\}$

  Would be negative for any log-concave density distribution (eg. Normal, Laplace)

The degree of differences in likelihoods agrees with the differences in variances



(a) Histogram of per-dimension means and variances (empirical).



(d) Train on ImageNet,
Test on CIFAR-10 / CIFAR-100 / SVHN

Conclusion

- Have shown that comparing the likelihoods of deep generative models alone cannot identify the training set or inputs like it
- Caution against using the density estimates from deep generative models to identify inputs outside the training distribution
- Need for further work on generative models and their evaluation
- Deep generative models can detect out-of-distribution inputs when
  - Using alternative metrics: computing the Watanabe- Akaike information criterion
  - Outlier Exposure

Q & A

Related Work

- Solutions:
  - Mitigate the CIFAR-10 vs SVHN issue by exposing the model to outliers during training: *Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep Anomaly Detection with Outlier Exposure. In International Conference on Learning Representations (ICLR), 2019.*
  - Propose training an ensemble of generative models with an adversarial objective and testing for out-of-training-distribution inputs by computing the Watanabe- Akaike information criterion via the ensemble: *Hyunsun Choi and Eric Jang. Generative Ensembles for Robust Anomaly Detection. ArXiv e-Print arXiv:1810.01392, 2018.*
  - Propose a likelihood ratio method: *Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., DePristo, M. A., Dillon, J. V., and Lakshmi- narayanan, B. Likelihood ratios for out-of-distribution detection. arXiv preprint arXiv:1906.02845, 2019.*

- Confirmation:
  - *Hyunsun Choi and Eric Jang. Generative Ensembles for Robust Anomaly Detection. ArXiv e-Print arXiv:1810.01392, 2018.*