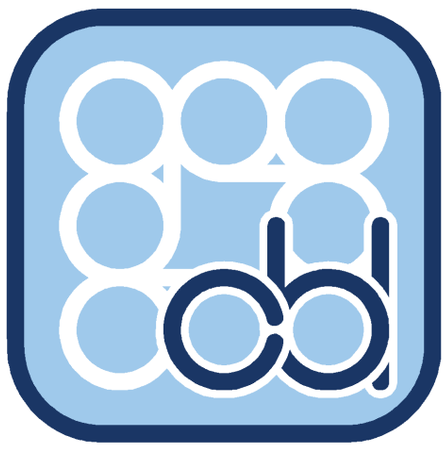
Do Deep Generative Models Know What They Don't Know?

Eric Nalisnick

LTL Seminar
14.2.19



Computational and Biological Learning
University of Cambridge

Outline

1 **Motivation:** Protecting Predictive Models

2 **An Experiment:** Chicken or Seven?

3 **Flow-Based Generative Models:** Background

4 **More Experiments**

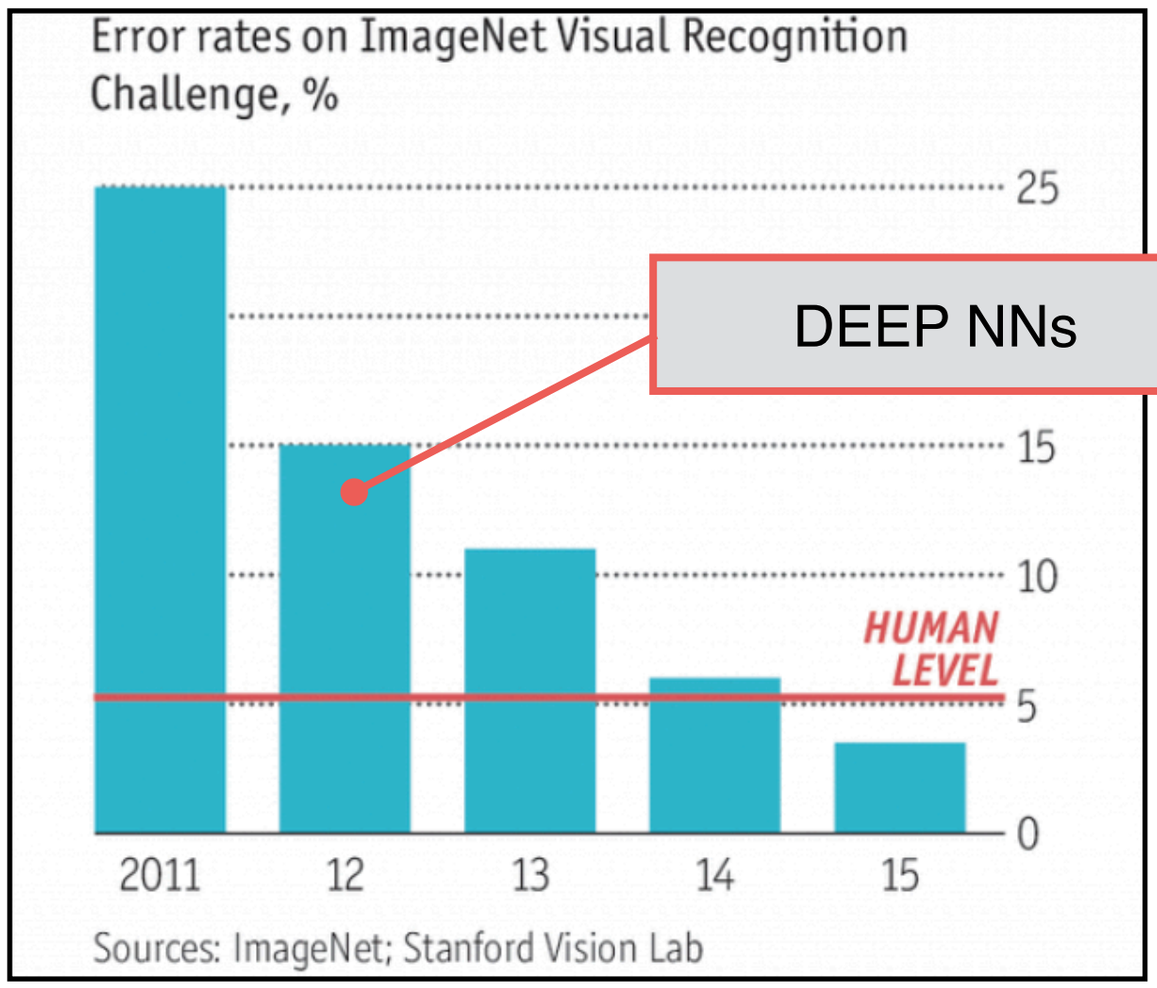
5 **A Mathematical Characterization**

6 **Parting Thoughts and Future Work**

Protecting Predictive Models

Deep Learning Results

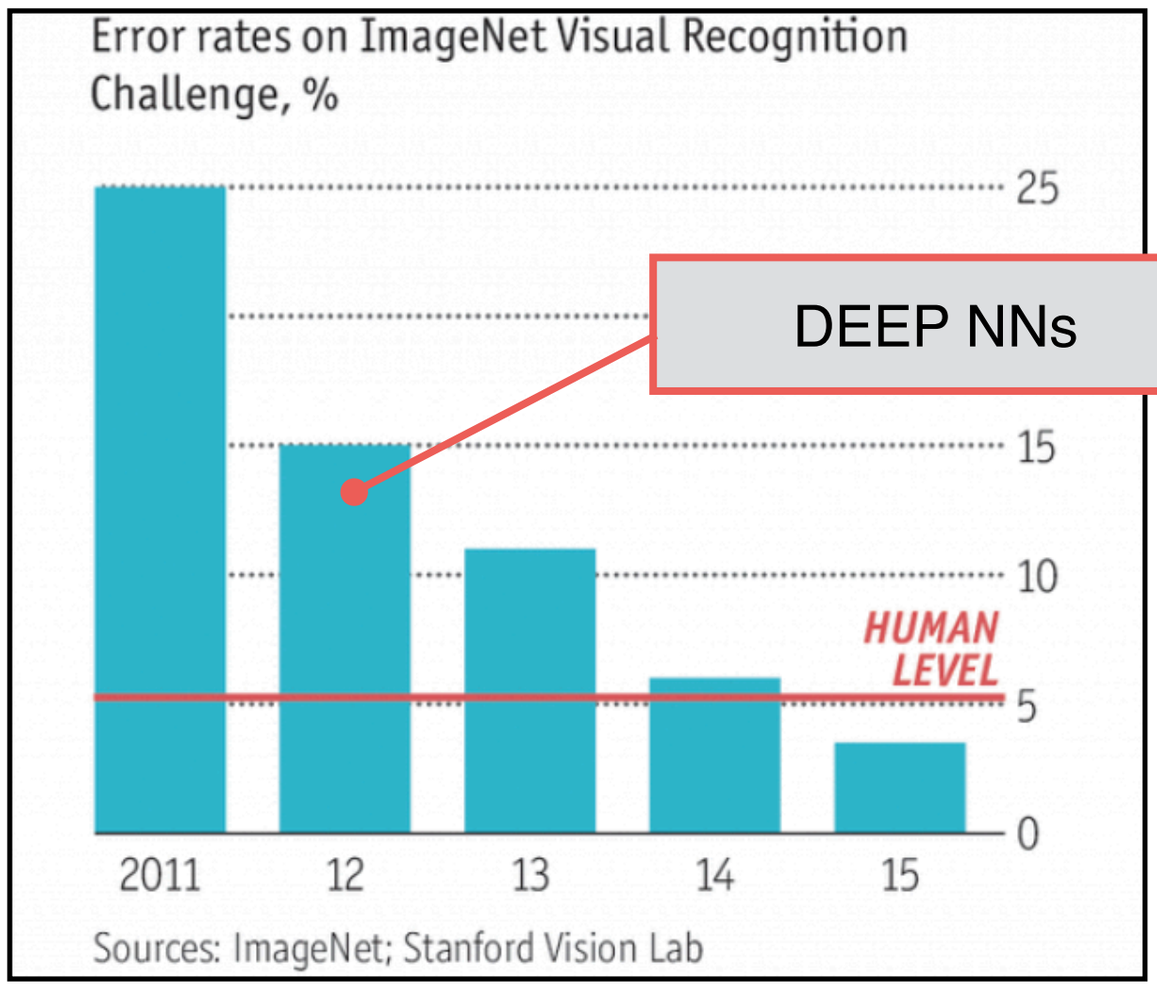
- **Computer Vision**: Results on ImageNet object classification dataset.



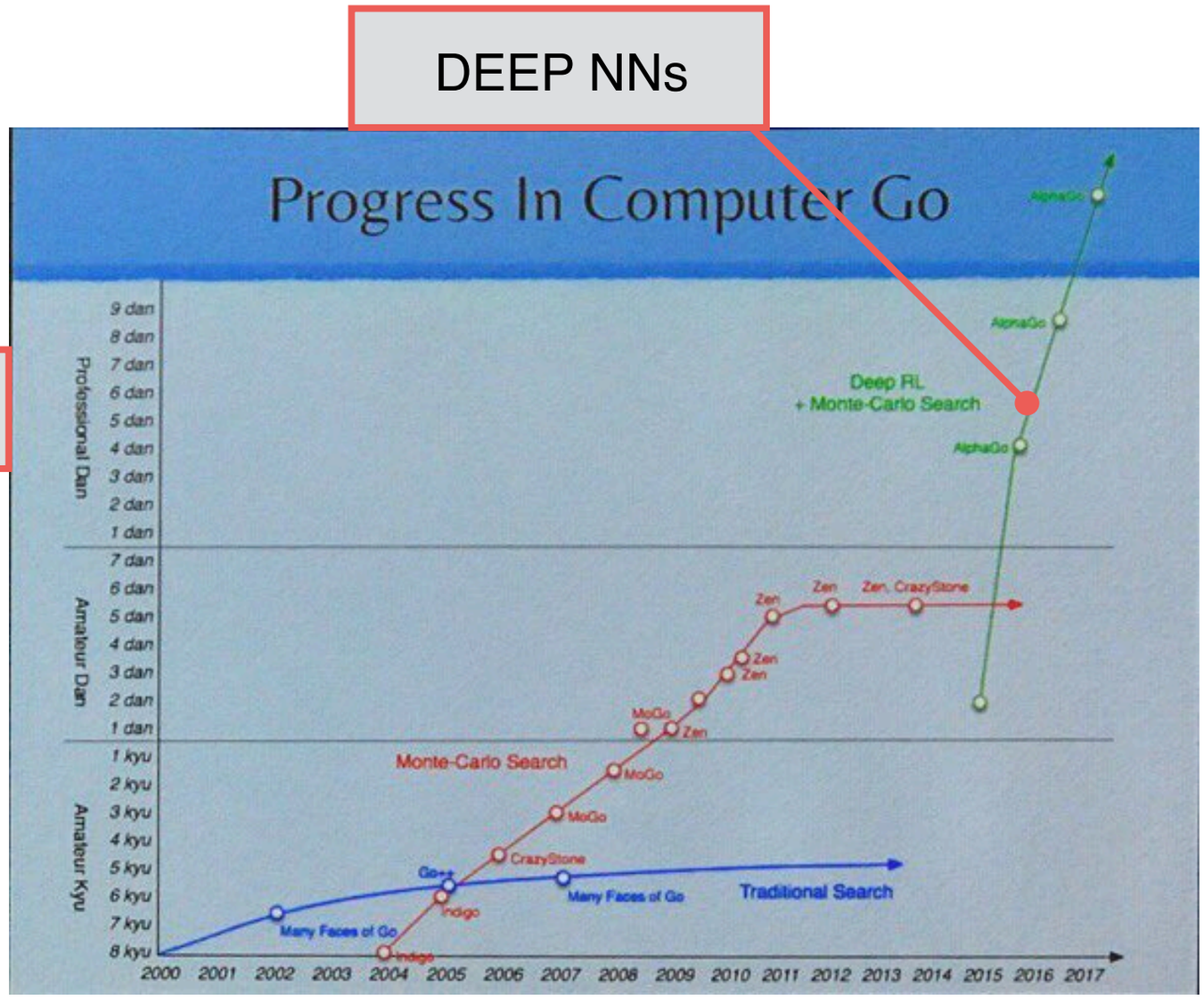
https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

Deep Learning Results

- **Computer Vision**: Results on ImageNet object classification dataset.



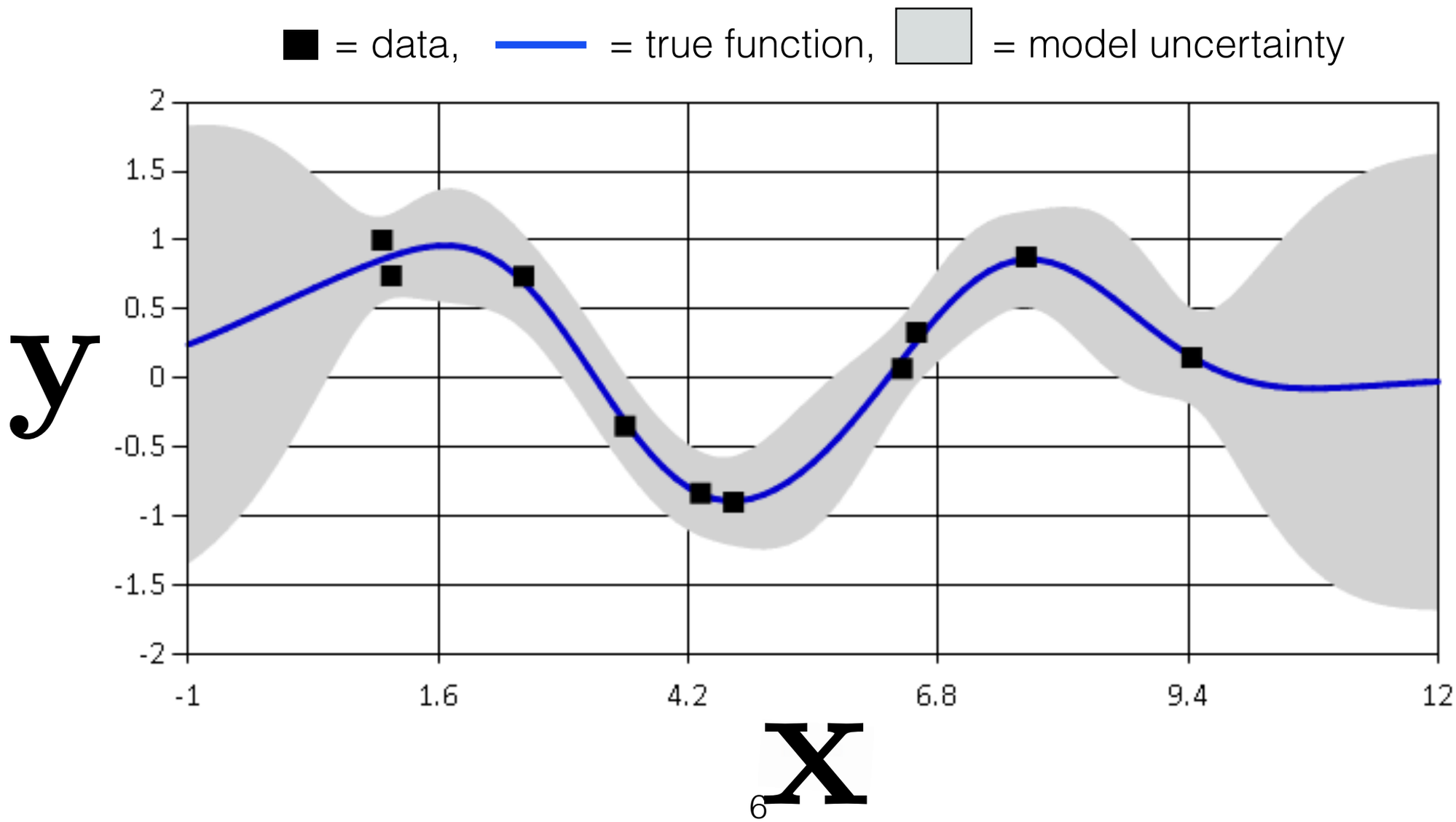
- **Reinforcement Learning**: Results in playing Go.

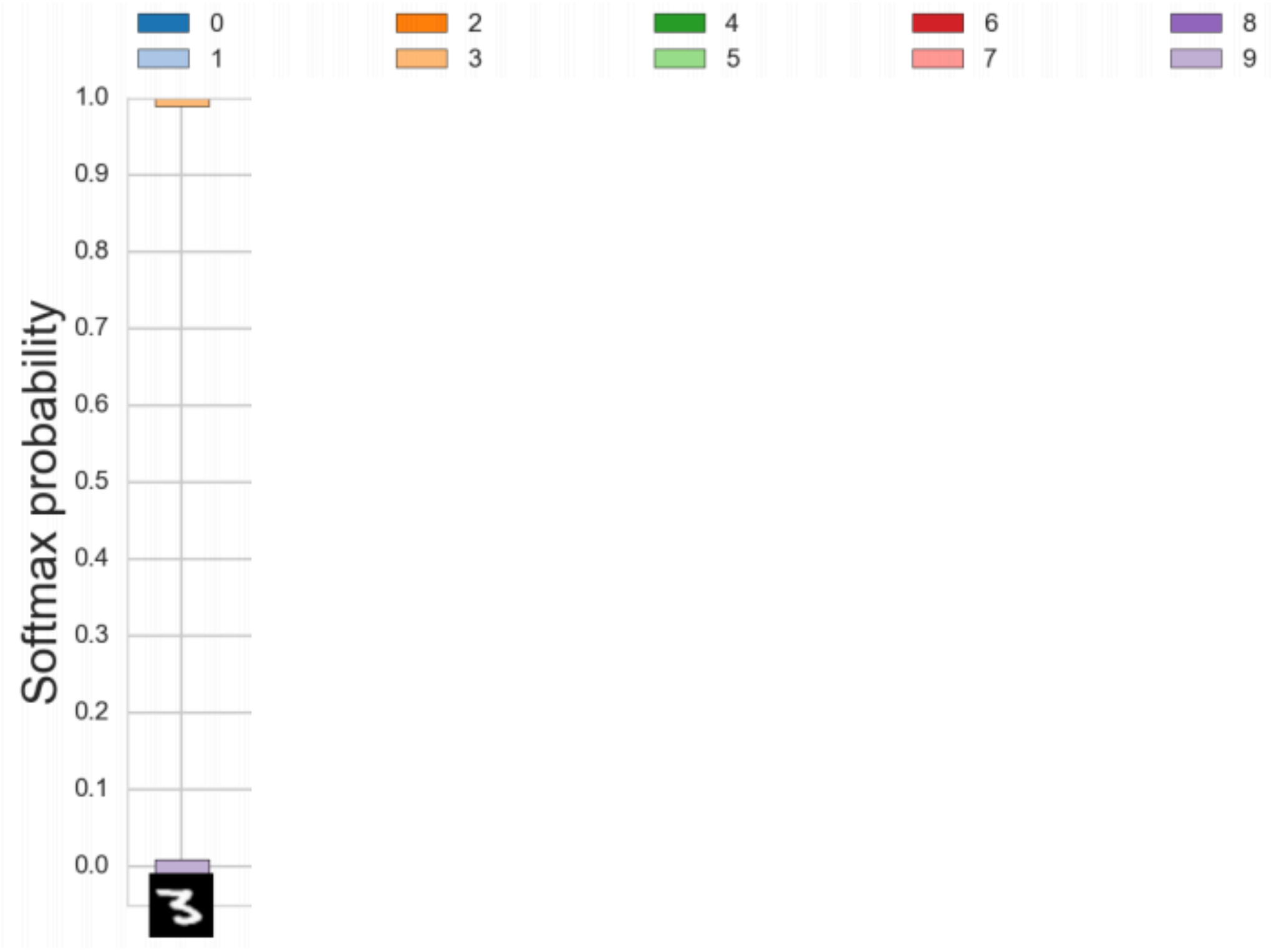


https://www.economist.com/news/special-report/21700756-artificial-intelligence-boom-based-old-idea-modern-twist-not

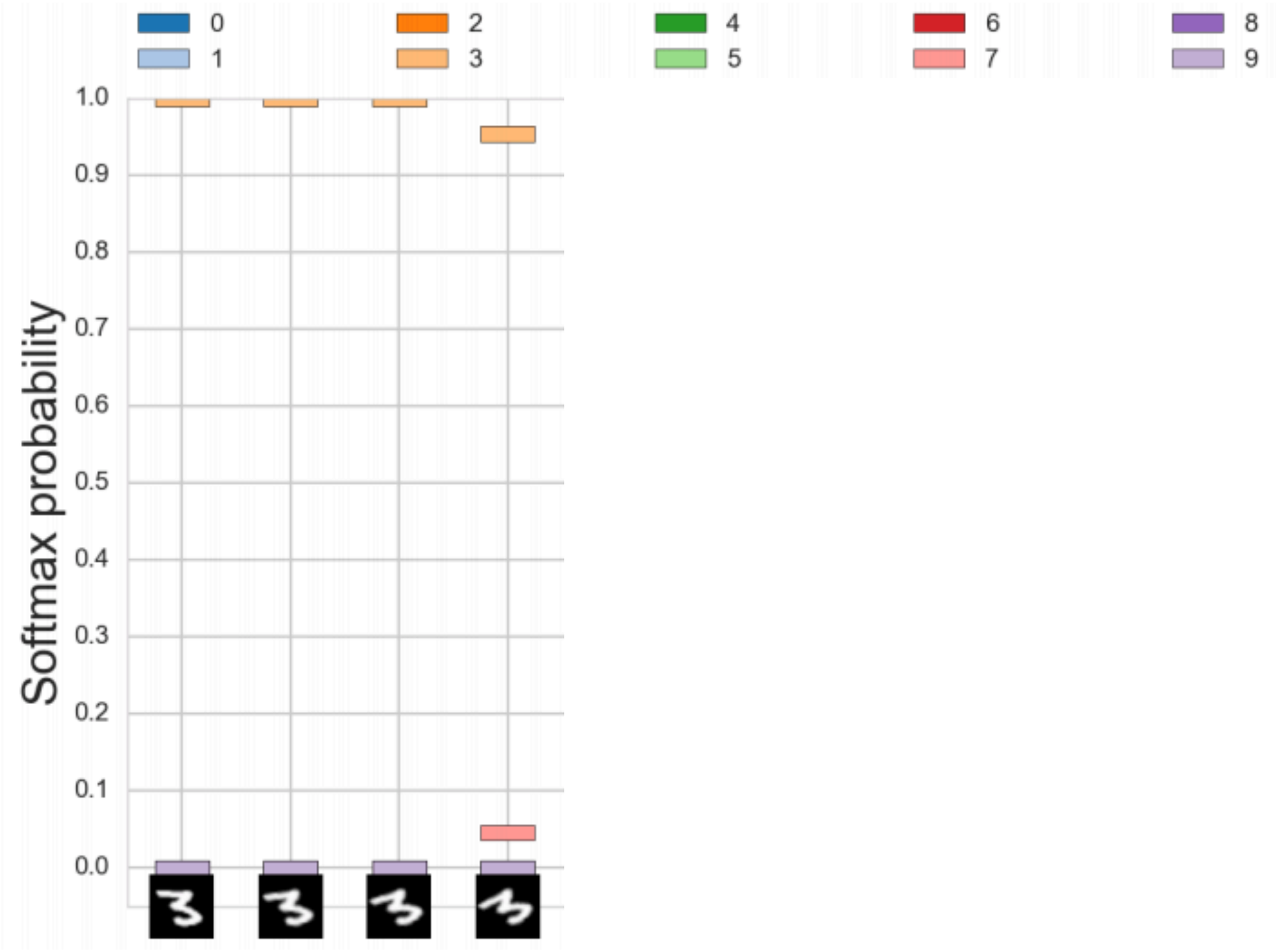
$$p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta})$$

Labels **Features** **Parameters**

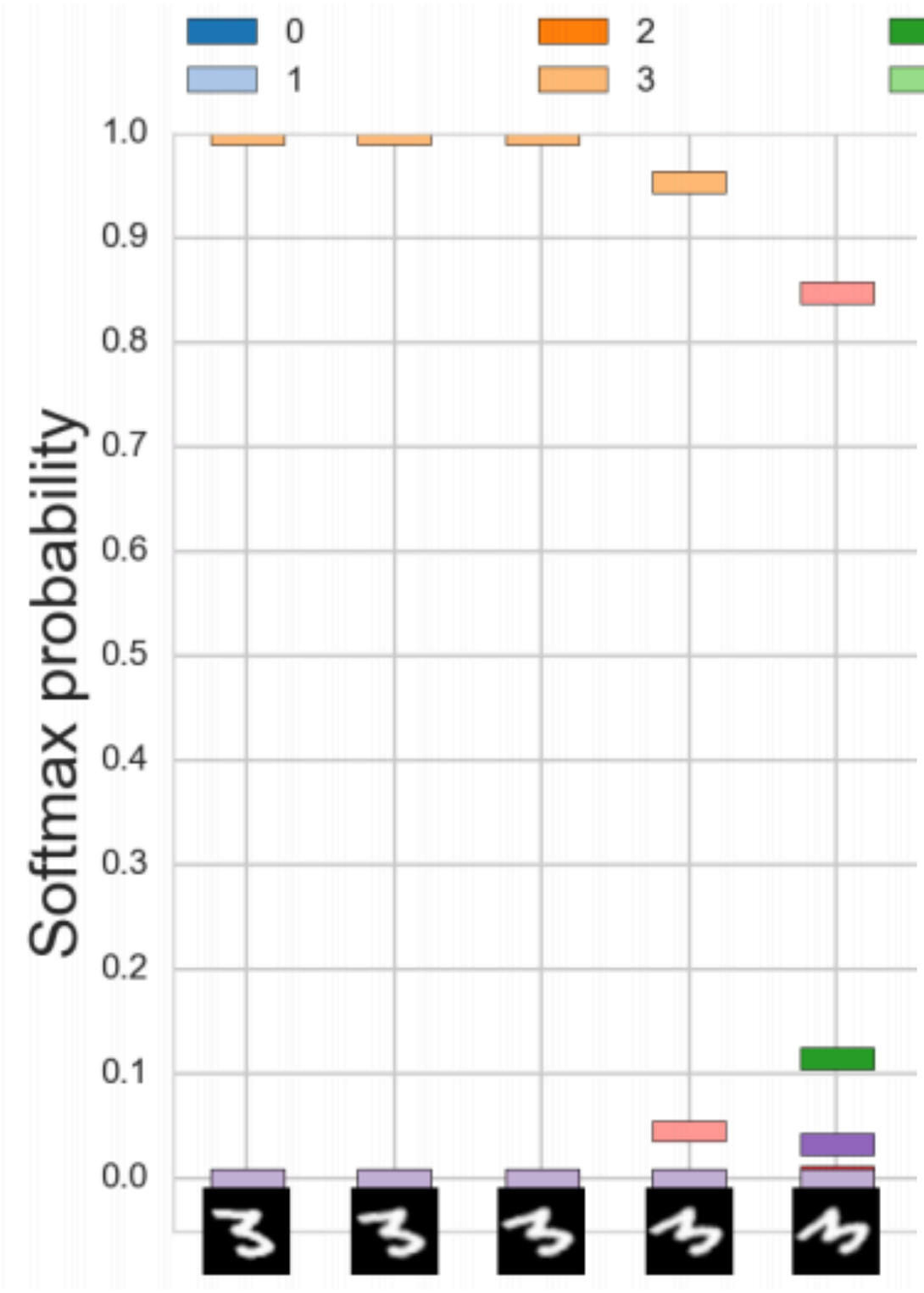




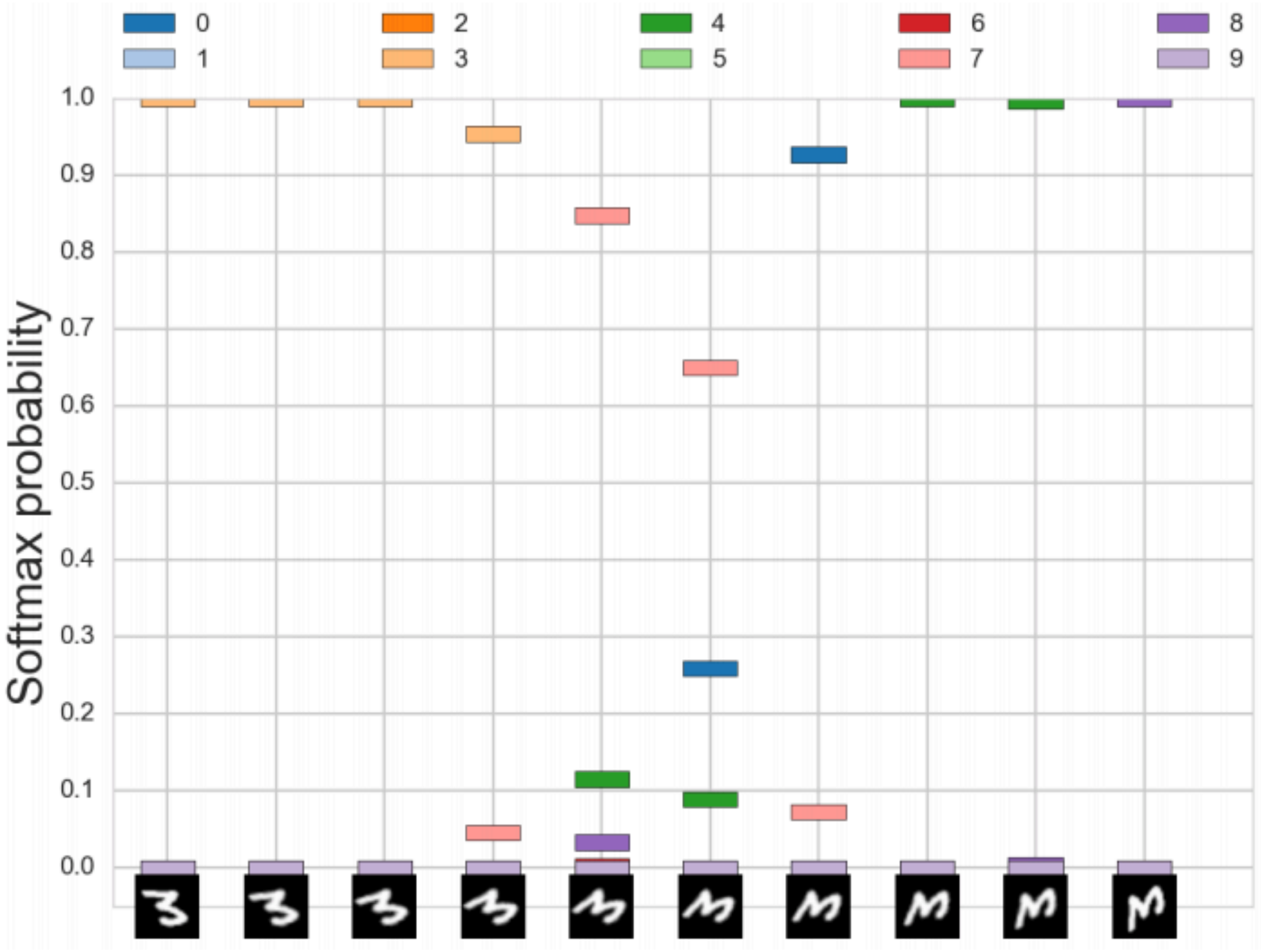
[Louizos & Welling, ICML 2017]



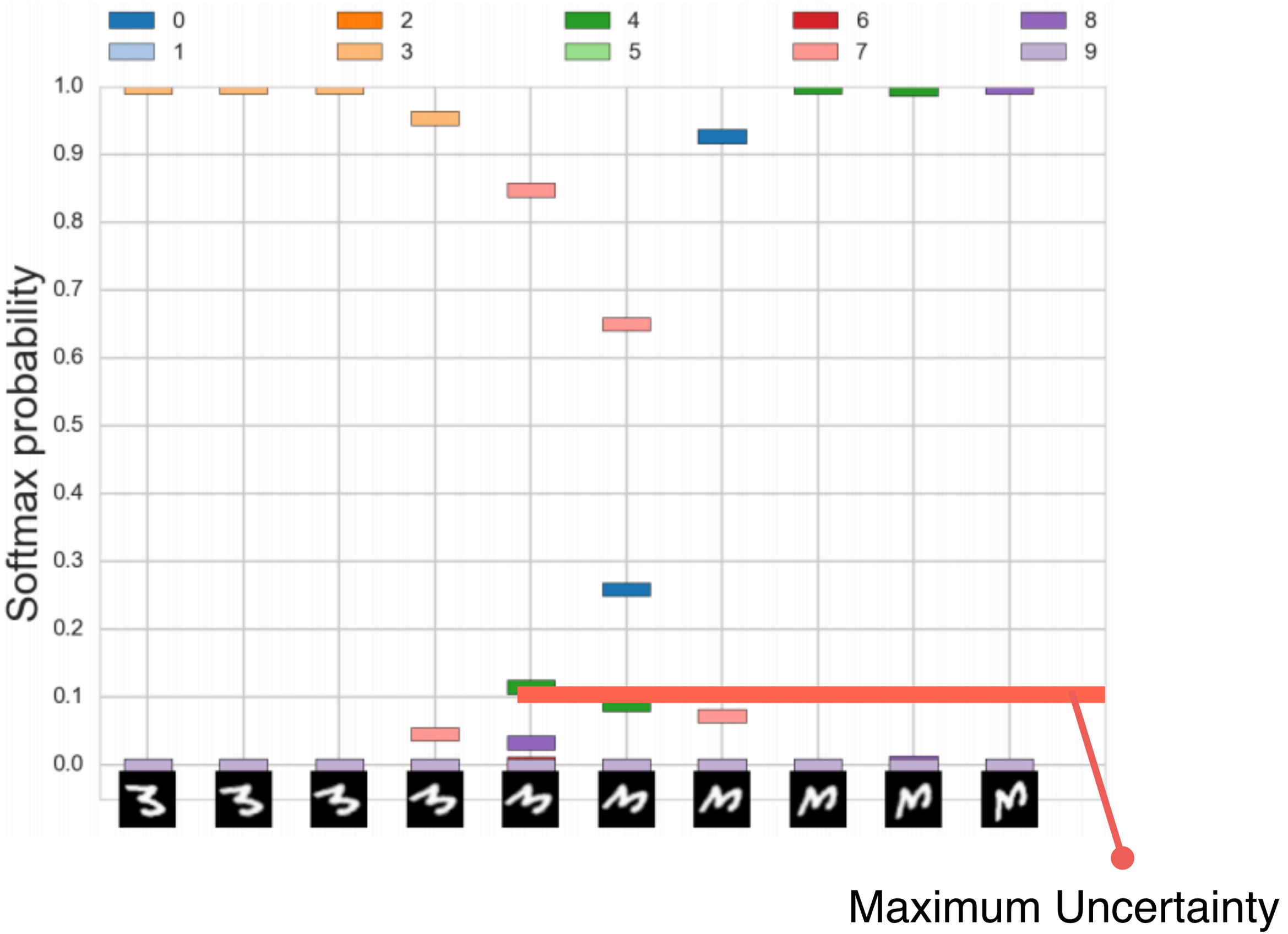
[Louizos & Welling, ICML 2017]



[Louizos & Welling, ICML 2017]



[Louizos & Welling, ICML 2017]

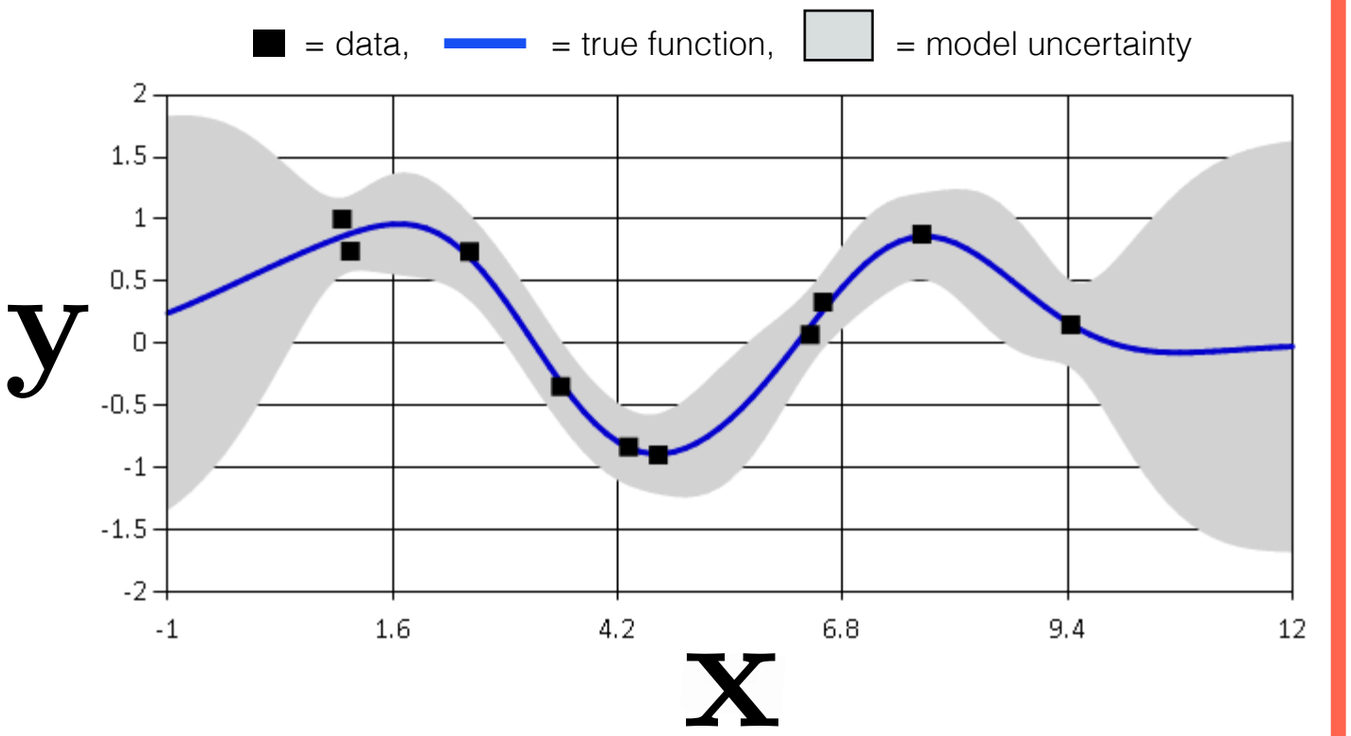


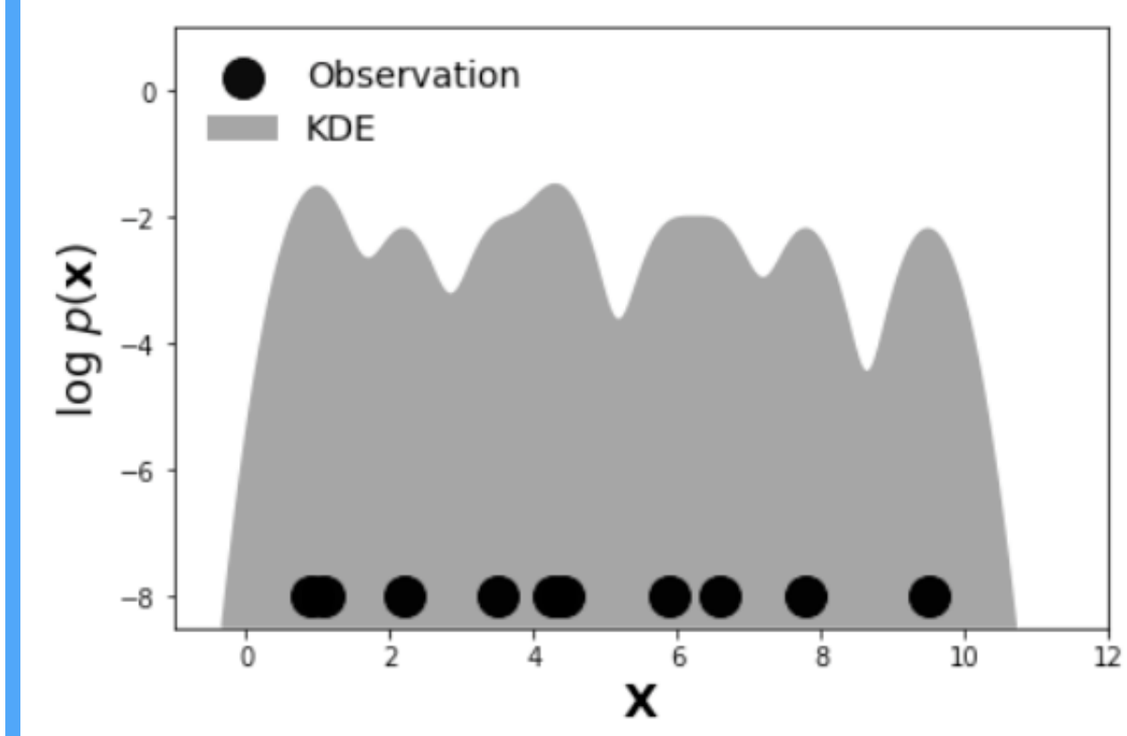
[Louizos & Welling, ICML 2017]

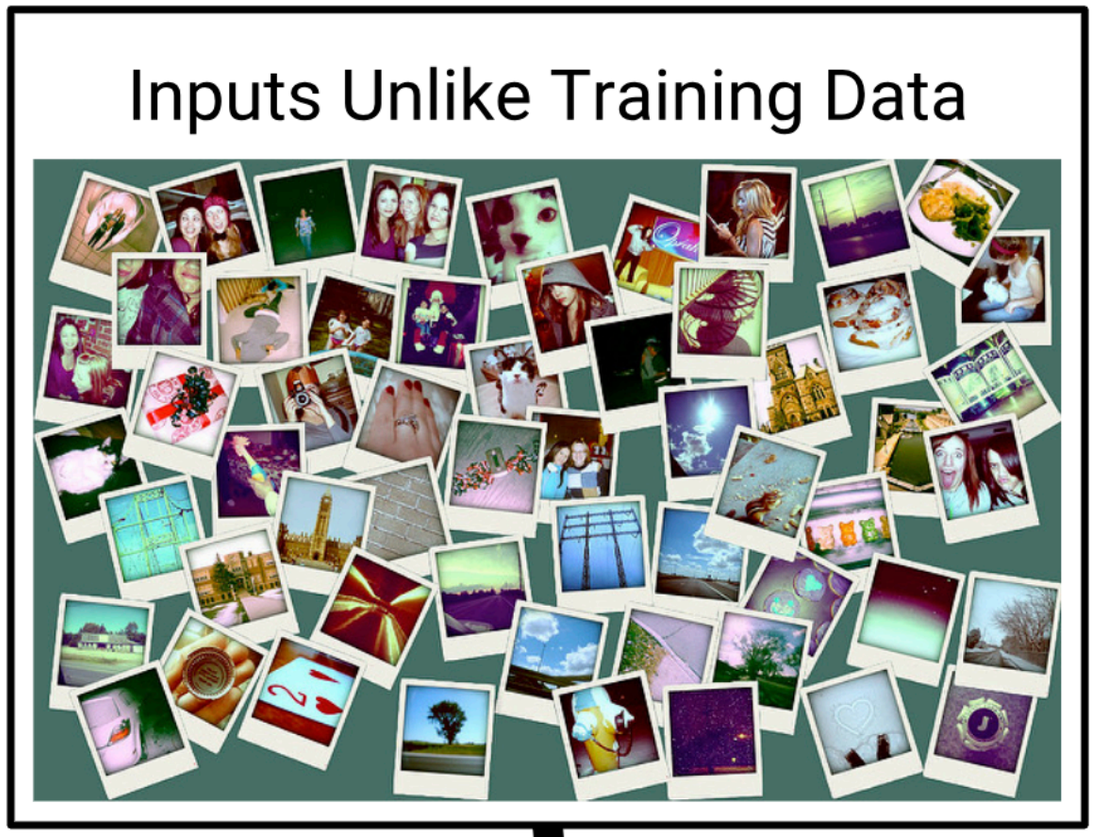
$$p(\mathbf{y}, \mathbf{X}) = p(\mathbf{y}|\mathbf{X}; \boldsymbol{\theta})\ p(\mathbf{X}; \boldsymbol{\phi})$$

Predictive Model

Generative Model







$$\text{if } p(\boldsymbol{x}^*; \boldsymbol{\phi}) < \tau,$$

$$\text{then } \texttt{reject } \boldsymbol{x}^*$$

Use p(**X**) model to reject inputs with density below some threshold [Bishop, 1994].

Novelty Detection and Neural Network Validation

Chris M. Bishop (May 1994)

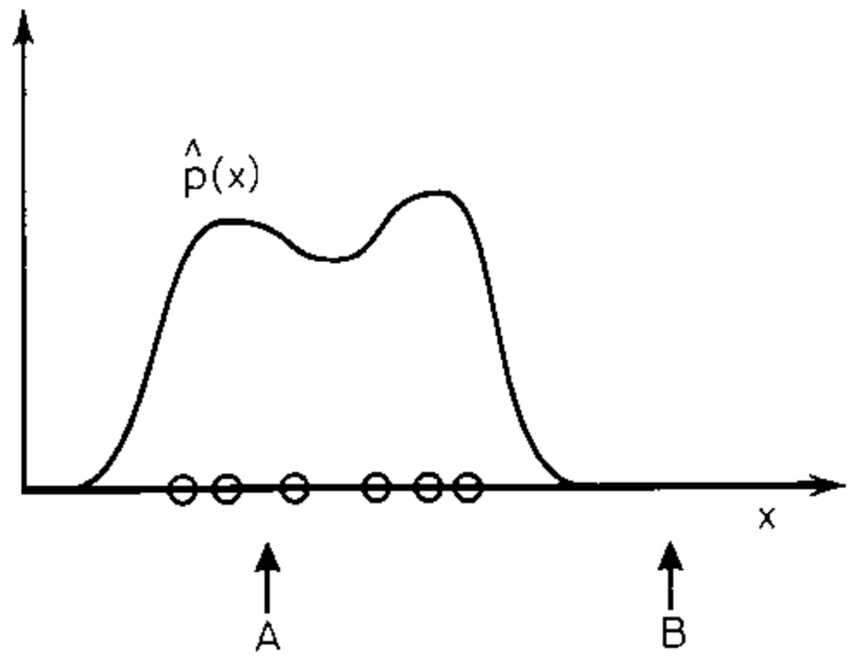


Fig. 1 *Use of the unconditional probability density to measure the degree of novelty of new input vectors*

Novelty Detection and Neural Network Validation

Chris M. Bishop (May 1994)

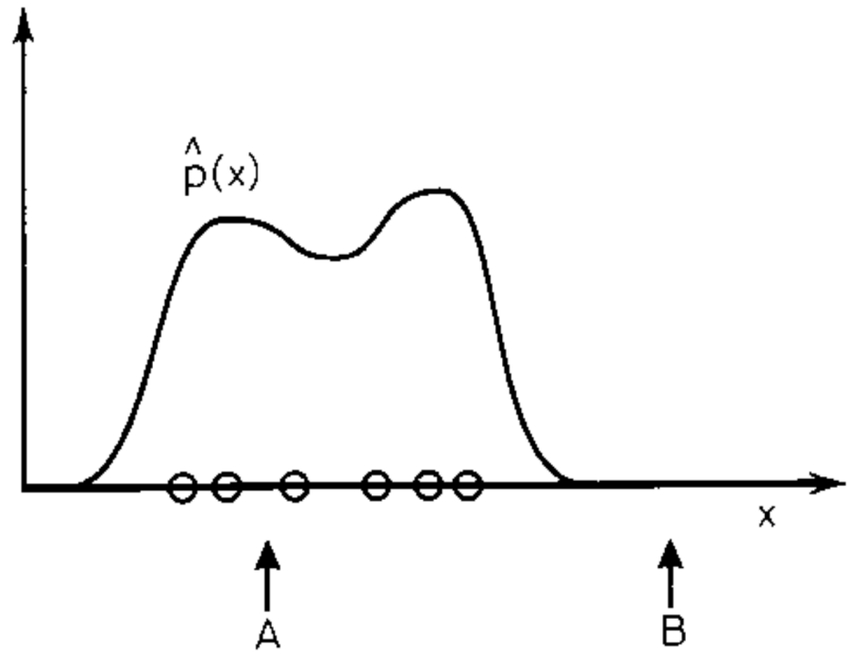


Fig. 1 *Use of the unconditional probability density to measure the degree of novelty of new input vectors*

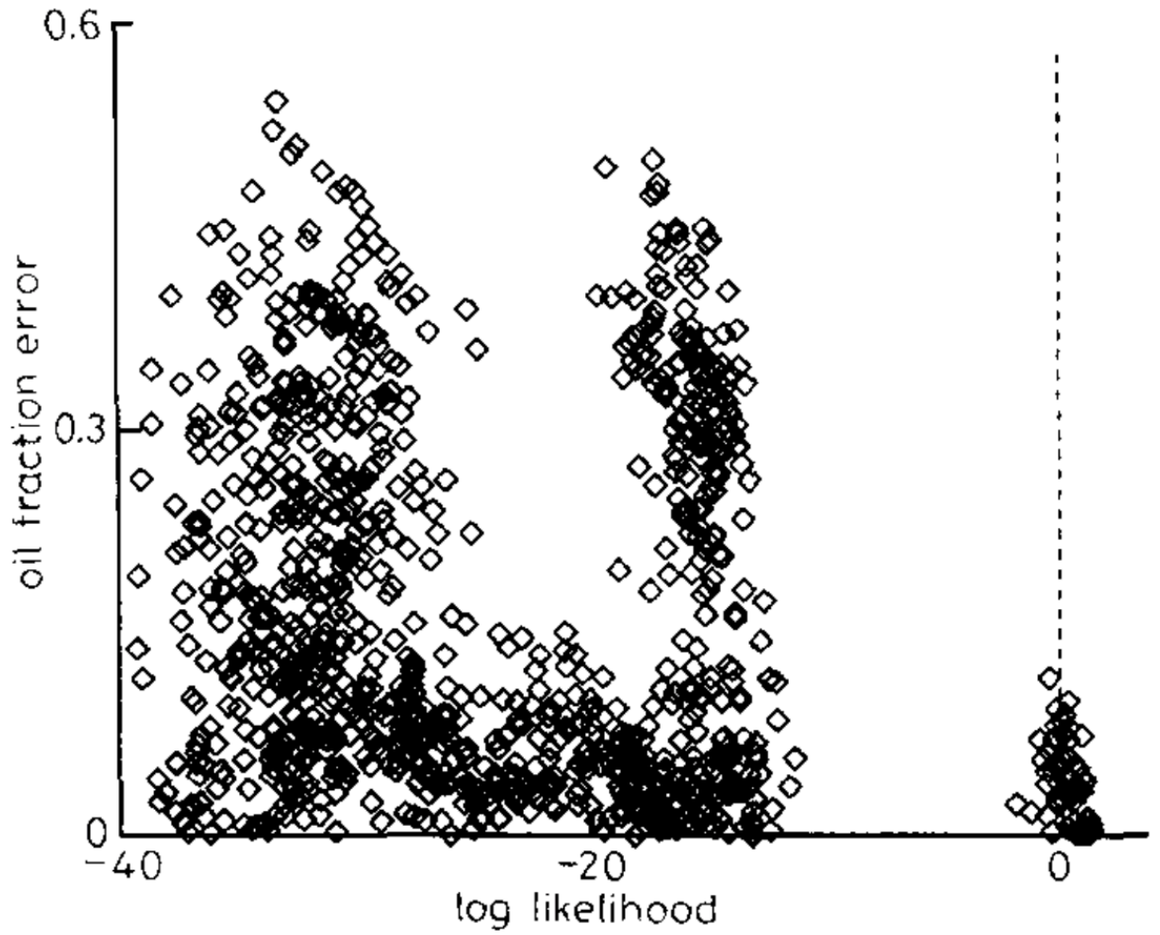


Fig. 7 *Magnitude of the oil fraction error from the neural network prediction against the log likelihood from the novelty detector*

Panel Discussion

Advances in Approximate Bayesian Inference, Dec 2017

Panel Discussion

Advances in Approximate Bayesian Inference, Dec 2017

Panel Discussion

Advances in Approximate Bayesian Inference, Dec 2017

Panel Discussion

Advances in Approximate Bayesian Inference, Dec 2017

Deep Generative Models

- [ ] **Autoregressive Models** (e.g. WaveNet)

  [Larochelle & Murray, AISTATS 2011; van den Oord, ICLM 2016; van den Oord, NeurIPS 2016]

- [ ] **Variational Autoencoders** (VAEs)

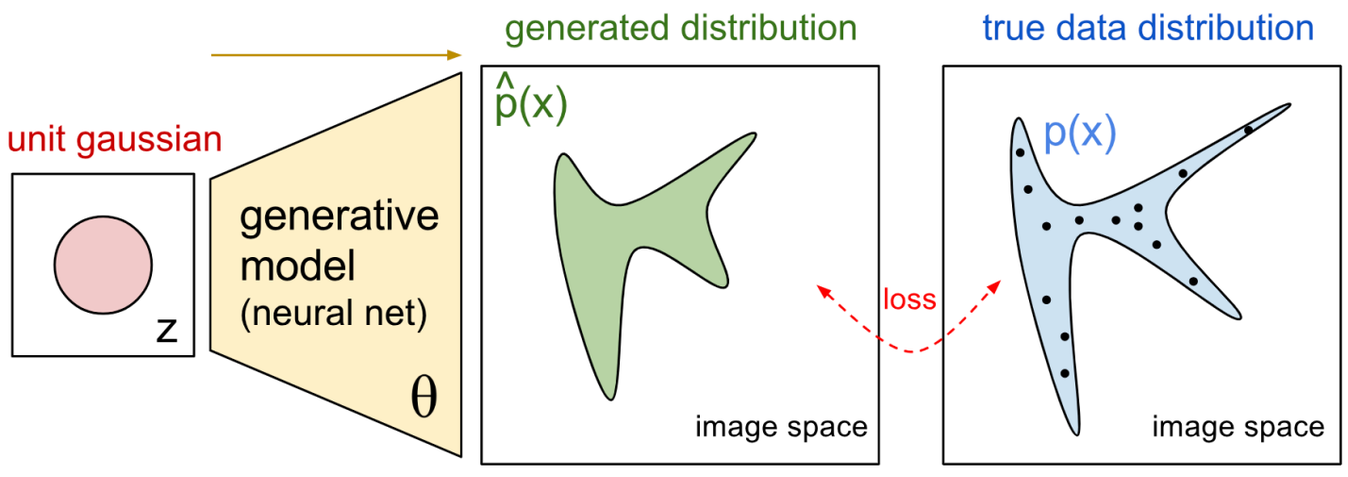
  [Kingma & Welling, ICLR 2014; Rezende et al., ICML 2014]

- [ ] **Generative Adversarial Networks** (GANs)

  [Goodfellow et al., NeurIPS 2014]

- [ ] **Normalizing Flows**

  [Tabak & Turner, 2013; Rezende & Mohamed, ICML 2015; Dinh et al., ICLR 2017]



https://blog.openai.com/generative-models/

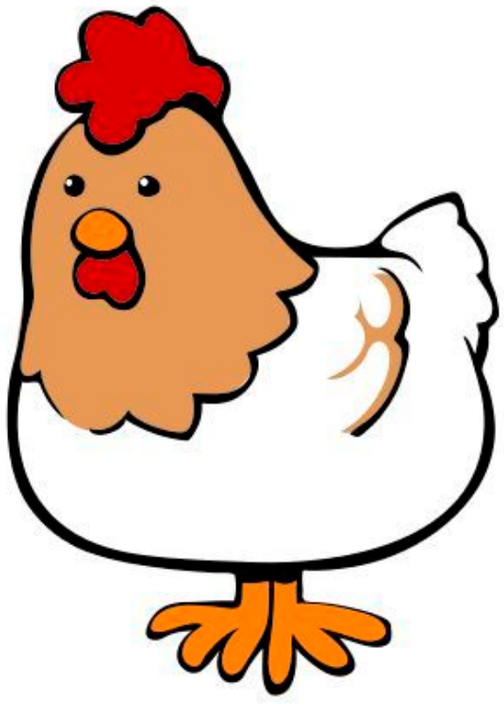
[Kingma & Dhariwal, NeurIPS 2018]

Other Applications

- [ ] **Active Learning**
- [ ] **Semi-Supervised Learning**
- [ ] **Generation and Super-Resolution**
- [ ] **Simulation for Reinforcement Learning**
- [ ] **Transfer Learning**
- [ ] **Open-Set Recognition**

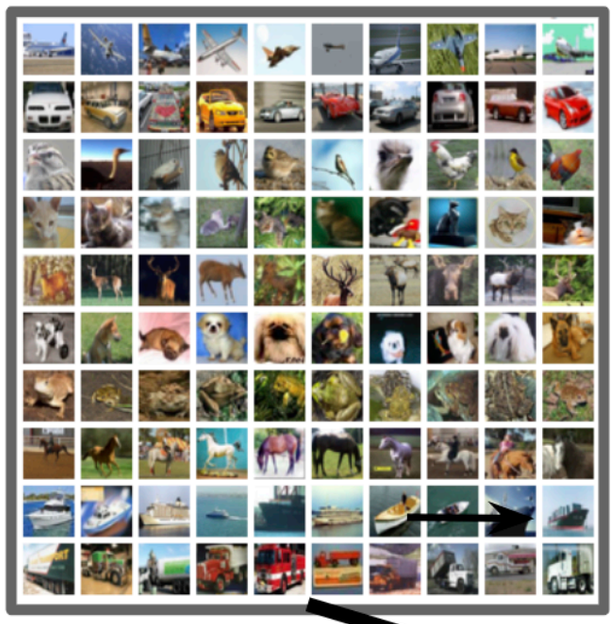
EXPERIMENT

Chicken or Seven?



Chicken or Seven?: Simulating the Scenario

Training: *CIFAR-10*



Testing: *SVHN*

GENERATIVE MODEL

$$p(\mathbf{x}_{\text{CIFAR-10}}) \overset{?}{>} p(\mathbf{x}_{\text{SVHN}})$$

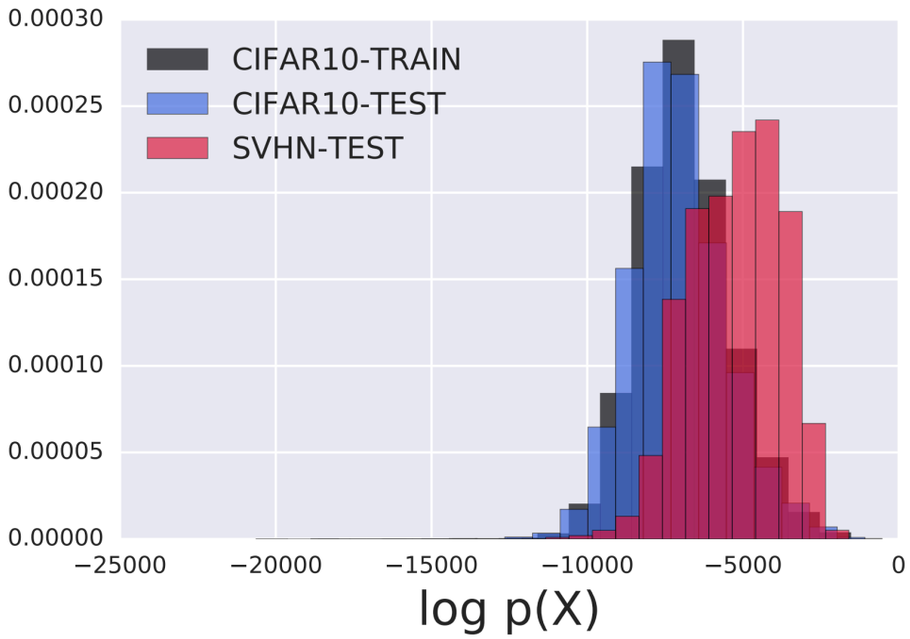
Experiment: CIFAR-10 vs SVHN

Variational Autoencoder

PixelCNN

Glow (Normalizing Flow)

Experiment: CIFAR-10 vs SVHN

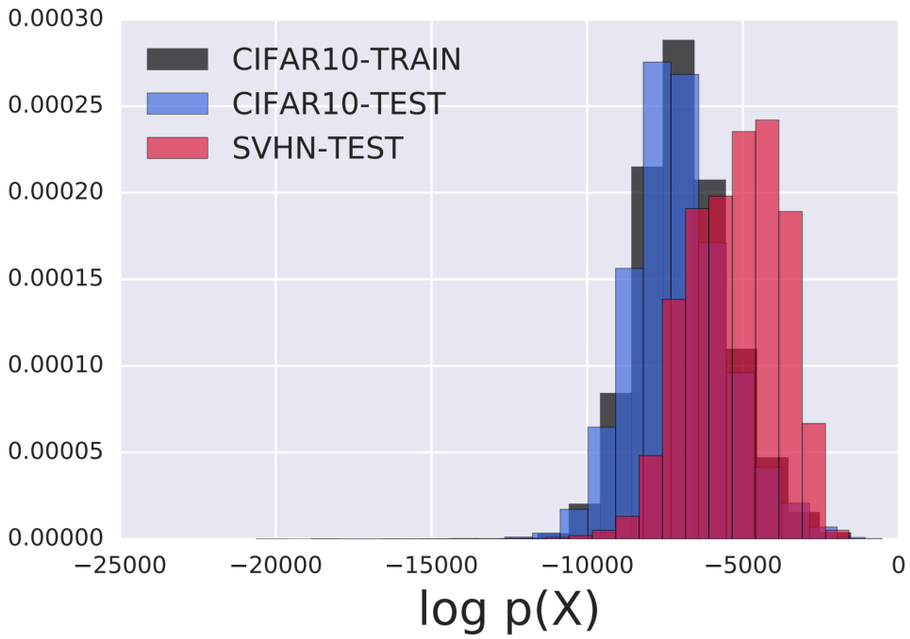


Variational Autoencoder

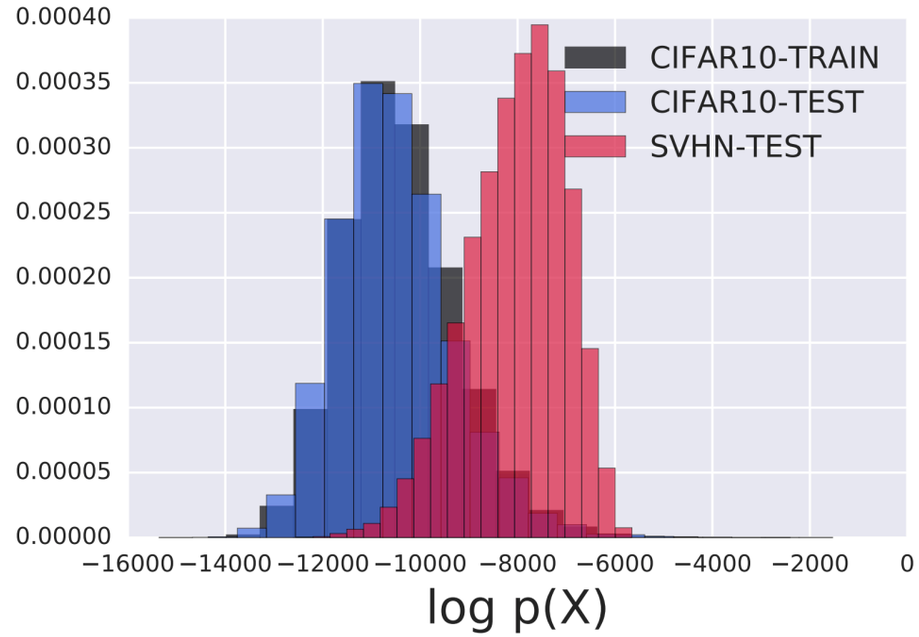
PixelCNN

Glow (Normalizing Flow)

Experiment: CIFAR-10 vs SVHN



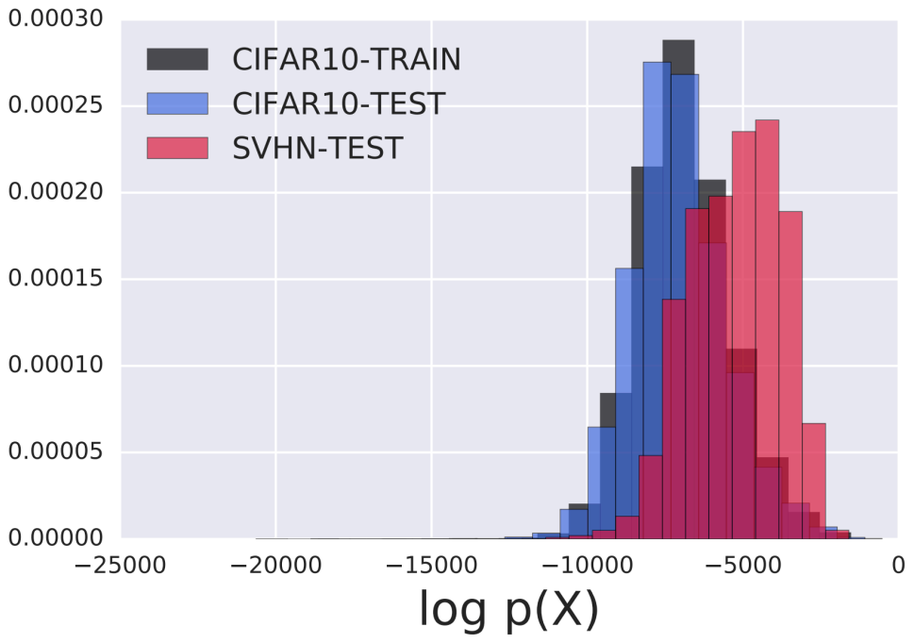
Variational Autoencoder



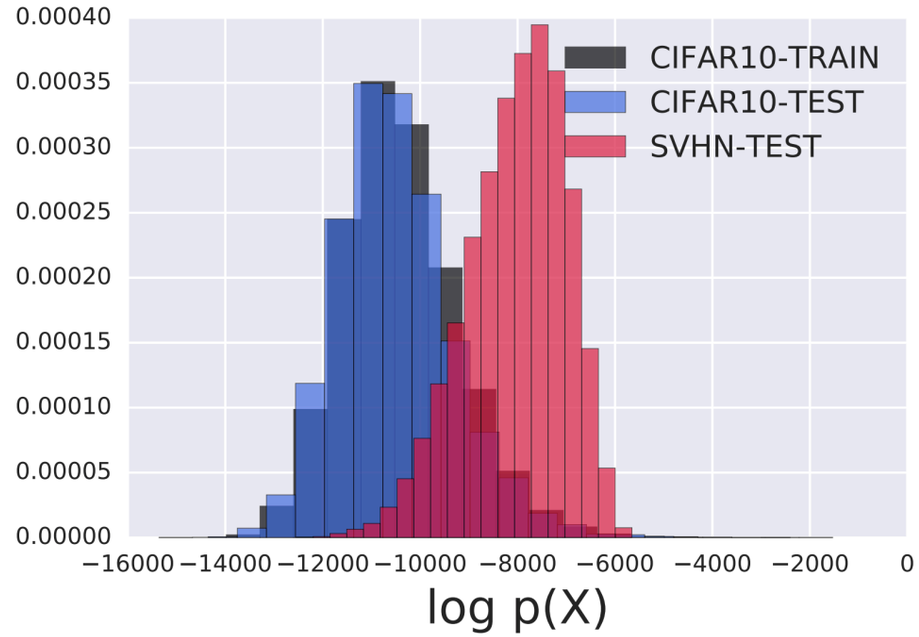
PixelCNN

Glow (Normalizing Flow)

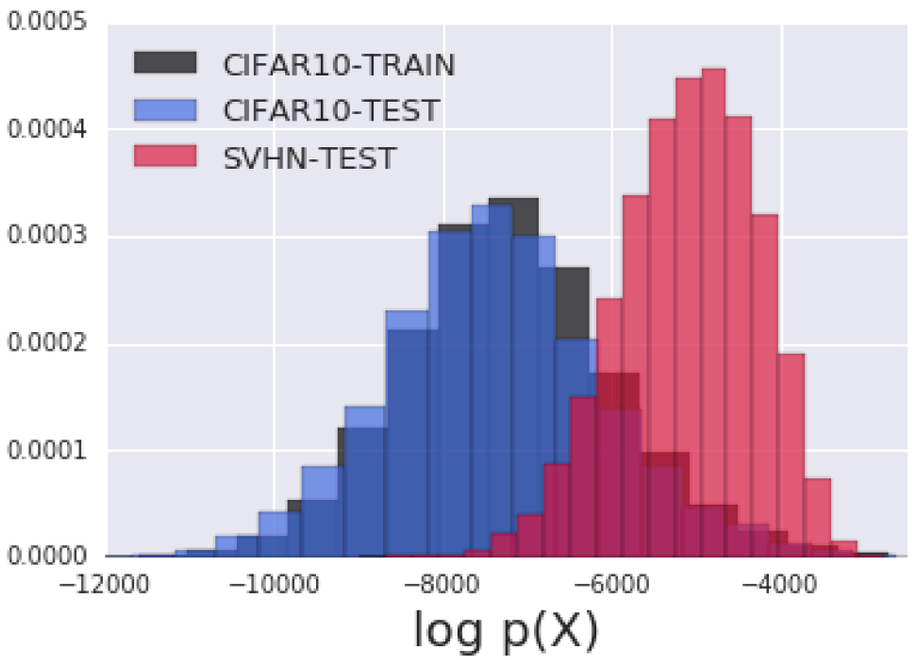
Experiment: CIFAR-10 vs SVHN



Variational Autoencoder

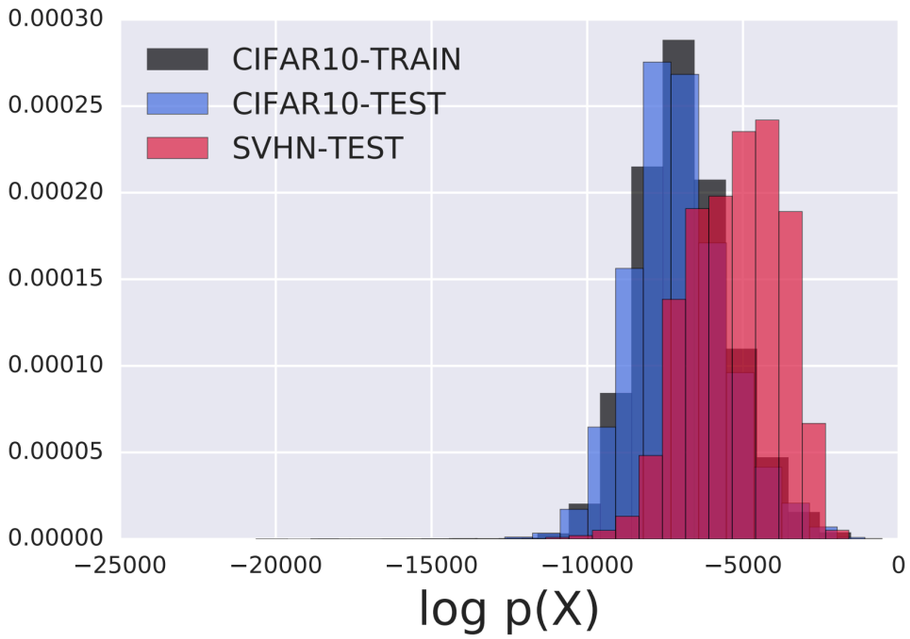


PixelCNN

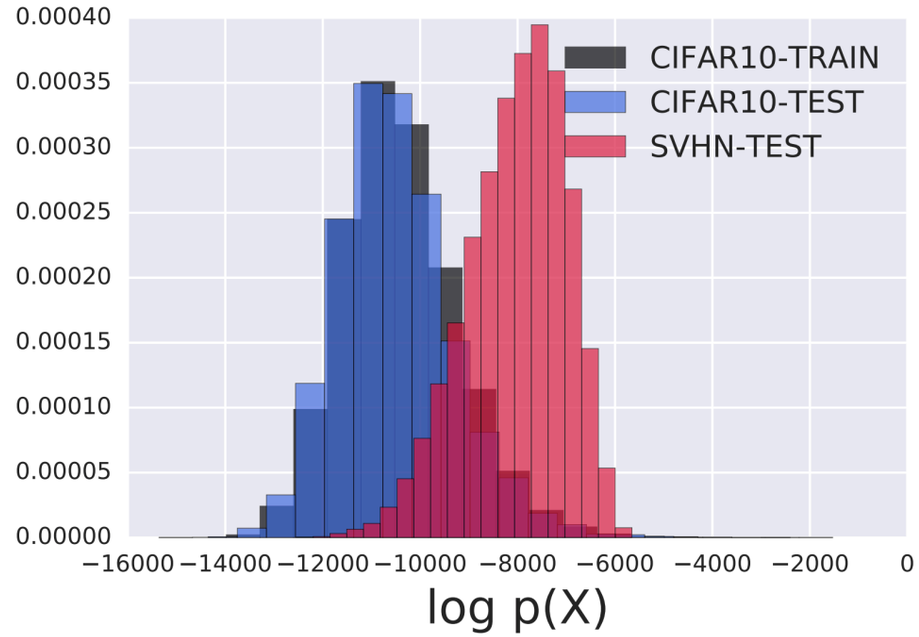


Glow (Normalizing Flow)

Experiment: CIFAR-10 vs SVHN

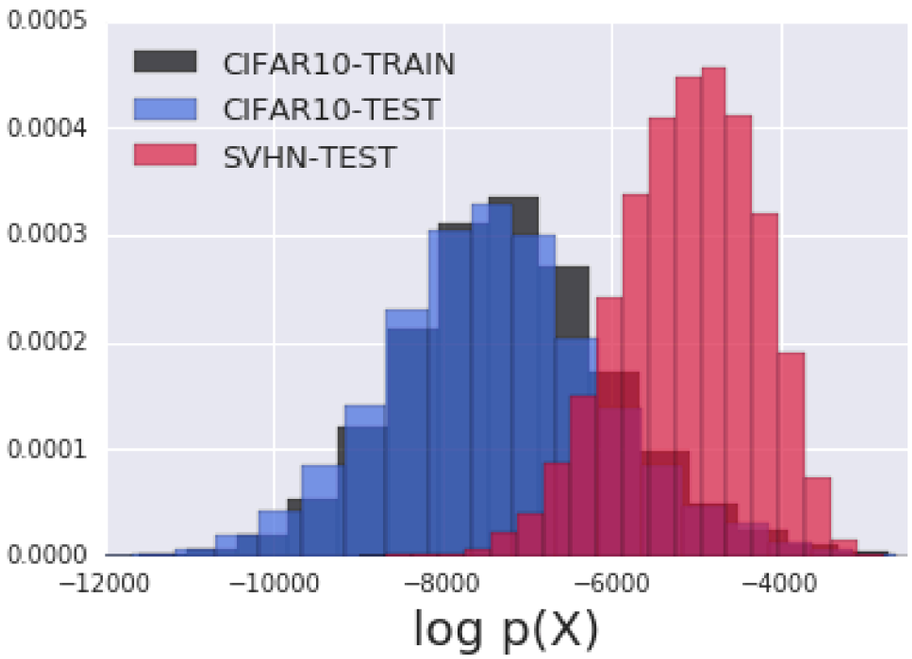


Variational Autoencoder



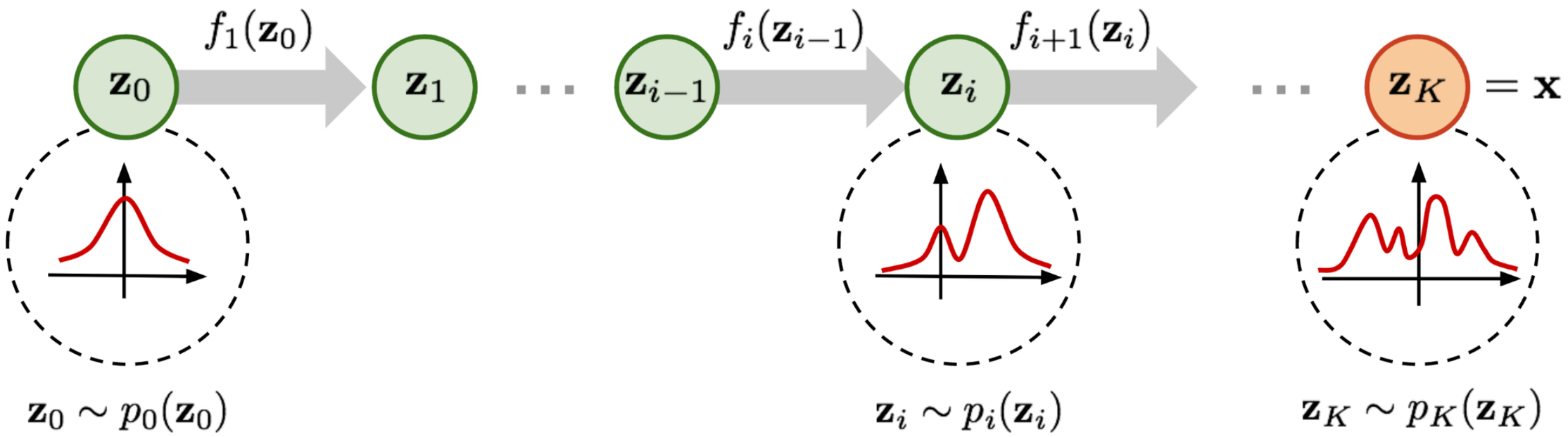
PixelCNN

- Exact likelihood calculations
- Additional structure (e.g. invertibility)



Glow (Normalizing Flow)

Digging Deeper into Flow-Based Models



Deriving Flow-Based Generative Models

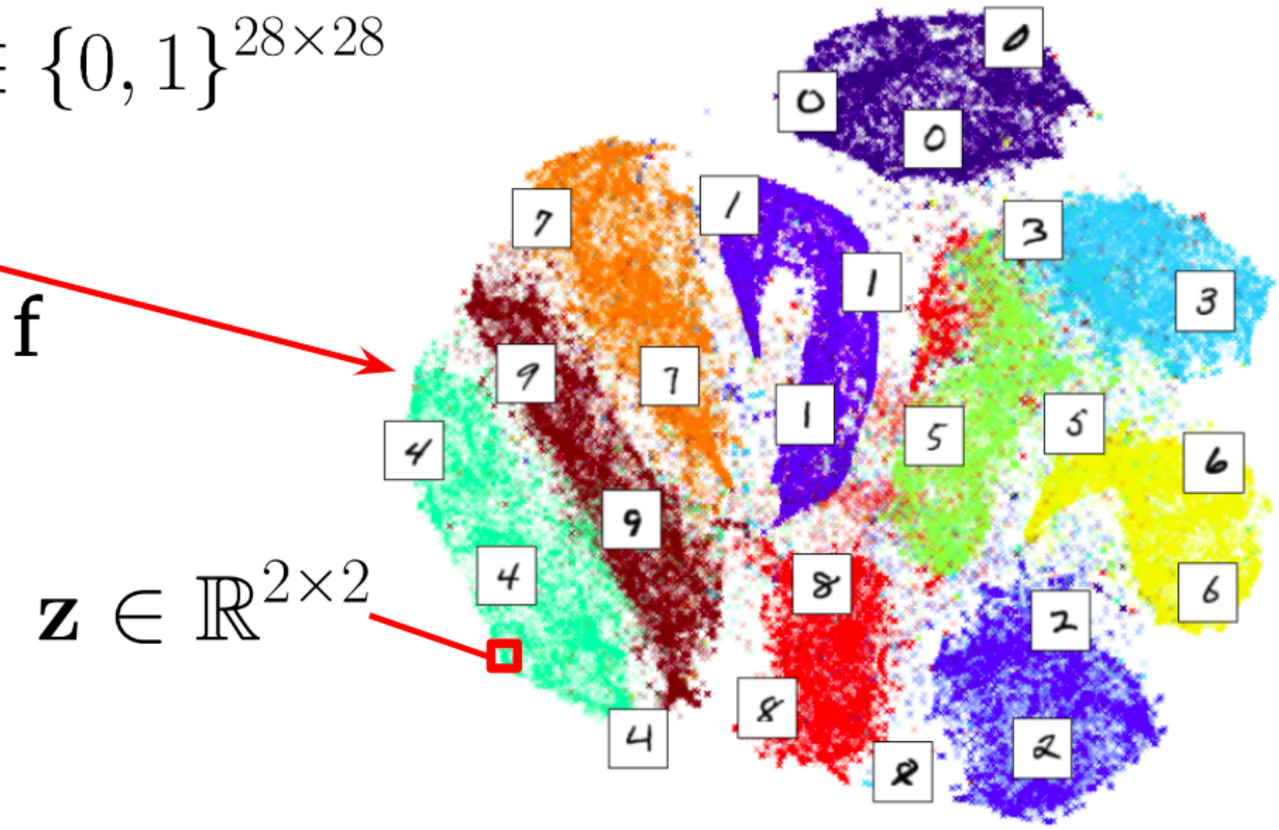
Latent variable models allow data to be represented in some auxiliary space that hopefully better captures the underlying semantics.

MNIST Dataset

$\mathbf{x} \in \{0, 1\}^{28 \times 28}$

$\mathbf{f}$

$\mathbf{z} \in \mathbb{R}^{2 \times 2}$



MNIST Latent Space

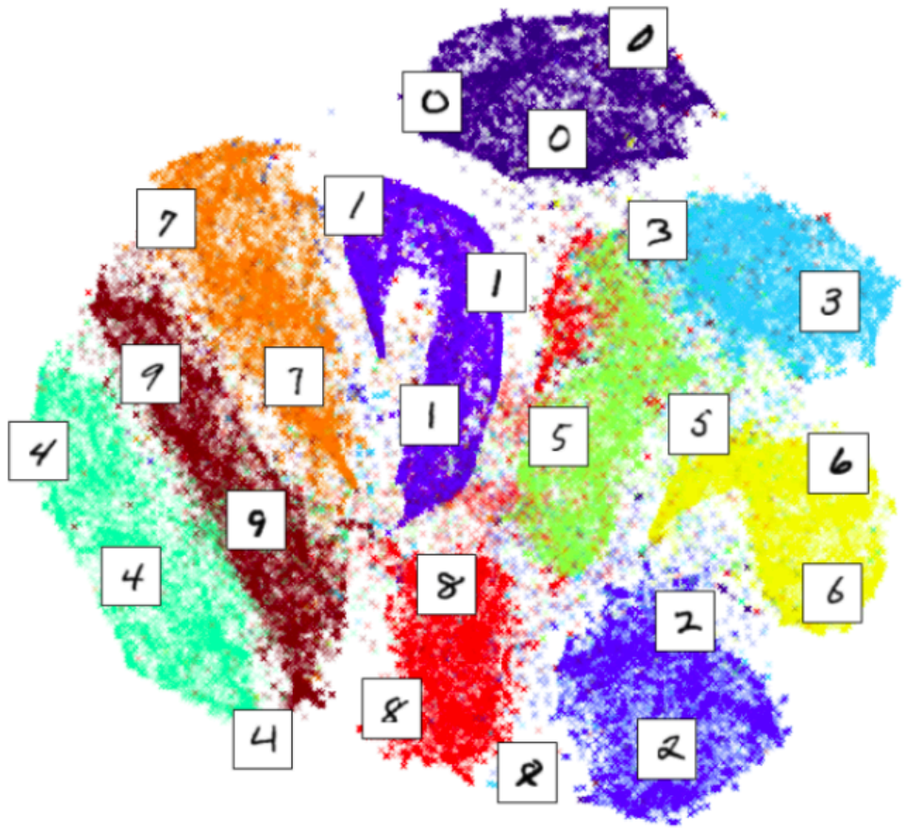
Deriving Flow-Based Generative Models

Latent variable models allow data to be represented in some auxiliary space that hopefully better captures the underlying semantics.

Most latent variable models define this latent / auxiliary space in terms of joint probabilities:

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) \, d\mathbf{z} = \int p(\mathbf{x}|\boldsymbol{\mu} = f(\mathbf{z})) \, p(\mathbf{z}) \, d\mathbf{z}$$

Examples:
(Probabilistic) *Principal Components Analysis* (PCA)
Deep Belief Networks
Topic Models (LDA)
Variational Autoencoders (VAEs)



Deriving Flow-Based Generative Models

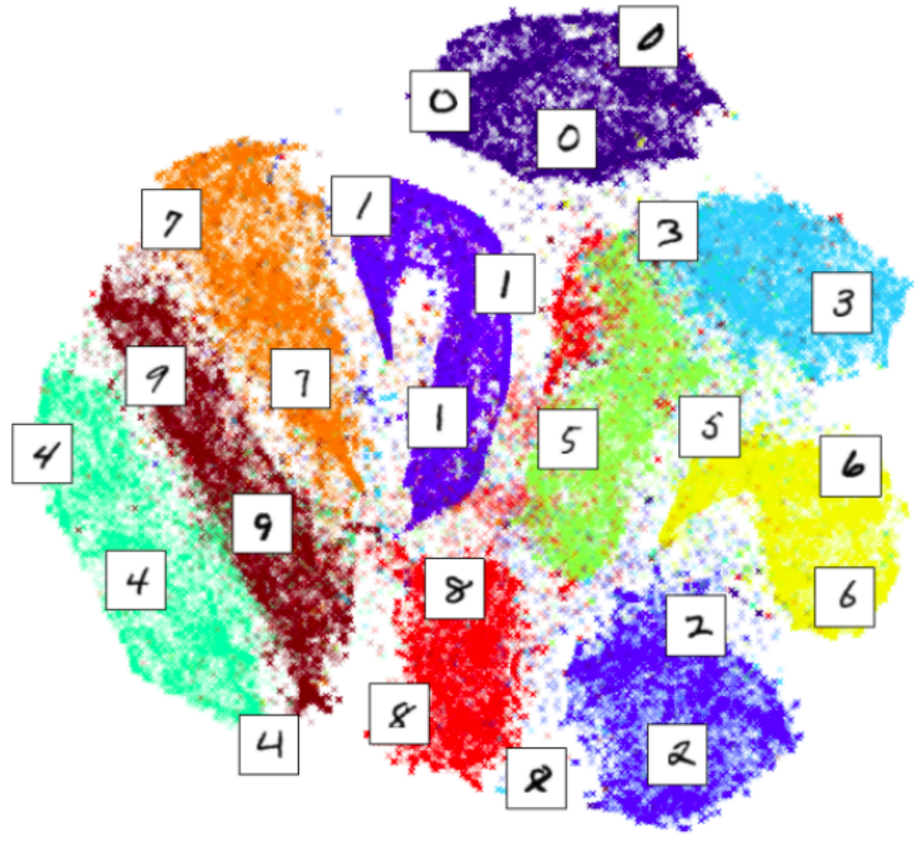
Latent variable models allow data to be represented in some auxiliary space that hopefully better captures the underlying semantics.

However, that's not the only way to define a latent variables model. We can define ***X*** as the transformation itself:

$$p(\mathbf{x}|\boldsymbol{\mu} = f(\mathbf{z})) \qquad \mathbf{x} = f(\mathbf{z})$$

Examples:

Standard Normal Transform (i.e. **(x-μ)/σ**)
Independent Component Analysis (ICA)
Generative Adversarial Networks (GANs)
Copulas
Flow-Based Generative Models



Deriving Flow-Based Generative Models

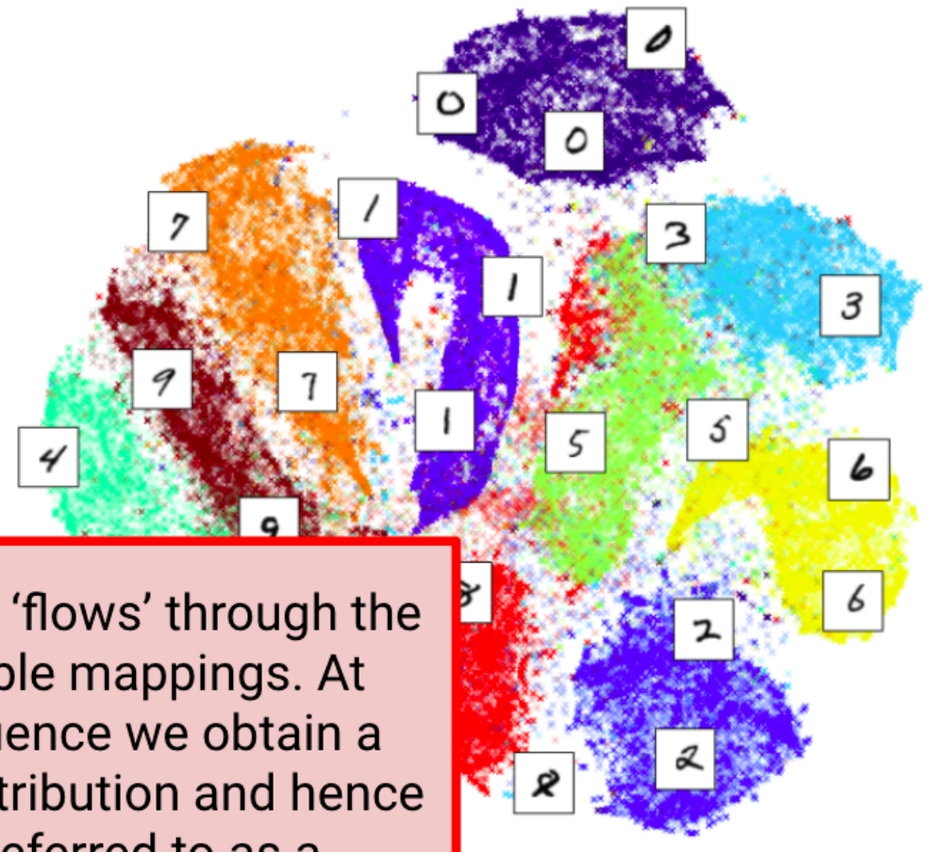
Latent variable models allow data to be represented in some auxiliary space that hopefully better captures the underlying semantics.

However, that's not the only way to define a latent variables model. We can define ***X*** as the transformation itself:

$$p(\mathbf{x}|\boldsymbol{\mu} = f(\mathbf{z})) \qquad \mathbf{x} = f(\mathbf{z})$$

Examples:

Standard Normal Transform (i.e. **(x-μ)/σ**)
Independent Component Analysis (ICA)
Generative Adversarial Networks (GANs)
Copulas
Flow-Based Generative Models



"...the initial density 'flows' through the sequence of invertible mappings. At the end of this sequence we obtain a valid probability distribution and hence this type of flow is referred to as a normalizing flow." [Rezende & Mohamed, 2016]

Deriving Flow-Based Generative Models

Definition of a probability density function (PDF):

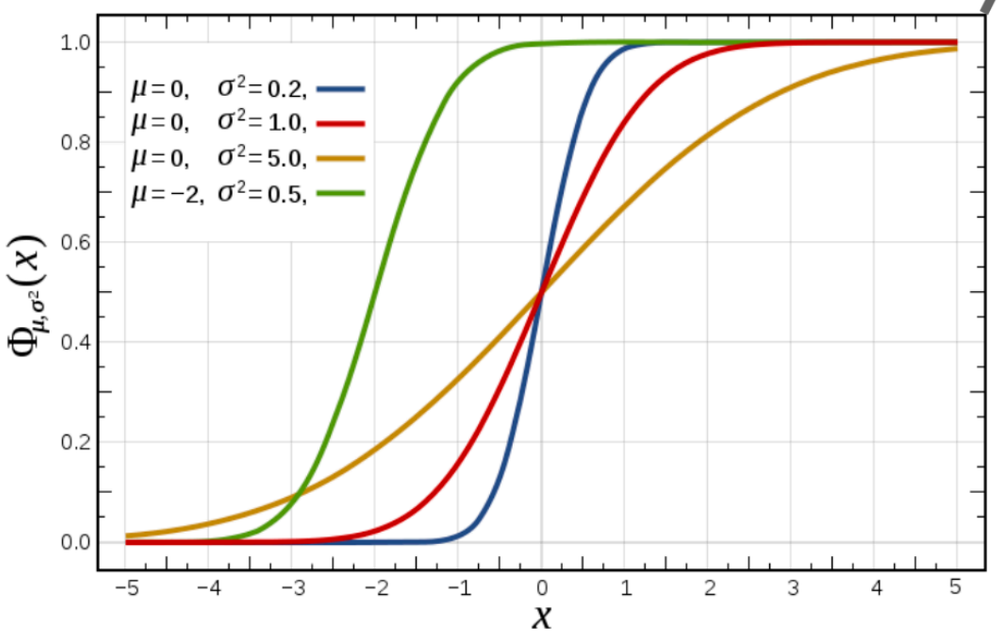
$$\frac{d}{dZ}P(Z) = p(Z)$$

Deriving Flow-Based Generative Models

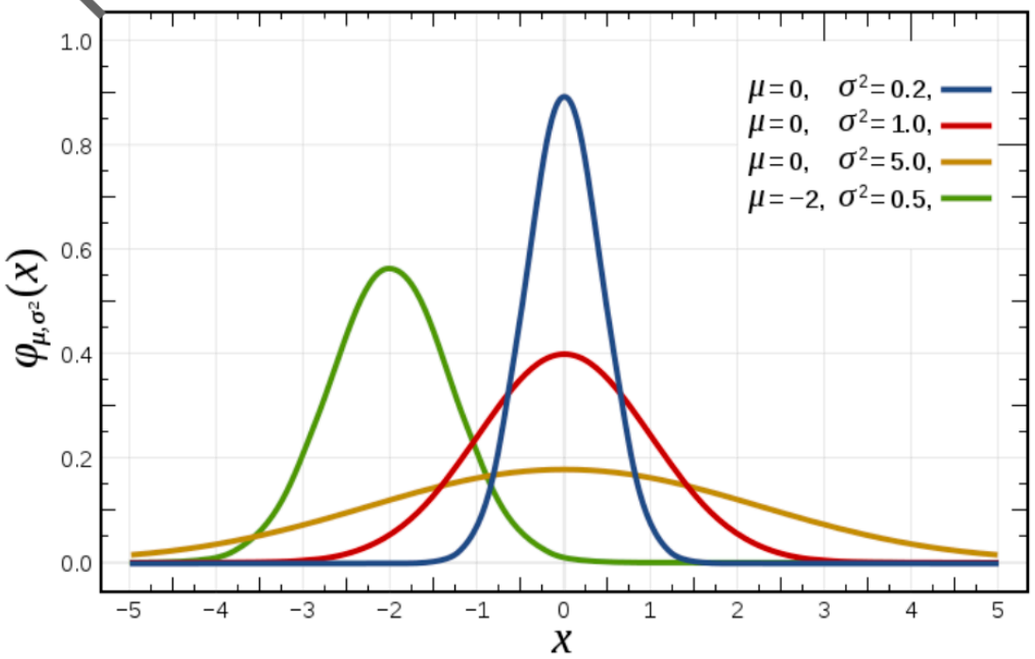
Definition of a probability density function (PDF):

$$\frac{d}{dZ}P(Z) = p(Z)$$

Gaussian CDF



Gaussian PDF



Deriving Flow-Based Generative Models

Definition of a probability density function (PDF):

$$\frac{d}{dZ}P(Z) = p(Z)$$

Consider defining Z by a transformation of another variable X:

$$Z = f(X)$$

Deriving Flow-Based Generative Models

Definition of a probability density function (PDF):

$$\frac{d}{dZ}P(Z) = p(Z)$$

Consider defining Z by a transformation of another variable X:

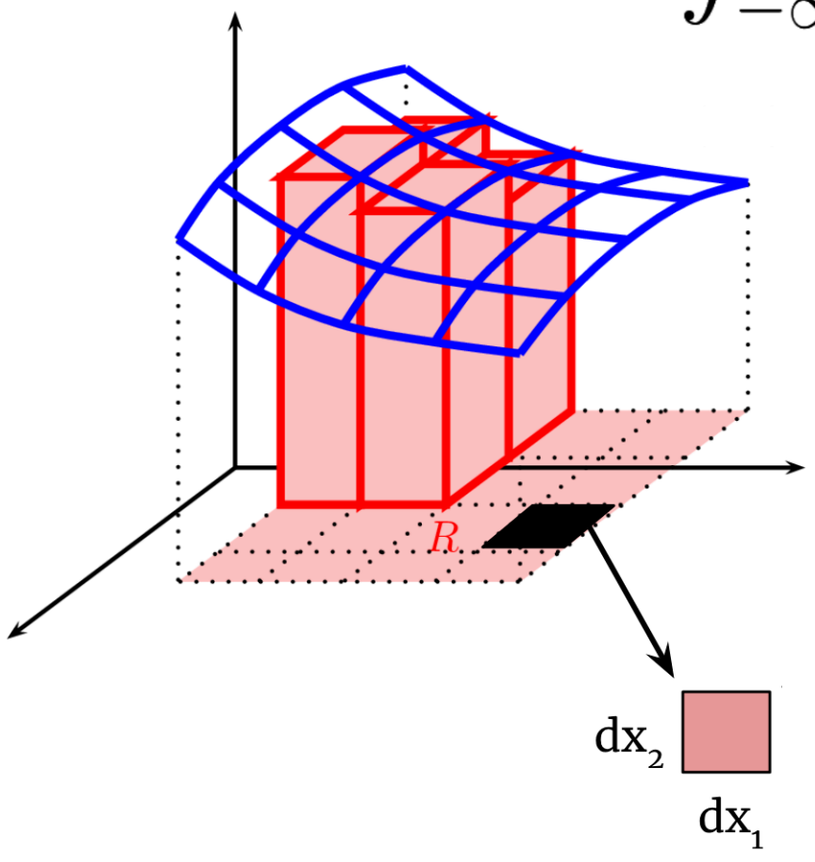
$$Z = f(X)$$

Change of Variables Formula (X → Z):

$$\frac{d}{dX}P_z(f(X)) = p_z(f(X))\left|\frac{df(X)}{dX}\right| = p(X)$$

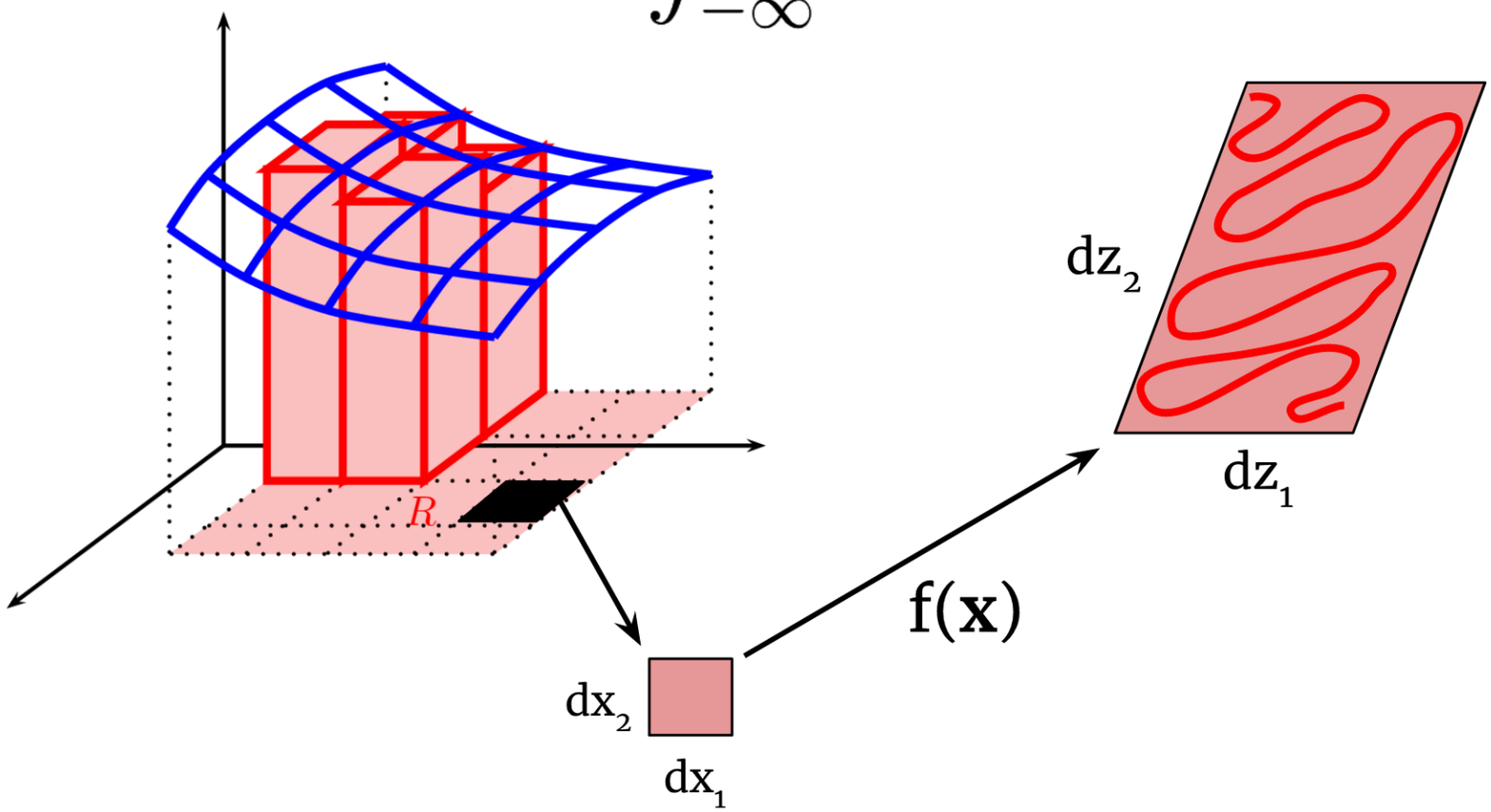
Deriving Flow-Based Generative Models

$$P(\mathbf{x}_0) = \int_{-\infty}^{\mathbf{x}_0} p(\mathbf{x})\ d\mathbf{x}$$



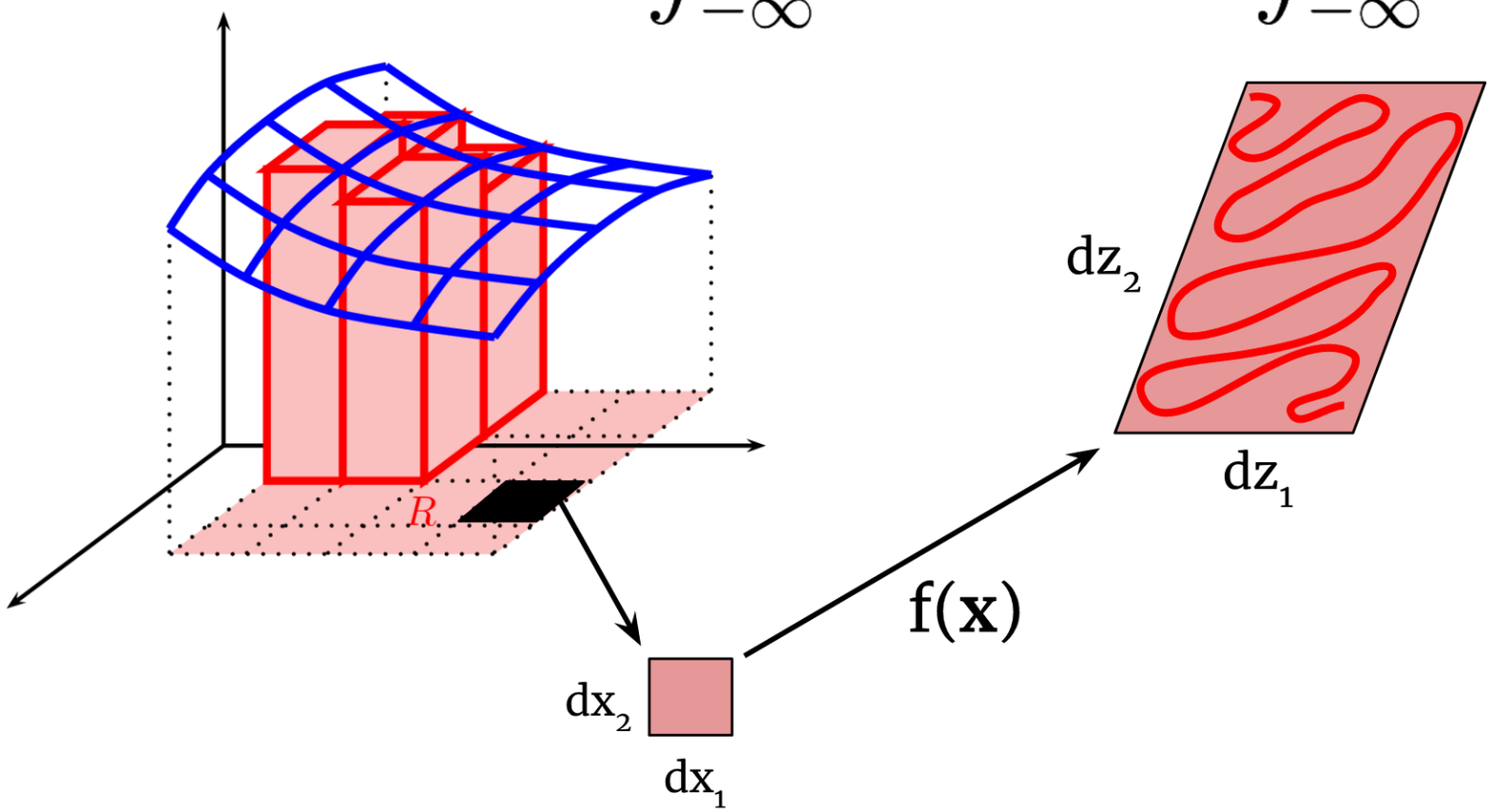
Deriving Flow-Based Generative Models

$$P(\mathbf{x}_0) = \int_{-\infty}^{\mathbf{x}_0} p(\mathbf{x})\ d\mathbf{x}$$



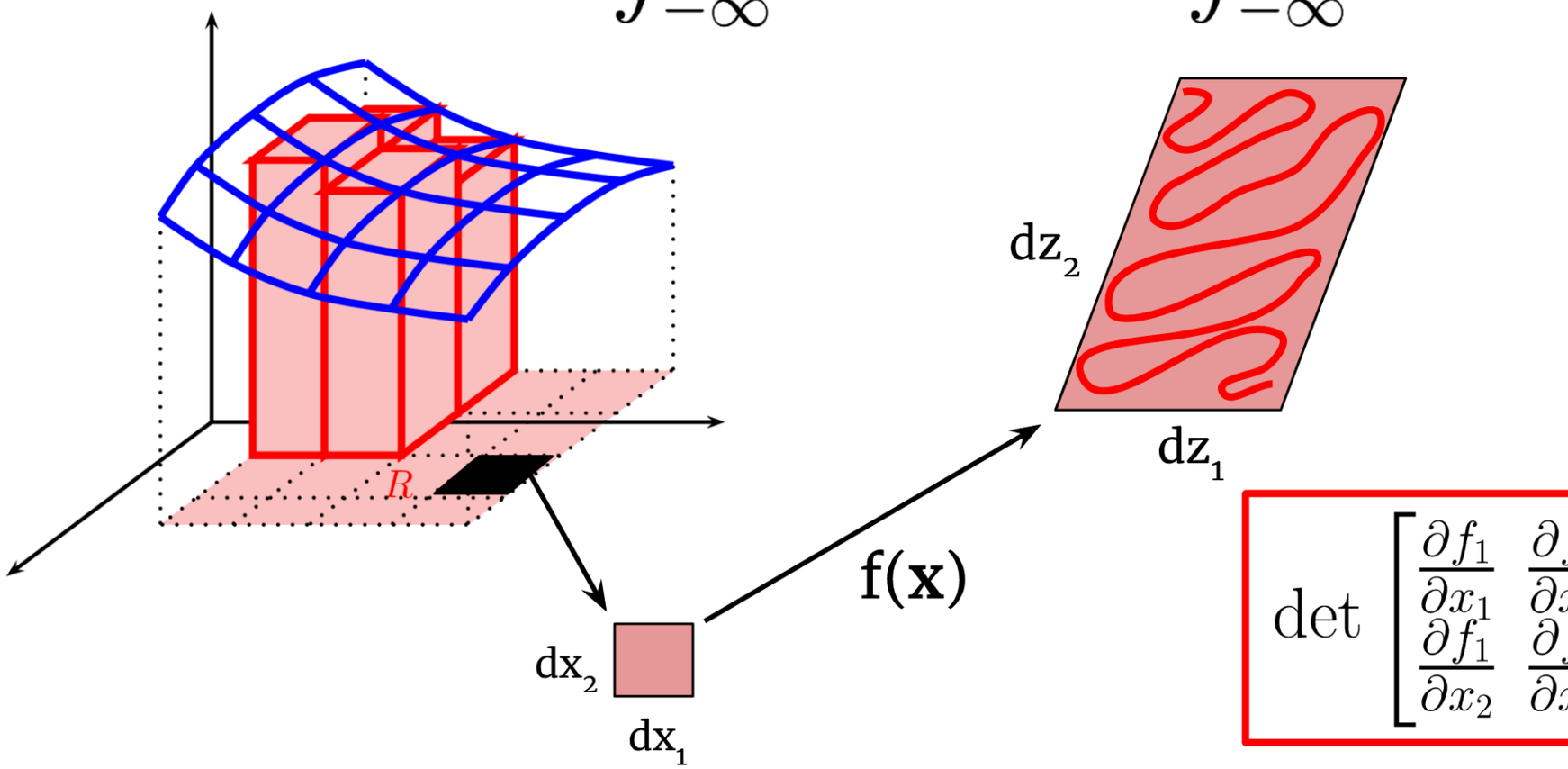
Deriving Flow-Based Generative Models

$$P(\mathbf{x}_0) = \int_{-\infty}^{\mathbf{x}_0} p(\mathbf{x})\, d\mathbf{x} = \int_{-\infty}^{\mathbf{x}_0} p_Z(f(\mathbf{x})) \underbrace{\left|\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|}_{\text{Volume Correction}} d\mathbf{x}$$



Deriving Flow-Based Generative Models

$$P(\mathbf{x}_0) = \int_{-\infty}^{\mathbf{x}_0} p(\mathbf{x})\, d\mathbf{x} = \int_{-\infty}^{\mathbf{x}_0} p_Z(f(\mathbf{x})) \underbrace{\left|\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|}_{\text{Volume Correction}} d\mathbf{x}$$



$$\det \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \frac{\partial f_1}{\partial x_1}\frac{\partial f_2}{\partial x_2} - \frac{\partial f_2}{\partial x_1}\frac{\partial f_1}{\partial x_2}$$

Deriving Flow-Based Generative Models

Change of Variables Formula (X → Z):

$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

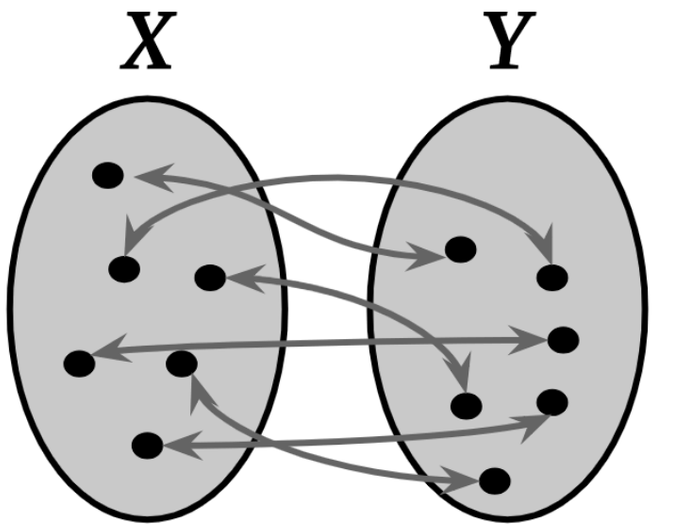
Deriving Flow-Based Generative Models

Change of Variables Formula (X → Z):

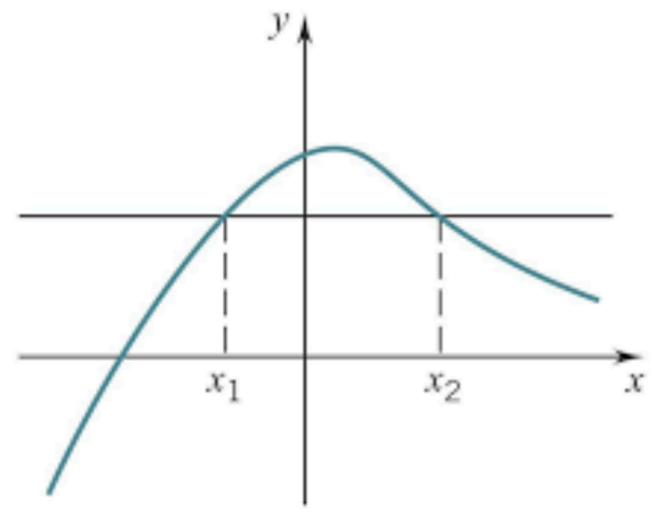
$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

So what's the catch?

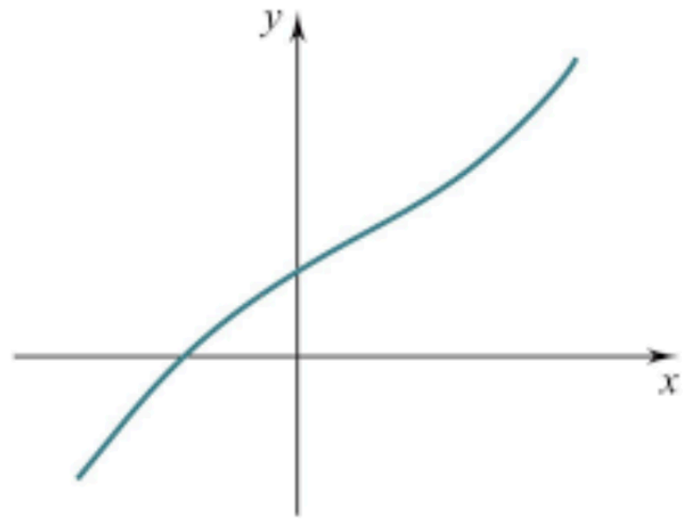
f(**x**) must be a *bijection*



$\mathbf{x} = \mathbf{f}^{-1}(\mathbf{y})$ $\mathbf{y} = \mathbf{f}(\mathbf{x})$



f is not one-to-one: $f(x_1) = f(x_2)$



f is one-to-one:

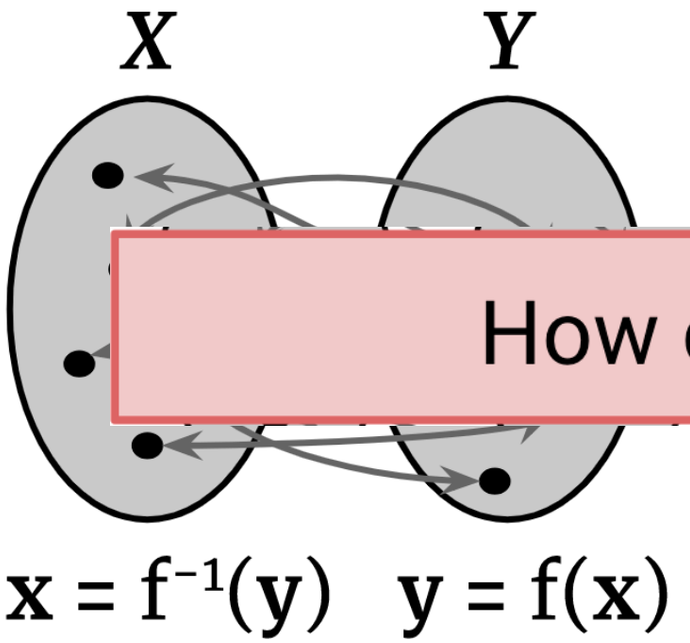
Deriving Flow-Based Generative Models

Change of Variables Formula (X → Z):

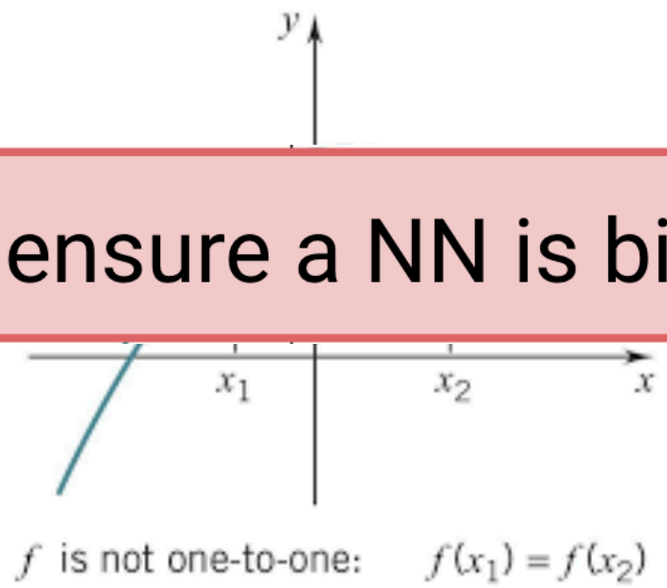
$$p_z(f(X)) \left| \frac{df(X)}{dX} \right| = p(X)$$

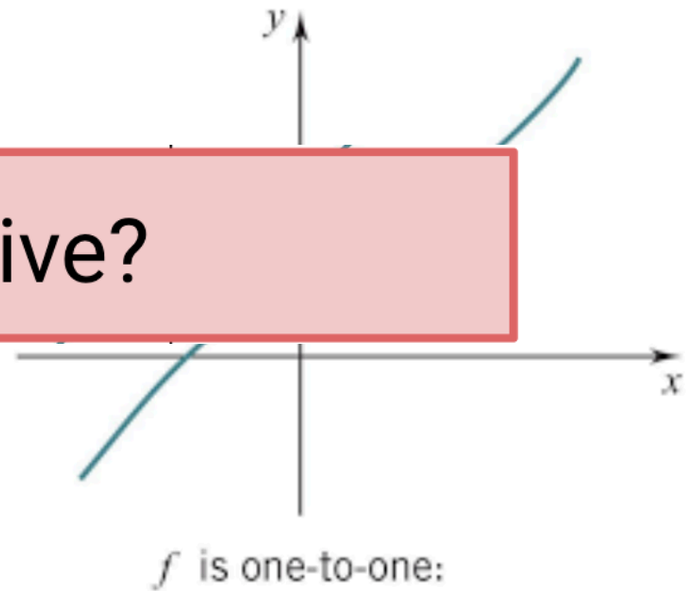
So what's the catch?

f(**x**) must be a *bijection*



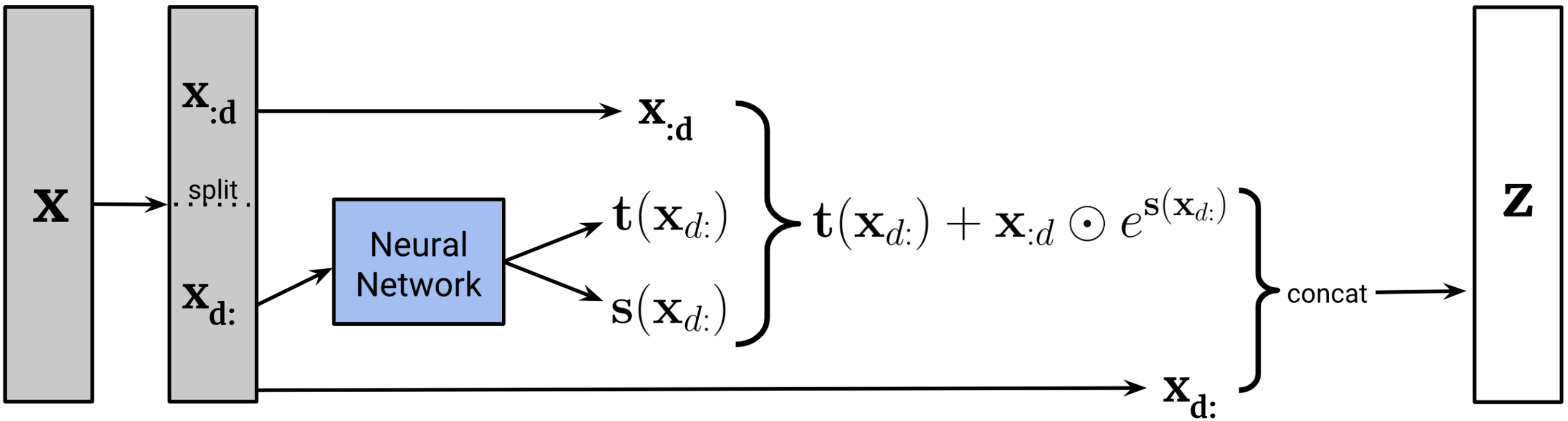
How can we ensure a NN is bijective?





The Affine Coupling Layer

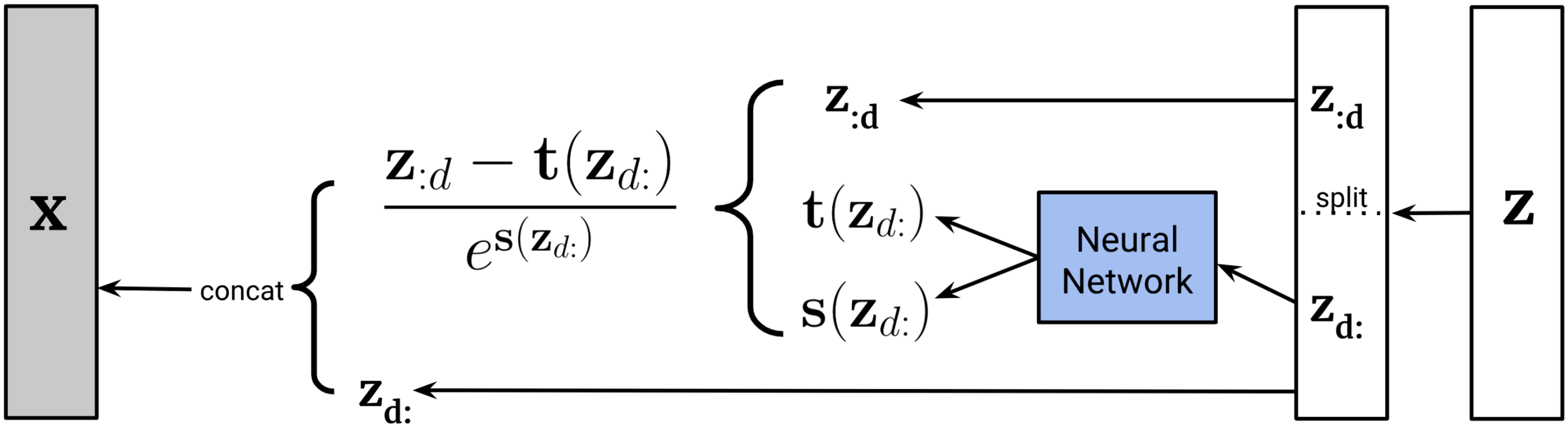
[Dinh et al., ICLR 2017]



$$\mathbf{z} = \left[\mathbf{t}(\mathbf{x}_{d:}) + \mathbf{x}_{:d} \odot e^{\mathbf{s}(\mathbf{x}_{d:})}, \mathbf{x}_{d:}\right]$$

The Affine Coupling Layer: Inverse

[Dinh et al., ICLR 2017]



$$\mathbf{x} = \left[\left(\mathbf{z}_{:d} - \mathbf{t}(\mathbf{z}_{d:}) \right) \odot e^{-\mathbf{s}(\mathbf{z}_{d:})}, \mathbf{z}_{d:} \right]$$

The Affine Coupling Layer: Volume Change

[Dinh et al., ICLR 2017]

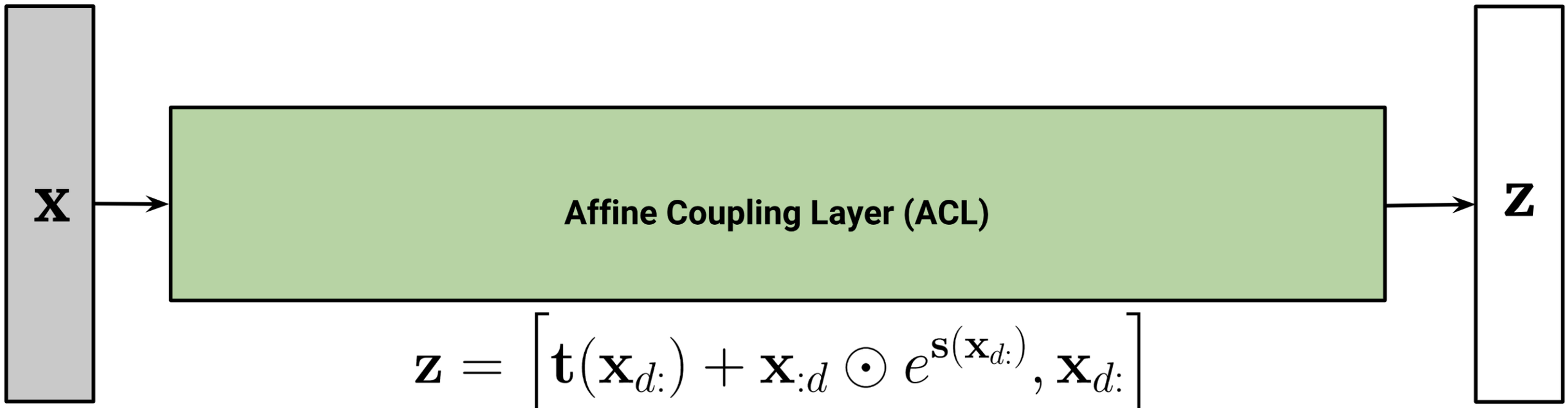
$$\mathbf{z} = \left[\mathbf{t}(\mathbf{x}_{d:}) + \mathbf{x}_{:d} \odot e^{\mathbf{s}(\mathbf{x}_{d:})}, \mathbf{x}_{d:}\right]$$

$$\begin{matrix} & \mathbf{Z}_{:d} & \mathbf{Z}_{d:} \\ \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = & \mathbf{X}_{:d} & \begin{bmatrix} e^{s(\mathbf{x}_{d:})} & \mathbf{0} \\ \frac{\partial f(\mathbf{x}_{:})}{\partial \mathbf{x}_{d:}} & \mathbf{1} \end{bmatrix} \\ & \mathbf{X}_{d:} & \end{matrix}$$

$$\left|\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right| = \prod_j e^{s_j(\mathbf{x}_{d:})} \cdot 1 = e^{\sum_j s_j(\mathbf{x}_{d:})}$$

The Affine Coupling Layer: Stacking

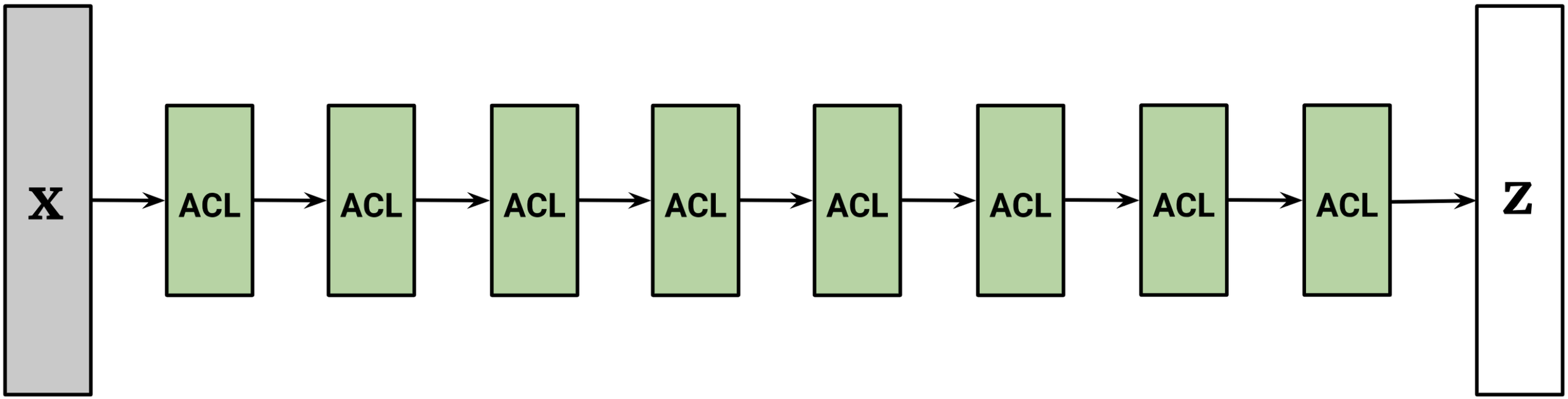
[Dinh et al., ICLR 2017]



$$\mathbf{z} = \left[\mathbf{t}(\mathbf{x}_{d:}) + \mathbf{x}_{:d} \odot e^{\mathbf{s}(\mathbf{x}_{d:})}, \mathbf{x}_{d:}\right]$$

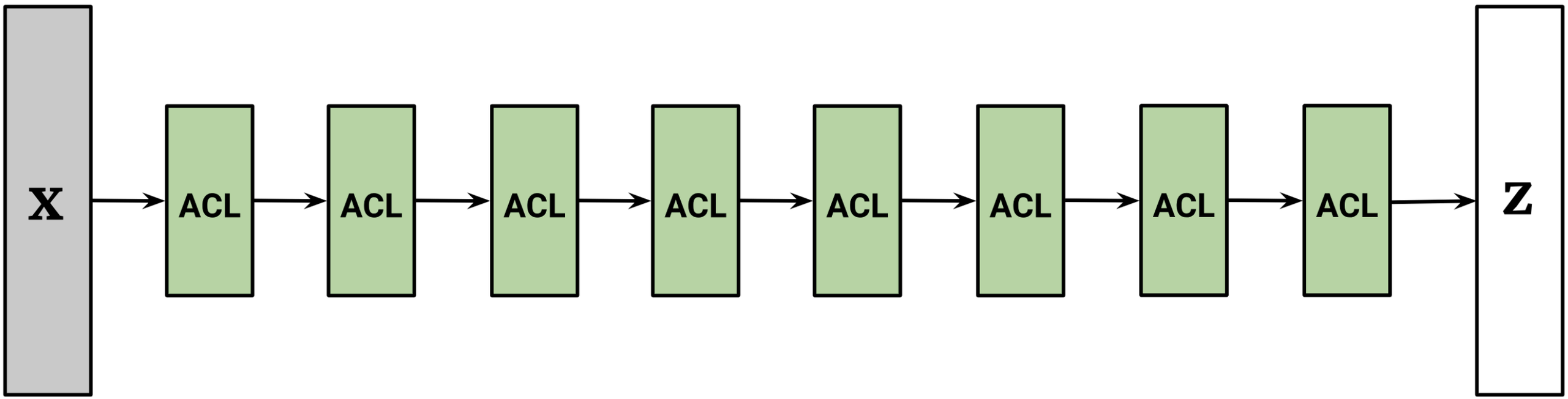
The Affine Coupling Layer: Stacking

[Dinh et al., ICLR 2017]



The Affine Coupling Layer: Stacking

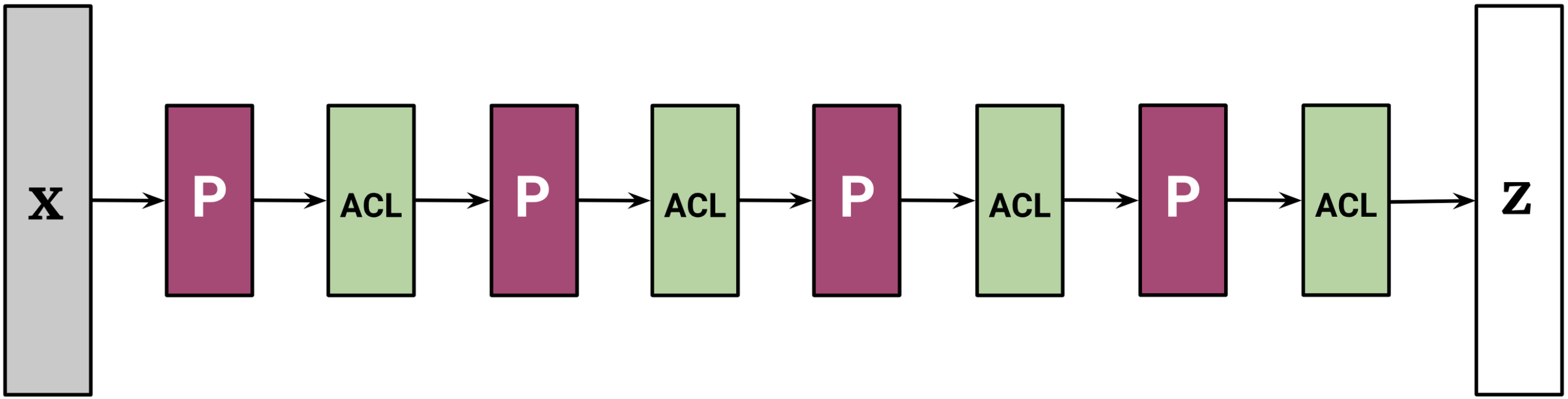
[Dinh et al., ICLR 2017]



Problem!: $\mathbf{x}_{:d} = \mathbf{z}_{:d}$

The Affine Coupling Layer: Stacking

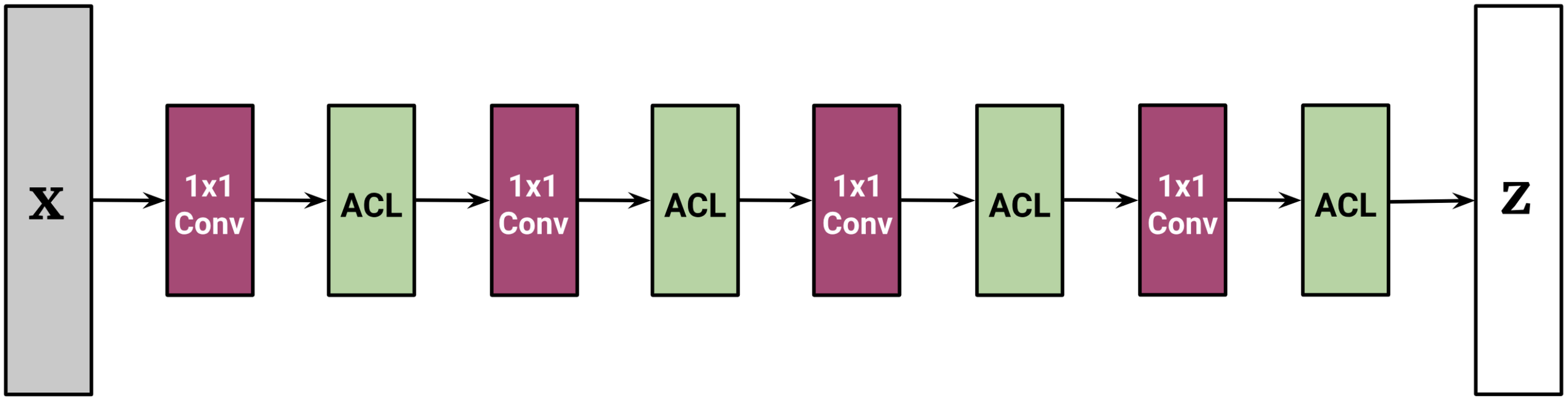
[Dinh et al., ICLR 2017]



Solution: permutation layers

The Affine Coupling Layer: Stacking

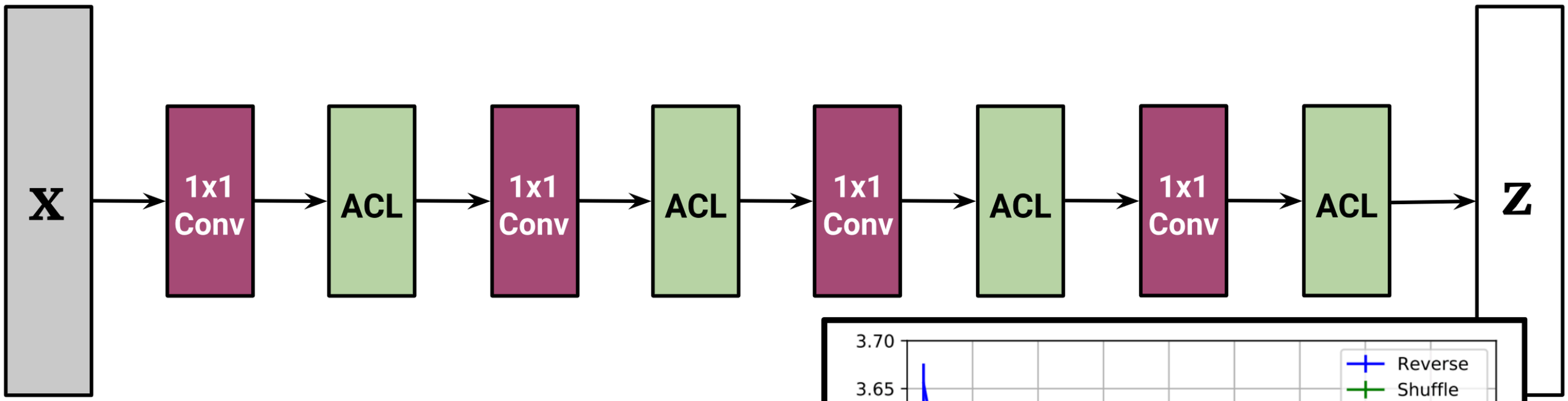
[Kingma & Dhariwal, NeurIPS 2018]



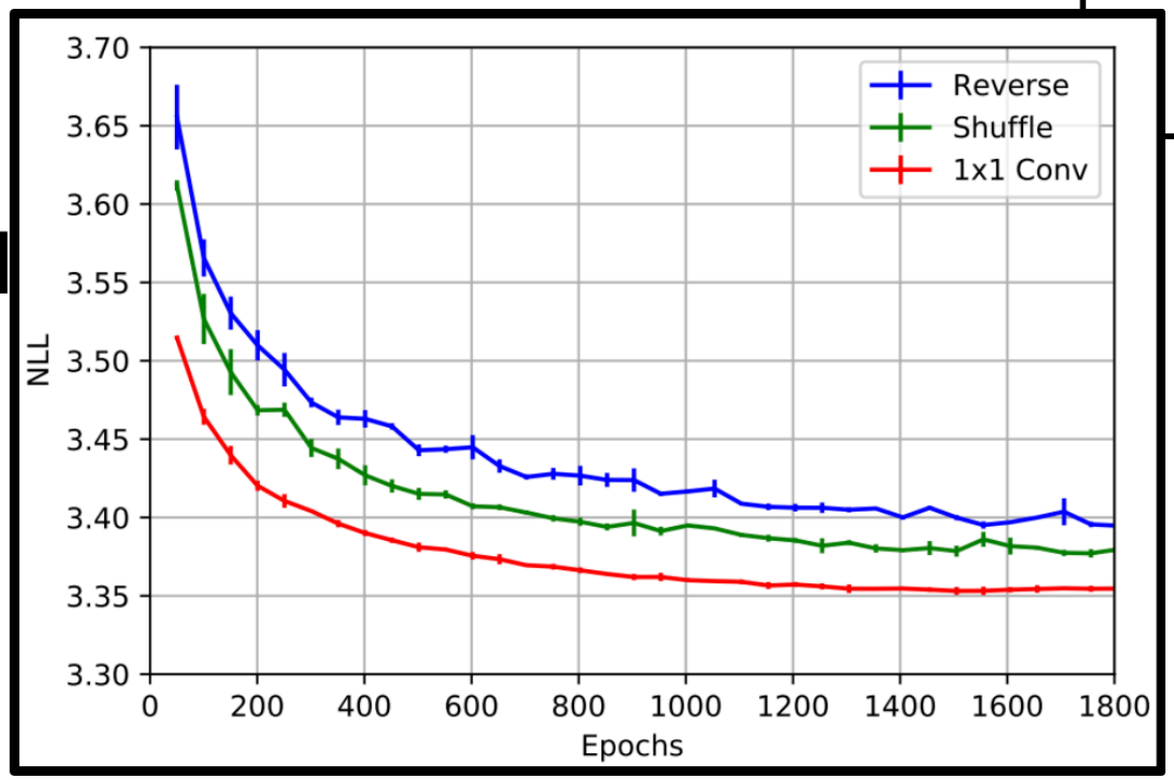
Solution: permutation layers

The Affine Coupling Layer: Stacking

[Kingma & Dhariwal, NeurIPS 2018]



Solution: permu

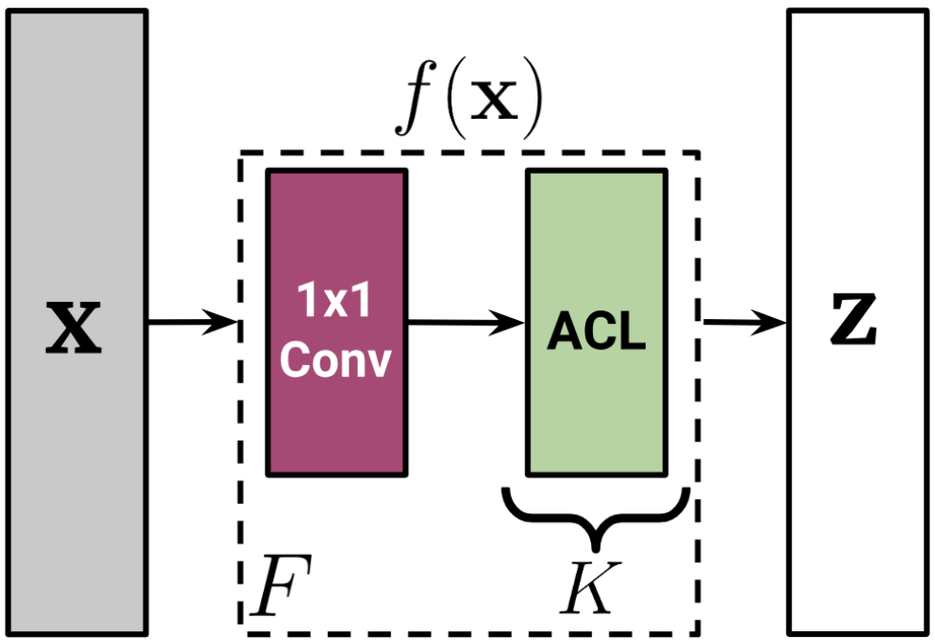


[Kingma & Dhariwal, NeurIPS 2018]

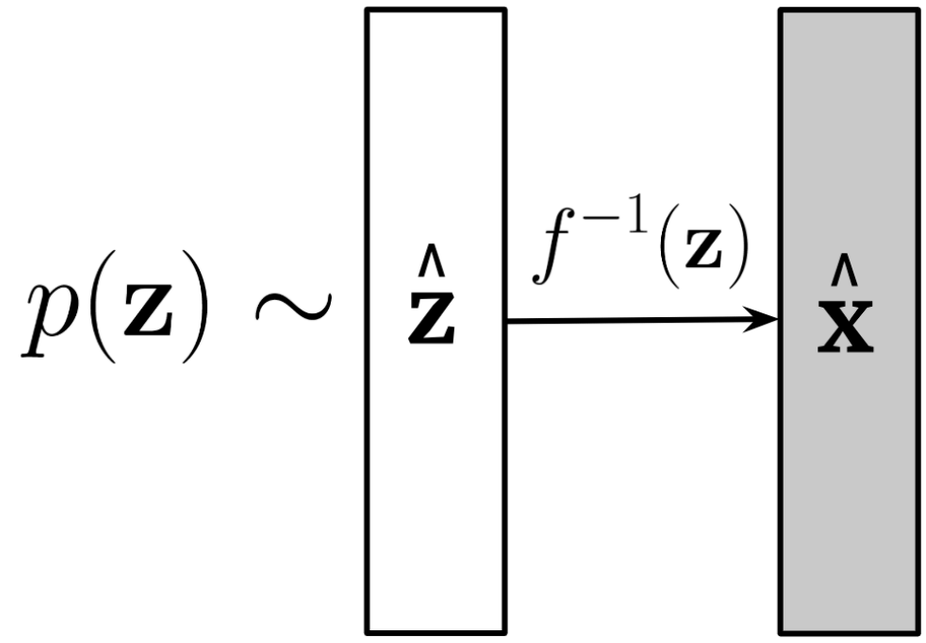
Flow-Based Generative Model

[Kingma & Dhariwal, NeurIPS 2018]

Architecture:



Sampling:



Training Objective:

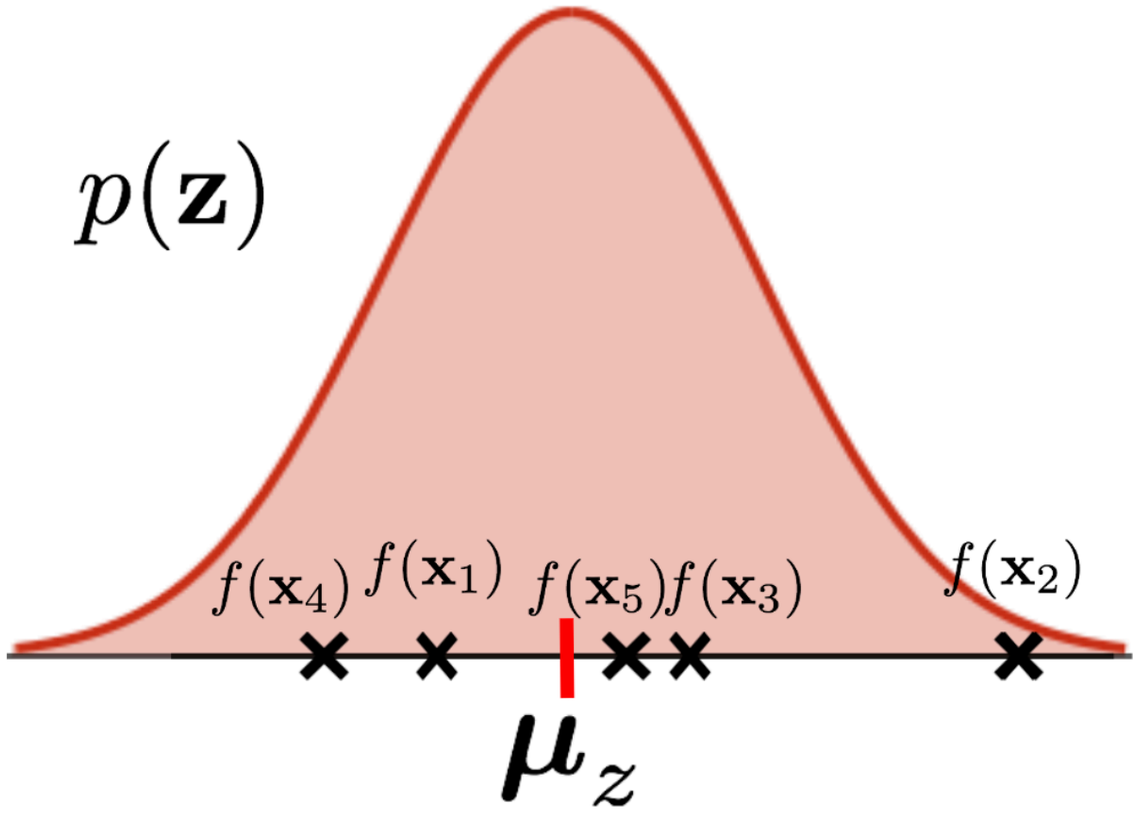
$$\boldsymbol{\phi}^* = \text{argmax}_{\boldsymbol{\phi}} \sum_{i=1}^{N} \log p(\mathbf{x}_i; \boldsymbol{\phi}) = \text{argmax}_{\boldsymbol{\phi}} \sum_{i=1}^{N} \log p_z(f(\mathbf{x}_i; \boldsymbol{\phi})) + \log \left| \frac{\partial \mathbf{f}_{\boldsymbol{\phi}}}{\partial \mathbf{x}_i} \right|$$

Flow-Based Generative Model

Training Objective:

$$\boldsymbol{\phi}^* = \text{argmax}_{\boldsymbol{\phi}} \sum_{i=1}^{N} \log p(\mathbf{x}_i; \boldsymbol{\phi}) = \text{argmax}_{\boldsymbol{\phi}} \sum_{i=1}^{N} \log p_z(f(\mathbf{x}_i; \boldsymbol{\phi})) + \log \left| \frac{\partial \mathbf{f}_{\boldsymbol{\phi}}}{\partial \mathbf{x}_i} \right|$$

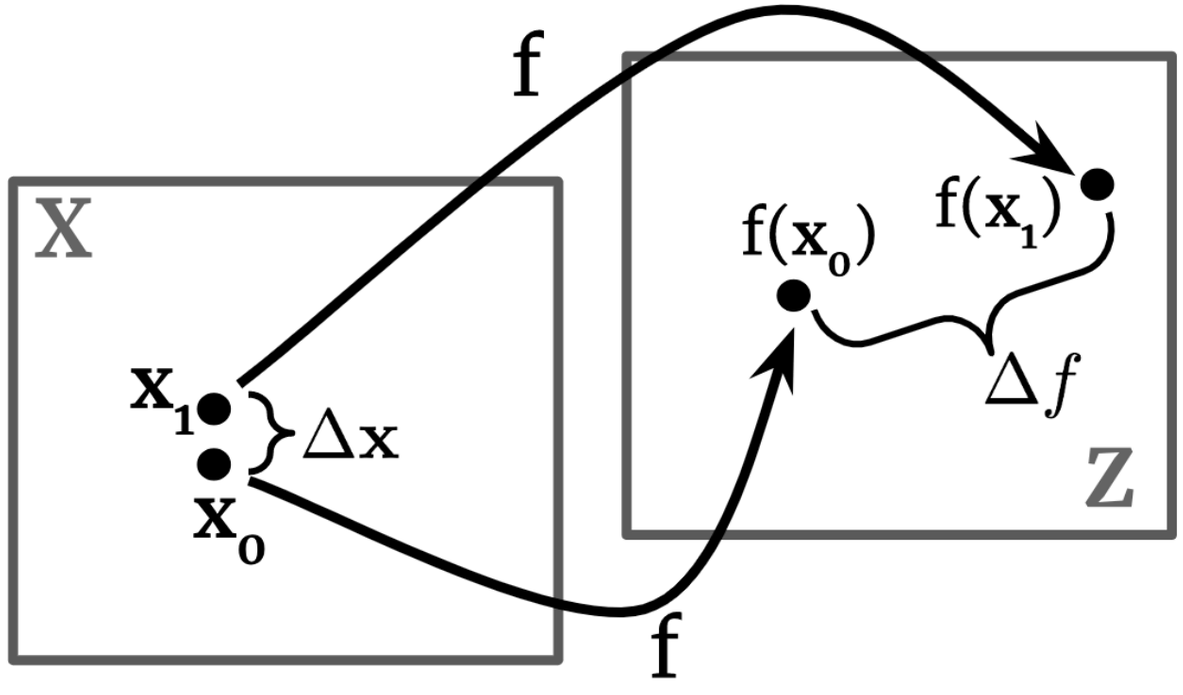
Encourages f(x)'s to distribute according to p(z)



Flow-Based Generative Model

Training Objective:

$$\boldsymbol{\phi}^* = \text{argmax}_{\boldsymbol{\phi}} \sum_{i=1}^{N} \log p(\mathbf{x}_i; \boldsymbol{\phi}) = \text{argmax}_{\boldsymbol{\phi}} \sum_{i=1}^{N} \log p_z(f(\mathbf{x}_i; \boldsymbol{\phi})) + \underbrace{\log \left| \frac{\partial \mathbf{f}_{\boldsymbol{\phi}}}{\partial \mathbf{x}_i} \right|}$$



Encourages an increase in the ‘sensitivity’ of f(x)

Flow-Based Generative Model

Training Objective:

$$\boldsymbol{\phi}^* = \text{argmax}_{\boldsymbol{\phi}} \sum_{i=1}^{N} \log p(\mathbf{x}_i; \boldsymbol{\phi}) = \text{argmax}_{\boldsymbol{\phi}} \sum_{i=1}^{N} \log p_z(f(\mathbf{x}_i; \boldsymbol{\phi})) + \underbrace{\log \left| \frac{\partial \mathbf{f}_{\boldsymbol{\phi}}}{\partial \mathbf{x}_i} \right|}$$

Encourages an increase in the 'sensitivity' of f(x)

Penalties on NN derivatives have been used for decades:

- *Regularization Theory and Neural Network Architectures*, Girosi et al., 1990
- *Training with Noise is Equivalent to Tikhonov Regularization*, C. Bishop., 1995
- *Contractive Autoencoders: Explicit Invariance During Feature Extraction*, Rifai et al., 2011

$$\mathcal{R} = \left|\left| \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|\right|_F^2$$

More Experiments with Flows

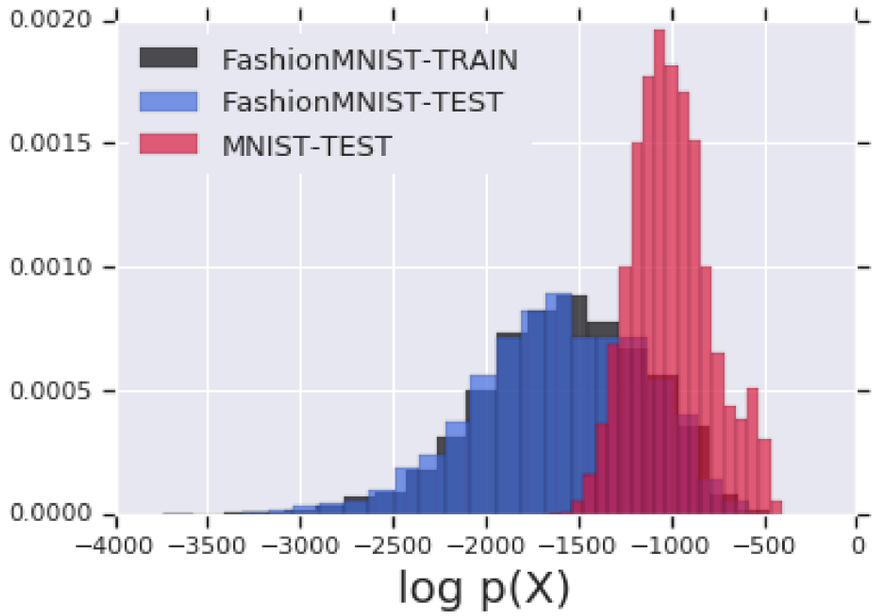
Additional Out-of-Distribution Tests

FashionMNIST vs **MNIST**

CelebA vs **SVHN**

ImageNet vs **CIFAR-10** vs **SVHN**

Additional Out-of-Distribution Tests

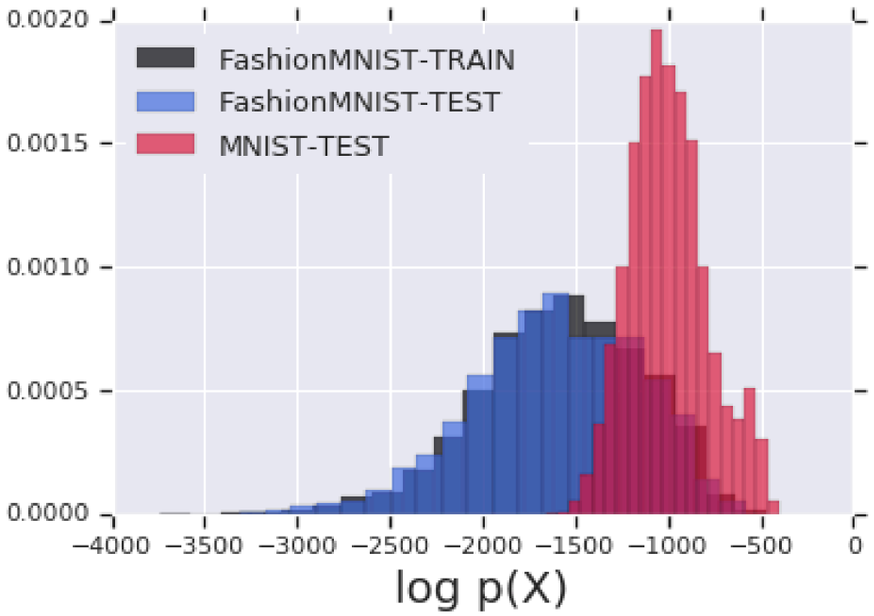


FashionMNIST vs MNIST

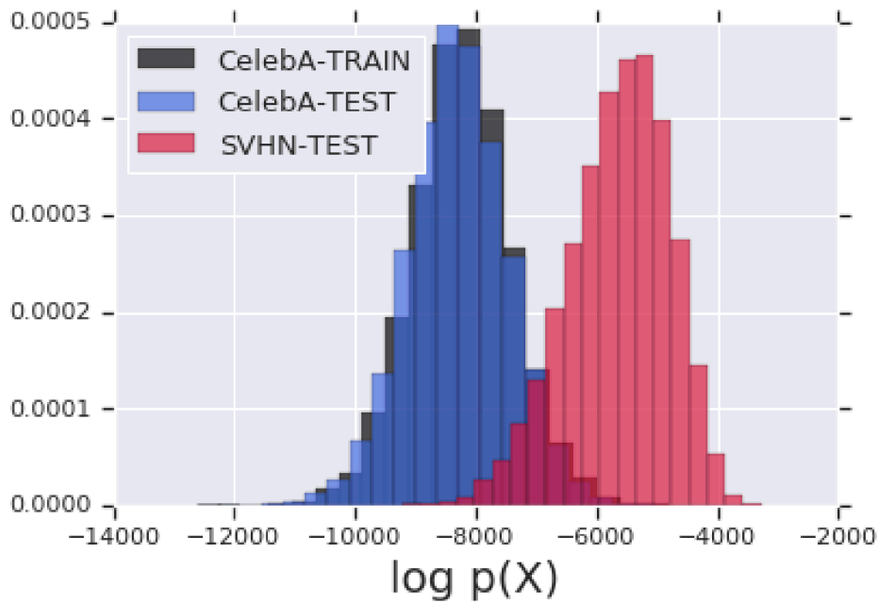
CelebA vs SVHN

ImageNet vs CIFAR-10 vs SVHN

Additional Out-of-Distribution Tests



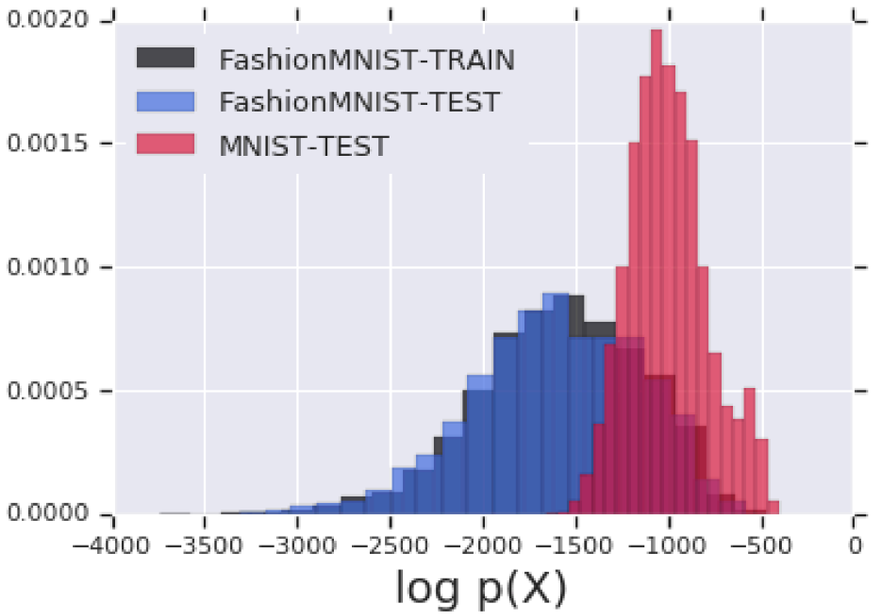
FashionMNIST vs MNIST



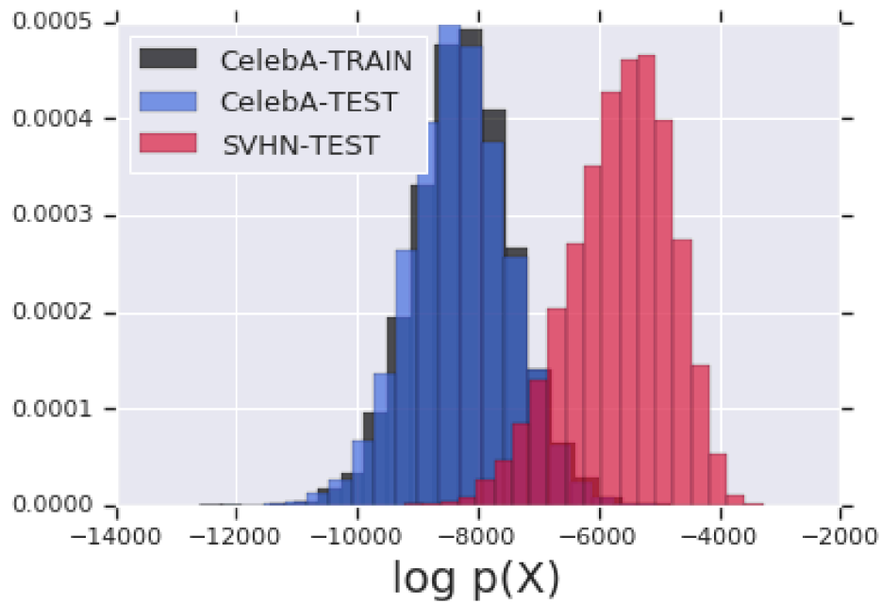
CelebA vs SVHN

ImageNet vs CIFAR-10 vs SVHN

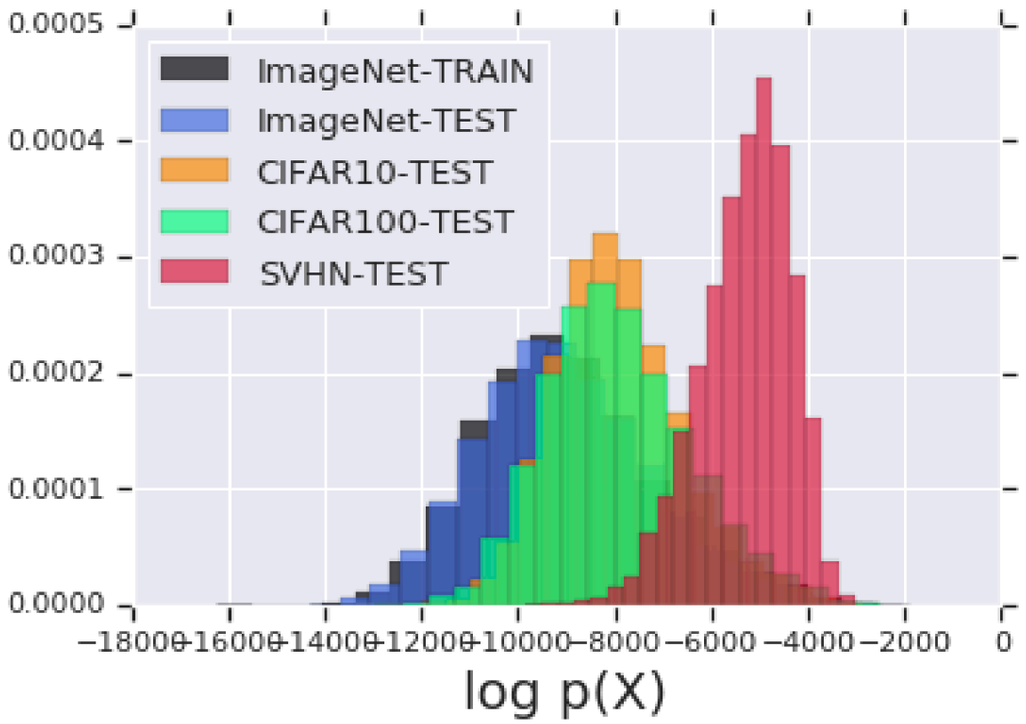
Additional Out-of-Distribution Tests



FashionMNIST vs MNIST

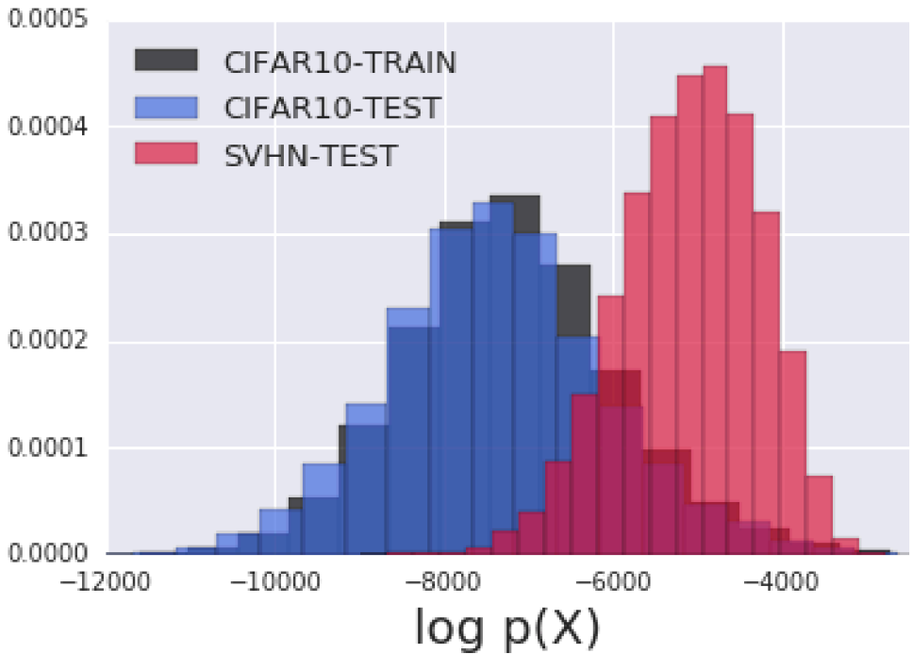


CelebA vs SVHN

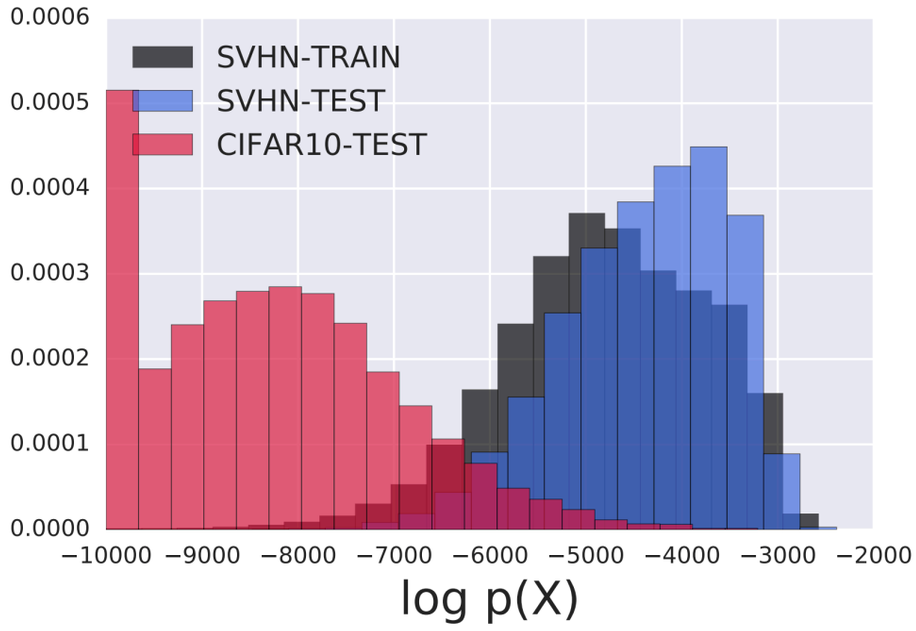


ImageNet vs CIFAR-10 vs SVHN

The Phenomenon is not Symmetric

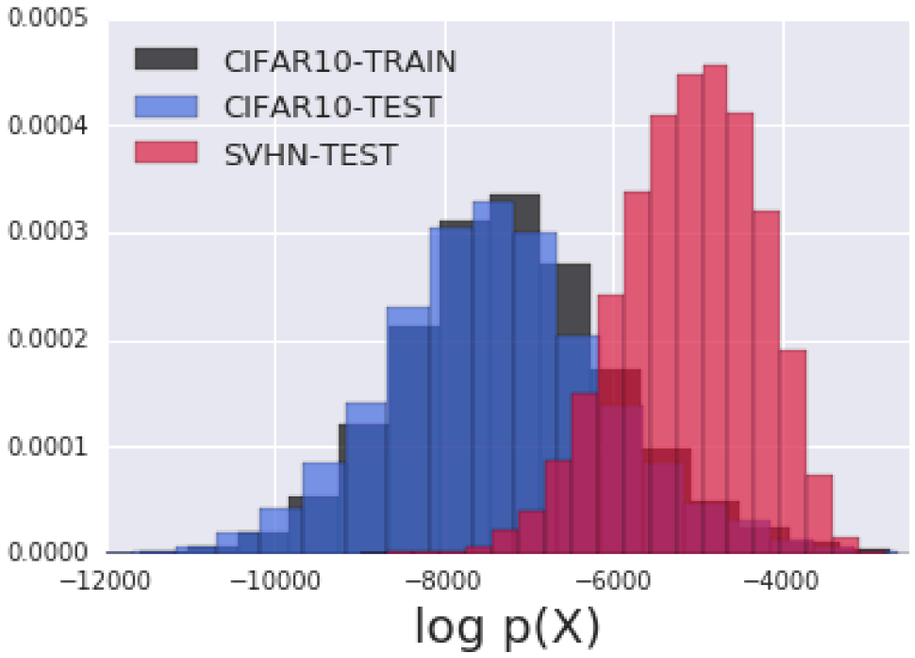


CIFAR-10 vs SVHN

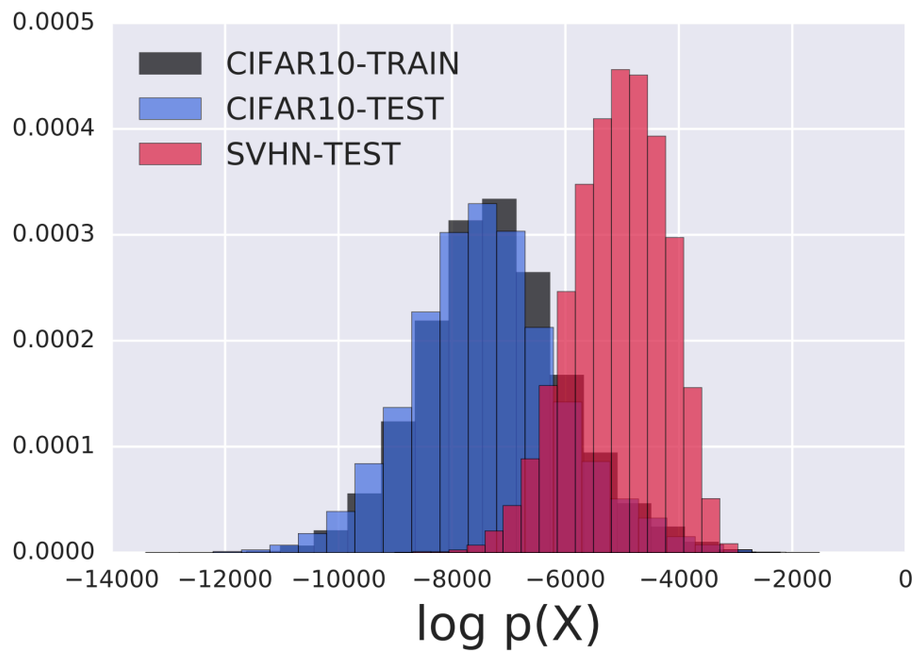


SVHN vs CIFAR-10

Ensembles Do Not Help



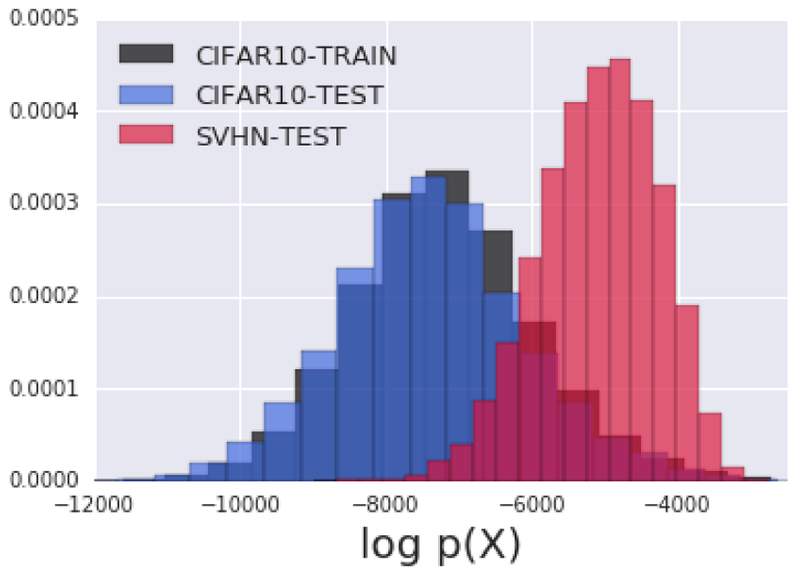
CIFAR-10 vs SVHN



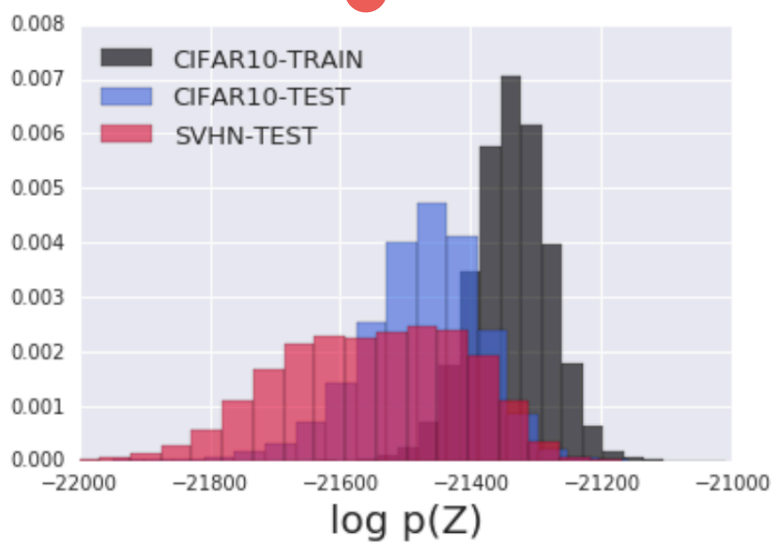
CIFAR-10 vs SVHN

Ensemble of 10 Glows

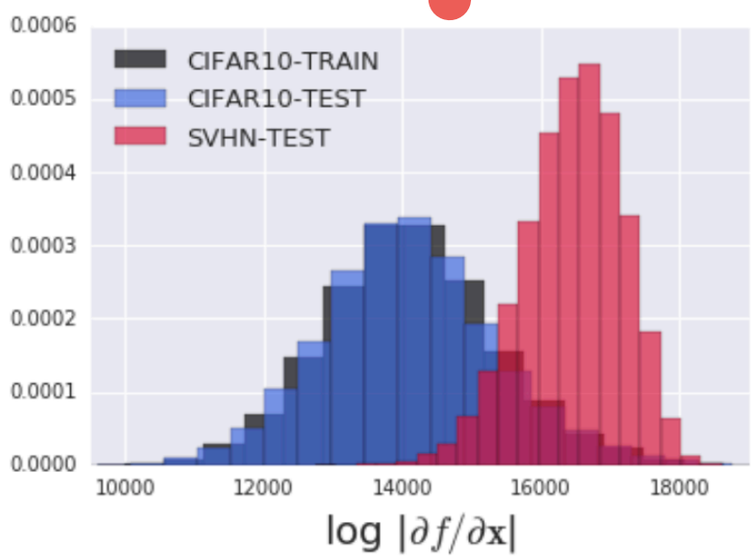
Decomposing the Change-of-Variables



CIFAR-10 vs SVHN

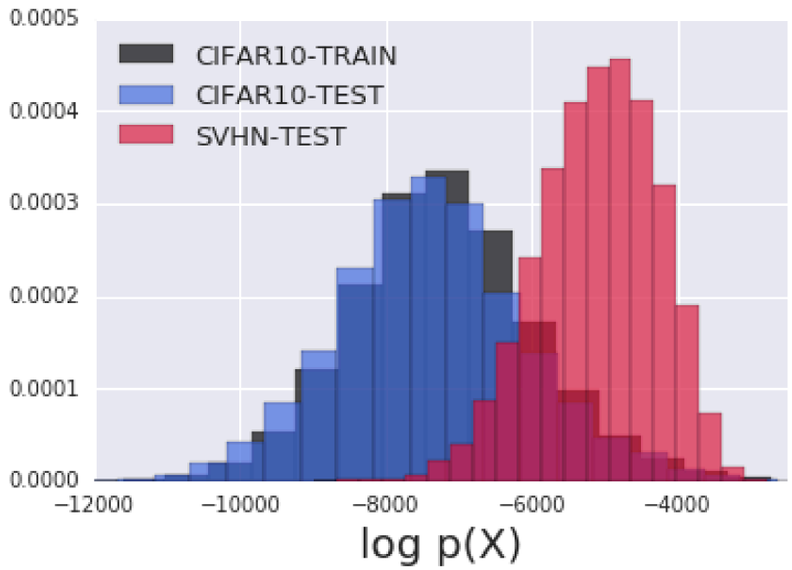


Distribution Term



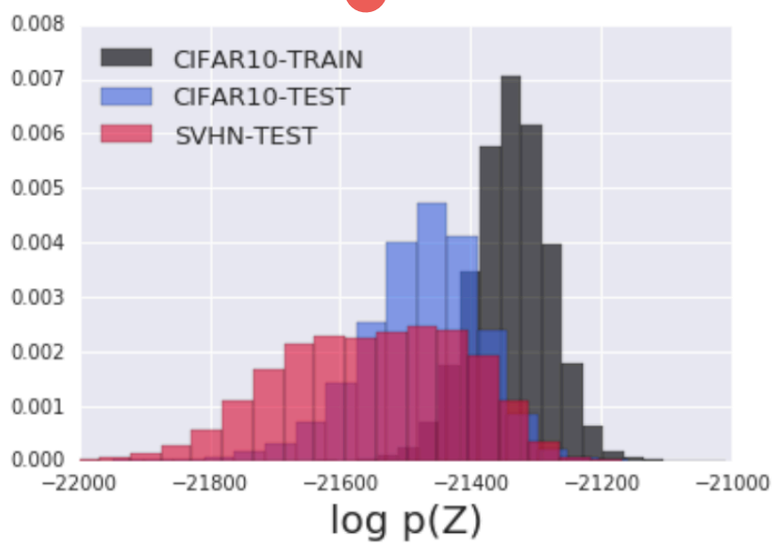
Volume Term

Decomposing the Change-of-Variables

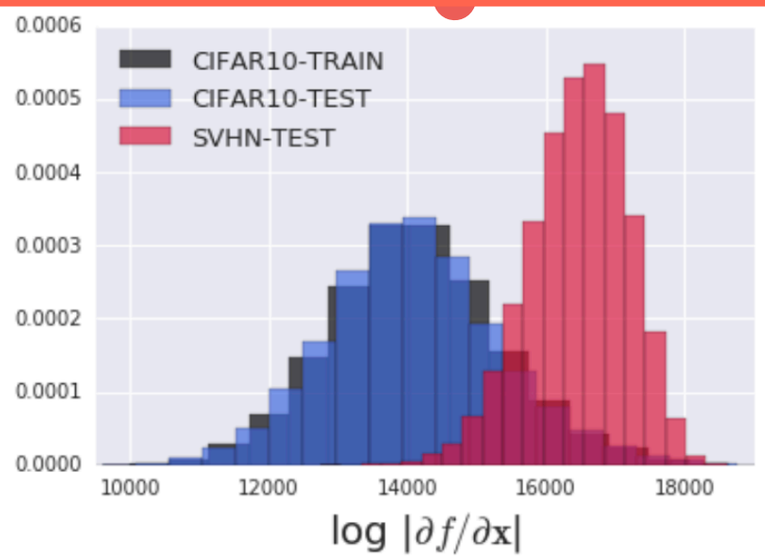


CIFAR-10 vs SVHN

- **Looks to be the cause of the phenomenon**



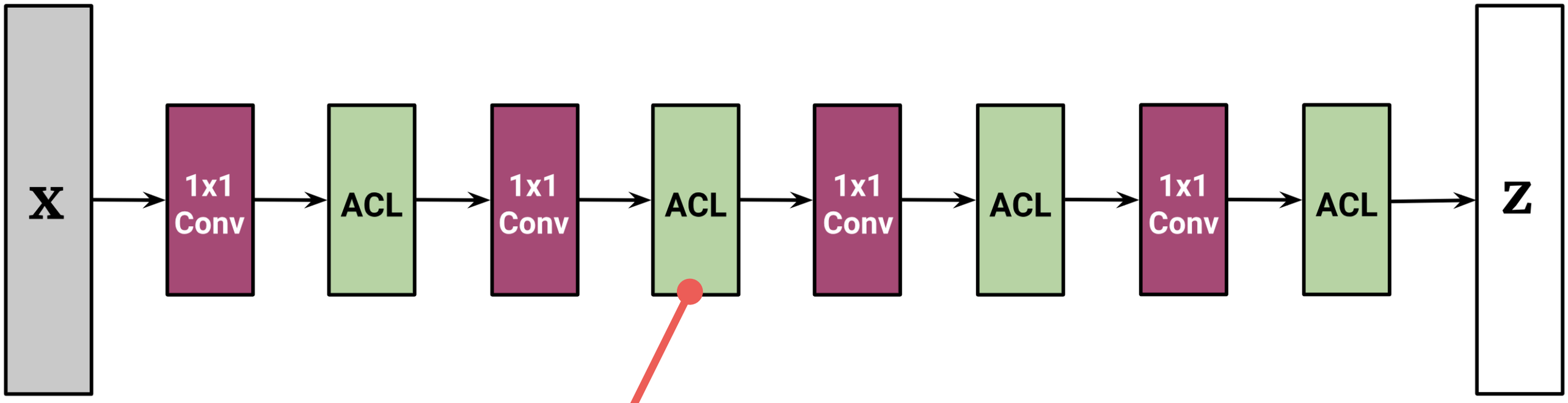
Distribution Term



Volume Term

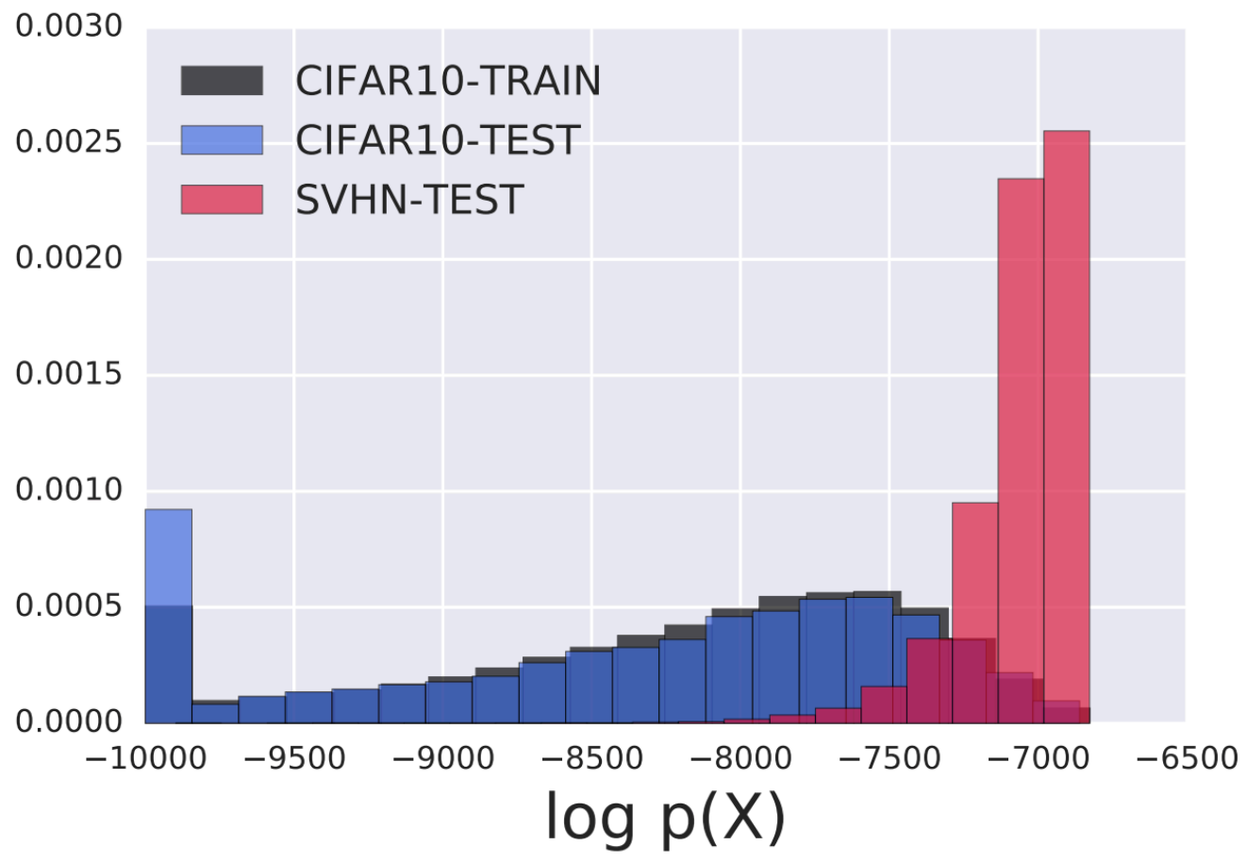
Constant-Volume Flows

To isolate the effect of the volume term, we define *constant-volume* (w.r.t. input) flows.



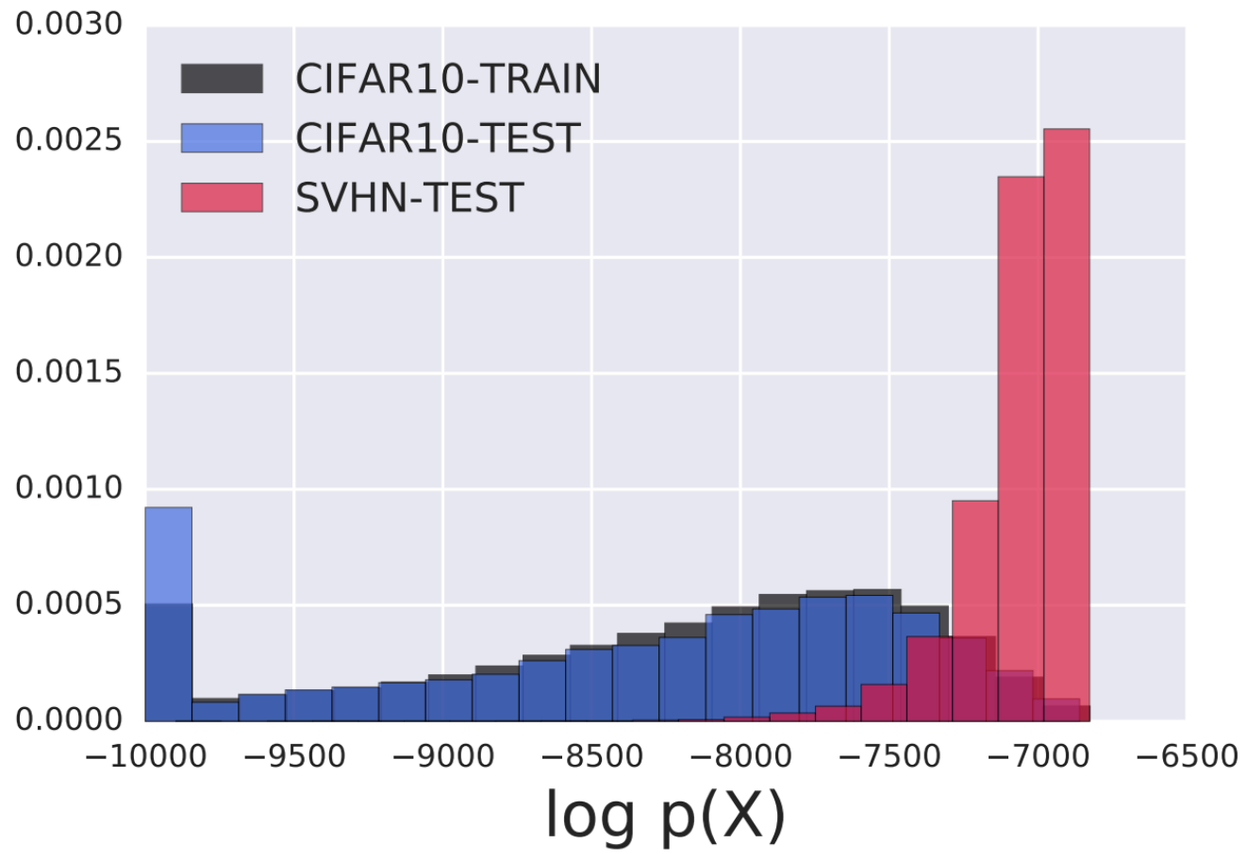
- **Use only translation operations.**

Constant-Volume Experiments

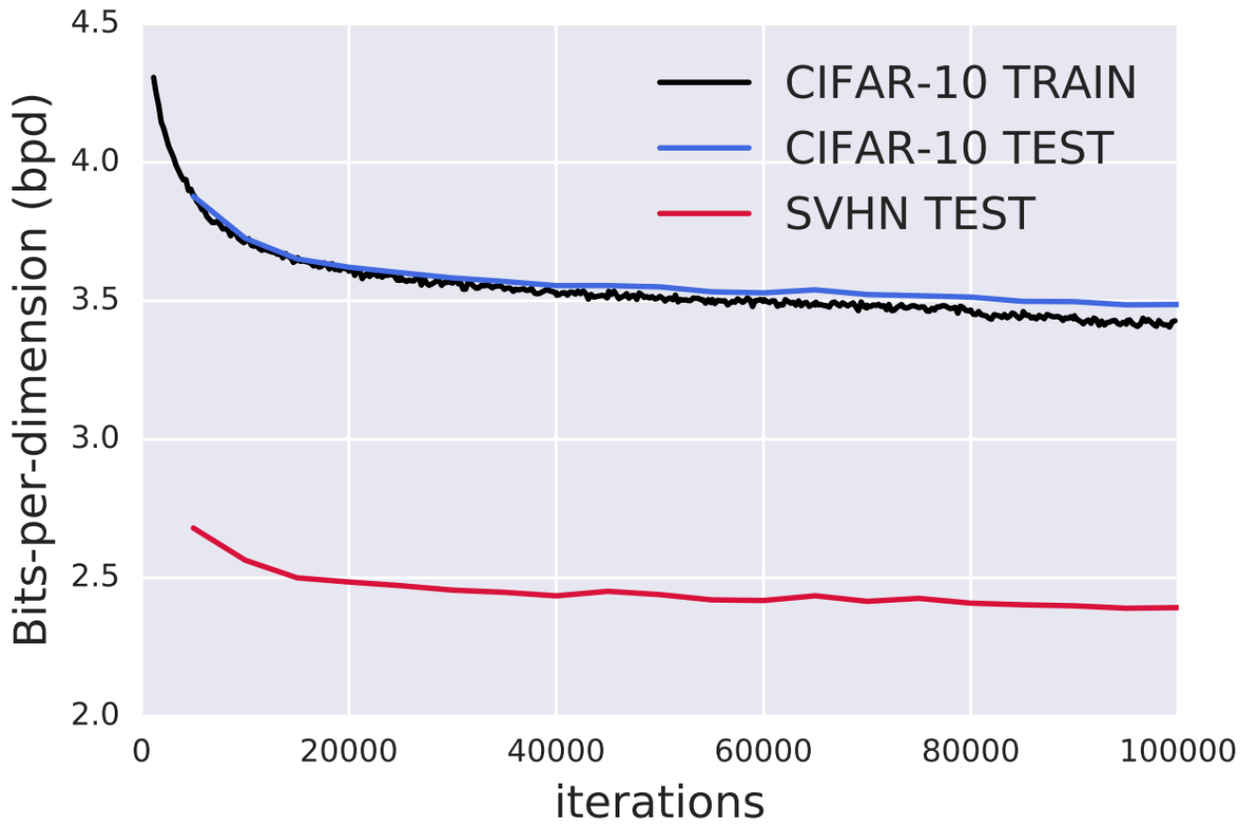


CIFAR-10 vs SVHN

Constant-Volume Experiments



CIFAR-10 vs SVHN



During Optimization

Analytical Investigation

Constant-Volume Flow Analysis

Mathematical characterization:

$$0 < \underbrace{\mathbb{E}_q[\log p(\boldsymbol{x};\boldsymbol{\theta})]}_{\text{Non-Training Distribution}} - \underbrace{\mathbb{E}_{p^*}[\log p(\boldsymbol{x};\boldsymbol{\theta})]}_{\text{Training Distribution}}$$

Constant-Volume Flow Analysis

Mathematical characterization:

$$0 < \underbrace{\mathbb{E}_q}_{\text{Non-Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \underbrace{\mathbb{E}_{p^*}}_{\text{Training Distribution}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2}\operatorname{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0}\log p_z(f(\boldsymbol{x}_0;\boldsymbol{\phi})) + \nabla^2_{\boldsymbol{x}_0}\log\left|\frac{\partial \boldsymbol{f}_{\boldsymbol{\phi}}}{\partial \boldsymbol{x}_0}\right|\right](\boldsymbol{\Sigma}_q - \boldsymbol{\Sigma}_{p^*})\right\}$$

Constant-Volume Flow Analysis

Mathematical characterization:

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

Non-Training Distribution

Training Distribution

Second Moment of Training Distribution

$$\approx \frac{1}{2}\operatorname{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0}\log p_z(f(\boldsymbol{x}_0;\boldsymbol{\phi})) + \nabla^2_{\boldsymbol{x}_0}\log\left|\frac{\partial \boldsymbol{f}_{\boldsymbol{\phi}}}{\partial \boldsymbol{x}_0}\right|\right](\underline{\boldsymbol{\Sigma}_q} - \overline{\boldsymbol{\Sigma}_{p^*}})\right\}$$

Second Moment of Non-Training Distribution

Constant-Volume Flow Analysis

Plugging in the CV-Glow transform:

$$\mathrm{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0}\log p(\boldsymbol{x}_0;\boldsymbol{\theta})\right](\boldsymbol{\Sigma}_q-\boldsymbol{\Sigma}_{p^*})\right\}$$

$$=\frac{\partial^2}{\partial z^2}\log p(\boldsymbol{z};\boldsymbol{\psi})\sum_{c=1}^{C}\left(\prod_{k=1}^{K}\sum_{j=1}^{C}u_{k,c,j}\right)^2\sum_{h,w}(\sigma^2_{q,h,w,c}-\sigma^2_{p^*,h,w,c})$$

Constant-Volume Flow Analysis

Plugging in the CV-Glow transform:

$$\mathrm{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0}\log p(\boldsymbol{x}_0;\boldsymbol{\theta})\right](\boldsymbol{\Sigma}_q-\boldsymbol{\Sigma}_{p^*})\right\}$$

$$=\frac{\partial^2}{\partial z^2}\log p(\boldsymbol{z};\boldsymbol{\psi})\sum_{c=1}^{C}\left(\prod_{k=1}^{K}\sum_{j=1}^{C}\underline{u_{k,c,j}}\right)^2\sum_{h,w}(\sigma^2_{q,h,w,c}-\sigma^2_{p^*,h,w,c})$$

1x1 Conv. Params

Constant-Volume Flow Analysis

Plugging in the CV-Glow transform:

$$\mathrm{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0} \log p(\boldsymbol{x}_0;\boldsymbol{\theta})\right](\boldsymbol{\Sigma}_q - \boldsymbol{\Sigma}_{p^*})\right\}$$

$$= \frac{\partial^2}{\partial z^2} \log p(\boldsymbol{z};\boldsymbol{\psi}) \sum_{c=1}^{C} \left(\prod_{k=1}^{K} \sum_{j=1}^{C} u_{k,c,j} \right)^2 \sum_{h,w} (\sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c})$$

1x1 Conv. Params

Sums over channel dimensions

Product over steps in flow

Sum over spatial dimensions

Constant-Volume Flow Analysis

Plugging in the CV-Glow transform:

$$\mathrm{Tr}\left\{\left[\nabla^2_{\boldsymbol{x}_0} \log p(\boldsymbol{x}_0;\boldsymbol{\theta})\right](\boldsymbol{\Sigma}_q - \boldsymbol{\Sigma}_{p^*})\right\}$$

$$= \frac{\partial^2}{\partial z^2} \log p(\boldsymbol{z};\boldsymbol{\psi}) \sum_{c=1}^{C} \left(\prod_{k=1}^{K} \sum_{j=1}^{C} u_{k,c,j}\right)^2 \sum_{h,w} (\sigma^2_{q,h,w,c} - \sigma^2_{p^*,h,w,c})$$

< 0 for all log-concave densities (e.g. Gaussian)

Non-negative due to square

Constant-Volume Flow Analysis

Plugging in the CIFAR-10 and SVHN statistics:

$$\mathbb{E}_{\text{SVHN}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \mathbb{E}_{\text{CIFAR10}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2\sigma_{\boldsymbol{\psi}}^2}\left[\alpha_1^2 \cdot 12.3 + \alpha_2^2 \cdot 6.5 + \alpha_3^2 \cdot 14.5\right] \geq 0 \quad \text{where } \alpha_c = \prod_{k=1}^{K}\sum_{j=1}^{C} u_{k,c,j}$$

Differences in variances in the three spatial dimensions

Constant-Volume Flow Analysis

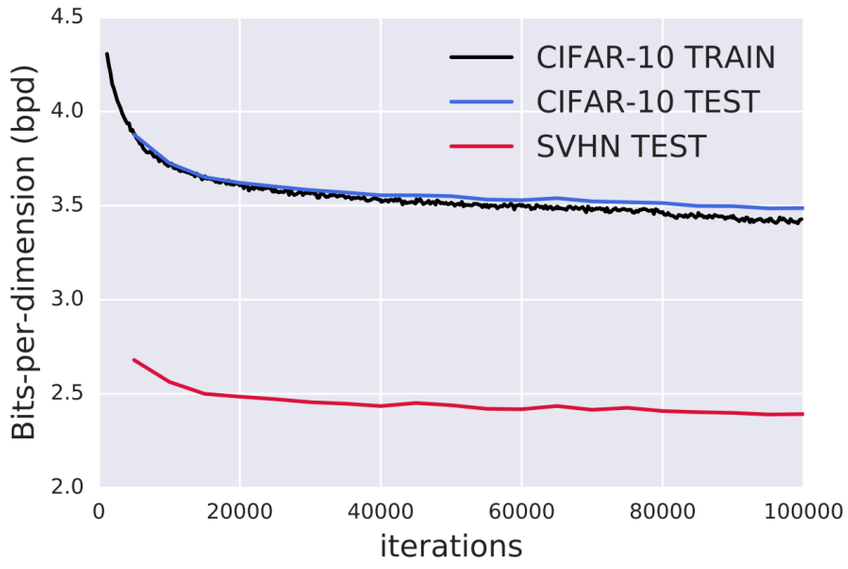
Plugging in the CIFAR-10 and SVHN statistics:

$$\mathbb{E}_{\text{SVHN}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \mathbb{E}_{\text{CIFAR10}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2\sigma_{\boldsymbol{\psi}}^2}\left[\alpha_1^2 \cdot 12.3 + \alpha_2^2 \cdot 6.5 + \alpha_3^2 \cdot 14.5\right] \geq 0 \quad \text{where } \alpha_c = \prod_{k=1}^{K}\sum_{j=1}^{C} u_{k,c,j}$$

Differences in variances in the three spatial dimensions

The expression will be non-negative **for any** parameter setting of the CV flow….



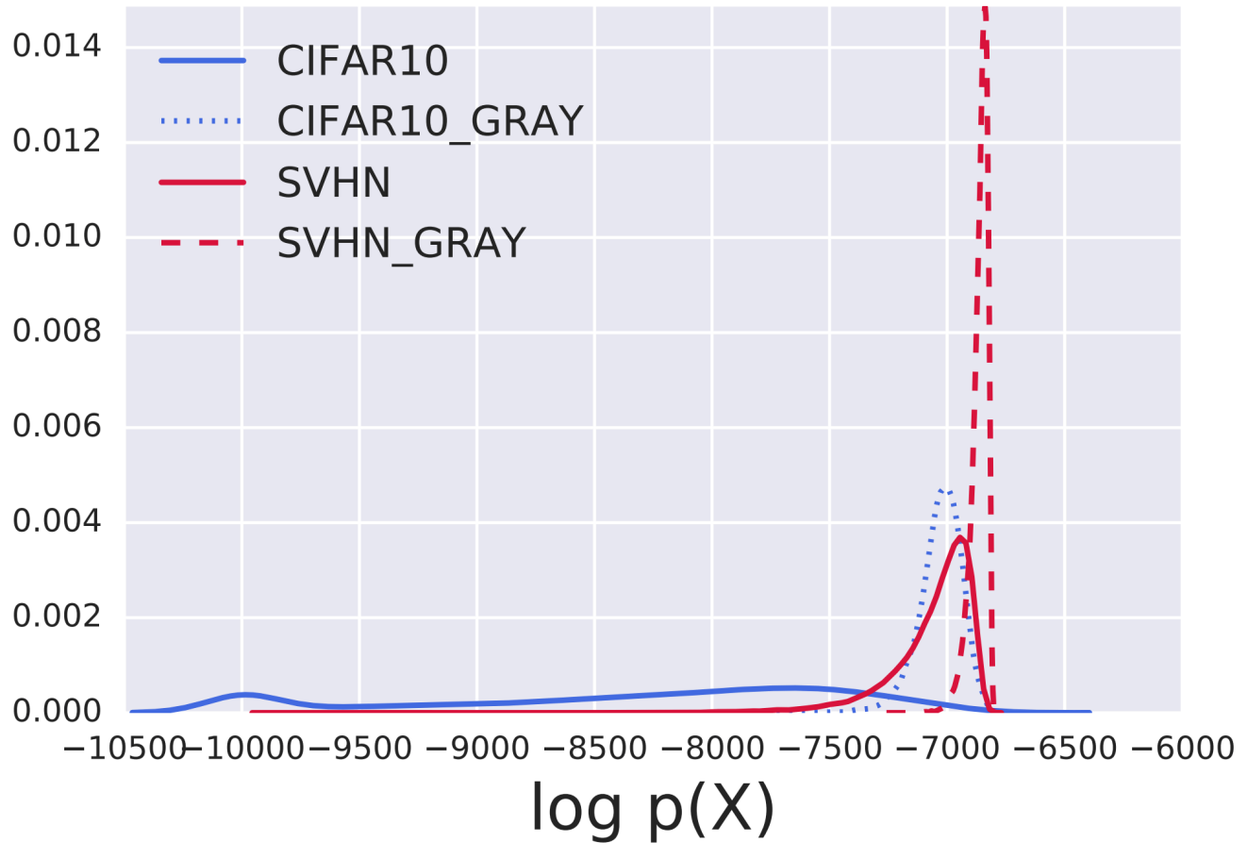
Constant-Volume Flow Analysis

Plugging in the CIFAR-10 and SVHN statistics:

$$\mathbb{E}_{\text{SVHN}}[\log p(\boldsymbol{x};\boldsymbol{\theta})] - \mathbb{E}_{\text{CIFAR10}}[\log p(\boldsymbol{x};\boldsymbol{\theta})]$$

$$\approx \frac{1}{2\sigma_{\boldsymbol{\psi}}^2}\left[\alpha_1^2 \cdot 12.3 + \alpha_2^2 \cdot 6.5 + \alpha_3^2 \cdot 14.5\right] \geq 0 \quad \text{where} \;\; \alpha_c = \prod_{k=1}^{K} \sum_{j=1}^{C} u_{k,c,j}$$

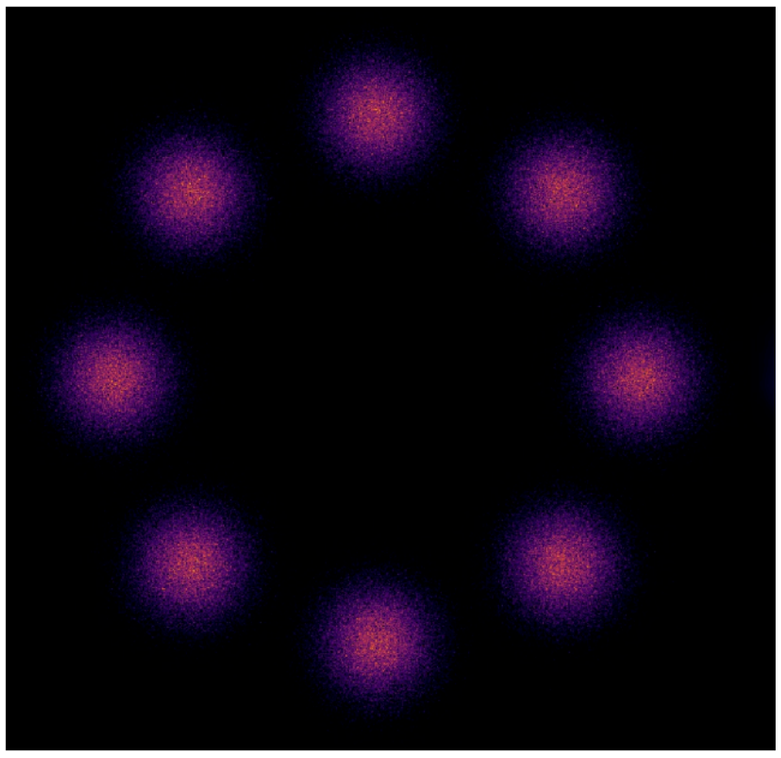
This also means that we can manipulate the relative log likelihoods just by changing the variance of the data. For natural images, this amounts to **graying**…



Parting Thoughts

Parting Thoughts

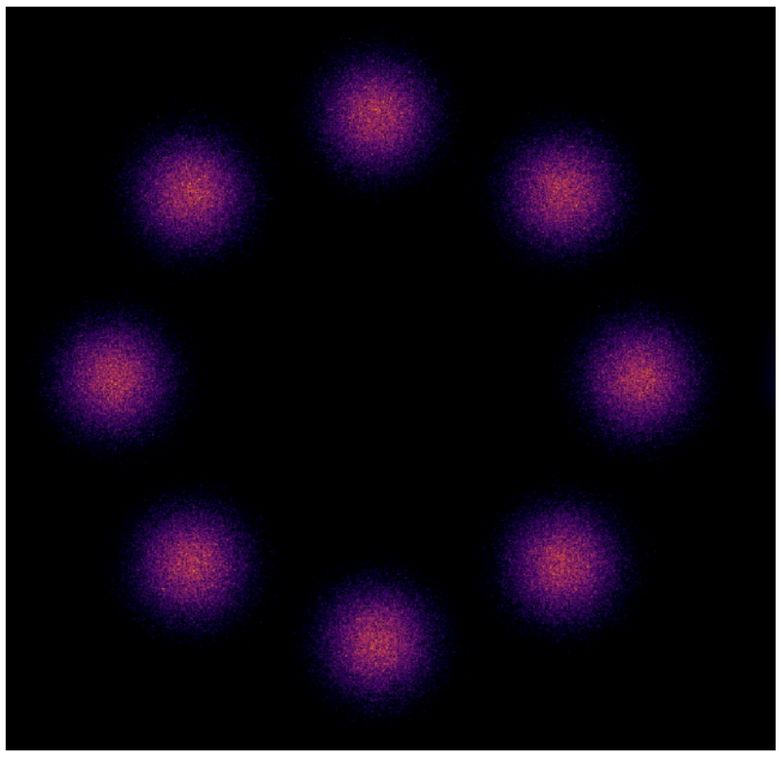
What does it mean for a deep generative model to *generalize*?



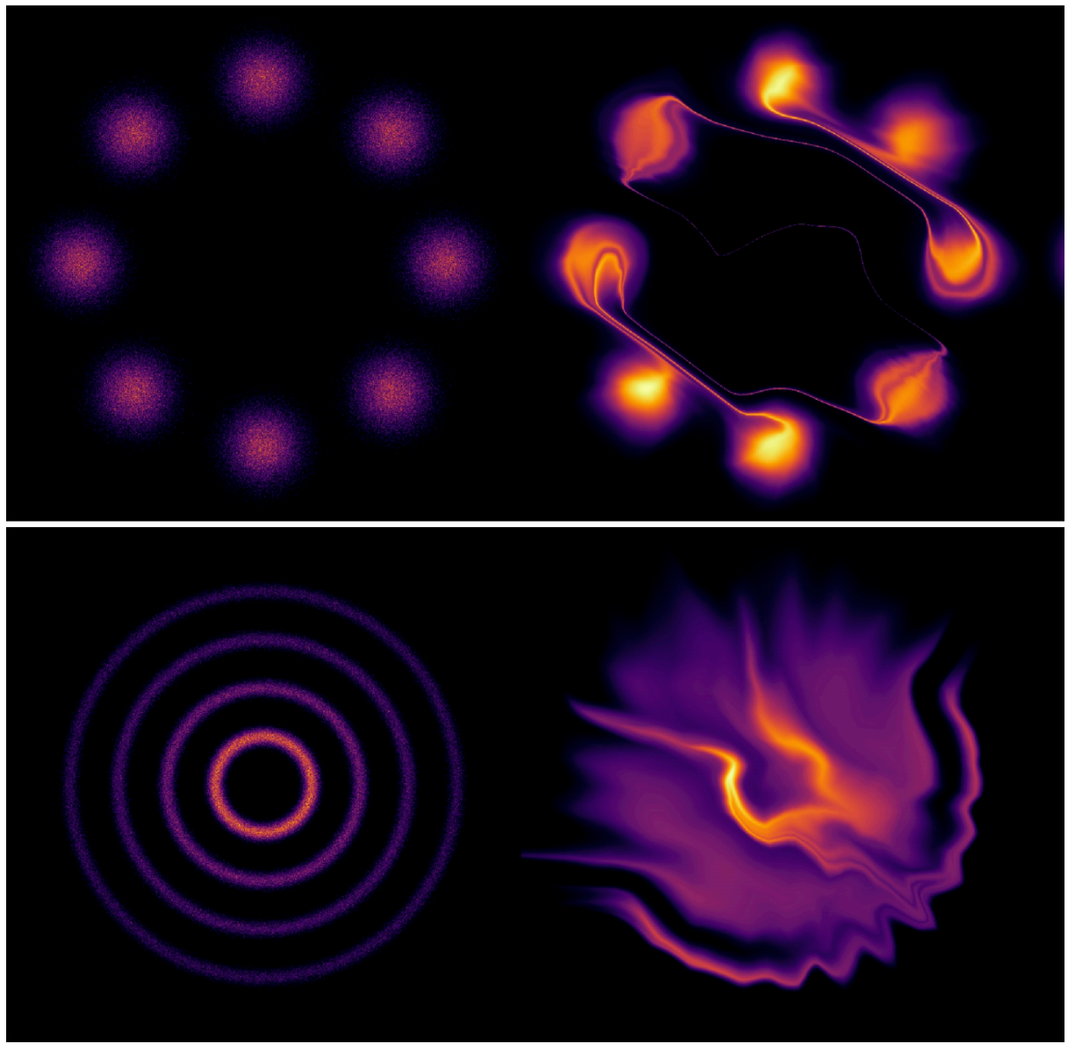
Data Samples

Parting Thoughts

What does it mean for a deep generative model to *generalize*?



Data Samples



Data Samples Glow

[Behrmann et al., ArXiv 2019]

How do we get Glow (and other deep gen. models) to ‘smear’ in the right directions?

Parting Thoughts

Do deep generative models need a better notion of semantics?

	CIFAR-10 TEST (BITS/DIM)	SVHN TEST (BITS/DIM)
BIVA $\mathcal{L}_1^{>13}$	79.36	121.04
BIVA $\mathcal{L}_{100}^{>13}$	54.91	71.38
BIVA $\mathcal{L}_1^{>11}$	35.34	58.82
BIVA $\mathcal{L}_1^{>9}$	20.93	26.76
BIVA $\mathcal{L}_1^{>0}$	3.12	2.28

Table 6. The $\mathcal{L}^{>k}$ for $k = 13, 11, 9, 0$ evaluated in bits/dim by a model trained on the CIFAR-10 train set and evaluated on the CIFAR-10 test set and the SVHN test set. It is possible to use BIVA for anomaly detection of high-level semantics in complex datasets when using a high k.

[Maaløe et al., ArXiv 2019]

Parting Thoughts

Does there exist analogous behavior for language?

Parting Thoughts

Conclusion: We should temper the enthusiasm with which we preach the benefits of deep generative models until their out-of-distribution behavior is better understood.

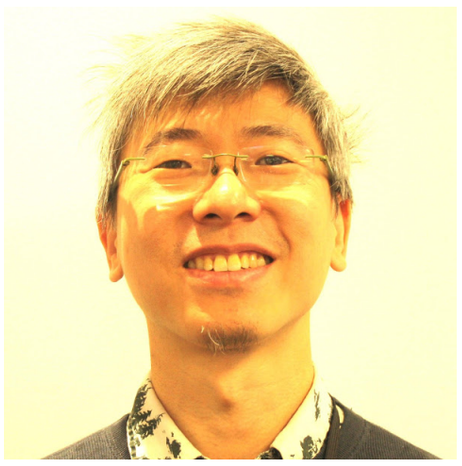
Thank you. Questions?

In collaboration with…

Aki Matsukawa

Balaji Lakshminarayanan

Dilan Gorur



Yee Whye Teh