Bayesian Deep Learning

Advances and Perspective

Jun Zhu

Tsinghua-Bosch ML Center
The Institute for Artificial Intelligence
Department of Computer Science and Technology
Tsinghua University

Uncertainty Modeling and Inference

- Uncertainty in Data/Environment: Ubiquitous, sometimes Adversarial!

Road conditions

Traffics

Pedestrian behaviors

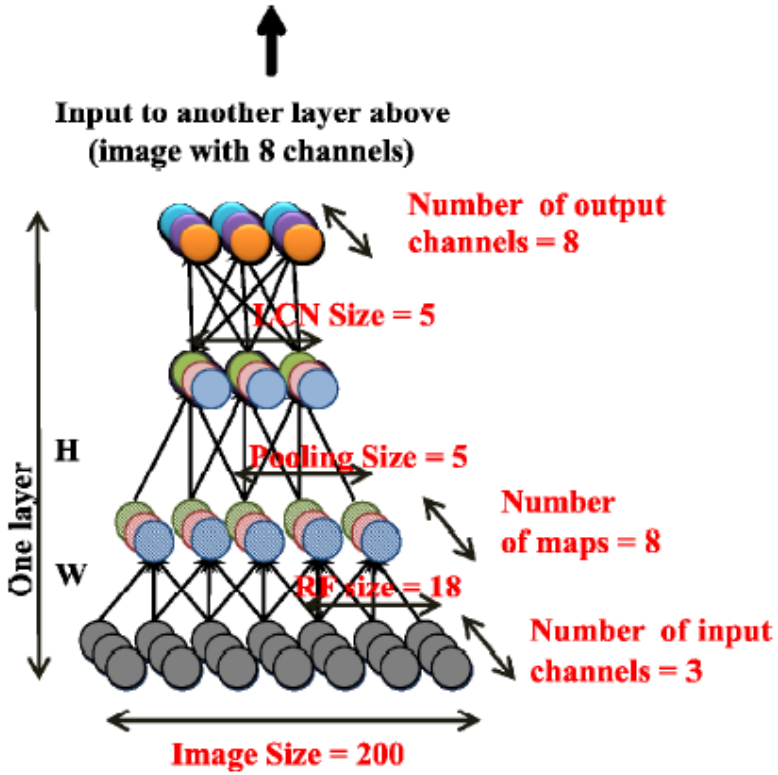
AlexNet: lionfish， confidence 81.3%
VGG-16: lionfish， confidence 93.3%
ResNet-18: lionfish， confidence 95.6%

[Dong et al., CVPR (2018, 2019, 2020); Pang et al., ICML (2018, 2019), ICLR 2020; Cheng et al., NeurIPS 2019]

Uncertainty Modeling and Inference

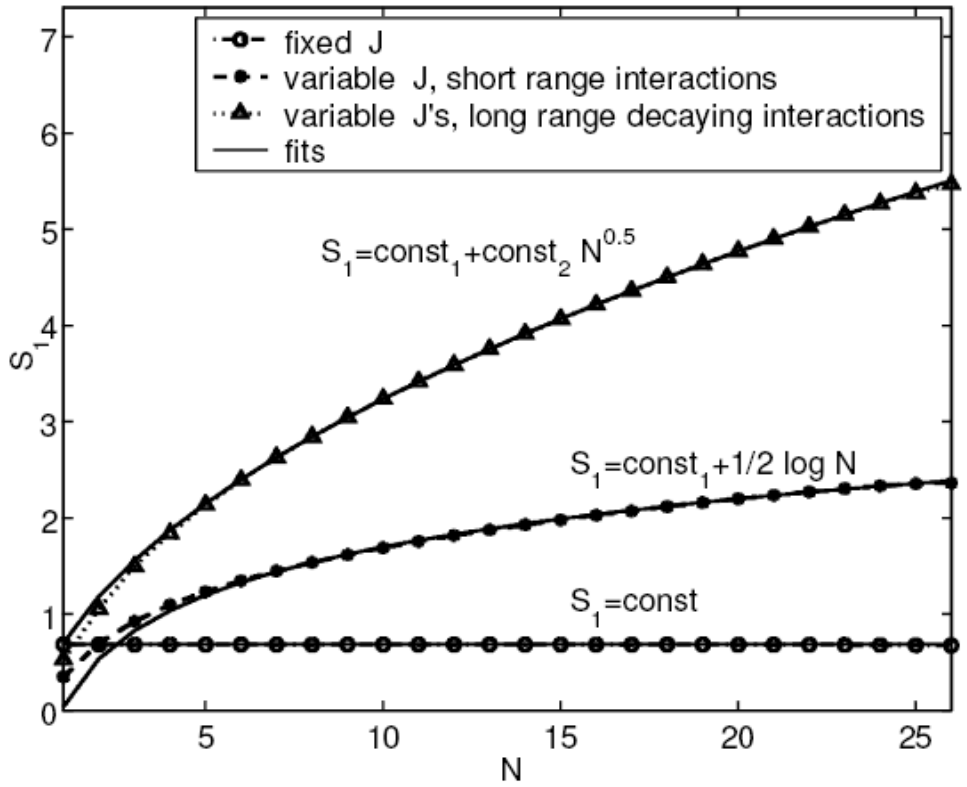
- Uncertainty of Models: need to concern, especially when models are huge!

Big Models



VS

Large Redundancy in Data



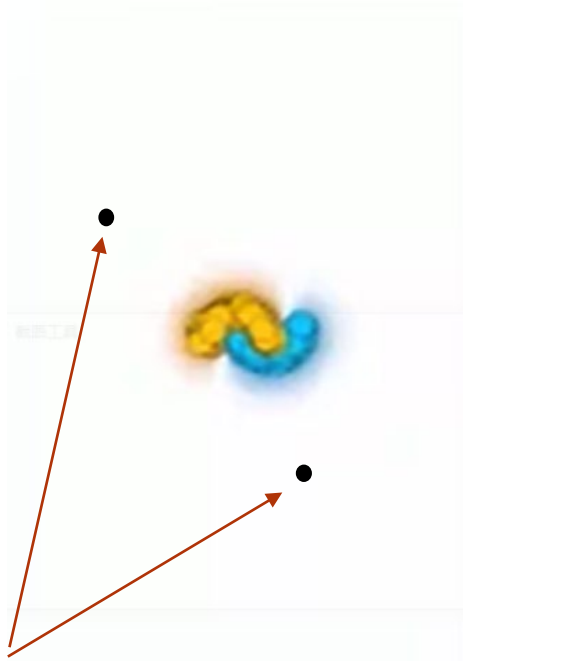
- **1B parameters** (connections); **10M** 200x200 images

(Bialek et al., 2001)

Uncertainty Modeling and Inference

- Uncertainty of Models: need to concern, especially when models are huge!
  - The output of DNN $p(y|x)$ doesn't well capture model uncertainty (**over-optimistic**)!

Over-confident



Uncertain prediction: reject? Or Transform

[Gal & Ghahramani, Dropout as a Bayesian Approximation, ICML 2016]

Bayesian (Probabilistic) Machine Learning

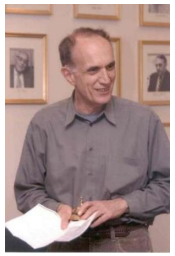
- The core is Bayes' rule: prior knowledge, uncertainty inference

posterior ← likelihood model → prior

$$p(\theta|D) = \frac{p(D|\theta)\pi(\theta)}{p(D)}$$

Thomas Bayes (1702 – 1761)

- Bayes' Theorem in the 21st Century (Year 2013 is the 250th Anniversary of Bayes' theorem)
  - Bradley Efron, *Science 7 June 2013: Vol. 340 no. 6137 pp. 1177-1178*



"There are two potent arrows in the statistician's quiver; there is no need to go hunting armed with only one."

REVIEW

doi:10.1038/nature14541

Probabilistic machine learning and artificial intelligence

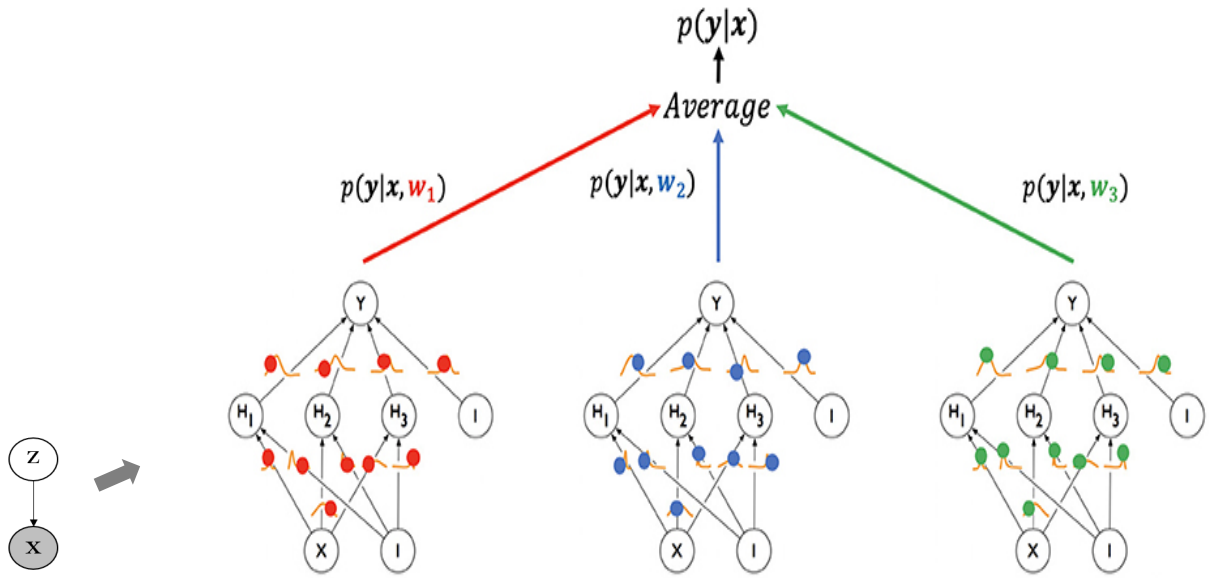
Zoubin Ghahramani[1]

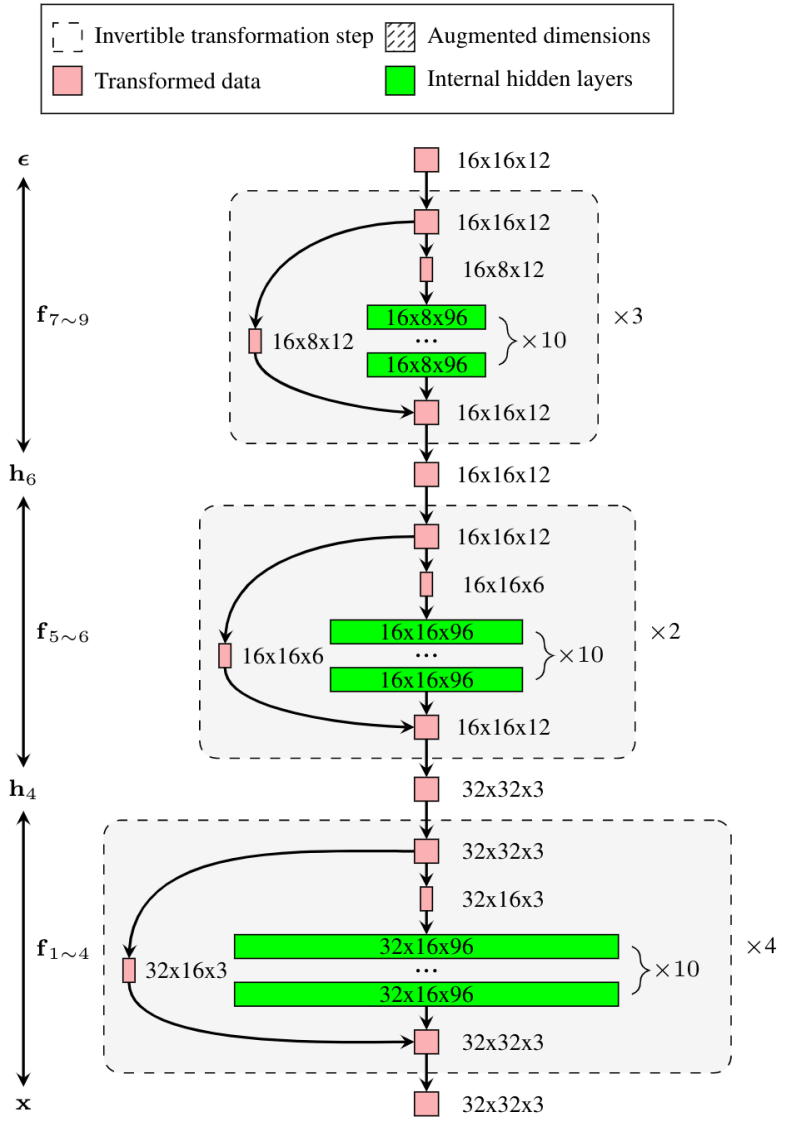
How can a machine learn from experience? Probabilistic modelling provides a framework for understanding what learning is, and has therefore emerged as one of the principal theoretical and practical approaches for designing machines that learn from data acquired through experience. The probabilistic framework, which describes how to represent and manipulate uncertainty about models and predictions, has a central role in scientific data analysis, machine learning, robotics, cognitive science and artificial intelligence. This Review provides an introduction to this framework, and discusses some of the state-of-the-art advances in the field, namely, probabilistic programming, Bayesian optimization, data compression and automatic model discovery.

Deep Resolution of Bayesian ML

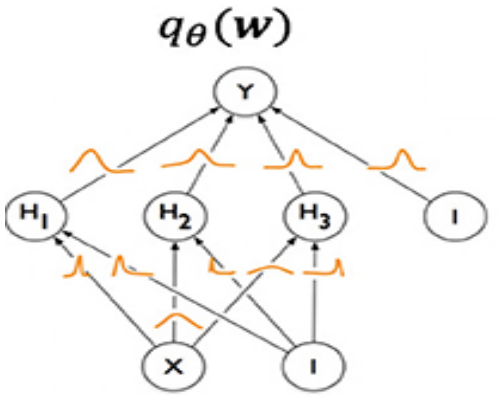
Type-1
(Bayes -> DNN)

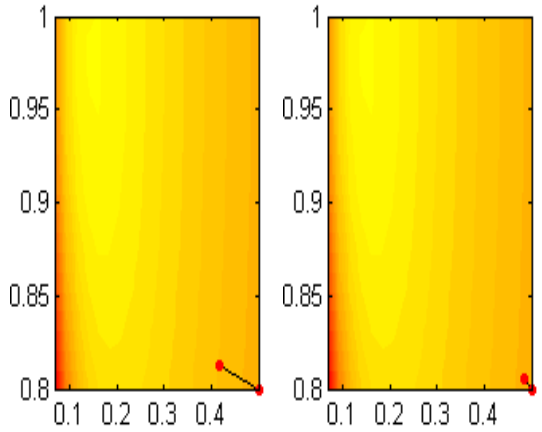
Type-2
(DNN -> Bayes)

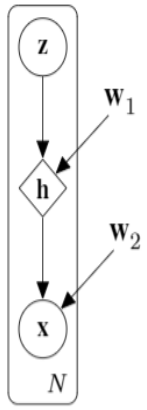




Deep Resolution of Bayesian ML





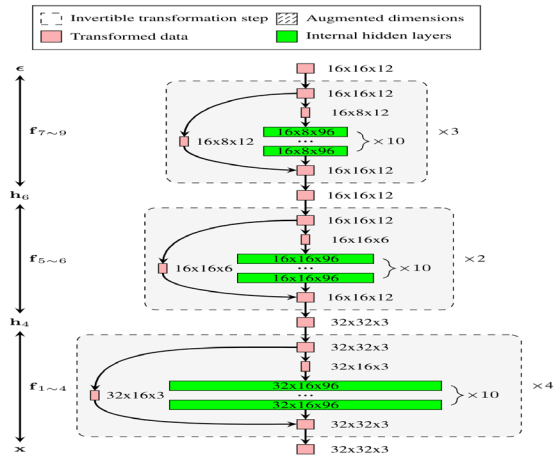


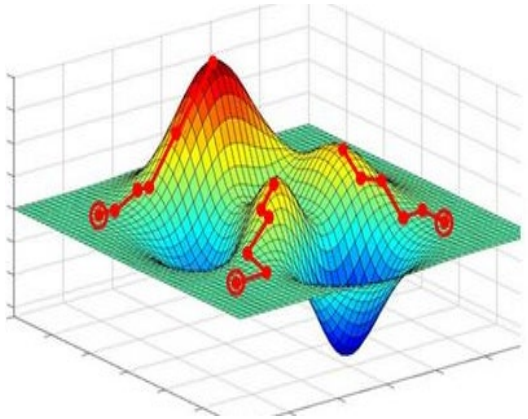
(a)

```python
@zs.meta_bayesian_net("vae", reuse_variables=True)
def build_vae(N, D):
    bn = zs.BayesianNet()
    z_mean = tf.zeros([N, D])
    z = bn.normal('z', z_mean, std=1., group_ndims=1)
    h = tf.layers.dense(z, 500,
                        activation=tf.nn.relu)
    x_logits = tf.layers.dense(h, 784)
    bn.bernoulli('x', x_logits, group_ndims=1)
    return bn
```

(b)

modeling → algorithms → programming





变分推理	梯度估计器	实现
elbo()	SGVB	.sgvb()
	Reinforce	.reinforce()
klpq()	Importance Sampling	.importance()
iw_objective()	SGVB	.sgvb()
	VIMCO	.vimco()

马尔科夫链蒙特卡洛	实现
Hamiltonian Monte Carlo	HMC()
Stochastic Gradient MCMC	(Preconditioned) SGLD	SGLD(), PSGLD()
	SGHMC	SGHMC()
	SGNHT	SGNHT()

Modeling

(How to do Bayesian inference for DNNs?

How to learn hierarchically structured Bayesian models?)

Marriage between Bayesian Methods and Neural Networks

- Dates back to 1990's …

Bayesian Methods for Adaptive Models

Thesis by

David J.C. MacKay

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

(Submitted December 10, 1991)

David MacKay
Fellow of Royal Society

John J. Hopfield
Dirac Medal
Albert Einstein World Award of Science

Bayes embodies Occam's Razor and Apply to Bayesian neural networks

Marriage between Bayesian Methods and Neural Networks

- Dates back to 1990's …

Bayesian Learning for Neural Networks

Radford M. Neal

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy,
Graduate Department of Computer Science,
in the University of Toronto
Convocation of March 1995

Abstract

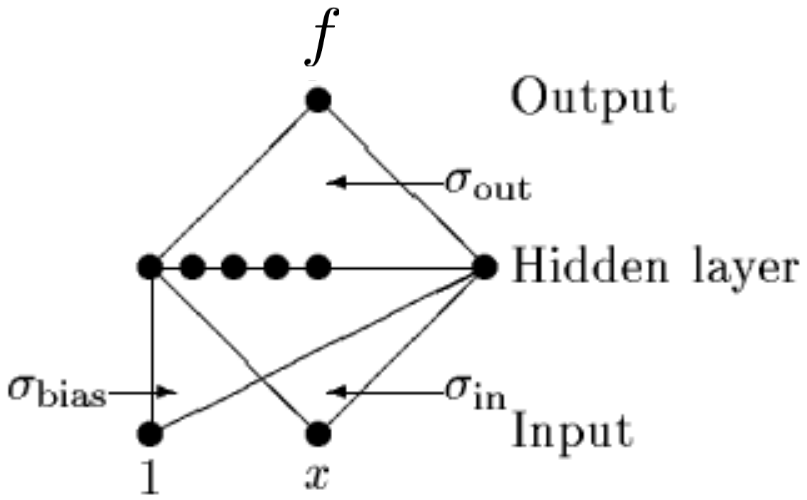
Two features distinguish the Bayesian approach to learning models from data. First, beliefs derived from background knowledge are used to select a prior probability distribution for the model parameters. Second, predictions of future observations are made by integrating the model's predictions with respect to the posterior parameter distribution obtained by updating this prior to take account of the data. For neural network models, both these aspects present difficulties — the prior over network parameters has no obvious relation to our prior knowledge, and integration over the posterior is computationally very demanding.

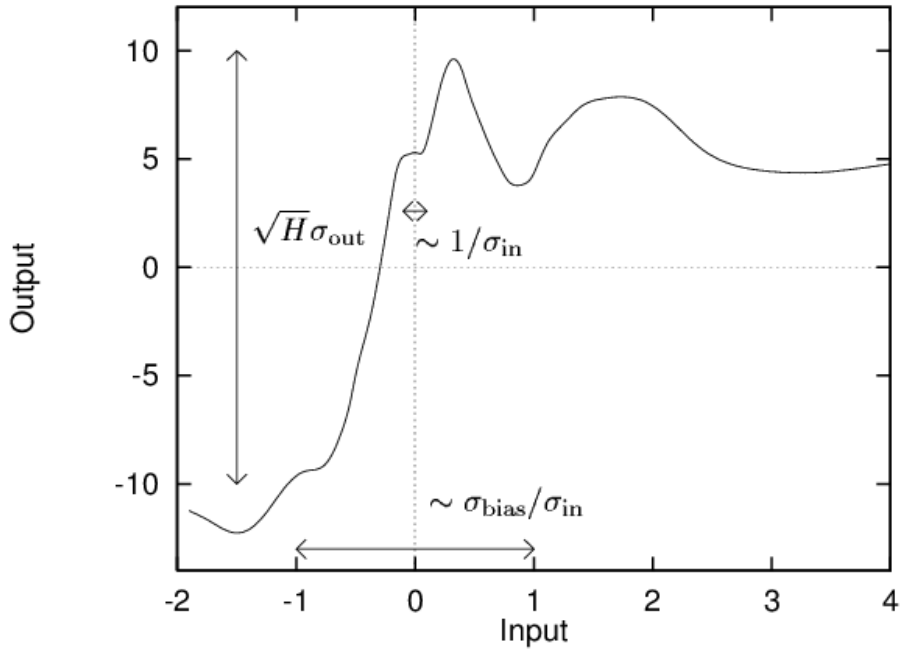
R.M. Neal
Lindley Award

G. Hinton
2018 Turing Award

Bayesian Neural Networks (BNNs)

- Two-layer Bayesian neural networks, zero-mean Gaussian prior



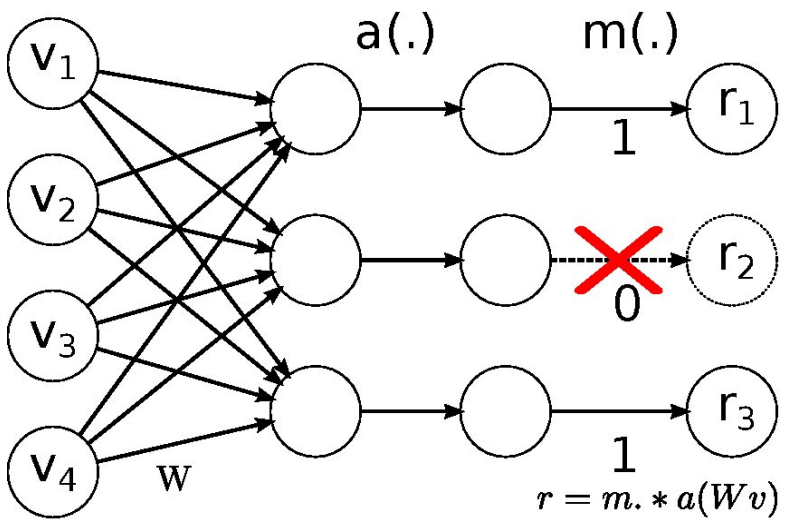


- Converge to Gaussian Processes (GPs) when the number of hidden units gets infinity [Neal, PhD thesis, 1995]
- Generally, Bayesian neural networks improve generalization, avoid overfitting!

[MacKay, Gaussian Process: a Replacement for Supervised Neural Networks?1997]
[Neal, Bayesian Learning for Neural Networks.1995]

Dropout as an Approximate Bayesian Inference

- **Dropout Training** (Hinton, 2012)



Variational posterior over the parameters

$$\mathbf{W}_i = \mathbf{M}_i \cdot \text{diag}([\mathbf{z}_{i,j}]_{j=1}^{K_i})$$
$$\mathbf{z}_{i,j} \sim \text{Bernoulli}(p_i) \text{ for } i = 1, ..., L,\ j = 1, ..., K_{i-1}$$

Minimize the KL-divergence between q and the true posterior of dGP

$$-\int q(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} + \text{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})).$$

- **MC-Dropout**: estimate the predictive uncertainty

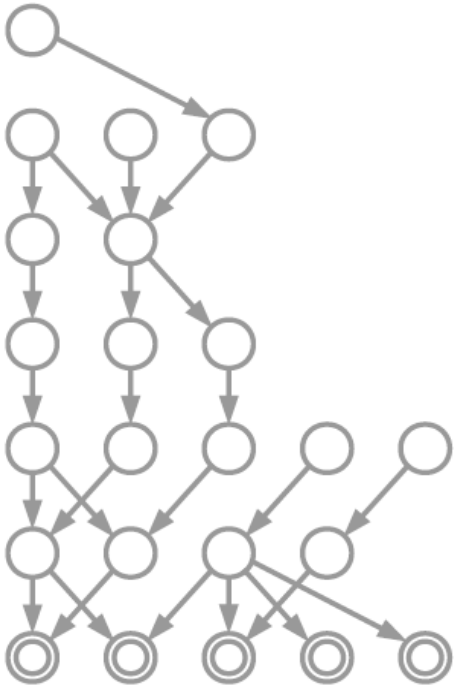
$$\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}((\mathbf{y}^*)^T(\mathbf{y}^*)) \approx \tau^{-1}\mathbf{I}_D + \frac{1}{T}\sum_{t=1}^{T} \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{W}_1^t, ..., \mathbf{W}_L^t)^T \widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{W}_1^t, ..., \mathbf{W}_L^t)$$

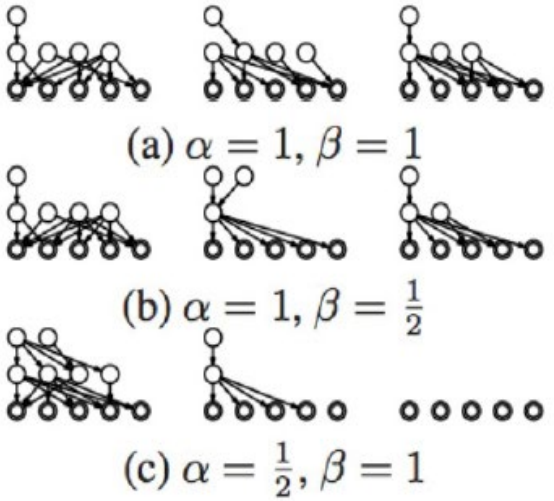
[Gal & Ghahramani, Dropout as a Bayesian Approxiamtion, ICML 2016]

Bayesian Methods for Inferring Architectures

- Nonparametric Bayesian neural networks (Adams et al., 2010, AISTATS Best Paper)

# of layers
# of units at each layer
Edges between units
Type of hidden units (discrete or continuous?)

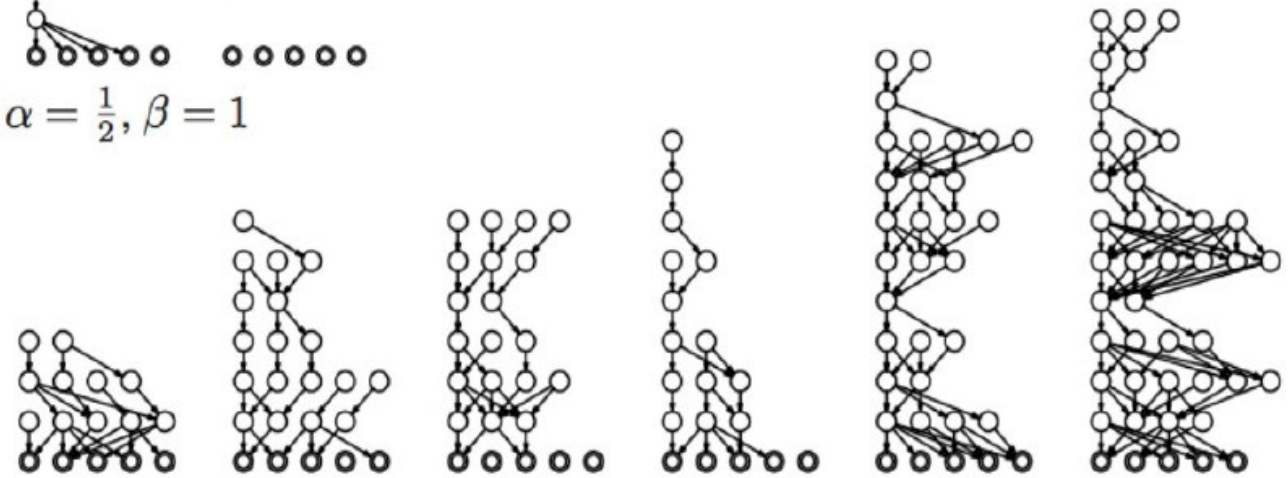




(a) $\alpha = 1, \beta = 1$

(b) $\alpha = 1, \beta = \frac{1}{2}$

(c) $\alpha = \frac{1}{2}, \beta = 1$



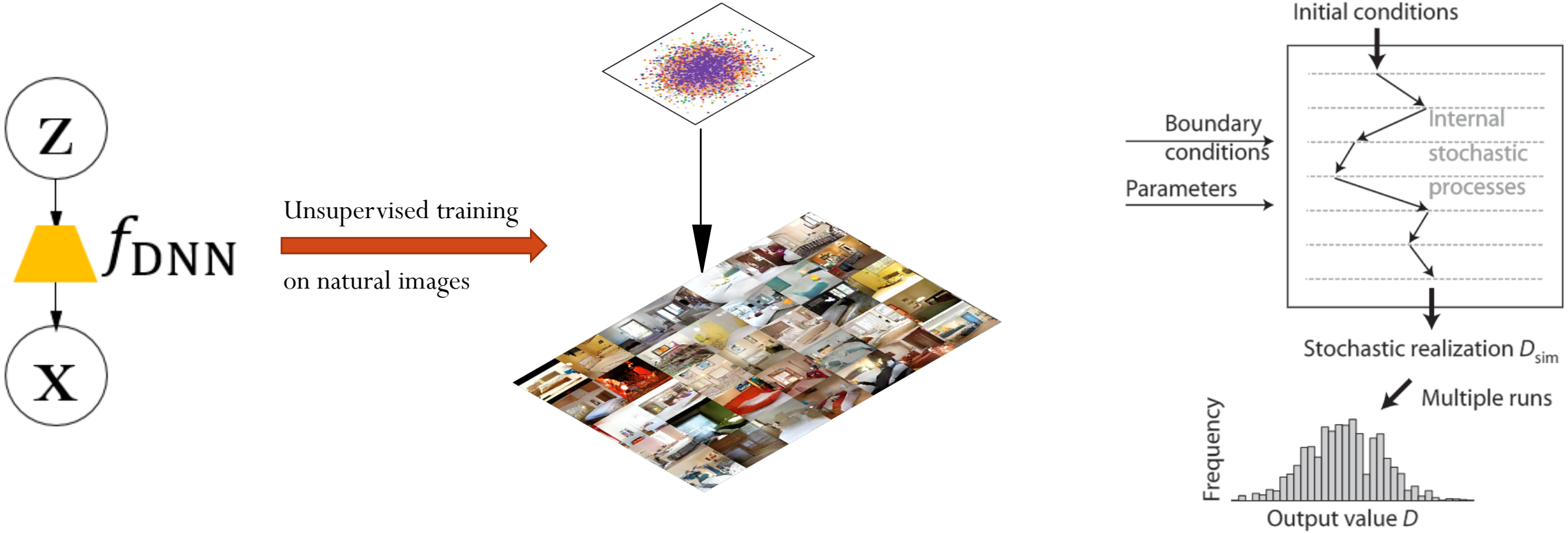
(d) $\alpha = 1, \beta = 2$

(e) $\alpha = \frac{3}{2}, \beta = 1$

Cascading Indian Buffet Process (IBP)

Deep Bayesian Learning

- Use DNNs to fit (learn) the complex relationships between random variables



(Hartig et al., 2011)

- Two types: Explicit models (e.g., VAE, Flow-based Models), Implicit models (e.g., GAN)

Flow-based Models

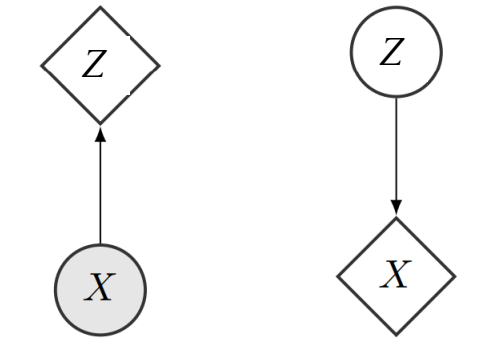
- A generative process with invertible (or bijective) function g:

$$z \sim p(z)$$

$$x = g(z)$$

- The inference network for the latent variable is:

$$z = f(x) = g^{-1}(x)$$



(a) Inference (b) Sampling

- The density:

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

- Desirae of the function: easy determinant of Jacobian, easy inverse

[Dinh, Krueger, Bengio. Nice: non-linear independent components estimation, workshop at ICLR 2015]

An Example of the Invertible Function

- A simple idea: split *x* into two parts and define

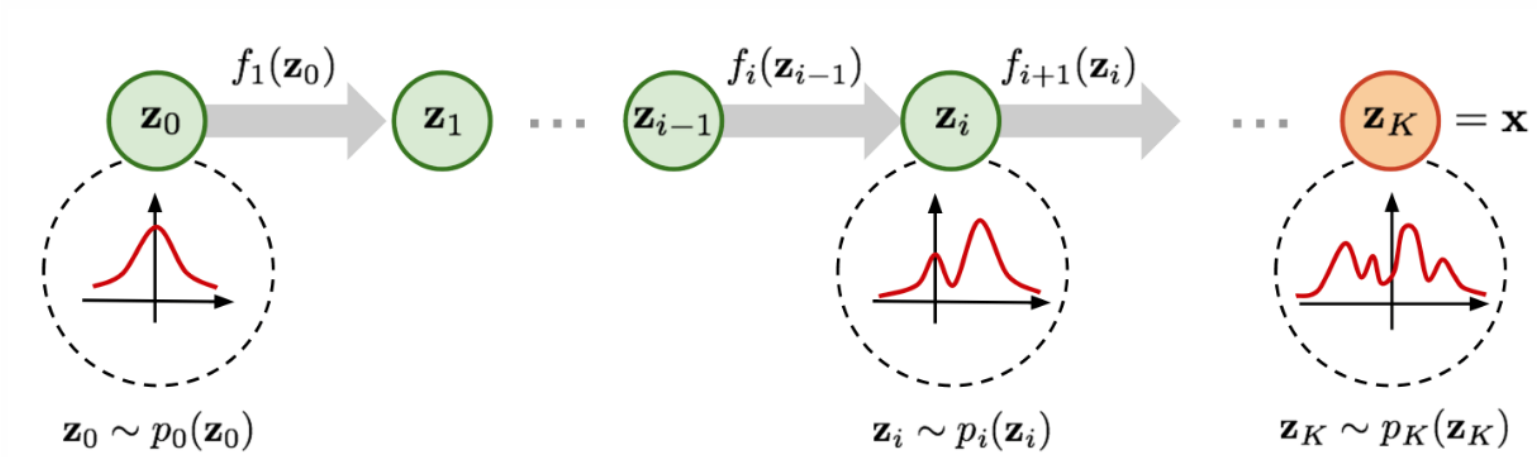
$$y_1 = x_1$$
$$y_2 = x_2 + m(x_1)$$

  - *m* is an arbitrarily complex function (e.g., ReLU networks)

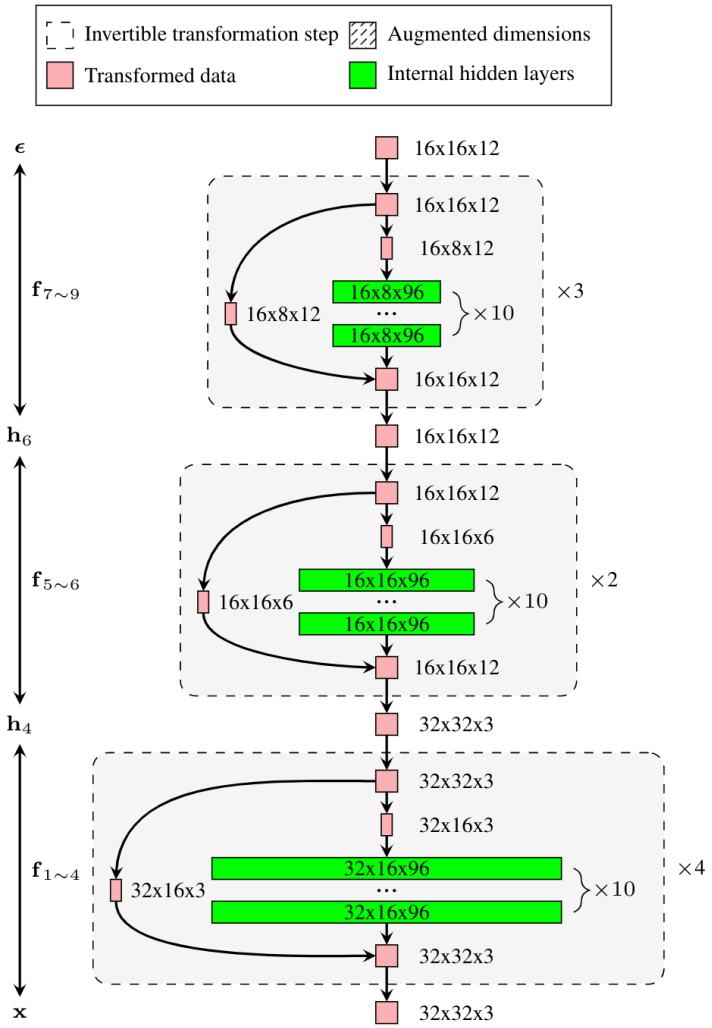
- Then the determinant-Jacobian 1!
- The inverse is

$$x_1 = y_1$$
$$x_2 = y_2 - m(y_1)$$

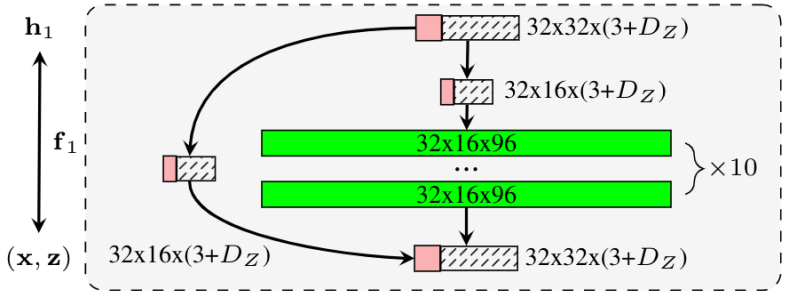
- Define the invertible transform function as a composition of simple functions
  - Thus, the variable change is a sequence, called a ***flow*** (the sequence on density change is a ***normalizing flow***)



Bottleneck Problem in Flow++



Invertibility imposes constraints on architectures!



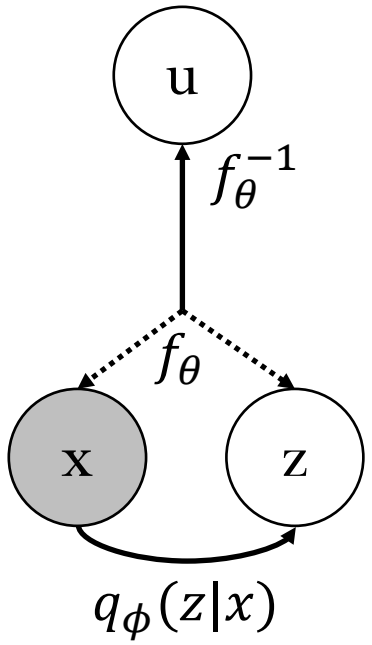
(b)

Figure 1. (a) Bottleneck problem in a Flow++ (Ho et al., 2019) for CIFAR-10. Dimensionality of the transformed data (red) limits the model capacity. (b) Our solution VFlow, where D_Z is the dimensionality of the augmented random variable. Only the transformation step $\mathbf{f}_1$ is shown due to space constraint.

[Chen, Lu, Chenli, Zhu, Tian. *VFlow: More Expressive Generative Flows with Variational Data Augmentation*. ICML 2020]

VFlow: more Expressive Generative Flows through Variational Data Augmentation

learn a generative flow $p(x, z)$ in the augmented data space jointly with the augmented data distribution



$$u \sim \mathcal{N}(0, I)$$

$$y = concat([x, z])$$

$$\log p(x, z) = \log p(u) + \log \det\left(\frac{du}{dy}\right)$$

$$\log p(x) \geq \mathrm{E}_{q(z|x)}\left[\log p(x, z) - \log q(z|x)\right]$$

$$\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathbb{E}_{\hat{p}(\mathbf{x}) q(\mathbf{z}|\mathbf{x};\boldsymbol{\phi})}\left[\log p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) - \log q(\mathbf{z}|\mathbf{x}; \boldsymbol{\phi})\right]$$

Provably better than generative flows! Also subsumes VAEs as a special case.

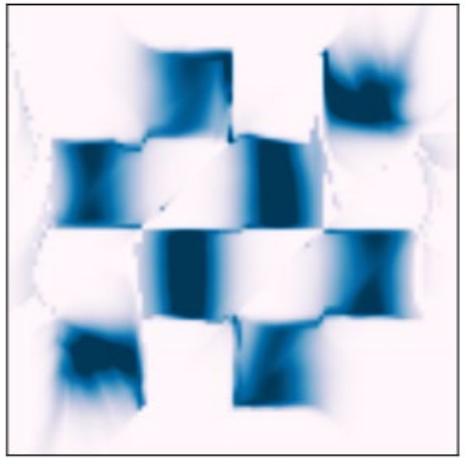
[Chen, Lu, Chenli, Zhu, Tian. *VFlow: More Expressive Generative Flows with Variational Data Augmentation*. ICML 2020]

Results on Toy 2D Data

- VFlow significantly outperforms Glow under similar model size (e.g., 3-step)
- The 3-layer, 10-dimensional VFlow even outperforms a much larger 20-step Glow

(a) Data (-3.47)

(b) 3-step, 10-dim VFlow (-3.51)



(c) 3-step Glow (-3.66)

(d) 20-step Glow (-3.52)

Results on Toy 2D Data

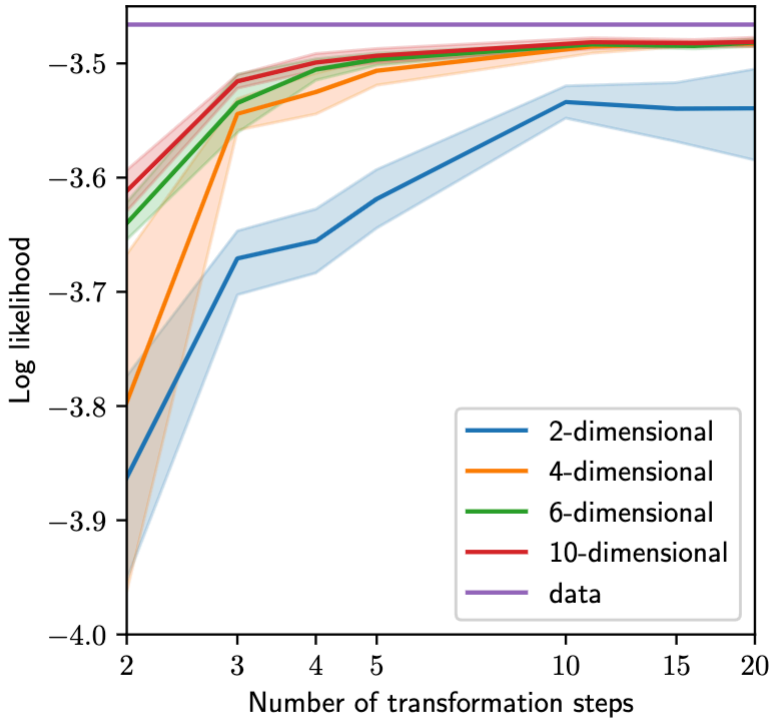


Figure 4. Impact of the dimensionality on the toy dataset.

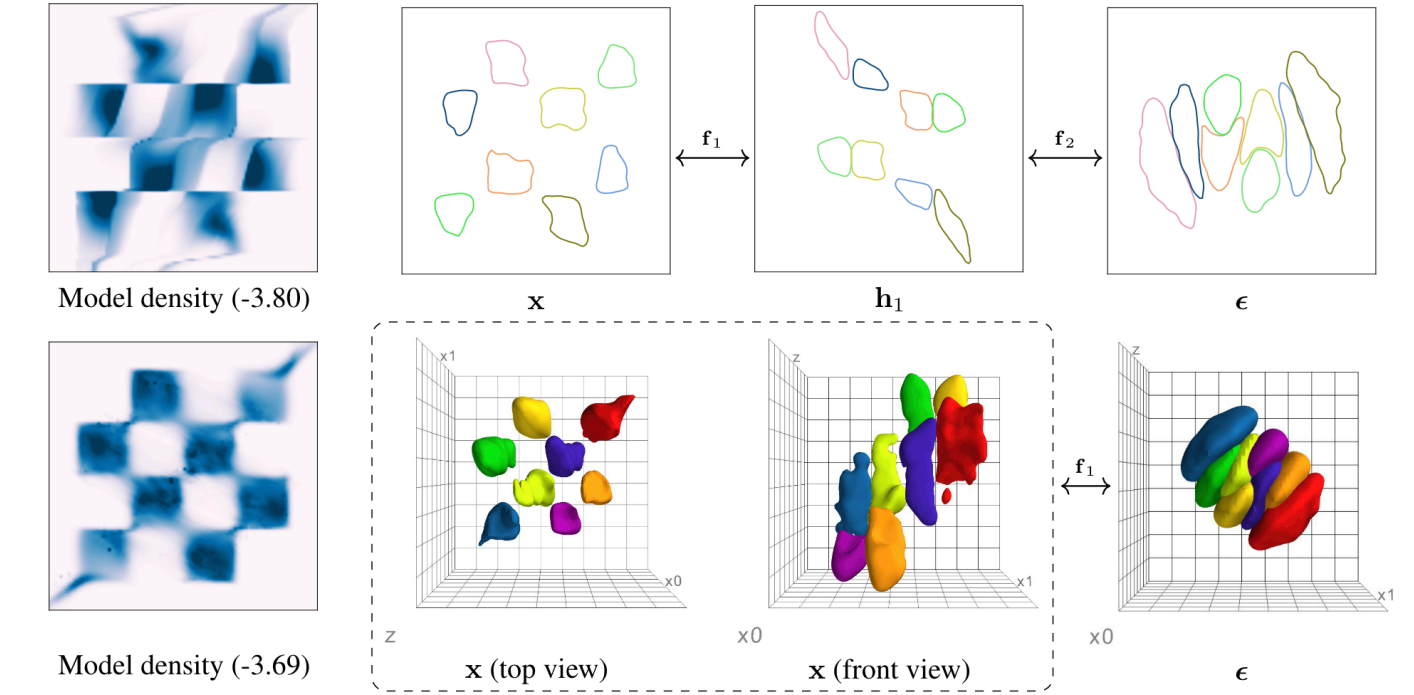


Figure 3. Visualization of learnt transformation on toy data. Top row: 2-step Glow. Bottom row: 2-step, 3-dimensional VFlow. Log-likelihood is shown in parenthesis. We sample ϵ and visualize the transformed density in $\mathbf{x}$, $\mathbf{h}_1$ and ϵ space. The density is estimated from samples by kernel density estimation, and we show the 50% probability contour / isosurface for each mode in different color.

[Chen, Lu, Chenli, Zhu, Tian. *VFlow: More Expressive Generative Flows with Variational Data Augmentation*. ICML 2020]

Results on CIFAR-10

Table 1. Density modeling result on the CIFAR-10 dataset.

Model	bpd
Glow (Kingma & Dhariwal, 2018)	3.35
FFJORD (Grathwohl et al., 2019)	3.40
Residual Flow (Chen et al., 2019)	3.28
MintNet (Song et al., 2019)	3.32
Flow++ (Ho et al., 2019)	3.08
VFlow	**2.98**

Table 3. Parameter efficiency.

Model	bpd	Parameters	D_H	B
3-channel Flow++	3.08	31.4M	96	10
6-channel VFlow	**2.98**	37.8M	96	10
6-channel VFlow	3.03	16.5M	64	10
6-channel VFlow	3.08	**11.9M**	56	10

Table 2. Impact of dimensionality on the CIFAR-10 dataset.

Model	bpd	Parameters	D_H	B
3-channel Flow++	3.21	4.02M	32	13
4-channel VFlow	3.15	4.03M	32	11
6-channel VFlow	**3.12**	**4.01M**	32	10

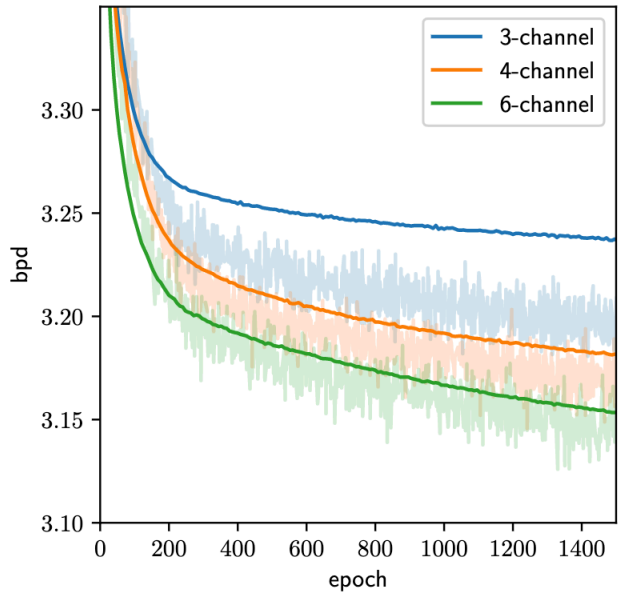


Figure 6. Bpd on training (light) and testing (dark) dataset of Flow++ and VFlow under a 4-million parameter budget. Here bpd is only a upper bound because we evaluate it with ELBO as Eq. (7) instead of the marginal likelihood.

[Chen, Lu, Chenli, Zhu, Tian. *VFlow: More Expressive Generative Flows with Variational Data Augmentation*. ICML 2020]

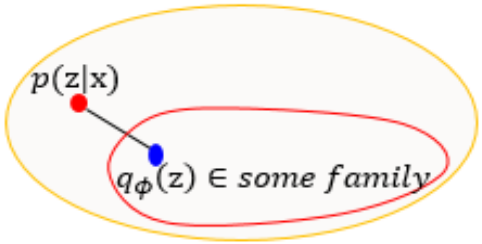
Algorithms

(How to compute posteriors efficiently? How to learn parameters?)

Two Types of Algorithms

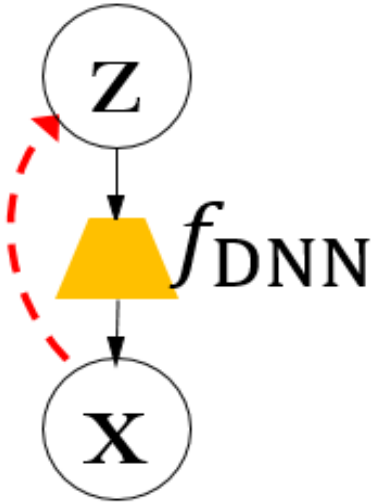
Variational Method

(Too much math!!!)



$$\min_{\phi} \mathrm{KL}(q(z;\phi)\|p(z|x))$$

Variational posterior — True posterior

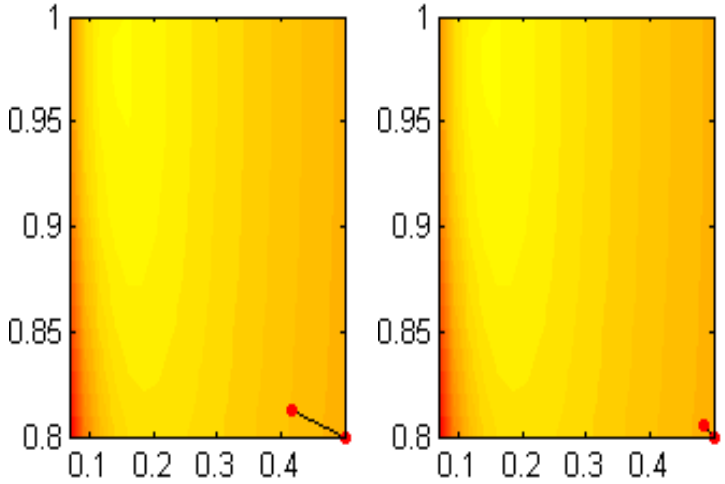


[Wainwright & Jordan, 2008]

Efficient, but with Approximation Error in general

Monte Carlo Methods

(Many dynamics!!!)



$$\begin{cases} \mathrm{d}q = G^{-1}p\mathrm{d}t \\ \mathrm{d}p = -\nabla_q U\mathrm{d}t - \frac{1}{2}\nabla_q \log|G|\mathrm{d}t - \xi p\,\mathrm{d}t - \frac{1}{2}\nabla_q\left[p^{\top}G^{-1}p\right]\mathrm{d}t + \mathcal{N}(0, 2CG\mathrm{d}t) \\ \mathrm{d}\xi = (\frac{1}{m}p^{\top}G^{-1}p - 1)\mathrm{d}t \end{cases}$$

Leverage geometry to improve efficiency of MCMC [GC11]

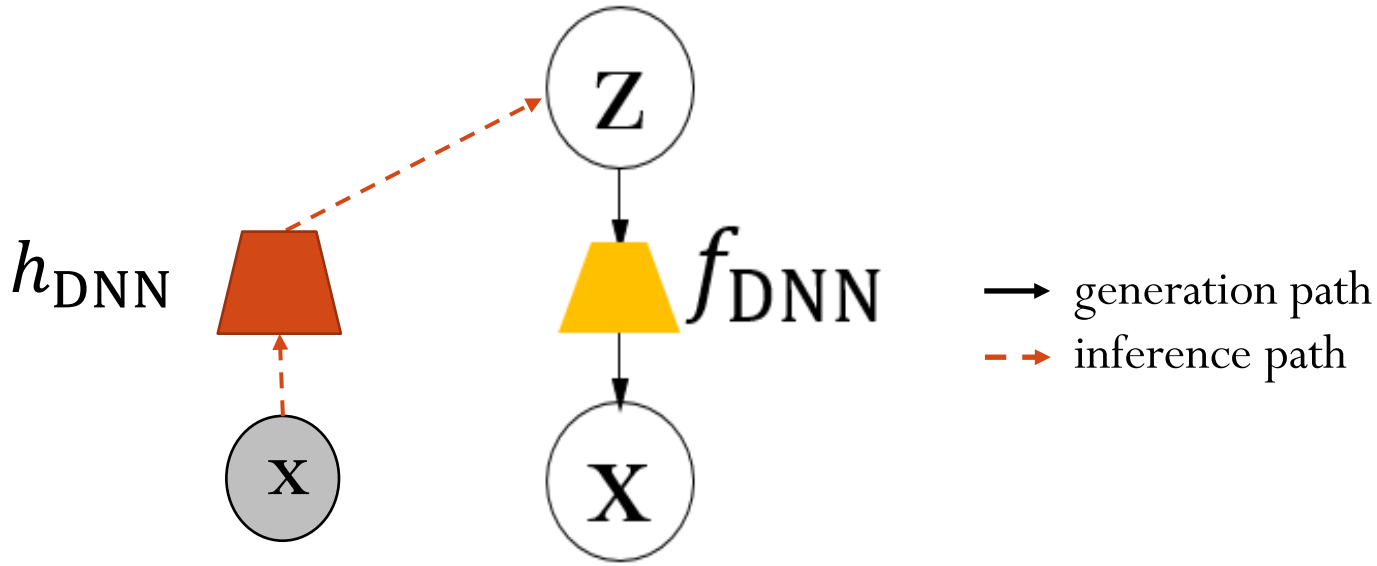
Accurate, but Low Efficiency in Speed and Particles

"Explicitly" Define Variational Distribution q via an Encoder

- Variational Auto-Encoders (VAE, Kingma & Welling, 2013)

$$q(\mathrm{z}|\mathrm{x};\phi) \approx p(\mathrm{z}|\mathrm{x};\theta)$$

ELBO: $$L(\theta,\phi,\mathrm{x}) = \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{x}|\mathrm{z};\theta)] - \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log q(\mathrm{z}|\mathrm{x};\phi)]$$



- Jointly optimize the parameters of decoder and encoder networks θ, ϕ: SGD

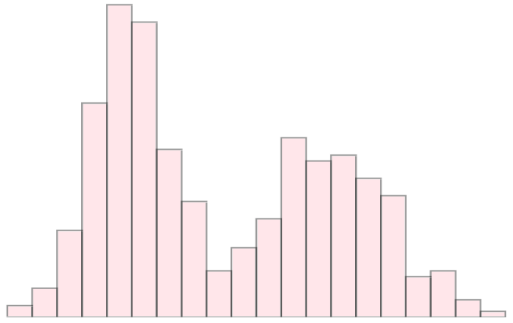
Variational inference with Implicit q

- Can we do variational inference when q is an implicit distribution?

$$L(\theta, \phi, \mathrm{x}) = \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{x}|\mathrm{z};\theta)] - \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log q(\mathrm{z}|\mathrm{x};\phi)]$$

  - The objective can be estimated via Monte Carlo, but we can't computer gradients!

- **A fundamental task**: Can we directly estimate the gradient (score) function from samples of some unknown density?

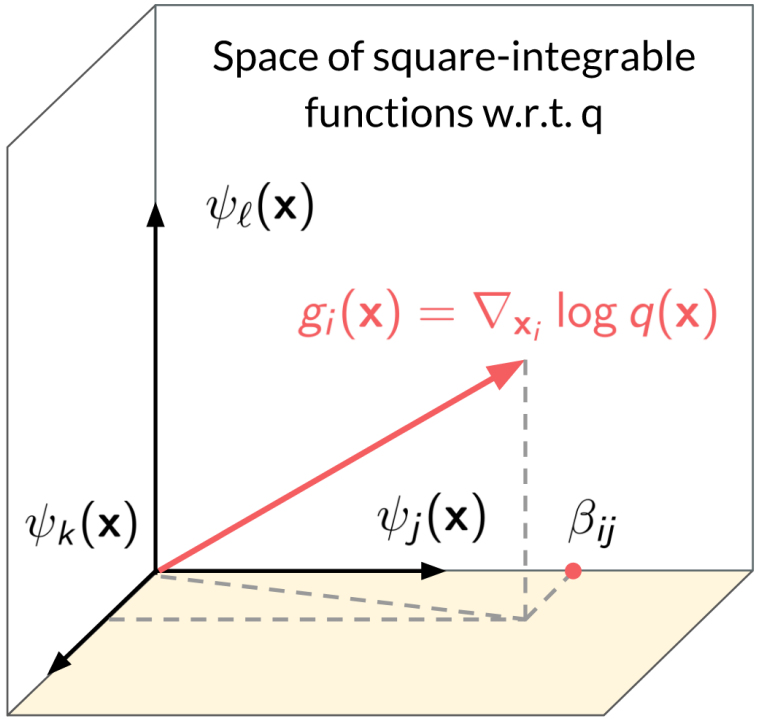


$$x^{1:M} \sim q$$

$$g(\mathbf{x}) = \nabla_{\mathbf{x}} \log q(\mathbf{x})$$

[A Spectral Approach to Gradient Estimation for Implicit Distributions. Shi et al., ICML 2018]

A Spectral Approach to Gradient Estimation for Implicit Distributions



We proved that (under mild assumptions)

$$\nabla_{x_i} \log q(\mathbf{x}) = -\sum_{j=1}^{\infty} \Big[\mathbb{E}_q \nabla_{x_i} \psi_j(\mathbf{x})\Big] \psi_j(\mathbf{x})$$

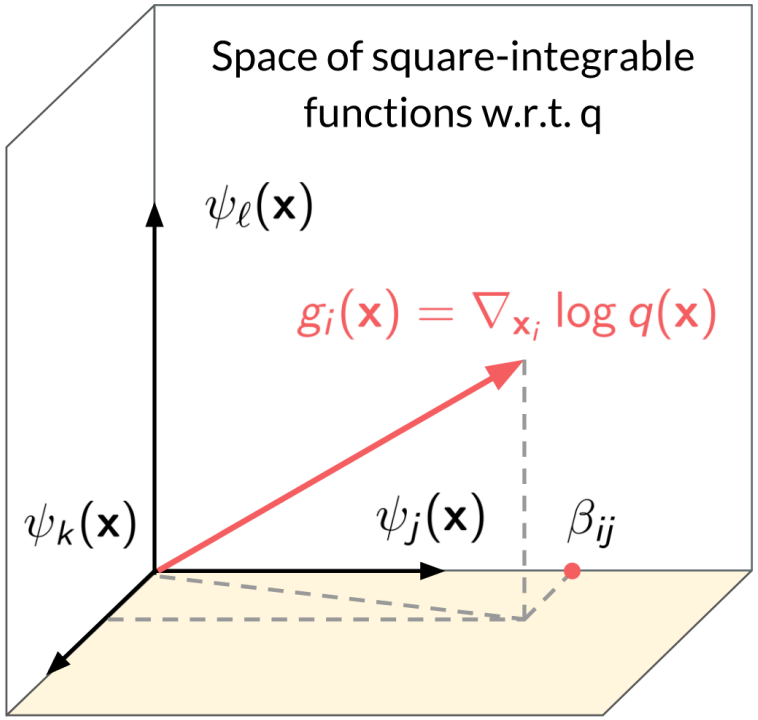
This orthonormal basis can be constructed by spectral decomposition of a p.d. kernel

$$\int k(\mathbf{x}, \mathbf{y}) \psi_j(\mathbf{y}) q(\mathbf{y}) d\mathbf{y} = \mu_j \psi_j(\mathbf{x})$$

[A Spectral Approach to Gradient Estimation for Implicit Distributions. Shi et al., ICML 2018]

A Spectral Approach to Gradient Estimation for Implicit Distributions

- Approximate the eigenfunctions by Nyström method [Nyström, 1930]
- Truncate the series with a finite number of basis functions, according to large eigenvalues



Monte Carlo Nyström approximation

$$\nabla_{x_i} \log q(\mathbf{x}) = -\sum_{j=1}^{J} \left[\mathbb{E}_q \nabla_{x_i} \bar{\psi}_j(\mathbf{x})\right] \bar{\psi}_j(\mathbf{x})$$

Spectral Stein Gradient Estimator

[A Spectral Approach to Gradient Estimation for Implicit Distributions. Shi et al., ICML 2018]

A Spectral Approach to Gradient Estimation for Implicit Distributions

Theorem (Error Bound) Given mild assumptions, the error

$$\int |\hat{g}_i(\mathbf{x}) - g_i(\mathbf{x})|^2 q(\mathbf{x})\, d\mathbf{x}$$

Our estimator True gradient

is bounded by

$$J^2 \left(O_p\left(\frac{1}{M}\right) + O_p\left(\frac{C}{\mu_J \Delta_J^2 M}\right)\right) + J O_p\left(\frac{C}{\mu_J \Delta_J^2 M}\right) + \|g_i\|_{\mathcal{H}}^2 O(\mu_J),$$

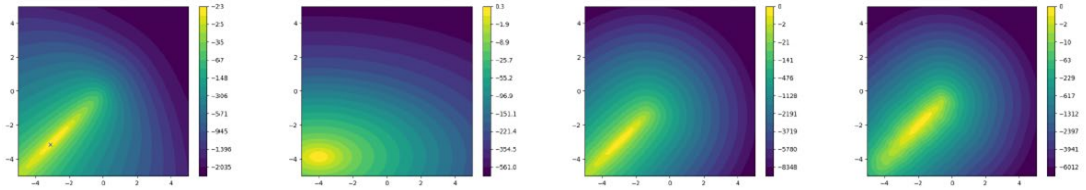
Estimation error

Approximation error due to truncation

where $\Delta_J = \min_{1 \le j \le J} |\mu_j - \mu_{j+1}|$

[A Spectral Approach to Gradient Estimation for Implicit Distributions. Shi et al., ICML 2018]

A Spectral Approach to Gradient Estimation for Implicit Distributions

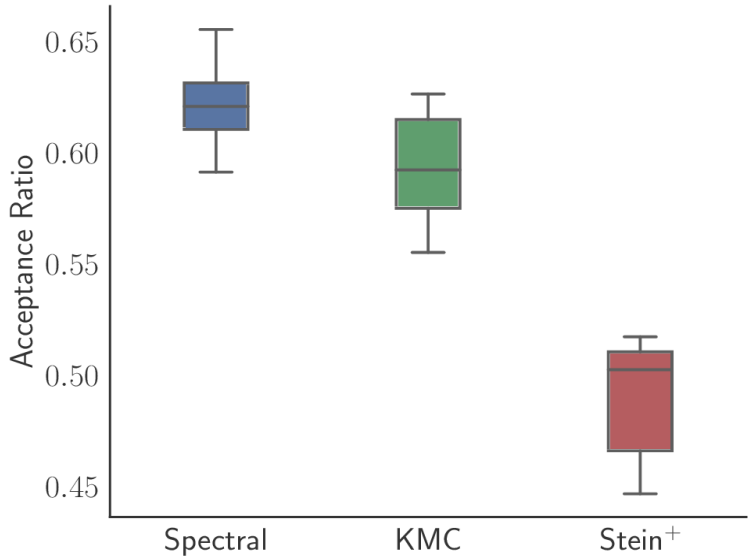


(b) True posterior (c) VI (factorized) (d) HMC (e) KIVI

Variational Inference with Implicit Distributions

(b) Implicit VAE, w/o entropy (c) Implicit VAE, Spectral

Improving Deep Generative Models



Gradient-free Hamiltonian Monte Carlo

[A Spectral Approach to Gradient Estimation for Implicit Distributions. Shi et al., ICML 2018]

Nonparametric Score Estimators
A Unifying Theory with Improved Results

- **Main Result1**: a table summary of existing estimators under the unifying framework

Table 1. Existing nonparametric score estimators, their kernel types, and regularization schemes. ϕ is from $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x} - \mathbf{y})$.

ALGORITHM	KERNEL	REGULARIZER
SSGE	$k(\mathbf{x}, \mathbf{y})\mathbf{I}_d$	$\mathbf{1}_{\{\sigma \geq \lambda\}}\sigma^{-1}$
Stein	$k(\mathbf{x}, \mathbf{y})\mathbf{I}_d$	$\mathbf{1}_{\{\sigma > 0\}}(\lambda + \sigma)^{-1}$
KEF	$-\nabla^2\phi(\mathbf{x} - \mathbf{y})$	$(\lambda + \sigma)^{-1}$
NKEF	$-\nabla^2\phi(\mathbf{x} - \mathbf{y})$	$\mathbf{1}_{\{\sigma > 0\}}(\lambda + \sigma)^{-1}$

[Nonparametric Score Estimators. Zhou et al., ICML 2020]

Nonparametric Score Estimators
A Unifying Theory with Improved Results

- **Main Result2**: the convergence rate of the estimator

$$\lambda = M^{-\frac{1}{2r+2}}, \qquad \|\hat{s}^g_{p,\lambda} - s_p\|_{\mathcal{H}_\mathcal{K}} = O_p\left(M^{-\frac{r}{2r+2}}\right)$$

$$r \in [0, \bar{r} - 1/2], \qquad \|\hat{s}^g_{p,\lambda} - s_p\|_\rho = O_p\left(M^{-\frac{r+1/2}{2r+2}}\right)$$

  - Improves over (Shi et al., ICML 2018)
  - Recover previous results of kernel exponential family (KEF) (Sriperumbudur et al., 2017)
  - Provide a bound for Li & Turner (2018)

[Nonparametric Score Estimators. Zhou et al., ICML 2020]

More Algorithms on Posterior Inference

Variational Inference:

- Variance reduction and quasi-Newton for particle-based variational inference (Zhu, Liu, Zhu, ICML 2020)
- SUMO: Unbiased Estimation of Log Marginal Probability for Latent Variable Models (Luo et al., ICLR 2020, spotlight)
- Understanding and Accelerating Particle-based Variational Inference (Liu et al., ICML 2019)
- Scalable Training of Inference Networks for Gaussian-Process Models (Shi, Khan, Zhu, ICML 2019)
- Riemannian Stein Variational Gradient Descent for Bayesian Inference (Liu, Zhu, AAAI 2018)
- Kernel Implicit Variational Inference (Shi, Sun, Zhu, ICLR 2018)

MCMC:

- Understanding MCMC Dynamics as Flows on the Wasserstein Space (Liu, Zhuo, Zhu, ICML 2019)
- Stochastic gradient Hamiltonian Monte Carlo with variance reduction for Bayesian inference (Li et al., Machine Learning, 2020)
- Function space particle optimization for Bayesian neural networks (Wang et al., ICLR 2019)

Learning Unnormalized Models:

- To Relieve Your Headache of Training an MRF, Take AdVIL (Li et al., ICLR 2020)
- A Wasserstein Minimum Velocity Approach to Learning Unnormalized Models (Wang et al., AISTATS 2020)
- Efficient learning of generative models via finite-difference score matching (Pang et al., NeurIPS 2020)

Probabilistic Programming Library

(how to auto/semi-auto implement Bayesian deep learning models?)

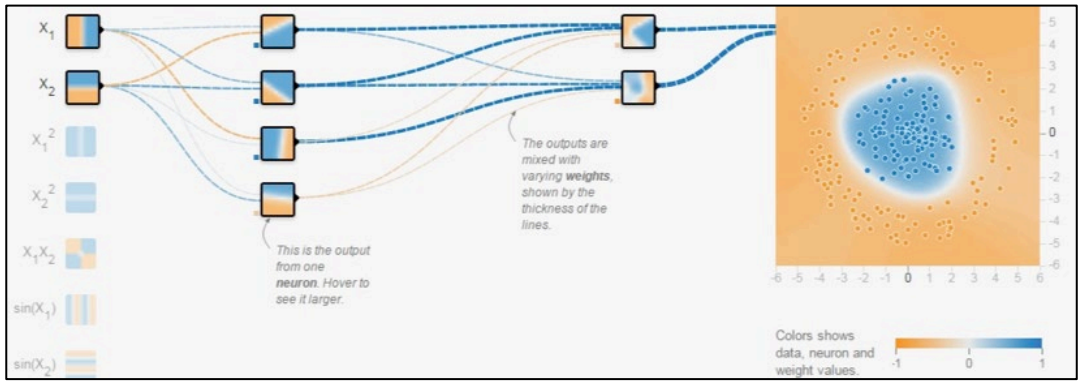
DNN Programming → Deep Probabilistic Programming (Deep PPL)

Differential programming

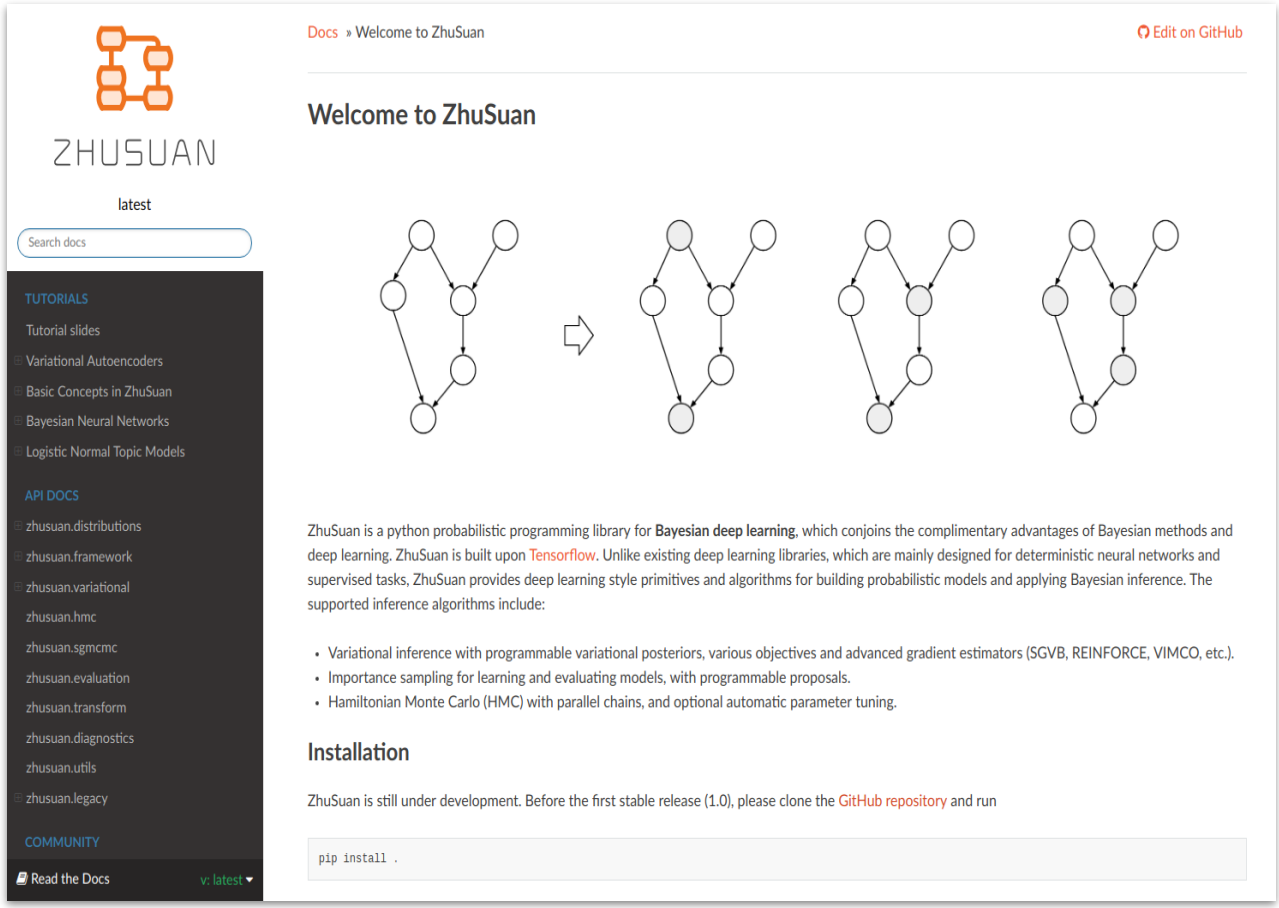
TensorFlow

PYTORCH

mxnet



Learning a mapping from x to y



github.com/thu-ml/zhusuan

“ZhuSuan” Deep Probabilistic Programming

ZHUSUAN

Improved modeling language

```
def build_model(...):
    bn = zs.BayesianNet()
    x = bn.normal(“x”, x_mean, std=...)
    y = bn.deterministic(“y”, tf.any_op(x))
Or  y = tf.any_op(x)
    ......
    return bn
```

1. Model build function
2. Create a Bayesian Net
3. Add stochastic node, e.g., a Gaussian variable x
4. Add deterministic nodes, can use any Tensorflow operation
5. Return the built Bayesian Net

“ZhuSuan” Deep Probabilistic Programming

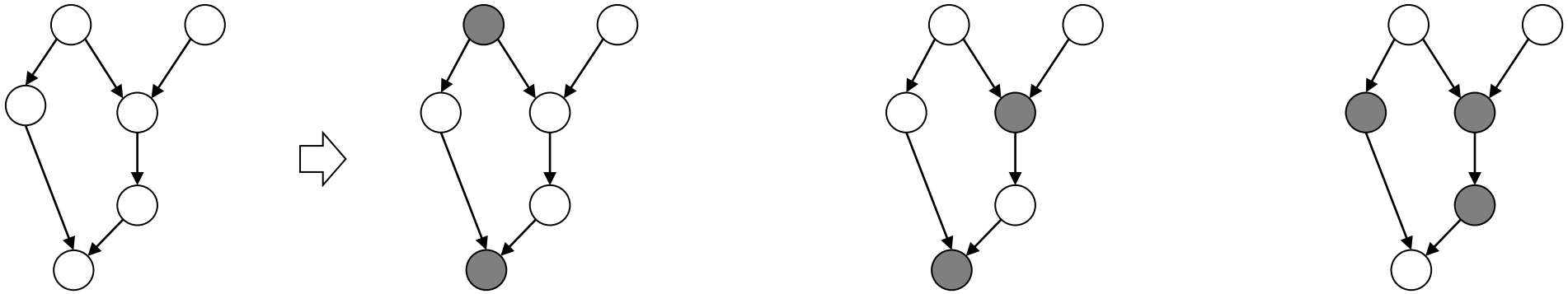
ZHUSUAN

New model reuse strategy

Just need a decorator `@zs.meta_bayesian_net` to the model-building function

```
@zs.meta_bayesian_net(scope="model", reuse_variables=True)
def build_model(...):
```

The returned is a reusable `zs.MetaBayesianNet` object, just call `observe()` to assign observed values to any subset of random variables



`zs.MetaBayesianNet` → `zs.BayesianNet`

```
bn = meta_bn.observe(x=x_obs, y=y_obs)
```

github.com/thu-ml/zhusuan

“ZhuSuan” Deep Probabilistic Programming

ZHUSUAN

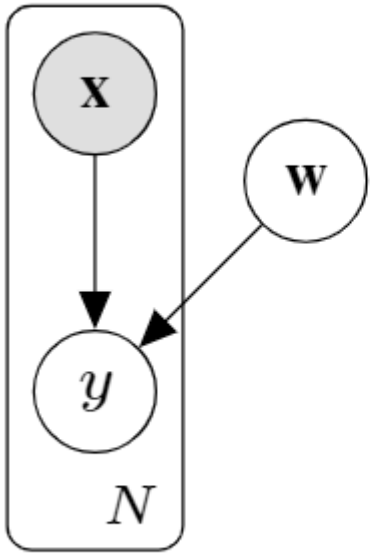
New Algorithms: Stochastic gradient MCMC

变分推理	梯度估计器	实现
elbo()	SGVB	.sgvb()
	Reinforce	.reinforce()
klpq()	Importance Sampling	.importance()
iw_objective()	SGVB	.sgvb()
	VIMCO	.vimco()

马尔科夫链蒙特卡洛		实现
Hamiltonian Monte Carlo		HMC()
Stochastic Gradient MCMC	(Preconditioned) SGLD	SGLD(), PSGLD()
	SGHMC	SGHMC()
	SGNHT	SGNHT()

github.com/thu-ml/zhusuan

Examples: Bayesian Logistic Regression



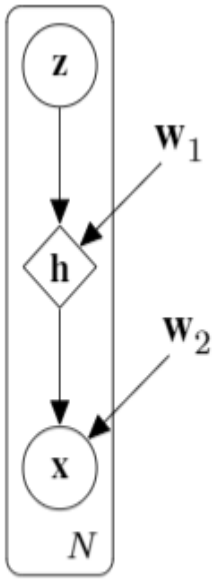
(a)

```python
import zhusuan as zs
import tensorflow as tf

@zs.meta_bayesian_net(scope="blr")
def build_blr(x, alpha, D):
    bn = zs.BayesianNet()
    w = bn.normal('w', mean=tf.zeros([D]),
                  std=alpha, group_ndims=1)
    y_logit = tf.reduce_sum(
        tf.expand_dims(w, 0)*x, axis=1)
    bn.bernoulli('y', y_logit)
    return bn
```

(b)

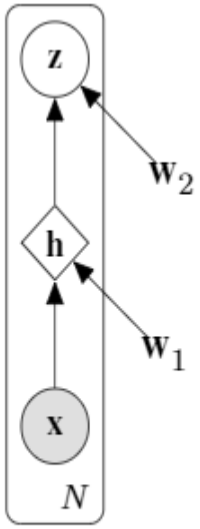
Example: Variational Auto-Encoder



(a)

```python
@zs.meta_bayesian_net("vae", reuse_variables=True)
def build_vae(N, D):
    bn = zs.BayesianNet()
    z_mean = tf.zeros([N, D])
    z = bn.normal('z', z_mean, std=1., group_ndims=1)
    h = tf.layers.dense(z, 500,
                        activation=tf.nn.relu)
    x_logits = tf.layers.dense(h, 784)
    bn.bernoulli('x', x_logits, group_ndims=1)
    return bn
```

(b)



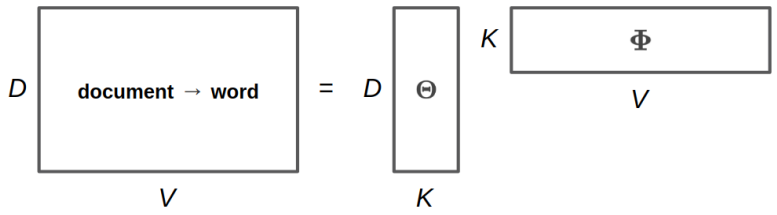
(a)

```python
@zs.reuse_variables("vae_q")
def build_q_net(h_dim, z_dim):
    bn = zs.BayesianNet()
    h = tf.layers.dense(x, h_dim,
                        activation=tf.nn.relu)
    z_mean = tf.layers.dense(h, z_dim)
    z_logstd = tf.layers.dense(h, z_dim)
    bn.normal('z', z_mean, logstd=z_logstd,
              group_ndims=1)
    return bn
```

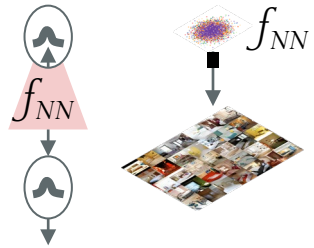
(b)

“ZhuSuan” Deep Probabilistic Programming

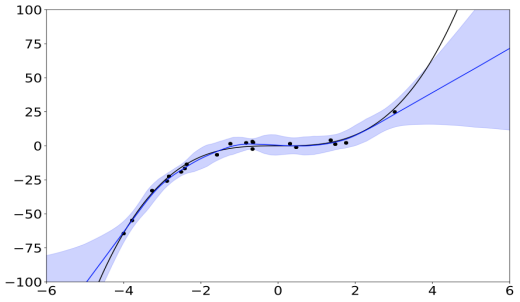
More models



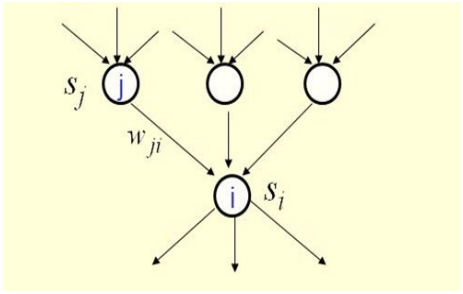
Topic Models; Probabilistic Matrix Factorization
for text analysis, Recommendor Sys.



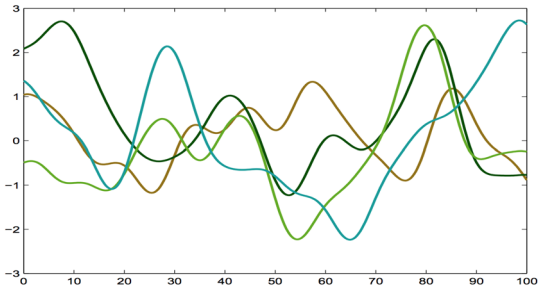
Variational Auto-encoder
for structured data generation, discrete representation learning, semi-supervised learning, etc.



Bayesian Neural Networks
for uncertainty in deep nets



Deep Belief Nets
Unsupervised pre-training
for DNNs



Gaussian Processes
Nonlinear regression problems

github.com/thu-ml/zhusuan

“ZhuSuan” Deep Probabilistic Programming

- Open-sourced in GitHub:

https://github.com/thu-ml/zhusuan

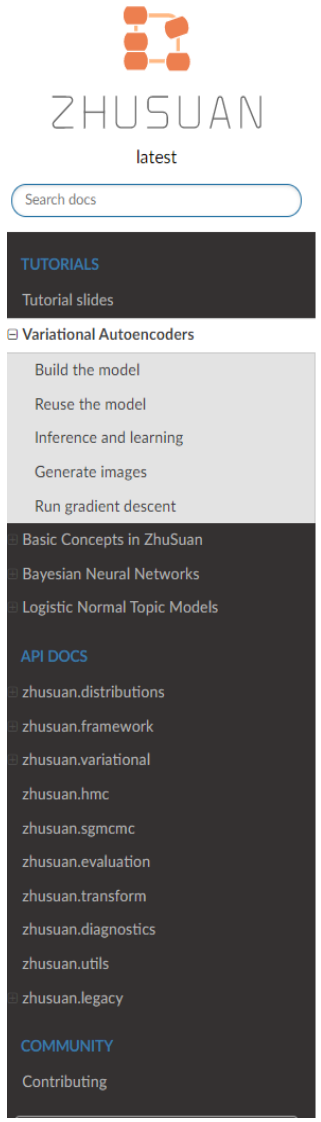
ZhuSuan: A Library for Bayesian Deep Learning

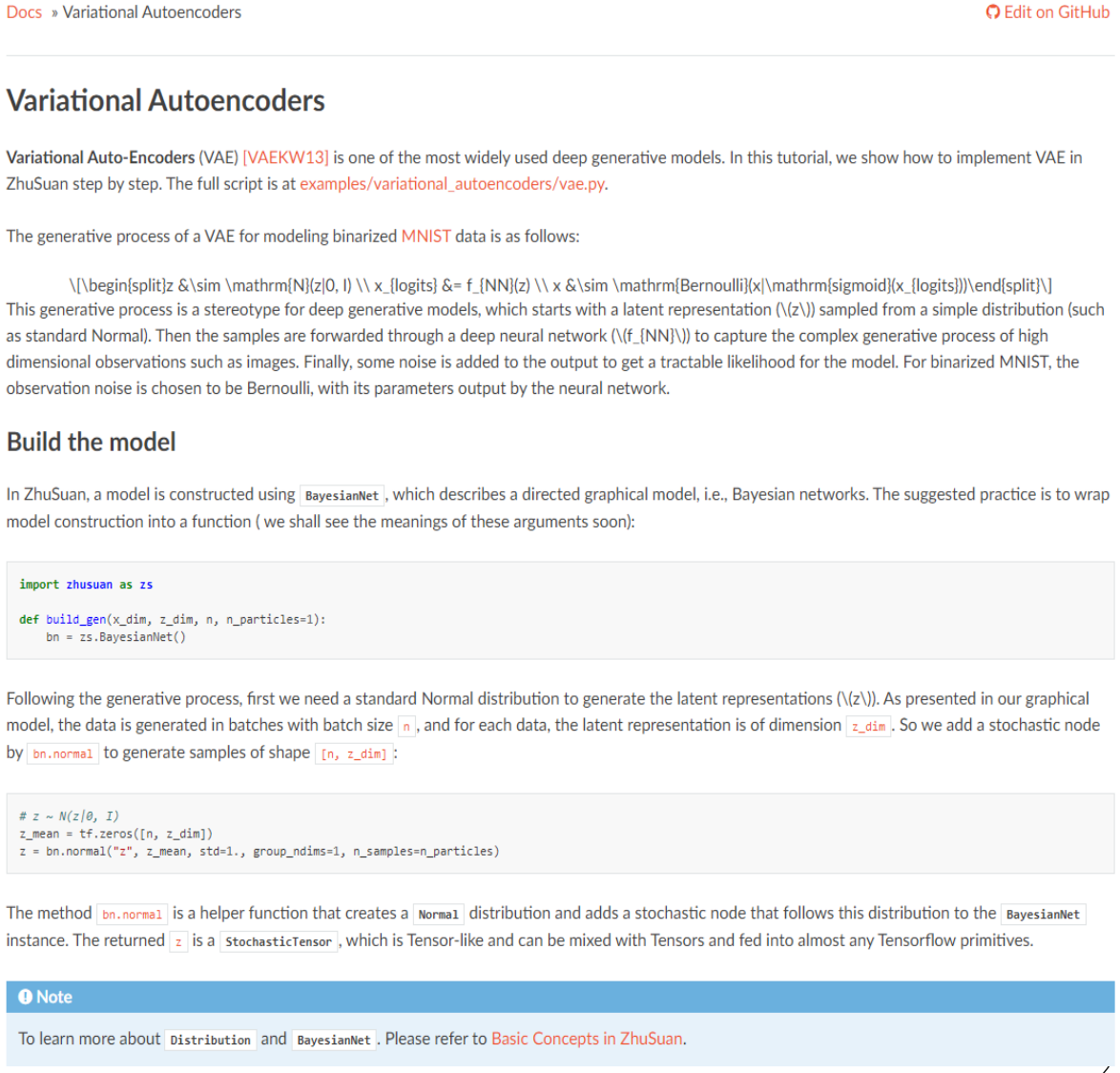
J. Shi, J. Chen, J. Zhu, S. Sun, Y. Luo, Y. Gu, Y. Zhou

arXiv preprint, arXiv:1709.05870 , 2017

Online Documents:

- http://zhusuan.readthedocs.io/





Joint with Huawei MindSpore

mindspore.nn.probability

mindspore.nn.probability.zhusuan

Ease

- Application oriented
- Seamlessly migrated to probabilistic learning
- For beginner users

Model and Toolbox

Probabilistic Programming

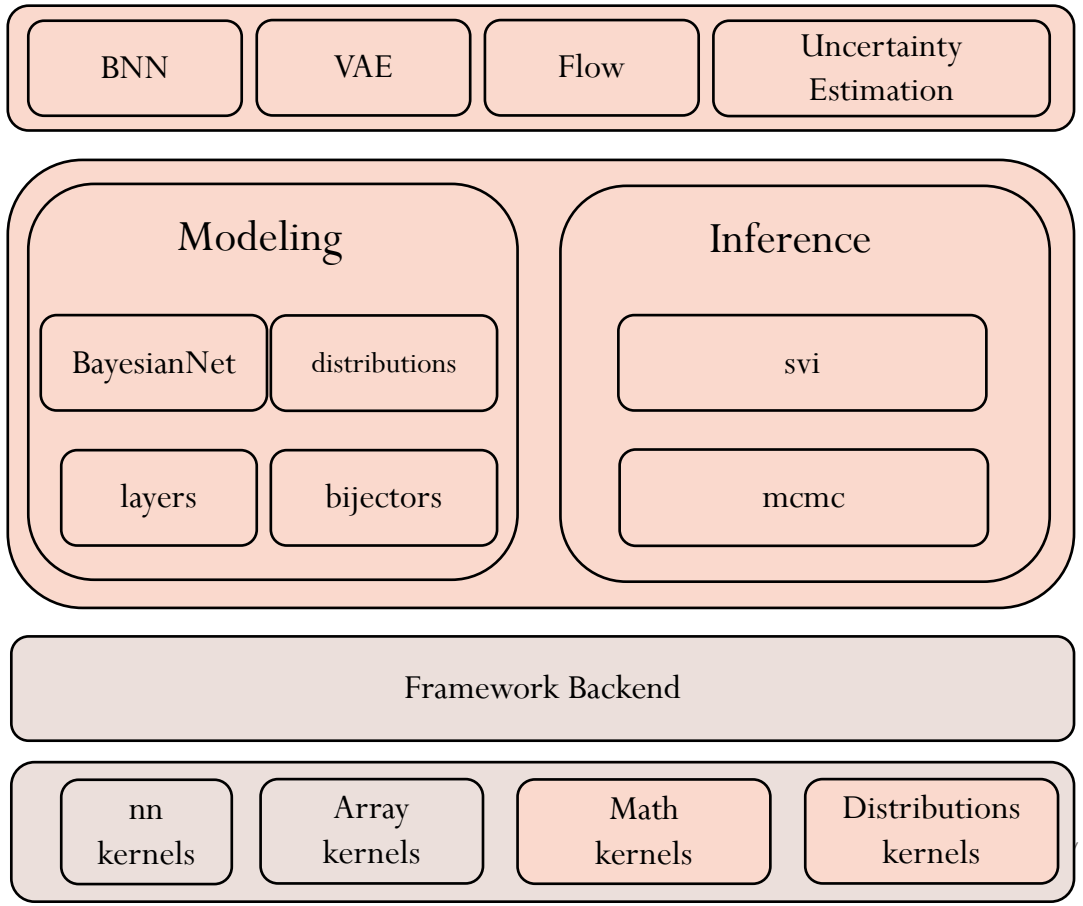
ZHUSUAN

Flexibility

- Fully-functional probabilistic programming
- For advanced users and developers

Framework

MindSpore



Examples

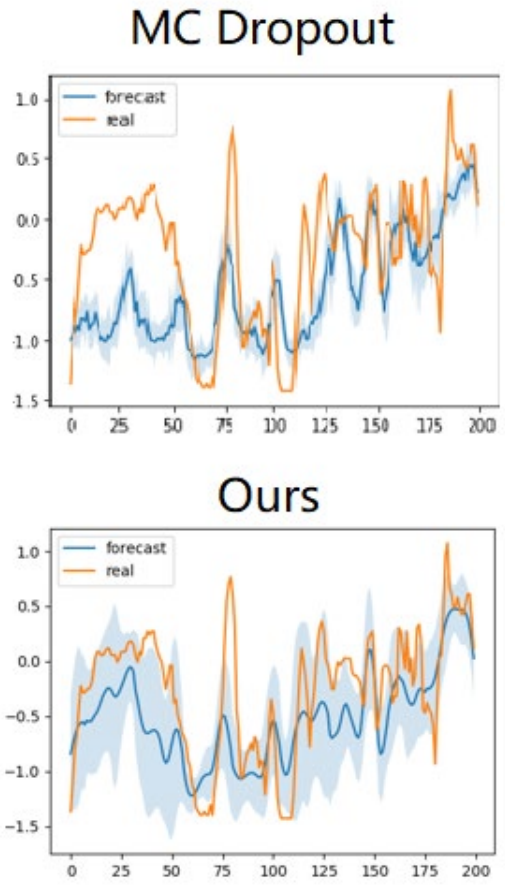
(what problems can be solved by Bayesian deep learning?)

Examples: Air Quality Prediction

- Bayesian inference for a baseline model with LSTM and attention
  - Calculate the uncertainty of prediction
  - Decrease the prediction error of the baseline model

- Function-space particle-based variational inference

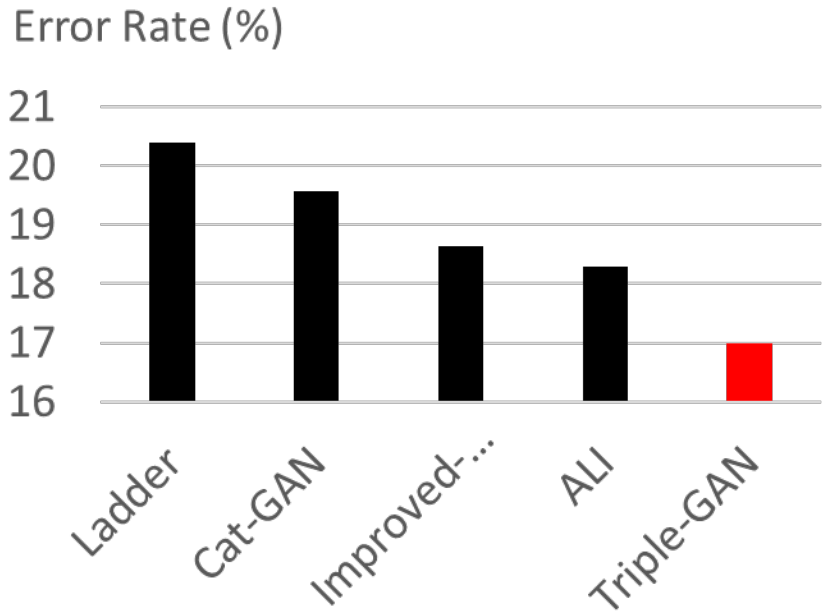
MSE (NO_2)	BNN	Baseline NN
+1 hr	**0.145**	0.160
+7 hr	**0.371**	0.423
+16 hr	**0.389**	0.508



[Function space particle optimization for Bayesian neural networks. Wang et al., ICLR 2019]

Examples: Semi-supervised Learning

Advance previous state-of-the-art results on natural images (CIFAR10) substantially

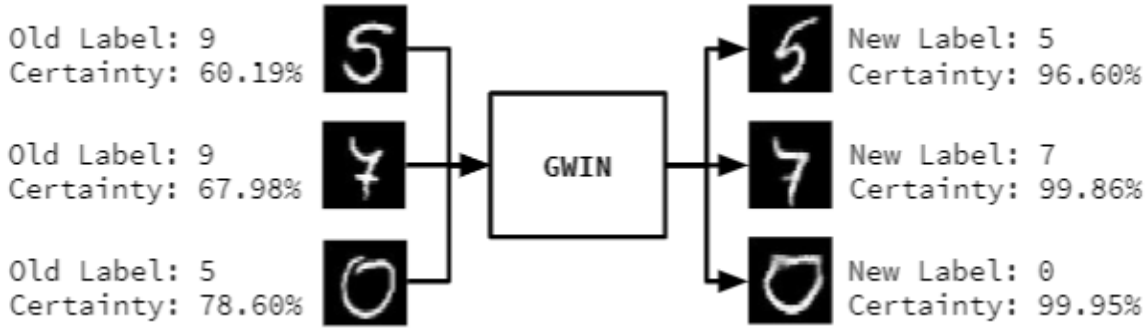


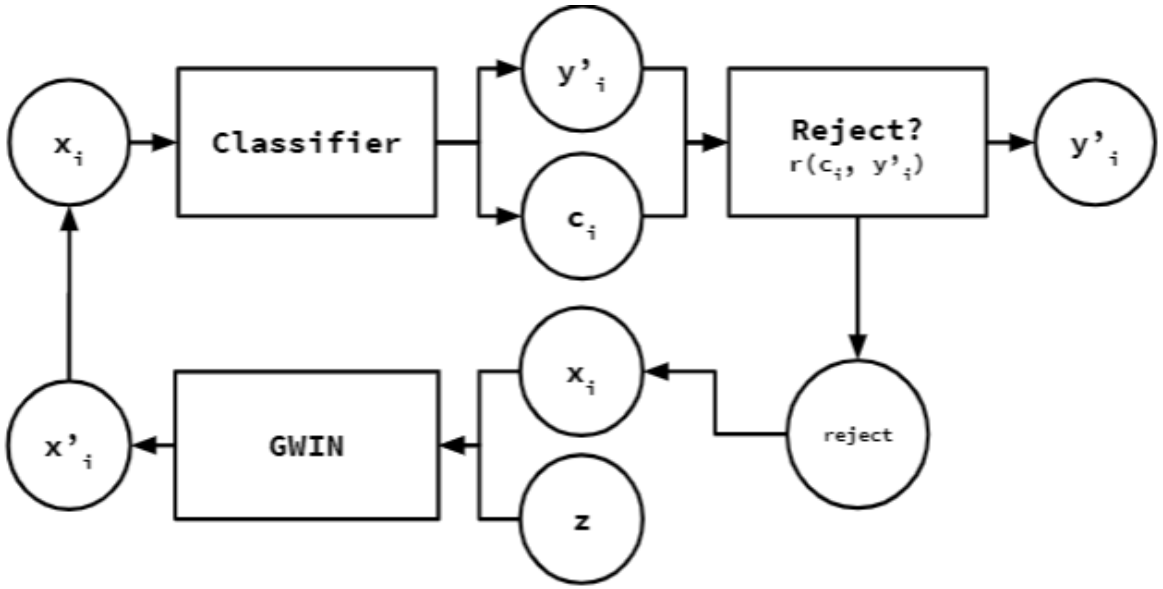
First GAN-based model to generate data in a specific class in SSL

Triple Generative Adversarial Nets (Li et al., NIPS 2017)

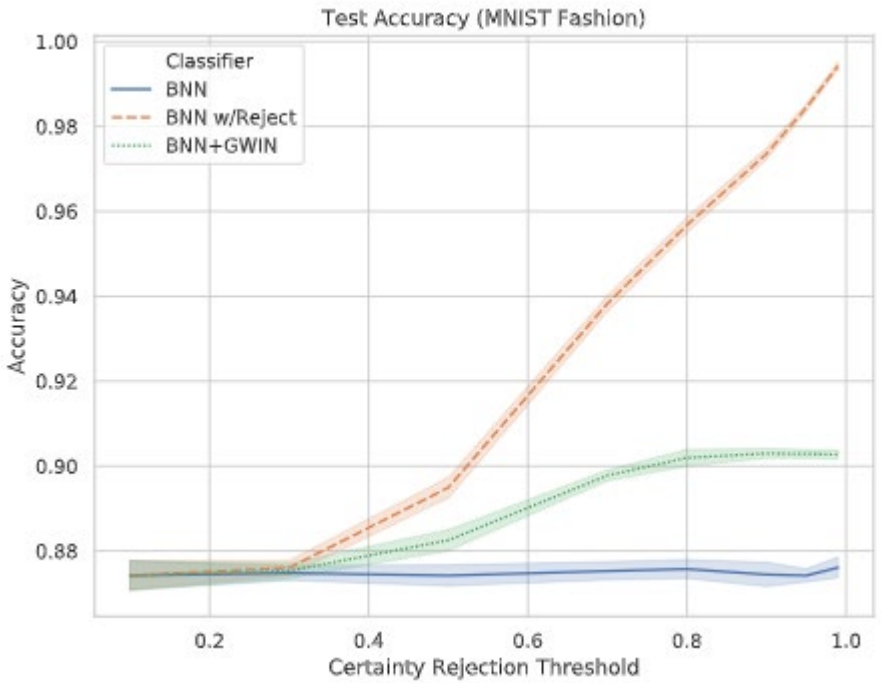
Example: Learning with Rejection

- Leverage deep generative models to transfer "rejected" samples into high-confidence region

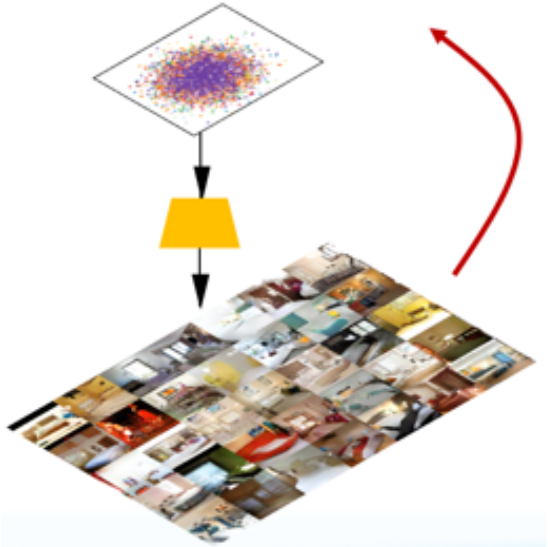




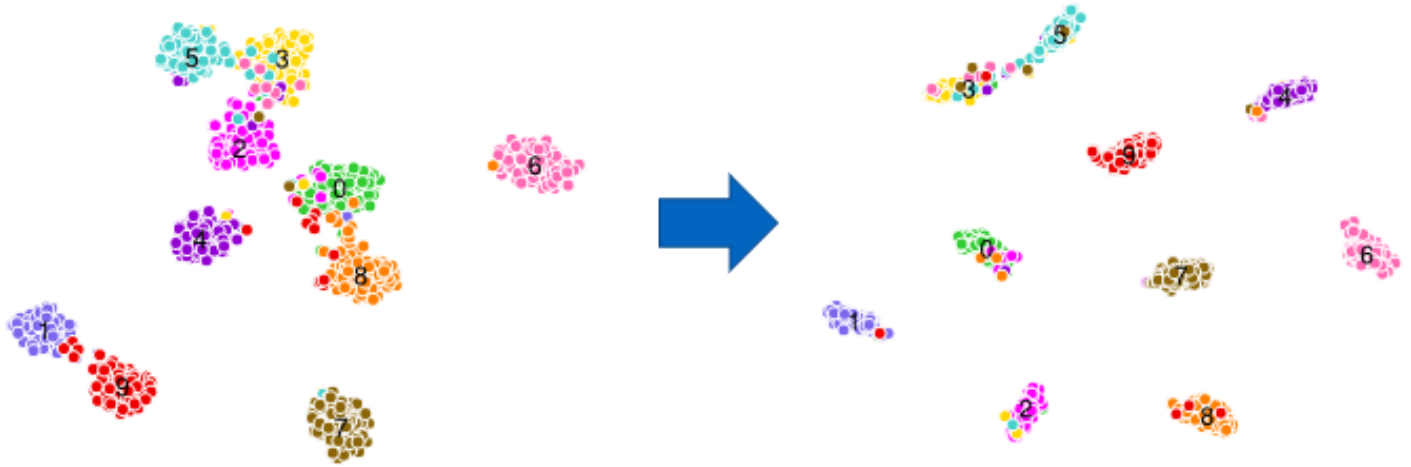
Generative Well-intentioned Networks (GWIN)
(Cosentino & Zhu, NeurIPS 2019)



Examples: Adversarial Robustness



Learn a deep network to map the data into a simple MoG distribution



MNIST CIFAR-10

- Theoretically show optimal robustness
- Algorithmically train MM-LDA network using SGD to minimize cross-entropy loss

No extra cost, as compared to vanilla DNN

Max-Mahalanobis Linear Discriminant Analysis Networks (Pang et al., ICML 2018)

Summary

- Uncertainty Models and Inference is Critical for AI Systems
- Bayesian Machine Learning provides a Powerful Language
- "ZhuSuan" Provides a User-Friendly Library for Deep Probabilistic Programming
- Examples
  - Sequential prediction, semi-supervised learning, few-shot learning
  - Bayesian methods improve adversarial robustness

Thank You!

- J. Zhu, C. Chen, W. Hu, B. Zhang. Big Learning with Bayesian Methods. *National Science Review*, 4(4): 627–651, 2017
- J. Zhu. Probabilistic Machine Learning: Models, Algorithms and a Programming Library. *Proc. of the 27th International Joint Conference on Artificial Intelligence*, Early Career. Pages 5754-5759.