

# Parametric Information Bottleneck to Optimize Stochastic Neural Networks

**Author:** Thanh T. Nguyen

**Supervisor:** Prof. Jaesik Choi

**Thesis Committee:** Prof. Jaesik Choi (SAIL Lab)  
Prof. Se Young Chun (BMITL Lab)  
Prof. Jun Moon (DCSL Lab)

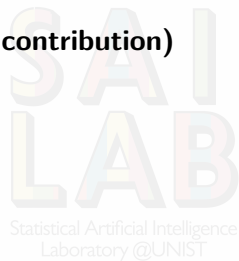
Statistical Artificial Intelligence Lab @ UNIST

Master thesis defense  
2017/11/30

Statistical Artificial Intelligence  
Laboratory @UNIST

# Contents

1. Motivation
2. Thesis statement
3. Information Bottleneck Principle
4. Parametric Information Bottleneck (our contribution)
5. Experiments
6. Conclusion



# Contents

## 1. Motivation

## 2. Thesis statement

## 3. Information Bottleneck Principle

## 4. Parametric Information Bottleneck (our contribution)

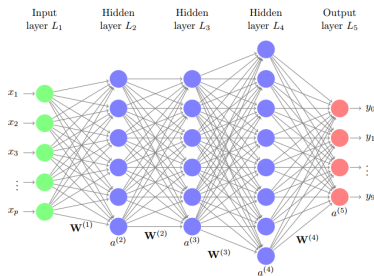
## 5. Experiments

## 6. Conclusion



# Motivation

- ▶ Deep neural networks (DNNs) have demonstrated competitive performance in several learning tasks;



1

- ▶ *Learning principle* is important to *guide the learning* in a model;
- ▶ The Maximum Likelihood Estimate (MLE) principle is widely used as a standard learning principle for DNNs;
- ▶ E.g., squared loss function, cross-entropy loss function.

<sup>1</sup>Figure credit: B.Efron and T.Hastie's

# The MLE principle

*model distribution*  $\xrightarrow{\text{match in KL divergence}}$  *empirical data distribution*,

## MLE

$$\begin{aligned}\theta_{ML} &= \arg \max_{\theta} \mathbb{E}_{\mathbf{x} \sim p_D(\mathbf{x})} [\log p_{\text{model}}(\mathbf{x}; \theta)] \\ &= \arg \min_{\theta} D_{KL} [p_D(\mathbf{x}) \| p_{\text{model}}(\mathbf{x}; \theta)] \\ &\approx \arg \max_{\theta} \sum_{i=1}^N \log p_{\text{model}}(\mathbf{x}_i; \theta)\end{aligned}$$

## So...

- ▶ Does MLE effectively and sufficiently exploit a neural network's representative power?
- ▶ Is there any better alternative?

Statistical Artificial Intelligence  
Laboratory @UNIST

# MLE principle for neural networks?

- ▶ MLE is **generic** for all models, **NOT specifically tailored for neural networks**;
- ▶ **MLE sees neural network as a whole**; hidden layers and neurons are not taken care of *explicitly*;
- ▶ As a result, a neural network with **redundant information** in hidden layers may have a good distribution match in a training set BUT show a poor generalization in a test set

## Occam's Razor

Among competing hypotheses, the one with the simplest description length should be selected!

# Contents

1. Motivation

**2. Thesis statement**

3. Information Bottleneck Principle

4. Parametric Information Bottleneck (our contribution)

5. Experiments

6. Conclusion



# Thesis statement

## Our main thesis (proposition)

To better exploit a neural network's representation requires internal information within hidden layers to be considered explicitly during learning phase.

→ e.g., increasing informativeness and compactness within a representation captures more regularities.



# This dissertation includes

## What we did ...

- ▶ Proposed an approximate information-theoretic learning framework, **Parametric Information Bottleneck (PIB)** framework that is specifically tailored for neural networks;
- ▶ PIB better exploited a neural network's representation by jointly and explicitly considering layer complexity and predictive power of each layer and of entire network

## Outcome

- ▶ An gradient-based algorithm that learns all parameters of a neural network using our PIB framework;
- ▶ PIB has some interesting connections with MLE in neural networks;
- ▶ PIB outperforms MLE in terms of generalization, multi-modal learning, and representation learning.

# Contents

1. Motivation
2. Thesis statement
- 3. Information Bottleneck Principle**
4. Parametric Information Bottleneck (our contribution)
5. Experiments
6. Conclusion



# Information Bottleneck Principle

## [Tishby et al., 1999]

- ▶ A principled way of extracting relevant information in one variable,  $X$  about another variable,  $Y$ .
  - ▶ Knowing some values of  $X$  gives some guidance in prediction of  $Y$ , i.e.,  $X$  provides information about  $Y$  (intrinsically via  $p(X, Y)$ ).

### Example 1

Whether  $X_1$  or  $X_2$  provides “more information” about  $Y$ ? Given

$$\begin{cases} \mathcal{Y} = \mathcal{X}_1 = \mathcal{X}_2 = \{0, 1\} \\ p(y) = p(x_1) = p(x_2) = 0.5 \quad \forall y \in \mathcal{Y}, x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2, \end{cases}$$

$$p(y, x_1) = \begin{cases} 1 & \text{if } y = x_1 = 0 \\ 0 & \text{otherwise,} \end{cases}; p(y, x_2) = \frac{1}{4} \quad \forall y, x_2$$

Intuition  $\rightarrow X_2$  provides **no** information about  $Y$ ;  $X_1$  provides **all** information about  $Y$ .

# Information Bottleneck Principle

## [Tishby et al., 1999]

### IB core idea

Encode  $X$  into an intermediate representation,

$$X \xrightarrow{p(z|x)} Z,$$

in such a way that  $Z$  preserves as much of **relevant information** about  $Y$  as possible.

### Markov assumption

$$Y \rightarrow X \rightarrow Z, \text{ i.e., } Y \perp Z | X$$

Q:

- ▶ How to quantify **relevant information** in  $Z$ ?
- ▶ How to quantify **the representation quality** of  $Z$ ?

A: **mutual information!**

# Mutual Information

## Definition (Mutual Information)

$$I(Z, Y) = \int p(\mathbf{z}, \mathbf{y}) \log \frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})p(\mathbf{y})} d\mathbf{z} d\mathbf{y}$$

## Intuition

$I(Z, Y)$  = the amount of information that  $Z$  contains about  $Y$ .

Back to **Example 1**:

$I(X_1, Y) = 2 = H(X) \rightarrow X_1$  provides ALL information about  $Y$

$I(X_2, Y) = 0 \rightarrow X_2$  contains no information about  $Y$

Statistical Artificial Intelligence  
Laboratory @UNIST

# Information Bottleneck Problem

## Definition

- ▶ **Compression** (measure) = representation complexity =  $I(Z, X)$
- ▶ **Relevance** = predictive power =  $I(Z, Y)$

## IB Problem

= Trade-off between **compression** and **relevance**

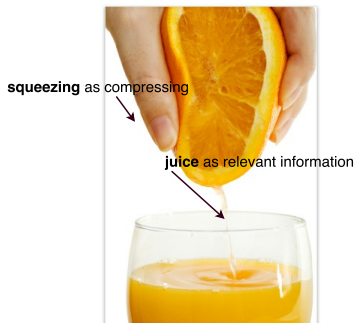
Why compression-relevance tradeoff? → think of this analogy:

- ▶ representation  $Z$  = some container (e.g., orange)
- ▶  $I(Z, X)$  = the container's capacity (e.g., the orange's volume)
- ▶  $I(Z, Y)$  = amount of useful items in the container (e.g., orange juice)

Statistical Artificial Intelligence  
Laboratory @UNIST

# Compression-Relevance tradeoff

## Orange's Analogy:



- ▶ Reducing capacity may cause loss of relevant information
- ▶ Larger capacity gives room for more relevant information

Rigorously,

**Extreme 1: Maximum Compression  $\rightarrow$  Minimum Relevance**

$$I(Z, X) = 0 \implies Z \perp X \implies Z \perp Y \implies I(Z, Y) = 0$$

**Extreme 2: Minimum Compression  $\rightarrow$  Maximum Relevance**

$$I(Z, X) = H(X) \implies Z = f(X) \implies I(Z, Y) = I(X, Y)$$

where  $f(\cdot)$  is a bijective (reversible) function

**Proofs:** It follows the Markov chain assumption, rules of probability, and definitions of mutual information.

LAB  
Statistical Artificial Intelligence  
Laboratory @UNIST

# IB Optimization Problem

## Compression-Relevance Function [Slonim and Weiss, 2002]

$$R(D) := \min_{Z=Z(X): I(Z, Y) \geq D} I(Z, X),$$

- Equivalent to the Lagrangian multiplier (to be minimized):

## Lagrangian multiplier [Tishby et al., 1999]

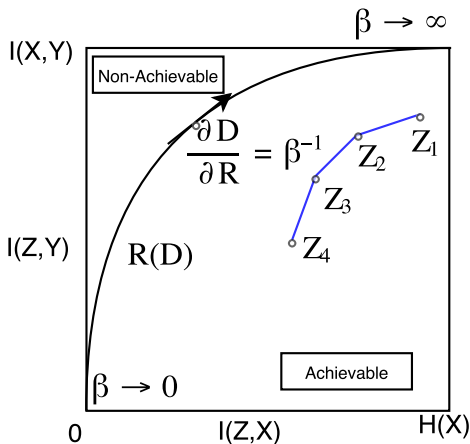
$$\mathcal{L}_{IB} = I(Z, X) - \beta I(Z, Y)$$

where  $\beta$  is a positive Lagrange multiplier that determines the trade-off between compression and relevance in  $Z$

→ is in general **NP hard!**



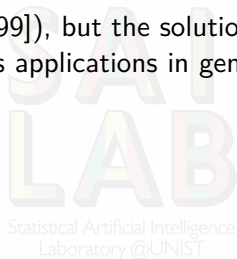
# Compression-Relevance tradeoff in action



**Figure:** The information curve  $R(D)$  is a non-decreasing concave curve that divides the information plane into achievable region and non-achievable region.

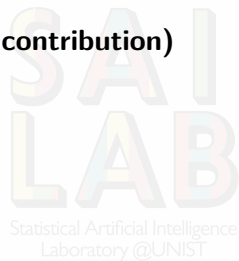
# About the IB principle

- ▶ No model assumption is required
- ▶ Assumption:  $Y \rightarrow X \rightarrow Z$  forms a Markov chain, and  $p(X, Y)$  is known (though one can simply use empirical joint distribution  $\hat{p}(X, Y)$  in practice)
- ▶ Exact solution is known ([Tishby et al., 1999]), but the solution is implicit and highly non-linear that limits its applications in general cases  $\rightarrow$  NP-hard in general!



# Contents

1. Motivation
2. Thesis statement
3. Information Bottleneck Principle
- 4. Parametric Information Bottleneck (our contribution)**
5. Experiments
6. Conclusion



# Parametric Information Bottleneck (PIB)

[Nguyen and Choi, 2017]

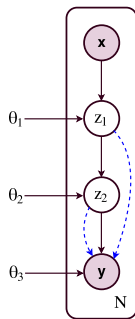
## What is this?

PIB = neural network architecture + IB approximations,

- ▶ A learning framework to **better exploit the neural network's intermediate representation**
- ▶ Learning a neural network's representation is now interpreted as **jointly** and **explicitly inducing compression and relevance into every level of neural networks**

# Our perspective: Neural network as sequential encodings

Neural network architecture = **series of stochastic encoders** that **sequentially modify the original data space**



- ▶  $Z_{l-1} \xrightarrow{p(z_l|z_{l-1})} Z_l$  where  $Z_l$  is the  $l^{th}$  layer <sup>2</sup>
- ▶ **Layer transform**  $p(z_{l+1}|z_l)$ : induces a **soft partitioning** of the space,  $\mathcal{Z}_l$ , into a new encoding space,  $\mathcal{Z}_{l+1}$
- ▶ Markov Assumption:

$$Y \rightarrow X \rightarrow Z_l \rightarrow Z_{l+1}$$

Statistical Artificial Intelligence  
Laboratory @UNIST

# Why is this perspective useful?

## 1. Property 1:

Enables easy sampling of  $Z_I$  given  $X$ .

→ useful for estimating mutual information in the later parts.

## 2. Property 2:

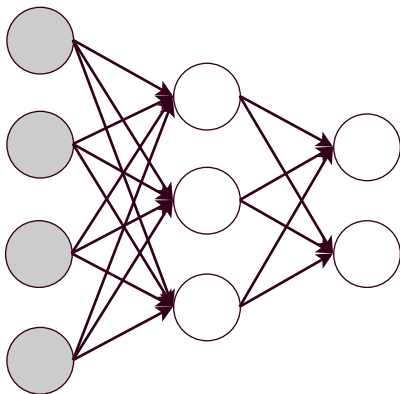
Resembles Bayesian Nets (Belief Nets)

## 3. Property 3:

Makes a **lossy representation** (due to Data-Processing Inequality):

$$\begin{aligned} H(X) &\geq I(X, Z_I) \geq I(X, Z_{I+1}) \\ I(X, Y) &\geq I(Y, Z_I) \geq I(Y, Z_{I+1}) \end{aligned}$$

# Resembling Bayesian Nets



- Represents conditional independences of random variables
- Bayesian Nets are well studied

## Property 3: Lossy is useful

- ▶ The original data space,  $\mathcal{X}$  is continuous  $\rightarrow$  requires infinite precision to represent it precisely
- ▶ **preciseness  $\neq$  usefulness!**
- ▶ Let it (representation) be lossy, but in a beneficial manner: **lossy in irrelevant information and gain in relevant information**

Q:

How to *mathematically* “implement” (quantify) this idea?

A: Borrow the idea of the IB principle!



# Parametric Information Bottleneck

## PIB objective

$$\mathcal{L}_{PIB}(Z) := \mathcal{L}_{PIB}(\theta) := \sum_{l=0}^L \left[ \beta_l^{-1} I(Z_l, Z_{l-1}) - I(Z_l, Y) \right]$$

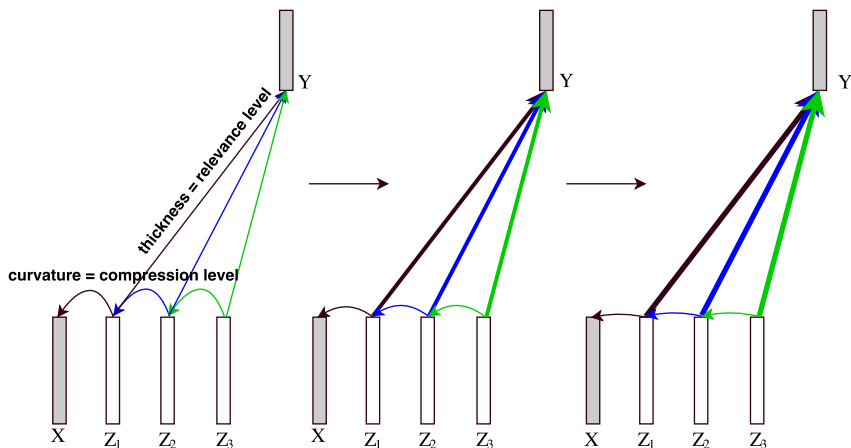
where the layer-specific Lagrangian multiplier  $\beta_l^{-1}$  controls the tradeoff between relevance and compression in each bottleneck.

- ▶  $\mathcal{L}_{PIB}$  = a **joint** version of the theoretical analysis in [Tishby and Zaslavsky, 2015]
- ▶  $\mathcal{L}_{PIB}$  **encourages compression and relevance simultaneously within each layer and within the entire network as a whole.**

Statistical Artificial Intelligence  
Laboratory @UNIST

# PIB Objective as Tightening the information knots

Think of minimizing the PIB objective as **tightening the information knots** of a network architecture:



# Solving for $\min J_{PIB}(Z)$ ?

- ▶ Much harder than the original IB problem since the information terms are inter-dependent
- ▶ Still inherits the intractability of mutual information



# Intractability of Mutual Information

- **Relevance:**

$$I(Z, Y) = H(Y) - H(Y|Z)$$

$$H(Y|Z) = - \int p(\mathbf{y}, \mathbf{z}) \log p_{\text{true}}(\mathbf{y}|\mathbf{z}) d\mathbf{y} d\mathbf{z}$$

Relevance decoder (def):

$$p_{\text{true}}(\mathbf{y}|\mathbf{z}_I) = \int p_D(\mathbf{x}, \mathbf{y}) \frac{p(\mathbf{z}_I|\mathbf{x})}{p(\mathbf{z}_I)} d\mathbf{x}$$

→ intractable!

- **Compression:**

$$I(Z_I, Z_{I-1}) = H(Z_I) - H(Z_I|Z_{I-1})$$

$$H(Z_I) = - \int p(\mathbf{z}_I) \log p(\mathbf{z}_I) d\mathbf{z}_I$$

$$p(\mathbf{z}_I) = \mathbb{E}_{p(\mathbf{z}_{I-1})}[p(\mathbf{z}_I|\mathbf{z}_{I-1})]$$

→ remains exponentially in terms of  $|\mathcal{Z}_{I-1}|$

# Approximate Relevance

- ▶ Use **variational approximation** to derive a lower bound on  $I(Z_I, Y)$  (as in [Alemi et al., 2016])
- ▶ Variational approximation = to posit a family of distributions and then find a member of that family that is closest to the intractable distribution in terms of KL divergence
- ▶ However,

here we propose to **re-use the network architecture** to define variational distributions

# Variational Approximation to Mutual Information

$$\begin{aligned} H(Y|Z_I) &= - \int p(\mathbf{z}_I) p_{\text{true}}(\mathbf{y}|\mathbf{z}_I) \log \boxed{p_{\text{true}}(\mathbf{y}|\mathbf{z}_I)} d\mathbf{y} d\mathbf{z}_I \\ &\quad \text{relevance decoder} \downarrow \\ &= - \int p(\mathbf{z}_I) p_{\text{true}}(\mathbf{y}|\mathbf{z}_I) \log \boxed{p_v(\mathbf{y}|\mathbf{z}_I)} d\mathbf{y} d\mathbf{z}_I - \int p(\mathbf{z}_I) D_{\text{KL}}[p_{\text{true}}(\mathbf{y}|\mathbf{z}_I) || p_v(\mathbf{y}|\mathbf{z}_I)] d\mathbf{z}_I \\ &\quad \text{variational relevance decoder} \downarrow \\ &\leq - \int p(\mathbf{z}_I) p_{\text{true}}(\mathbf{y}|\mathbf{z}_I) \log p_v(\mathbf{y}|\mathbf{z}_I) d\mathbf{y} d\mathbf{z}_I \\ &= - \int p(\mathbf{y}, \mathbf{z}_I) \log p_v(\mathbf{y}|\mathbf{z}_I) d\mathbf{y} d\mathbf{z}_I \\ &= - \int p(\mathbf{x}, \mathbf{y}, \mathbf{z}_I) \log p_v(\mathbf{y}|\mathbf{z}_I) d\mathbf{y} d\mathbf{z}_I d\mathbf{x} \\ &= - \int p_D(\mathbf{x}, \mathbf{y}) p(\mathbf{z}_I|\mathbf{x}, \mathbf{y}) \log p_v(\mathbf{y}|\mathbf{z}_I) d\mathbf{y} d\mathbf{z}_I d\mathbf{x} \\ &= - \int p_D(\mathbf{x}, \mathbf{y}) p(\mathbf{z}_I|\mathbf{x}) \log p_v(\mathbf{y}|\mathbf{z}_I) d\mathbf{z}_I d\mathbf{x} d\mathbf{y} \\ &= - \mathbb{E}_{p_D(\mathbf{x}, \mathbf{y})} [\mathbb{E}_{p(\mathbf{z}_I|\mathbf{x})} [\log p_v(\mathbf{y}|\mathbf{z}_I)]] =: \tilde{H}(Y|Z_I) \end{aligned}$$

# Approximate Relevance

- ▶ Re-use the higher-level part of the existing network architecture at each layer to define the *variational relevance decoder* for that layer, i.e.,

$$p_v(\mathbf{y}|\mathbf{z}_I) = p(\mathbf{y}|\mathbf{z}_I)$$

where  $p(\mathbf{y}|\mathbf{z}_I)$  is determined by the network architecture

- ▶ Then,

$$p_v(\mathbf{y}|\mathbf{z}_I) = \int \prod_{i=I}^{L+1} p(\mathbf{z}_{i+1}|\mathbf{z}_i) d\mathbf{z}_L \dots d\mathbf{z}_{I+1} = \mathbb{E}_{p(\mathbf{z}_L|\mathbf{z}_I)} [p(\mathbf{y}|\mathbf{z}_L)]$$

## Variational Conditional Relevance (VCR) (def.)

$$\tilde{H}(Y|Z_I) = -\mathbb{E}_{p_D(\mathbf{x}, \mathbf{y})} \left[ \mathbb{E}_{p(\mathbf{z}_I|\mathbf{x})} \left[ \log \mathbb{E}_{p(\mathbf{z}_L|\mathbf{z}_I)} [p(\mathbf{y}|\mathbf{z}_L)] \right] \right]$$

→ Use Monte Carlo sampling to estimate VCR because sampling is very easy in Bayesian Nets!

# VCR vs. MLE

## Proposition 1

$\tilde{H}(Y|Z_{I=0})$  = negative log-likelihood (NLL) function

## Proposition 2

VCR for the *compositional* bottleneck  $Z = (Z_1, Z_2, \dots, Z_L)$  places an upper bound on the NLL function, i.e.,

$$\begin{array}{ccc} \text{VCR for } Z_L & & \text{VCR for } Z \\ \downarrow & & \downarrow \\ \boxed{\tilde{H}(Y|Z_L)} & = & \boxed{\tilde{H}(Y|Z)} \geq -\mathbb{E}_{p_D(\mathbf{x}, \mathbf{y})} [\log p(\mathbf{y}|\mathbf{x})] \end{array}$$

**Proof of Prop. 1:** a direct result of VCR.

**Proof of Prop. 2:** It follows the Markov assumption and Jensen's inequality



# VCR vs. MLE: Interpretation

- ▶ VCR for the entire network (i.e.,  $l = 0$ ) = NLL of  $p(\mathbf{y}|\mathbf{x})$
- ▶ VCR for layer  $l$  = NLL of  $p(\mathbf{y}|\mathbf{z}_l)$



# Approximate Compression

- Decomposition:  $I(Z_1, Z_0) = H(Z_1) - H(Z_1|Z_0)$
- Conditional entropy:

$$H(Z_1|Z_0) = \mathbb{E}_{p(\mathbf{z}_0)} \left[ \sum_{i=1}^{N_1} H(Z_{1,i}|Z_0 = \mathbf{z}_0) \right]$$

- Resort to empirical samples of  $\mathbf{z}_1$  generated by Monte Carlo sampling to estimate the entropy:

$$H(Z_1) = -\mathbb{E}_{p(\mathbf{z}_1)}[\log p(\mathbf{z}_1)] \approx -\frac{1}{M} \sum_{k=1}^M \log p(\mathbf{z}_1^{(k)}) =: \hat{H}_{MLE}(Z_1)$$

$$\mathbf{z}_1^{(k)} \sim p(\mathbf{z}_1) = \mathbb{E}_{p(\mathbf{z}_0)}[p(\mathbf{z}_1|\mathbf{z}_0)]$$

→ However,  $\log p(\mathbf{z}_1)$  is numerically unstable since  $p(\mathbf{z}_1) \approx 0$  for high-dimensional  $\mathbf{z}_1$ !

# Approximate Compression

Numerically-stable estimation of entropy:

Jensen's inequality

$$H(Z_1) \overset{\downarrow}{\leq} -\mathbb{E}_{p(\mathbf{z}_1)} \left[ \mathbb{E}_{p(\mathbf{z}_0)} [\log p(\mathbf{z}_1|\mathbf{z}_0)] \right] := \tilde{H}(Z_1)$$

→ more stable since  $p(\mathbf{z}_1|\mathbf{z}_0)$  factorizes!

# Monte Carlo Sampling

$$s = \int p(x)f(x)dx = \mathbb{E}_{p(x)}[f(x)] \approx \frac{1}{N} \sum_{i=1}^N f(x^{(i)}) =: \hat{s}_N$$

where  $x^{(i)} \sim p(x)$ .

- ▶ Unbiased mean:  $\mathbb{E}[\hat{s}_N] = s$
- ▶ Variance:  $\text{Var}[\hat{s}_N] = \frac{\text{Var}[f(x)]}{N}$



# Application to Network Architecture Domain

We applied our PIB framework to feed-forward neural networks with binary bottlenecks,

- ▶  $Z_l \in \{0, 1\}^{|Z_l|}$
- ▶ stochastic encoder  $p(\mathbf{z}_{l+1}|\mathbf{z}_l)$  is simply modeled by network weights and **sigmoid** function for **binary**  $\mathbf{z}_{l+1}$
- ▶  $H(Z_{1:l}|\mathbf{Z}_0 = \mathbf{z}_0)$  can be computed analytically

**Q:**

Derivative wrt discrete-valued variables is impossible. How to compute gradients via sampling of binary variables?

**A:**

Use Raiko gradient estimator!

# Raiko gradient estimator [Raiko et al., 2014]

**Idea:** Decompose the binary variable into the sum of a deterministic term and a noise term (to be ignored when taking derivatives)

$$\mathbf{z}_l = (z_{l,1}, z_{l,2}, \dots, z_{l,k}),$$

$$\mathbf{a}^{(l)} = (a_1^{(l)}, a_2^{(l)}, \dots, a_k^{(l)}) = W^{(l)} \mathbf{z}_{l-1} + \mathbf{b}_i^{(l)}$$

$$p(z_{l,i} = 1 | \mathbf{z}_{l-1}) = \sigma(a_i^{(l)})$$

Rewrite  $z_i^{(l)}$  as if it is continuous:  $z_{l,i} = \sigma(a_i^{(l)}) + \epsilon_i^{(l)}$

where

$$\epsilon_i^{(l)} = \begin{cases} 1 - \sigma(a_i^{(l)}) & \text{with probability } \sigma(a_i^{(l)}) \\ -\sigma(a_i^{(l)}) & \text{with probability } 1 - \sigma(a_i^{(l)}) \end{cases}$$

Statistical Artificial Intelligence  
Laboratory @UNIST

# Gradient-based Algorithm: GRAD-PIB

---

**Algorithm 1** Minibatch version of training PIB, we use  $M = 16$  for training (and  $M = 32$  for testing).

---

- 1: **procedure** GRAD-PIB
  - 2: **Input:** Labeled training dataset  $S_D$
  - 3:  $\theta \leftarrow$  Initialize parameters
  - 4: *repeat*:
  - 5:    $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N \leftarrow$  Random minibatch of  $N$  samples drawn from  $S_D$
  - 6:   Generate  $M$  samples of  $\mathbf{z}_i$  per each sample of  $\mathbf{z}_{i-1}$  for  $1 \leq i \leq L$
  - 7:   Use the generated samples above and the Approximate Relevance and Approximate Compression to approximate  $\tilde{\mathcal{L}}_{PIB}(\theta)$
  - 8:    $\mathbf{g} \leftarrow \frac{\partial}{\partial \theta} \tilde{\mathcal{L}}_{PIB}(\theta)$  using Raiko estimator
  - 9:    $\theta \leftarrow$  Update parameters using the approximate gradients  $\mathbf{g}$  and SGD
  - 10: until convergence of parameters  $\theta$
  - 11: **Output:**  $\theta$
  - 12: **end procedure**
- 

→ **Main advantage:** only one pass of (hierarchical ancestral) sampling for computation of all information terms in  $J_{PIB}$

# Contents

1. Motivation
2. Thesis statement
3. Information Bottleneck Principle
4. Parametric Information Bottleneck (our contribution)
- 5. Experiments**
6. Conclusion





# Experiment Setup

- ▶ Used the same neural network architecture for PIB and MLE;
- ▶ The architecture guided by the MLE is named as Stochastic Feed-forward Neural Networks (SFNN) (e.g., [Tang and Salakhutdinov, 2013])
- ▶ Three tasks in the MNIST dataset [LeCun et al., 1998]:
  - ▶ **Image classification**: to evaluate generalization,
  - ▶ **Odd-even decision problem**: to visualize the network's representation in terms of mutual information,
  - ▶ **Multi-modal learning**: to evaluate capability to learn a multi-modal distribution.

# Exp. 1: Classification

- **Task:** To classify digit images of the MNIST dataset into 10 categories
- **Input:** A 28x28 gray (digit) image
- **Output:** One of 10 labels
- **Architecture:**  $784 \times 512 \times 512 \times 10$

We compared PIB with 4 other models (A), (B), (C), (D) and (E):

	Model	Mean (%)	Std dev.
deterministic <sup>3</sup>	deterministic (A)	1.73	-
	SFNN as deterministic (B)	1.88	-
	PIB as deterministic (C)	<b>1.46</b>	-
stochastic <sup>4</sup>	deterministic as stochastic (D)	2.30	0.07
	SFNN (E)	1.94	0.036
	PIB ( $\beta^{-1} = 10^{-4}$ )	<b>1.47</b>	<b>0.034</b>

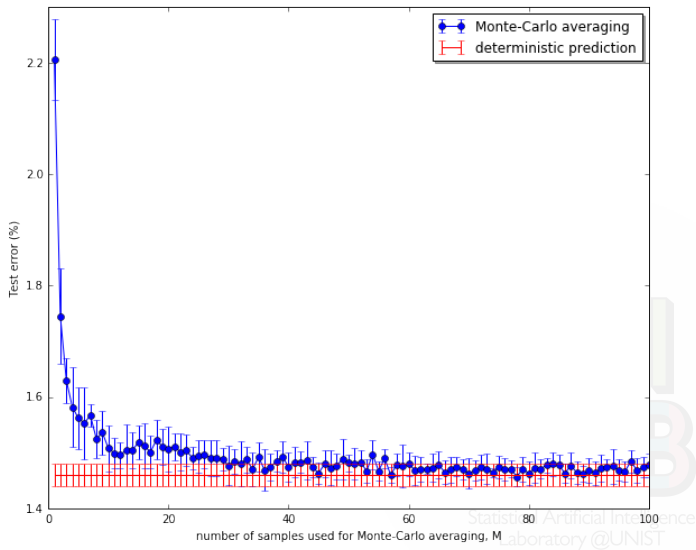
Statistical Artificial Intelligence  
Laboratory@UNIST

---

<sup>3</sup>hidden layers as **deterministic** during test time

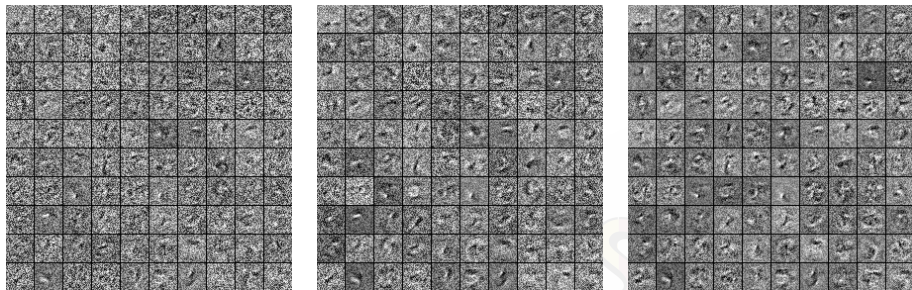
<sup>4</sup>hidden layers as **stochastic** during test time

# Classification: stochastic vs deterministic prediction



**Figure:** A comparison of Monte-Carlo averaging and deterministic prediction of PIB.

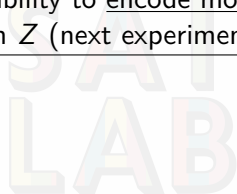
# Classification: Learned features



**Figure:** The learned weights of the first layer in MNIST classification for various models: deterministic neural networks (left), SFNN (middle), and PIB (right).

# Classification: To conclude

- ▶ PIB outperforms MLE by large margin (for the considered architecture, 1.47% vs. 1.94%)
- ▶ The generalization of PIB is attributed to, of course, compression (as a regularization), and its ability to encode more relevant information into the representation  $Z$  (next experiment).



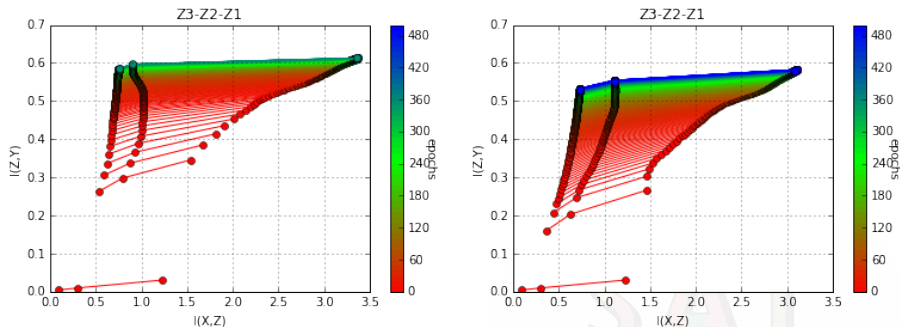
Statistical Artificial Intelligence  
Laboratory @UNIST

## Exp. 2: Learning dynamics

- ▶ **Task:** To determine if the digit in an MNIST image is odd or even.
- ▶ **Input:** A  $28 \times 28$  gray (digit) MNIST image
- ▶ **Output:** 0/1 (i.e., even/odd)
- ▶ **Architecture:**  $784 \times 10 \times 10 \times 10 \times 1$
- ▶ **Purpose:** To visualize a neural network's representation

→ Used a small network architecture so that we can compute  $I(Z_I, X)$  and  $I(Z_I, Y)$  precisely, and plot them on the *information plane* over training epochs

# Learning dynamics



**Figure:** The learning dynamic of PIB (left) and SFNN (right) in a decision problem are presented in the information plane. Each point represents a hidden layer while the color indicates epochs. Because of the Markov assumption, we have  $H(X) \geq I(Z_i, X) \geq I(Z_{i+1}, X)$  and  $I(X, Y) \geq I(Z_i, Y) \geq I(Z_{i+1}, Y)$ .

Laboratory@UNIST

# Learning dynamics: To conclude

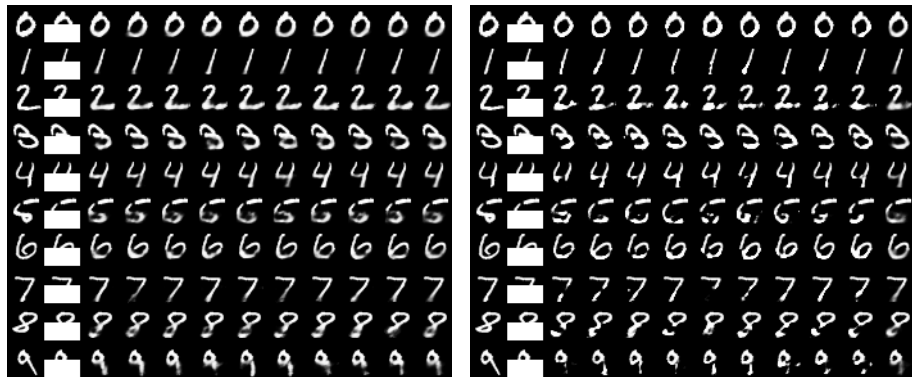
- ▶ Both PIB and SFNN enable the network to gradually encode more information into their hidden layers at the beginning as  $I(Z_i, X)$  increases
- ▶ The encoded information at the beginning also contains some relevant information for the target variable as  $I(Z_i, Y)$  increases as well.
- ▶ However, information encoding in the PIB is more selective as it quickly encodes more relevant information (it reaches higher  $I(Z, Y)$  but in lesser number of epochs) while keeps the layers concise at higher epochs



## Exp. 3: Multi-modal learning

- ▶ **Task:** To predict the lower haft of a digit image given the upper haft
- ▶ **Input:** The upper half of a digit image
- ▶ **Output:** The lower half of a digit image
- ▶ **Architecture:**  $392 \times 512 \times 512 \times 392$
- ▶ **Purpose:** To evaluate the ability of learning multi-modal distribution (i.e., 1-to-many mapping, e.g., a mixture of Gaussian, mixture of Bernoulli)

# Multi-modal learning



**Figure:** Samples drawn from the prediction of the lower half of the MNIST test data digits based on the upper half for PIB (left, after 300 epochs) and SFNN (right, after 1000 epochs). The leftmost column is the original MNIST test digit followed by the masked out digits and nine samples. The rightmost column is obtained by averaging over all generated samples of bottlenecks drawn from the prediction. The figures illustrate the capability of modeling structured output space using PIB and SFNN.

# Multi-modal learning: To conclude

- ▶ PIB can model multi-modal distributions better than MLE (e.g., the generated digits in PIB are more complete and recognizable)
- ▶ Multi-modal learning requires information of different modes from all layers:

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}, \mathbf{z}|\mathbf{x}) d\mathbf{z} = \int p(\mathbf{y}|\mathbf{z})p(\mathbf{z}|\mathbf{x}) d\mathbf{z} = \int \prod_{l=1}^{L+1} p(\mathbf{z}_l|\mathbf{z}_{l-1}) d\mathbf{z}$$

while PIB assures **selective encodings** occur at all levels.

- ▶ PIB **exploits** the neural network's **representation** to an extent that MLE cannot

# Contents

1. Motivation
2. Thesis statement
3. Information Bottleneck Principle
4. Parametric Information Bottleneck (our contribution)
5. Experiments
- 6. Conclusion**



# Summary

- ▶ provided arguments about inefficiency of MLE for learning neural networks and encouraged a re-thinking of a new learning principle that is specifically tailored for neural networks;
- ▶ proposed Parametric Information Bottleneck (PIB) framework, to better exploit a neural network's representation by jointly and explicitly considering representation complexity and predictive power for every layer;
- ▶ PIB is followed by GRAD-PIB, the first algorithm that learns all parameters using Information Bottleneck principle;
- ▶ supported the effectiveness and robustness of our PIB with the qualitative empirical results.

# Why did we do all these?

- ▶ It is ALL about **information**:

information in machine learning = atoms in Chemistry

Learning and inference can be interpreted as acts of modeling (capturing), modifying it, and transferring it from one form to another.

- ▶ Understanding the **information flow** within a model gives us an idea of what's going on inside the model and how should we **modify/transfer the information**.
- ▶ DNNs are very good at modeling high-dimensional distribution space but they are notorious for being a black-box model and “not safe” (cannot capture *uncertainty*).
- ▶ We aim at ambitious goals:
  - ▶ To quantify the contribution of each neuron and each hidden layer within an architecture,
  - ▶ To understand and capture *uncertainty* in neural networks

# Limitations

- ▶ Hierarchical ancestral sampling in *GRAD-PIB* requires an exponential number of samples as the number of hidden layers grow, which currently causes computational burden in large neural network architectures;
- ▶ Since our algorithm is of gradient-based learning, it inherits the weakness of gradient-based learning which fails to guarantee the theoretical learning bound and underestimate the variance of the underlying data distribution; this property makes it difficult to analyze neural network architectures;
- ▶ Here only fully-connected feed-forward architecture with binary hidden layers are considered and larger neural network architecture is not yet exploited.

# Acknowledgement(s)

Prof. Jaesik Choi




Thank you for listening!


Q & A Section

SAI  
LAB  
Statistical Artificial Intelligence  
Laboratory @UNIST



# References

-  Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2016).  
Deep variational information bottleneck.  
*CoRR*, abs/1612.00410.
-  LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).  
Gradient-based learning applied to document recognition.  
*Proceedings of the IEEE*, 86(11):2278–2324.
-  Nguyen, T. T. and Choi, J. (2017).  
Parametric information bottlenecks to optimize stochastic neural networks.

-  Raiko, T., Berglund, M., Alain, G., and Dinh, L. (2014).

Techniques for learning binary stochastic feedforward neural networks.

*CoRR*, abs/1406.2989.

-  Slonim, N. and Weiss, Y. (2002).

Maximum likelihood and the information bottleneck.

In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 335–342.

-  Tang, Y. and Salakhutdinov, R. (2013).

Learning stochastic feedforward neural networks.

*In Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pages 530–538.*



Tishby, N., Pereira, F. C., and Bialek, W. (1999).

The information bottleneck method.

*In Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing.*



Tishby, N. and Zaslavsky, N. (2015).

Deep learning and the information bottleneck principle.

pages 1–5.

SAI  
LAB  
Statistical Artificial Intelligence  
Laboratory @UNIST