



### 저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

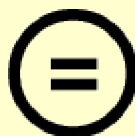
다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.




변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#) 

Master's Thesis

PARAMETRIC INFORMATION BOTTLENECK TO  
OPTIMIZE STOCHASTIC NEURAL NETWORKS

Thanh Nguyen Tang

Department of Computer Science and Engineering

Graduate School of UNIST

2018

# PARAMETRIC INFORMATION BOTTLENECK TO OPTIMIZE STOCHASTIC NEURAL NETWORKS

Thanh Nguyen Tang

Department of Computer Science and Engineering

Graduate School of UNIST

# Parametric Information Bottleneck to Optimize Stochastic Neural Networks

A thesis  
submitted to the Graduate School of UNIST  
in partial fulfillment of the  
requirements for the degree of  
Master of Science

Thanh Nguyen Tang

11. 30. 2017

Approved by

---

Advisor  
Jaesik Choi

# Parametric Information Bottleneck to Optimize Stochastic Neural Networks

Thanh Nguyen Tang

This certifies that the thesis of Thanh Nguyen Tang is approved.

11. 30. 2017

---

Advisor: Jaesik Choi

---

Committee Member: Se Young Chun

---

Committee Member: Jun Moon

In this thesis, we present a layer-wise learning of **Stochastic Neural Networks (SNNs)** in an information-theoretic perspective. In each layer of an **SNN**, the compression and the relevance are defined to quantify the amount of information that the layer contains about the input space and the target space, respectively. We jointly optimize the compression and the relevance of all parameters in an **SNN** to better exploit the neural network's representation. Previously, the **Information Bottleneck (IB)** ([1]) extracts relevant information for a target variable. Here, we propose **Parametric Information Bottleneck (PIB)** for a neural network by utilizing (only) its model parameters explicitly to approximate the compression and the relevance. We show that, the **PIB** framework can be considered as an extension of the **Maximum Likelihood Estimate (MLE)** principle to every layer level. We also show that, as compared to the **MLE** principle, **PIB** : (i) improves the generalization of neural networks in classification tasks, (ii) generates better samples in multi-modal prediction, (iii) is more efficient to exploit a neural network's representation by pushing it closer to the optimal information-theoretical representation in a faster manner. Our **PIB** framework, therefore, shows a great potential from an information-theoretic perspective for exploiting neural networks' representative power that have not yet been fully utilized.

# Contents

<b>Abstract</b>	<b>5</b>
<b>I. Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Thesis statement . . . . .	2
<b>II. Related Work</b>	<b>3</b>
<b>III. Background</b>	<b>4</b>
3.1 Notations . . . . .	4
3.2 Information Theory . . . . .	4
3.2.1 Entropy . . . . .	4
3.2.2 Relative Entropy . . . . .	7
3.2.3 Mutual Information . . . . .	8
3.2.4 Mutual Information versus Entropy . . . . .	9
3.2.5 Stochastic Encoding . . . . .	9
3.3 Neural Networks . . . . .	11
3.3.1 Back-propagation algorithms . . . . .	11
3.3.2 Loss functions as the MLE principle . . . . .	13
<b>IV. Information Bottleneck Principle</b>	<b>15</b>
4.1 The IB optimal representations . . . . .	15
4.2 The Information Plane . . . . .	17
<b>V. Parametric Information Bottleneck</b>	<b>19</b>
5.1 Neural Networks as Sequential Quantization . . . . .	19
5.2 PIB Framework . . . . .	20
5.2.1 Stochasticity . . . . .	20
5.2.2 Learning Principle . . . . .	21
5.3 Approximate learning . . . . .	23
5.3.1 Approximate Relevance . . . . .	23
5.3.2 Approximate Compression . . . . .	25
5.3.3 Approximate Gradients via Binary Bottlenecks . . . . .	26
<b>VI. Experiments</b>	<b>28</b>
6.1 MNIST Classification . . . . .	28
6.2 Learning dynamics . . . . .	30
6.3 Multi-modal learning . . . . .	31

6.4	Additional Experiment: Structure Analysis . . . . .	33
<b>VII.</b>	<b>Conclusion</b>	<b>36</b>
7.1	Summary . . . . .	36
7.2	Discussion . . . . .	36
7.2.1	Limitation . . . . .	36
7.2.2	Future work . . . . .	37
	<b>Acknowledgements</b>	<b>38</b>



## List of Figures

3.1	The entropy (in <i>bits</i> in this example) of Bernoulli( $p$ ) with various values of $p$ . The entropy $H(p)$ achieves its maximum with uniform distribution, i.e., $p = 0.5$ and is a concave functional with respect to $p$ . . . . .	6
3.2	Relationship between mutual information and entropy (Figure credit: [2]) . . . . .	10
3.3	This specific stochastic mapping that transforms space $\mathcal{X}$ of 2 dimensionality into a new code space $\mathcal{Z}$ of one dimensionality modifies the information content of the original space in a lossy way. A good learning principle should make this loss in a beneficial manner in which only irrelevant information is discarded and the relevant information is preserved. . . . .	10
3.4	A neuron is a computational unit in a neural network that squashes an affine transformation of the input vector by an non-linear activation function (figure credit: [3]). . . . .	11
3.5	An example of neural network architecture with 3 hidden layers (figure credit: [4]). . . . .	12
3.6	Two phases of back-propagation (figure credit: Prof. Sung Ju Hwang's lecture notes). . . . .	12
3.7	The backward pass propagates errors back to each neurons to fix up neuron weights (figure credit: Prof. Sung Ju Hwang's lecture notes). . . . .	13
4.1	The information curve $R(D)$ is a non-decreasing concave curve that divides the information plane into the achievable region and non-achievable region. Any point of compression-relevance $(I(Z,X), I(Z,Y))$ of a representation $Z$ is achievable iff it lies under the information curve in the information plane. The representation curve $Z_4 - Z_3 - Z_2 - Z_1$ of a neural network lies within the achievable region. The goal of learning a neural network is to move the representation curve close to the information curve. . . . .	17
5.1	A directed graphical representation of a PIB of two bottlenecks. The neural network parameters $\theta = (\theta_1, \theta_2, \theta_3)$ . The dashed blue arrows do not denote variable dependencies but the relevance decoders for each bottleneck. The relevance decoder $p_{true}(\mathbf{y} \mathbf{z}_i)$ , which is uniquely determined given the encoder $p_{\theta}(\mathbf{z}_i \mathbf{x})$ and the joint distribution $p_D(\mathbf{x}, \mathbf{y})$ , is intractable. We use $p_{\theta}(\mathbf{y} \mathbf{z}_i)$ as a variational approximation to each intractable relevance decoder $p_{true}(\mathbf{y} \mathbf{z}_i)$ . . . . .	21
5.2	Minimizing $J_{PIB}$ can be intuitively interpreted as tightening the information knots of a neural network architecture. Here a curvature of the curve connecting two consecutive layers represents compression while the thickness of the string connecting $Z_l$ to $Y$ indicates relevance level. . . . .	22
6.1	A comparison of Monte-Carlo averaging and deterministic prediction of PIB. . . . .	29
6.2	The learned weights of the first layer in MNIST classification for various models: deterministic neural networks (left), SFNN (middle), and PIB (right). . . . .	30

6.3 The learning dynamic of PIB (left) and SFNN (right) in a decision problem are presented in the information plane (the  $\log$  function is in the natural base  $e$ ). Each point represents a hidden layer while the color indicate epochs. Because of the Markov assumption (Equation 5.2), we have  $H(X) \geq I(Z_i, X) \geq I(Z_{i+1}, X)$  and  $I(X, Y) \geq I(Z_t, Y) \geq I(Z_{t+1}, Y)$ . 31

6.4 Classification errors during training and validation of PIB ( $\beta^{-1} = 10^{-4}$ ) and SFNN. . . . 32

6.5 Samples drawn from the prediction of the lower half of the MNIST test data digits based on the upper half for PIB (left, after 60 epochs) and SFNN (right, after 200 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. . . . . 33

6.6 The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-8-10-1. . 34

6.7 The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-6-10-1. . 34

6.8 The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-4-10-1. . 34

6.9 The learning dynamic of PIB (left) and MLE (right) in a decision problem in 10-10-8. . . 35

6.10 The learning dynamic of PIB (left) and MLE (right) in a decision problem in 10-10-6. . . 35

6.11 The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-10-1. . . 35

## List of Tables

6.1	The MNIST classification results of various models. . . . .	28
-----	---	----

## List of Abbreviations

- AEP** Asymptotic Equipartition Property. 9
- DNN** Deep Neural Network. 1, 3, 20
- DPI** Data Processing Inequality. 16, 19
- IB** Information Bottleneck. 1–3, 5, 6, 15–17, 20, 24, 30
- iff** if and only if. 6, 7, 24
- KL** Kullback-Leibler. 1, 7, 23, 43
- MLE** Maximum Likelihood Estimate. 1, 2, 5, 14, 24, 25, 28, 36, 43
- NLL** negative log-likelihood. 24, 25
- PIB** Parametric Information Bottleneck. 2, 5, 6, 8, 19–32, 36
- SFNN** Stochastic Feed-forward Neural Network. 28–32
- SGD** Stochastic Gradient Descent. 30, 31
- SNN** Stochastic Neural Network. 5
- VCR** Variational Conditional Relevance. 24, 25, 43

# Chapter I

## Introduction

### 1.1 Motivation

Deep Neural Networks (DNNs) have demonstrated competitive performance in several learning tasks including image recognition (e.g., [5], [6]), natural language translation (e.g., [7], [8]) and game playing (e.g., [9]). Specifically in supervised learning contexts, a common practice to achieve good performance is to train DNNs with the Maximum Likelihood Estimate (MLE) principle along with various techniques such as data-specific design of network architecture (e.g., convolutional neural network architecture), regularizations (e.g., early stopping, weight decay, dropout ([10]), and batch normalization ([11]), and optimizations (e.g., [12]). The learning principle in DNNs has therefore attributed to the MLE principle as a standard one for guiding the learning toward a beneficial direction. The question however is does the MLE principle effectively and sufficiently exploit a neural network's representative power and is there any better alternative? As an attempt to address this important question, this work investigates the learning of DNNs from the information-theoretic perspective.

An alternative principle is the Information Bottleneck (IB) framework ([1]) which extracts relevant information in an input variable  $X$  about a target variable  $Y$ . More specifically, the IB framework constructs a *bottleneck* variable  $Z = Z(X)$  that is an compressed version of  $X$  but preserves as much relevant information in  $X$  about  $Y$  as possible. In this information-theoretic perspective,  $I(Z, X)$ , the mutual information of  $Z$  and  $X$ , captures the compression of  $Z$  about  $X$  and  $I(Z, Y)$  represents the relevance of  $Z$  about  $Y$ . The optimal representation  $Z$  is determined via the minimization of the following Lagrangian:

$$\mathcal{L}_{IB}[p(z|x)] = I(Z, X) - \beta I(Z, Y) \quad (1.1)$$

where  $\beta$  is the positive Lagrangian multiplier that controls the trade-off between *representation complexity*,  $I(Z, X)$ , and *predictive power* in  $Z$ ,  $I(Z, Y)$ . The exact solution to the minimization problem above is found ([1]) with the implicit self-consistent equations:

$$\begin{cases} p(z|x) &= \frac{p(z)}{Z(x; \beta)} \exp(-\beta D_{KL}[p(y|x)||p(y|z)]) \\ p(z) &= \int p(z|x)p(x)dx \\ p(y|z) &= \int p(y|x)p(x|z)dx \end{cases} \quad (1.2)$$

where  $Z(x; \beta)$  is the normalization function, and  $D_{KL}[\cdot||\cdot]$  is the Kullback-Leibler (KL) divergence ([13]). Unfortunately, the self-consistent equations are highly non-linear and still non-analytic for most practical cases of interest. Furthermore, the general IB framework assumes that the joint distribution  $p(X, Y)$  is known and does not specify concrete models.

On the other hand, the goal of the MLE principle is to match the model distribution  $p_{model}$  as close to the empirical data distribution  $\hat{p}_D$  as possible (e.g., see Appendix I.B). The MLE principle treats the

neural network model  $p(\mathbf{x}; \boldsymbol{\theta})$  as a whole without explicitly considering the contribution of its internal structures (e.g., hidden layers and hidden neurons). In other words, the MLE principle is generic that is not specified to neural networks and does not explicitly make use of the hierarchical structure of neural networks during the learning phase. 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. In the MLE principle, we only need empirical samples of the joint distribution to maximize the likelihood function of the model given the data. The MLE principle is proved to be mathematically equivalent to the IB principle for the multinomial mixture model for clustering problem when the input distribution  $X$  is uniform or has a large sample size ([14]). However in general the two principles are not obviously related.

In this work, we propose a learning framework that is specifically tailored for neural networks. Particularly, we leverage neural networks and the IB principle by viewing neural networks as a set of encoders that sequentially modify the original data space. We then propose a new generalized IB-based objective that takes into account the compression and relevance of all layers in the network as an explicit goal for guiding the encodings in a beneficial manner. Since the objective is designed to optimize all parameters of neural networks and is mainly motivated by the IB principle for deep learning ([15]), we name this method the **Parametric Information Bottleneck (PIB)**. Because the generalized IB objective in the PIB is intractable, we approximate it using variational methods and Monte Carlo estimation. We propose re-using the existing neural network architecture as variational decoders for each hidden layers. The approximate generalized IB objective in turn presents interesting connections with the MLE principle. In practice, we empirically show that our PIBs have a better generalization and push the neural network's representation closer to the information-theoretical optimal representation as compared to the MLE principle.

## 1.2 Thesis statement

We start out by stating our main thesis (proposition) that we are maintaining in this work:

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

## Chapter II

### Related Work

Originally, the general **IB** framework is proposed in [1]. The framework provides a principled way of extracting the relevant information in one variable  $X$  about another variable  $Y$ . The authors represent the exact solution to the **IB** problem in highly-nonlinear self-consistent equations and propose the iterative Blahut Arimoto algorithm to optimize the objective. However, the algorithm is not applicable to neural networks. In practice, the **IB** problem can be solved efficiently in the following two cases only: (1)  $X, Y$  and  $Z$  are all discrete ([1]); or (2)  $X, Y$  and  $Z$  are mutually joint Gaussian ([16]) where  $Z$  is a bottleneck variable.

Recently, the **IB** principle is applied to **DNNs** ([15]). The work proposes to use mutual information of a hidden layer with the input layer and the output layer to quantify the performance of **DNNs**. By analyzing these measures with the **IB** principle, the authors establish an information-theoretic learning principle for **DNNs**. In theory, one can optimize the neural network by pushing up the network and all its hidden layers to the **IB** optimal limit in a layer-wise manner. Although the analysis offers a new perspective about optimality in neural networks, it proposes an general analysis of optimality rather than a practical optimization criteria. Furthermore, estimating mutual information between the variables transformed by network layers and the data variables poses several computational challenges in practice that the authors did not address in the work. A small change in a multi-layered neural network could greatly modify the entropy of the input variables. Thus, it is hard to analytically capture such modifications.

Recently, the authors in [17] have used variational methods to approximate the mutual information as an attempt to apply the **IB** principle to neural networks. Their approach however considers one single bottleneck and parameterizes the encoder  $p(\mathbf{z}|\mathbf{x};\boldsymbol{\theta})$  by an entire neural network. The encoder maps the input variable  $\mathbf{x}$  to a single bottleneck variable  $\mathbf{z}$  that is not a part of the considered neural network architecture. Therefore, their approach still treats a neural network as a whole rather than optimizing it layer-wise. Furthermore, the work imposes a variational prior distribution in the code space to approximate its actual marginal distribution. However, the variational approximate distribution for the code space may be too loose for a good approximation.

Our work, on the other hand, focuses on better exploiting intermediate representations of a neural network architecture using the **IB** principle. More specifically, our work proposes an optimization **IB** criteria for an existing neural network architecture in an effort of learning better the layers' representation to their **IB** optimality. In estimating mutual information, we adopt the variational method as in [17] for  $I(Z, Y)$  but use empirical estimation for  $I(Z, X)$ . Furthermore, we exploit the existing network architecture as variational decoders rather than resort to variational decoders that are not part of the neural network architecture.

# Chapter III

## Background

In this chapter, we provide preliminaries to the remaining chapters by reviewing the most relevant and fundamental concepts from Information Theory [2] and Neural Networks.

### 3.1 Notations

Throughout this thesis, we adopt the following notations. We use capital letters, e.g.,  $X, Y, Z$ , to represent random variables. Normal lowercase letters, e.g.,  $x, y, z$ , and bold lowercase letters, e.g.,  $\mathbf{x}, \mathbf{y}, \mathbf{z}$ , indicate specific values of univariate and multivariate random variables, respectively. Here  $p(X = \mathbf{x})$  is written as  $p(\mathbf{x})$  for short which indicates the probability of  $X$  at a specific value  $\mathbf{x}$ . Hence,  $p(\mathbf{x})$  and  $p(\mathbf{y})$  indicate different probability functions. Calligraphic letters, e.g.,  $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$  denotes the spaces in which the corresponding random variables live. We use notation  $|\mathcal{X}|$  to prefer to the cardinality of space  $\mathcal{X}$ . Probability distributions from data are denoted as  $p_D(\cdot)$ . Probability distributions from models are denoted as  $p(\cdot)$ . For simplicity, we use integral notations for expectation of a function of both discrete-valued and continuous random variables, i.e.,

$$\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})] = \int p(\mathbf{x})f(\mathbf{x})d\mathbf{x}.$$

We use notation  $\mathbf{x} \sim p(\mathbf{x})$  to indicate that  $\mathbf{x}$  is sampled according to distribution  $p(\mathbf{x})$ .

### 3.2 Information Theory

#### 3.2.1 Entropy

As an introduction of entropy, we consider a simple example of data compression as follows.

**Example 3.2.1.** Consider a discrete random variable  $X$  with the following distribution:

$$\begin{cases} p(X = 1) = \frac{1}{3} \\ p(X = 2) = \frac{1}{3} \\ p(X = 3) = \frac{1}{3} \end{cases}$$

Now assume that we wish to encode each element  $X = i \in \mathcal{X}$  into a binary string  $C(i)$  of various length  $l_i$ . One measure for the quality of the encoding is a *degree of compression*, i.e., the expected length of bits needed to describe the distribution,

$$L(C) = \sum p(X = i)l_i$$



Let's consider, for instance, the encoding function:

$$C(1) = 0$$

$$C(2) = 10$$

$$C(3) = 11$$

Note that this encoding function makes no confusion in recovery, e.g., decoded message 01011010 is uniquely decoded back into 12312 without any confusion. The expected length, i.e, the expected number of bits, to describe the distribution under the encoding function  $C$  is then,

$$L(C) = \frac{1}{3} \times 1 + \frac{1}{3} \times 2 + \frac{1}{3} \times 2 = \frac{5}{3} \approx 1.67 \text{ (bits)}$$

A natural question is what is the minimum expected number of bits needed to describe the distribution without confusion? No matter how we compress the signal  $X$ , there exists an irreducible complexity of the signal below which it cannot be further compressed. This irreducible complexity is the entropy of the signal.

Entropy is a fundamental concept in information theory ([2]) that measures the uncertainty of a random variable. Intuitively, the more uncertain an event, the more surprising, the more informative, and the greater its entropy. To establish a functional,  $H(X)$  or  $H(p)$ , that can quantify the amount of an uncertainty in a probability distribution of a discrete random variable,  $p(x)$ , Shannon ([18]) suggested three axioms that any such functional must follow.

**Axiom 1.**  $H(X)$  is continuous in  $p(x)$ .

**Axiom 2.** If  $p(x) = \frac{1}{|\mathcal{X}|}, \forall x$ , then  $H(X)$  is a monotonically increasing function of  $|\mathcal{X}|$ .

**Axiom 3.** For any grouping of  $\mathcal{X} = \{x_1, \dots, x_{|\mathcal{X}|}\}$  into the group  $\mathcal{T} = \{t_1, \dots, t_{|\mathcal{T}|}\}$ , the functional  $H(X)$  must satisfy:

$$H(X) = H(p(x)) = H(p(t)) + \sum_t p(t)H(p(x|t))$$

Interestingly, the entropy defined in the following definition is the only functional that satisfies three axioms above.

**Definition 3.2.1.** (Entropy) The entropy  $H(X)$  of a discrete random variable  $X$  is defined as

$$H(X) = - \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

Here we adopt the convention that  $0 \log 0 = 0$  according the property that  $x \log x \rightarrow 0$  as  $x \rightarrow 0$ . Function  $\log$  is either to the base 2 or the base logarithm  $e$ , and the entropy is measured in *bits* or *nats*, respectively. Note that  $\log$  in this chapter is all expressed in bits while it is based in logarithm  $e$  in the remaining chapters. For the example of data compression in the beginning, the entropy (in bits) of signal  $X$  can be computed by the above definition as,

$$H(X) = -3 \times \frac{1}{3} \log \frac{1}{3} \approx 1.58 < L(C) \approx 1.67$$

To illustrate the concavity property of entropy, we consider a random Bernoulli variable in the following simple example.

**Example 3.2.2.** Let  $X$  be a random Bernoulli variable  $\text{Bernoulli}(p)$ , i.e.,

$$X = \begin{cases} 1 & \text{with probability of } p \\ 0 & \text{with probability of } 1 - p \end{cases}$$

Then, the entropy of  $X$  is

$$H(X) = H(p) = -p \log p - (1 - p) \log(1 - p)$$

A plot of  $H(p)$  is illustrated in Figure 3.1. The plot shows concavity of  $H(p)$  in terms of  $p$  where  $H(p)$

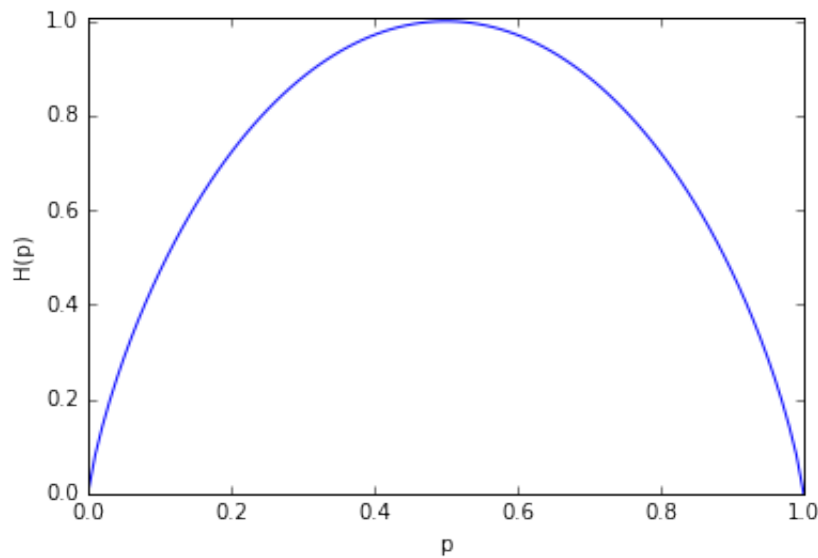


Figure 3.1: The entropy (in *bits* in this example) of  $\text{Bernoulli}(p)$  with various values of  $p$ . The entropy  $H(p)$  achieves its maximum with uniform distribution, i.e.,  $p = 0.5$  and is a concave functional with respect to  $p$ .

has its maximum when  $\text{Bernoulli}(p)$  becomes a uniform distribution, i.e.,  $p = 0.5$ . These two properties also hold for general cases as stated in the following propositions.

**Proposition 3.2.1.**  $H(X) \in [0, \log|\mathcal{X}|]$  and is a concave function of  $p(x)$ .

**Proposition 3.2.2.** For any discrete random variable  $X$ ,  $H(X)$  achieves its maximum *if and only if (iff)*  $p(x)$  is a uniform distribution, i.e.,  $p(x) = \frac{1}{|\mathcal{X}|}, \forall x \in \mathcal{X}$ .

A formal proof of Proposition 3.1 and Proposition 3.2 can be found in Theorem 2.7.3 (Convexity of entropy) and Theorem 2.6.4 in [2].

The concept of entropy can be extended to more than one random variables as stated in the following definition.

**Definition 3.2.2.** (Joint entropy) The joint entropy  $H(X, Y)$  of a pair of discrete random variables  $(X, Y)$  with a joint distribution  $p(x, y)$  is defined as

$$H(X, Y) = - \sum_{(x, y) \in (\mathcal{X}, \mathcal{Y})} p(x, y) \log p(x, y)$$

We also define the conditional entropy of a random variable given another variable as follows.

**Definition 3.2.3.** (Conditional entropy) If  $(X, Y) \sim p(x, y)$ , then the conditional entropy of  $Y$  given  $X$  is defined as

$$H(Y|X) = \sum_{x \in \mathcal{X}} p(x) H(Y|X = x) = - \sum_{(x, y) \in (\mathcal{X}, \mathcal{Y})} p(x) p(y|x) \log p(y|x)$$

Intuitively,  $H(Y|X)$  is the uncertainty in  $Y$  once  $X$  is known. Furthermore, the uncertainty in both  $X$  and  $Y$  equals the uncertainty in  $X$  plus the uncertainty in  $Y$  when  $X$  is given. This property is formally stated in Theorem 3.2.3.

**Theorem 3.2.3.** (Chain rule for entropy)

$$H(X, Y) = H(X) + H(Y|X)$$

**Proof:** This result is straightforward from the definition of entropy and the property of log that  $\log(xy) = \log(x) + \log(y)$ .

### 3.2.2 Relative Entropy

Another important concept from Information Theory that we used in this thesis is relative entropy or **KL** divergence,  $D_{KL}[\cdot||\cdot]$ . This concept measures the discrepancy between two distributions. More specifically,  $D_{KL}[p||q]$  is a measure of invalidity of assuming that the distribution is  $q$  while the true distribution is  $p$ . Its concise definition is shown in the following definition:

**Definition 3.2.4.** (Relative entropy)

$$D_{KL}[p||q] := \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$$

Especially, the relative entropy is always non-negative as stated in the following theorem.

**Theorem 3.2.4.** (Information inequality)

$$D_{KL}[p||q] \geq 0$$

with equality *iff*  $p(x) = q(x) \forall x \in \mathcal{X}$ .

The proof of Theorem 3.2.4 directly follows the result of Jensen's inequality:

**Theorem 3.2.5.** (*Jensen's inequality*) If  $f$  is a convex function and  $X$  is a random variable,

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$$

with equality if  $X$  is constant or  $f$  is linear.

**Zero-forcing property of KL divergence:** In practice, e.g., in *variational inference*, we usually approximate the intractable distribution  $p$  with a variational (tractable) distribution  $q$  by reducing their KL divergence  $D_{KL}[p||q]$  over a set of empirical samples drawn from  $p$ . An emergent effect of such approximation is that it encourages  $q$  to be close to zero whenever  $p$  is zero.

### 3.2.3 Mutual Information

Here we review the concept of mutual information, which is heavily used in this thesis. This time, we get started with a definition before considering a concrete example.

**Definition 3.2.5.** (Mutual information) Consider two random variable  $X, Y$  with joint distribution  $p(X, Y)$ , and marginal distributions  $p(X)$  and  $p(Y)$ . The mutual information,  $I(X, Y)$ , is then defined as

$$I(X, Y) = I(Y, X) = \sum_{(x,y) \in (\mathcal{X}, \mathcal{Y})} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

An intuitive meaning of mutual information  $I(X, Y)$  is that it measures *the amount of information that  $X$  contains about  $Y$* . The random variables  $X$  and  $Y$  have some degree of dependence (intrinsically determined via the joint distribution  $p(X, Y)$ ) that knowing some (sampled) values of  $X$  gives some guidance in predicting values of  $Y$ . How much is such guidance depends on how much information about  $Y$  that  $X$  has, i.e.,  $I(X, Y)$ . Note that the role of  $X$  and  $Y$  are exchangeable in mutual information  $I(X, Y)$ . As an illustration, consider a following simple example.

**Example 3.2.3.** Let  $X_1, X_2$ , and  $Y$  be three random variables such that

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

and

$$p(y, x_1) = \begin{cases} 1 & \text{if } y = x_1 = 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$p(y, x_2) = \frac{1}{4} \quad \forall y, x_2$$

It follows from  $p(y, x_1)$  and  $p(y, x_2)$  that the value of  $Y$  is totally determined once  $X_1$  is known while knowing  $X_2$  gives no clue about value of  $Y$ . Intuitively, we would expect that  $Y_1$  contains some

information about  $X$  while  $Y_2$  has no information about  $X$ . This intuition is indeed correctly quantified by mutual information since we have, from the definition, that

$$I(X_1, Y) = 2 = H(X)$$

$$I(X_2, Y) = 0$$

In the special case when  $X$  and  $Y$  are independent, their mutual information  $I(X, Y)$  becomes zero, i.e., knowing  $X$  does not enable a shorter description of  $Y$  and vice versa.

### 3.2.4 Mutual Information versus Entropy

The knowledge of  $X$  gives some information about  $Y$  which reduces the uncertainty of  $Y$ . Intuitively, mutual information  $I(X, Y)$  can be interpreted as the reduction of the uncertainty in  $Y$  when  $X$  is known. Theorem 3.2.6 rigorously describes the aforementioned intuitions followed by the Venn diagram in Figure 3.2 summarizing such relationships.

#### Theorem 3.2.6.

$$I(X, Y) = I(Y, X) \tag{3.1}$$

$$I(X, X) = H(X) \tag{3.2}$$

$$I(X, Y) = H(X) - H(X|Y) \tag{3.3}$$

$$I(X, Y) = H(Y) - H(Y|X) \tag{3.4}$$

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \tag{3.5}$$

### 3.2.5 Stochastic Encoding

Let  $X$  be a random signal with probability  $p(X)$ . We assume here that  $\mathcal{X}$  is a finite space. A stochastic encoder  $p(z|x)$  induces a soft partitioning of space  $\mathcal{X}$  into a new space  $\mathcal{Z}$  (so-called code space) with probability measure,

$$p(z) = \sum_{x \in \mathcal{X}} p(x)p(z|x)$$

The encoding may modify the structure and the information content of space  $\mathcal{X}$  (e.g., Figure 3.3). It follows from the **Asymptotic Equipartition Property (AEP)** ([2]) that there are on average  $2^{H(X|Z)}$  elements in  $\mathcal{X}$  that are mapped to the same code in  $\mathcal{Z}$ . Since the “typical” volume of  $\mathcal{X}$  is  $2^{H(X)}$ , the average volume of the partitioning in  $\mathcal{X}$  that is induced by  $p(z|x)$  is,

$$\frac{2^{H(X)}}{2^{H(X|Z)}} = 2^{I(X,Z)}.$$

To put it literally, we need on average  $I(X, Z)$  bits per element in  $\mathcal{X}$  to specify an element in  $\mathcal{Z}$  without confusion. This is also the rate of the encoding, one factor that quantifies the quality of the encoding.

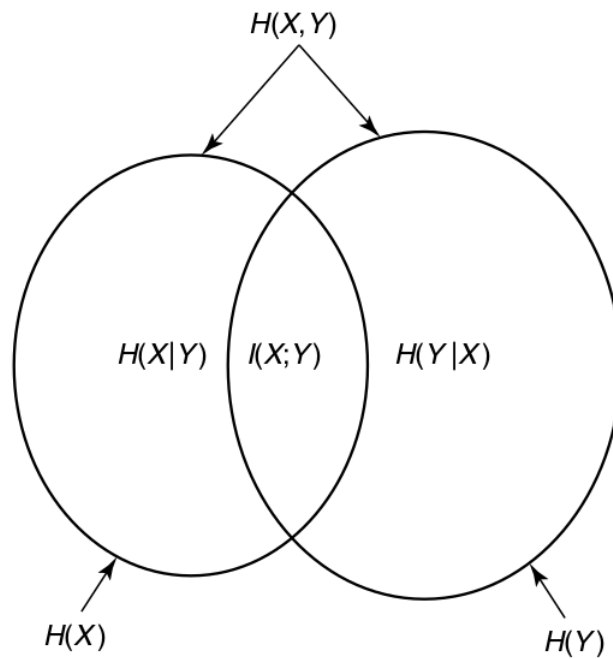


Figure 3.2: Relationship between mutual information and entropy (Figure credit: [2])

**Definition 3.2.6.** (Rate of encoding) The rate of an encoding,  $R[p(z|x)]$ , is the average number of bits per message needed to specify a code in the code space without confusion.

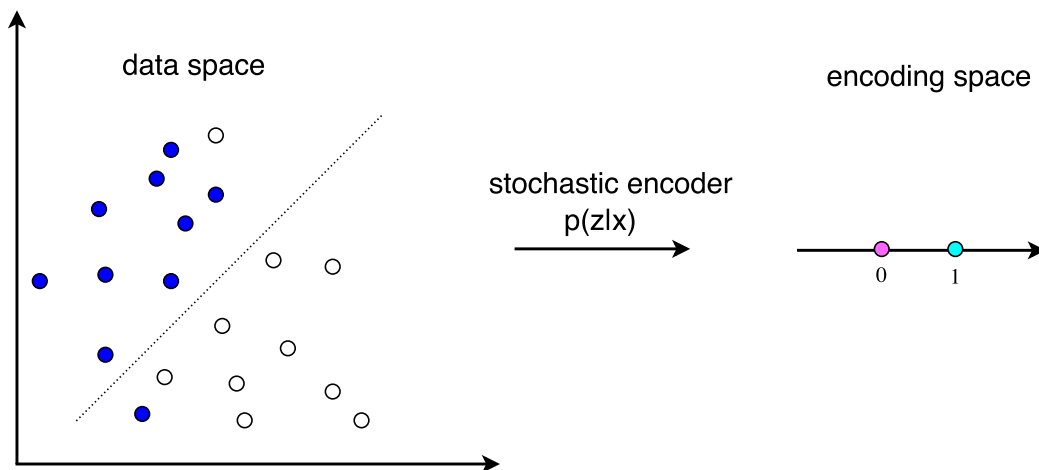


Figure 3.3: This specific stochastic mapping that transforms space  $\mathcal{X}$  of 2 dimensionality into a new code space  $\mathcal{Z}$  of one dimensionality modifies the information content of the original space in a lossy way. A good learning principle should make this loss in a beneficial manner in which only irrelevant information is discarded and the relevant information is preserved.

Mutual information,  $I(Z, X)$ , is a measure of the quality of the encoding since it bounds from below the rate of the encoding as stated in the following theorem.

**Theorem 3.2.7.**

$$R[p(z|x)] \geq I(Z, X)$$

A more rigorous argument for the correctness of Theorem 3.2.7 can be found in Chapter 7 of [2].

### 3.3 Neural Networks

A neural network is a parametrized model that is inspired by human brain's architecture. It is a *universal approximator* that can learn any smooth predictive relationship given sufficient data ([19]).

To describe neural networks, let's consider the supervised learning context where we have access to the labeled training set  $S_D := \{(\mathbf{x}^{(i)}, y^{(i)}) \in (\mathcal{X}, \mathcal{Y}) | 1 \leq i \leq N\}$ . Neural networks define a non-linear form of hypothesis  $f(\mathbf{x}; W, \mathbf{b})$  with learnable parameters  $W$  and  $\mathbf{b}$ . A neural network consists of neurons that are rearranged in a specific architecture (e.g., feed-forward neural network, convolutional neural networks [20], recurrent neural networks [21]). Each neuron (Figure 3.4) is a computational unit that makes an affine transformation of its input possibly followed by a squashing function, e.g., sigmoid, tanh, and ReLU. Figure 3.5 shows an specific example of neuron arrangement with feed-forward architecture.

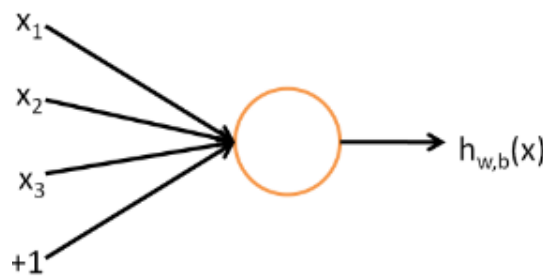


Figure 3.4: A neuron is a computational unit in a neural network that squashes an affine transformation of the input vector by an non-linear activation function (figure credit: [3]).

#### 3.3.1 Back-propagation algorithms

Back-propagation algorithm [22] is a very famous and successful algorithm to learn neural networks. Due to its popularity, we ignore its details, which can be found for instance in this tutorial [23], and focus instead on its meaning. Back-propagation consists of two phases for each iteration: forward and backward pass (Figure 3.6). In the forward pass, the network passes a batch of data examples through the network architecture and computes the loss at the output end according to the defined loss function. During the backward pass, the network computes the partial derivatives of the loss with respect to the network weights using the chain rule and then update the weights. Intuitively, the backward pass propagates the errors computed in the forward pass back to each neurons and updates the neuron weights to fix up the errors collectively (Figure 3.7). An advantage of back-propagation is that it enables large-scale learning in which a large set of data can be trained efficiently by gradient-based methods using a

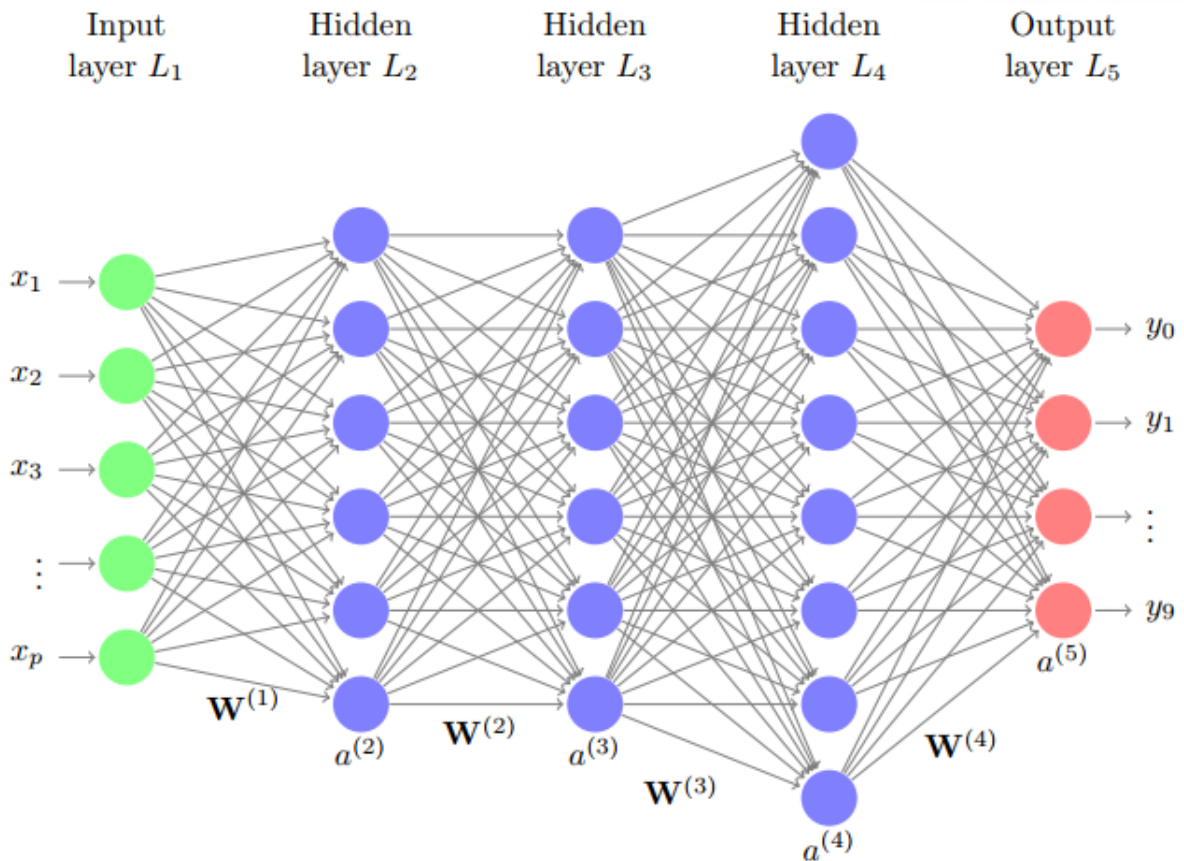


Figure 3.5: An example of neural network architecture with 3 hidden layers (figure credit: [4]).

small batch of data samples at each iteration. It is also shown to be efficient for non-convex loss function in the context of deep learning. The main disadvantage is however it does not guarantee to obtain the minimum, e.g., the obtained value may be suboptimal or local minimum.

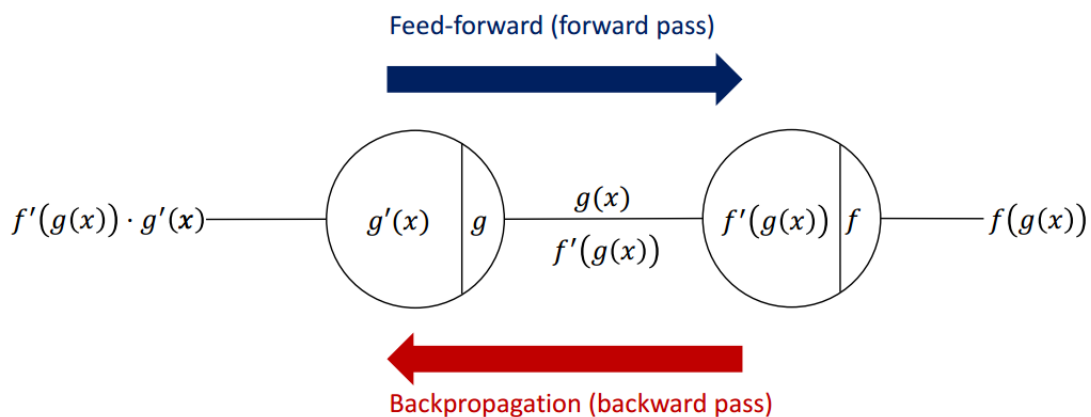


Figure 3.6: Two phases of back-propagation (figure credit: Prof. Sung Ju Hwang's lecture notes).



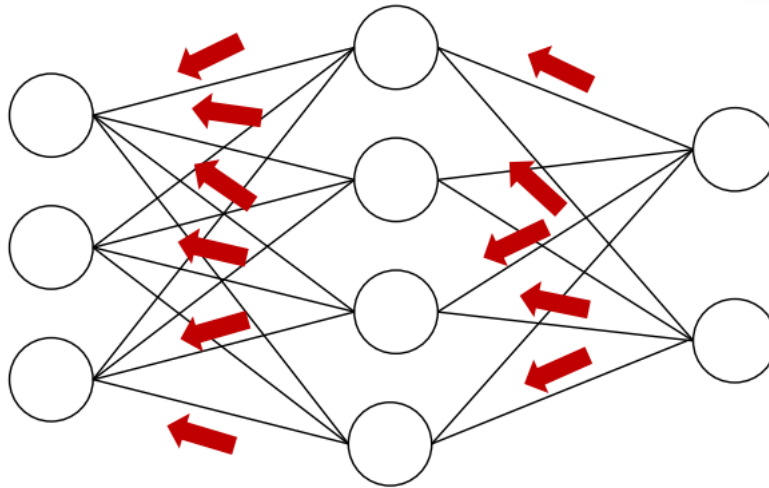


Figure 3.7: The backward pass propagates errors back to each neurons to fix up neuron weights (figure credit: Prof. Sung Ju Hwang’s lecture notes).

### 3.3.2 Loss functions as the MLE principle

We demonstrate that common loss functions in neural networks for supervised learning actually follow the MLE principle. Specifically, we consider here the two most common loss functions: the squared loss function and the cross-entropy loss function. We denote  $\hat{p}_D$  as empirical data distribution in which

$$\hat{p}_D(\mathbf{y}|\mathbf{x}^{(i)}) = \begin{cases} 1 & \text{if } \mathbf{y} = \mathbf{y}^{(i)}, \\ 0 & \text{otherwise.} \end{cases} \quad (3.6)$$

Here we consider deterministic neural networks only and explain how the conditional distribution  $p(\mathbf{y}|\mathbf{x})$  is defined. For stochastic neural networks, the conditional distribution  $p(\mathbf{y}|\mathbf{x})$  is made clear in Subsection 5.2.1 of Chapter 5. We denote  $\hat{\mathbf{y}}$  as the output of the last layer (before activation if any) when  $\mathbf{x}$  is fed into the network as an input.

#### Squared loss

In this case, we assume a multivariate Gaussian distribution with the neural network output as its mean vectors and unit covariance matrix, i.e.,

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}; \hat{\mathbf{y}}, I) \quad (3.7)$$

$$= (2\pi)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{y} - \hat{\mathbf{y}})^T(\mathbf{y} - \hat{\mathbf{y}})\right) \quad (3.8)$$

Then the squared loss function can be interpreted as the KL divergence between the empirical conditional data distribution and the conditional model distribution:

$$\mathcal{L}_{squared}(\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \frac{1}{2} \|\hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)}\|^2 \quad (3.9)$$

$$= -\hat{p}_D(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) + const \quad (3.10)$$

$$= -\sum_{y=1} \hat{p}_D(y|\mathbf{x}^{(i)}) \log p(y|\mathbf{x}^{(i)}) \quad (3.11)$$

### Cross-entropy loss

The cross-entropy loss is also easily rewritten as the KL divergence between two distributions:

$$p(y = k|\mathbf{x}^{(i)}) = [\text{softmax}(\hat{\mathbf{y}})]_k, 1 \leq k \leq C \quad (3.12)$$

$$\mathcal{L}_{cross-entropy}(\mathbf{x}^{(i)}; \boldsymbol{\theta}) = -\log p(y = y^{(i)}|\mathbf{x}^{(i)}) \quad (3.13)$$

$$= -\sum_{y=1}^C \hat{p}_D(y|\mathbf{x}^{(i)}) \log p(y|\mathbf{x}^{(i)}) \quad (3.14)$$

Thus, as we can see from the above examples, a common loss function in neural networks for a supervised learning context can be interpreted as an “implementation” of the **MLE** principle. This is made possible by distribution assumptions of the neural network model depending on the type of the output space and loss functions.

## Chapter IV

### Information Bottleneck Principle

This chapter serves an important background as well as motivation for our work in the remaining chapters. Here we present the **IB** principle proposed by [1]. The **IB** framework provides a principled way of extracting relevant information in one variable about another variable. In this chapter, we revisit this concept from our own perspectives.

Consider a general context of supervised learning in which we wish to learn a (deterministic or stochastic) mapping  $p(Y|X)$  from an input space  $\mathcal{D}_1^D \subseteq \mathbb{R}^D$  into a target space  $\mathcal{D}_2^K \subseteq \mathbb{R}^K$  such that the learned mapping is as close to some underlying unknown conditional distribution  $p_D(Y|X)$  as possible given a set of finite samples  $\{(x^{(i)}, y^{(i)}) \in (\mathcal{D}_1^D \times \mathcal{D}_2^K) | 1 \leq i \leq N\}$  drawn i.i.d. from some underlying unknown distribution  $p_D(X, Y)$ . Depending on the structure of the target space and the (unknown) underlying distribution  $p_D$ , the described context can become several tasks including classification, regression, image segmentation or multi-modal prediction problems. Moreover, the way that the input space presents the information and the relevant information about the target space can be implicitly complicated depending on the input space structure. Therefore, it is beneficial for subsequent tasks to represent such information in the input space in a manner such that they can disentangle underlying explanatory factors in the data [24]. Specifically for the context of supervised learning, one is interested in finding a good representation  $Z = Z(X)$  of the input that compresses the data but preserves the relevant information about  $Y$ . Such representation, closely related to the Minimum Description Length (MDL) Principle [25], presents useful regularities in the data about the target which in turn enables a minimal description of the model and better generalization.

#### 4.1 The **IB** optimal representations

An optimal representation  $Z = Z(X)$  of the input  $X$  with regard to the target  $Y$  can be defined in terms of *compression* and *relevance* level in the representation  $Z$ . Here, compression in a representation  $Z$  can be defined as the amount of information of the input data which is still present in  $Z$ . Respectively, relevance in a representation  $Z$  is defined as the amount of information that the representation  $Z$  contains about the target variable  $Y$ . A detailed definition is presented in Definition 4.1.1.

**Definition 4.1.1.** (Compression and Relevance) Given three random variables  $X, Y$ , and  $Z$  where  $Z = Z(X)$  is a representation of  $X$  from which  $Z$  is stochastically mapped.

- **Compression:** the averaged number of bits per signal in  $X$  to specify  $Z$  without confusion, i.e., the averaged amount of information that  $Z$  contains about  $X$ .
- **Relevance:** the averaged amount of information that  $Z$  contains about  $Y$ .

For such intuitive definitions of compression and relevance, mutual information naturally becomes a good measure that can quantify these concepts in a representation  $Z$ . Specifically, the compression and the relevance in the representation  $Z$  can be measured with  $I(Z, X)$  and  $I(Z, Y)$ , respectively. Recall that the mutual information  $I(Z, X)$  measures the amount of information that  $Z$  contains about  $X$ , or the decrease in the uncertainty of  $X$  when  $Z$  is known. Since  $Z$  is resulted from a data processing on  $X$ , the three variables  $Y, X$ , and  $Z$  form a Markov chain in that order which is denoted as  $Y \rightarrow X \rightarrow Z$ . It then follows from the **Data Processing Inequality (DPI)** [2] that

$$I(X, Y) \geq I(Z, Y) \quad (4.1)$$

$$I(X, X) = H(X) \geq I(Z, X). \quad (4.2)$$

Ideally, we wish  $Z = Z(X)$  to be the minimal sufficient statistics of  $X$  that preserves all the information in  $X$  about  $Y$ , i.e.,  $I(Z, Y) = I(X, Y)$  (maximum relevance) and that has the minimum description length, i.e.,  $I(Z, X)$  is minimized (maximum compression). The minimal sufficient statistics is formally determined via the optimization problem (see more on Theorem 5 of [26]):

$$\min_{Z=Z(X):I(Z,Y)=I(X,Y)} I(Z, X) \quad (4.3)$$

The authors in [1] relaxed this optimization problem to be approximate sufficient statistics by using the method of Lagrangian multipliers to ease the original minimization problem with the minimization of the following objective:

$$\mathcal{L}\{p(z|x)\} := I(Z, X) - \beta I(Z, Y) \quad (4.4)$$

where  $\beta$  is a positive Lagrange multiplier that determines the trade-off between compression and relevance in  $Z$ . As  $\beta \rightarrow 0$ , the minimization of the Lagrangian (4.4) results in the maximum compression,  $I(Z, X) = 0$  (e.g., it collapses all information in  $X$  into a single point) thus also loses all useful information,  $I(Z, Y) = 0$ . On the other hand, as  $\beta \rightarrow \infty$ , the minimization leads to the maximum relevance,  $I(Z, Y) = I(X, Y)$ , but the minimum compression,  $I(Z, X) = H(X)$ , i.e.,  $Z = \phi(X)$  where  $\phi(\cdot)$  is an invertible deterministic function. The relaxation in (4.4) has another interpretation as it maintains the relevance in  $Z$  above certain level, i.e.,  $I(Z, Y) \geq D$  for some  $0 < D < I(X, Y)$ , which is controlled by the Lagrange parameter  $\beta$ . The authors in [1] gave an exact yet implicit solution to the minimization problem (4.4) via the **IB** self-consistent equations:

$$p(z|x) = \frac{p(z)}{Z(x; \beta)} \exp(-\beta D_{KL}[p(y|x)||p(y|z)]) \quad (4.5)$$

where  $Z(x; \beta)$  is the normalization function, and  $p(z)$  and  $p(y|z)$  respects the constraints for a valid distribution and the Bayes' Rule:

$$p(z) = \int p(z|x)p(x)dx \quad (4.6)$$

$$p(y|z) = \int p(y|x)p(x|z)dx \quad (4.7)$$

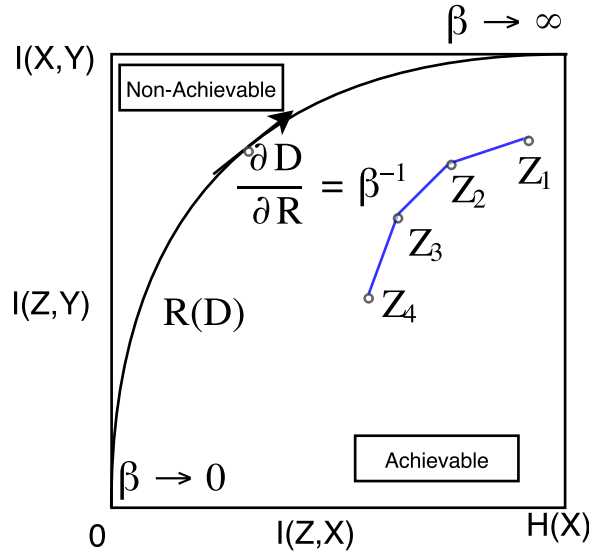


Figure 4.1: The information curve  $R(D)$  is a non-decreasing concave curve that divides the information plane into the achievable region and non-achievable region. Any point of compression-relevance  $(I(Z, X), I(Z, Y))$  of a representation  $Z$  is achievable iff it lies under the information curve in the information plane. The representation curve  $Z_4 - Z_3 - Z_2 - Z_1$  of a neural network lies within the achievable region. The goal of learning a neural network is to move the representation curve close to the information curve.

Note that once the joint distribution  $p(x, y)$  and an encoder  $p(z|x)$  are given, the corresponding decoder  $p(y|z)$  is uniquely determined (via Equation 4.7). Since  $p(y|z)$  maps space  $\mathcal{Z}$  onto the target space  $\mathcal{Y}$ , instead of reconstructing  $X$ , we call  $p(y|z)$  *relevance decoder* rather than decoder.

**Definition 4.1.2.** (Relevance decoder) Given  $p(x, y)$  and  $p(z|x)$ ,  $p(y|z)$  defined in Equation 4.7 is relevance decoder corresponding to encoder  $p(z|x)$ .

The IB self-consistent equations (4.5)-(4.6)-(4.7) poses a highly nonlinear functional of  $p(z|x)$  which is, unfortunately, very challenging to solve. In practice, the IB problem can be solved efficiently in the following two cases only: (1)  $X, Y$  and  $Z$  are all discrete [1]; or (2)  $X, Y$  and  $Z$  are mutually joint Gaussian [16].

## 4.2 The Information Plane

Developed by [1], the information plane is an information-theoretic plane that characterizes any representation  $Z = Z(X)$  in terms of the achievability of compression-relevance  $(I(Z, Y), I(Z, X))$  given the joint distribution  $I(X, Y)$ . The plane has  $I(Z, X)$  and  $I(Z, Y)$  as its horizontal axis and its vertical axis, respectively. In the plane, the information curve, which is given as

$$R(D) := \min_{Z=Z(X): I(Z, Y) \geq D} I(Z, X), \quad (4.8)$$

characterizes the achievability in the representation  $Z$ . Finding the so-called compression-relevance function  $R(D)$  [27] is an equivalent problem to the optimization of the Lagrangian (4.4). An interesting property of the compression-relevance function,  $R(D)$  is that it is a non-decreasing concave function of  $D$  with the slope determined by the Lagrangian multiplier [1], [27]:

$$\frac{\partial D}{\partial R} = \beta^{-1}, \quad (4.9)$$

and that it is an inherent characteristic of the joint distribution  $p(x, y)$  regardless of any model assumptions. The information plane and information curve is illustrated in Figure 4.1.

## Chapter V

### Parametric Information Bottleneck

This chapter presents our main contribution. Here we describe an information-theoretic perspective of neural networks and then define our **PIB** framework. This perspective paves a way for the soundness of constraining the compression-relevance trade-off into a neural network layers.

We first introduce notations used in this chapter. We denote  $X, Y$  as the input and the target (label) variables of the data, respectively;  $Z_l$  as a stochastic variable represented by the  $l^{\text{th}}$  hidden layer of a neural network where  $1 \leq l \leq L$ ,  $L$  is the number of hidden layers. We extend the notations of  $Z_l$  by using the convention  $Z_0 := X$  and  $Z_{-1} := \emptyset$ . The space of  $X, Y$  and  $Z_l$  are denoted as  $\mathcal{X}, \mathcal{Y}$  and  $\mathcal{Z}_l$ , respectively. Each respective space is associated with the corresponding probability measures  $p_D(\mathbf{x}), p_D(\mathbf{y})$  and  $p(\mathbf{z}_l)$  where  $p_D(\cdot)$  indicates the underlying probability distribution of the data and  $p(\cdot)$  denotes model distributions. Each  $Z_l$  is stochastically mapped from the previous stochastic variable  $Z_{l-1}$  via an encoder  $p(\mathbf{z}_l | \mathbf{z}_{l-1})$ . We name  $Z_l, 1 \leq l \leq L$  as a (information) bottleneck or code variable of the network. In this work, we focus on binary bottlenecks where  $Z_l \in \{0, 1\}^{n_l}$  and  $n_l$  is the dimensionality of the bottleneck space.

#### 5.1 Neural Networks as Sequential Quantization

An encoder  $p(\mathbf{z} | \mathbf{x})$  introduces a soft partitioning of the space  $\mathcal{X}$  into a new space  $\mathcal{Z}$  whose probability measure is determined as  $p(\mathbf{z}) = \int p(\mathbf{z} | \mathbf{x}) p_D(\mathbf{x}) d\mathbf{x}$ . The encoding can modify the information content of the original space possibly including its dimensionality and topological structure. On average,  $2^{H(X|Z)}$  elements of  $\mathcal{X}$  are mapped to the same code in  $\mathcal{Z}$ . Thus, the average volume of a partitioning of  $\mathcal{X}$  is  $2^{H(X)} / 2^{H(X|Z)} = 2^{I(X,Z)}$ . The mutual information  $I(Z, X)$  which measures the amount of information that  $Z$  contains about  $X$  can therefore quantify the quality of the encoding  $p(\mathbf{z} | \mathbf{x})$ . A smaller mutual information  $I(Z, X)$  implies a more compressed representation  $Z$  in terms of  $X$ .

Since the original data space is continuous, it requires infinite precision to represent it precisely. However, only some set of underlying explanatory factors among others in the the data space would be beneficial for a certain task. Therefore, lossy representation is often more helpful (and of course more efficient) than a precise representation. In this aspect, we view the hidden layers of a multi-layered neural network as a lossy representation of the data space. The neural network in this perspective consists of a series of stochastic encoders that sequentially encode the original data space  $\mathcal{X}$  into the intermediate code spaces  $\mathcal{Z}_l$ . These code spaces are lossy representations of the data space as it follows from the **DPI** ([2]) that

$$H(X) \geq I(X, Z_l) \geq I(X, Z_{l+1}) \quad (5.1)$$

where we assume that  $Y, X, Z_l$  and  $Z_{l+1}$  form a Markov chain in that order, i.e.,

$$Y \rightarrow X \rightarrow Z_l \rightarrow Z_{l+1} \quad (5.2)$$

A learning principle should compress irrelevant information and preserve relevant information in the lossy intermediate code spaces. In the next subsection, we describe in details how a sequential series of encoders, compression and relevance are defined in a neural network.

## 5.2 PIB Framework

Our **PIB** framework is an extension of the **IB** framework to optimize all parameters of neural networks. In neural networks, intermediate representations represent a hierarchy of information bottlenecks that sequentially extract relevant information for a target from the input data space. Existing **IB** framework for **DNNs** specifies a single bottleneck while our **PIB** preserves hierarchical representations which a neural network's expressiveness comes from. Our **PIB** also gives neural networks an information-theoretic interpretation both in network structure and model learning. In **PIBs**, we utilize only neural network parameters  $\theta$  for defining encoders and variational relevance decoders at every level, therefore the name *Parametric Information Bottleneck*. Our **PIB** is also a standard step towards better exploiting representational power of more expressive neural network models such as Convolutional Neural Networks ([20]) and ResNet ([28]).

### 5.2.1 Stochasticity

In this paper, we focus on binary bottlenecks in which the encoder  $p(\mathbf{z}_l | \mathbf{z}_{l-1})$  is defined as

$$p(\mathbf{z}_l | \mathbf{z}_{l-1}) = \prod_{i=1}^{n_l} p(z_{l,i} | \mathbf{z}_{l-1}) \quad (5.3)$$

where

$$p(z_{l,i} = 1 | \mathbf{z}_{l-1}) = \sigma(a_i^{(l)}) = \sigma(W_i^{(l)} \mathbf{z}_{l-1} + b_i^{(l)}), \quad (5.4)$$

$\sigma(\cdot)$  is the sigmoid function, and  $W^{(l)}$  is the weights connecting the  $l^{\text{th}}$  layer to the  $(l+1)^{\text{th}}$  layer. Depending on the structure of the target space  $\mathcal{Y}$ , we can use an appropriate model for output distributions as follows: (1) For classification, we model the output distribution with softmax function,  $p(Y = i | \mathbf{z}_L) = \text{softmax}(W_i^{(L+1)} \mathbf{z}_L + \mathbf{b}_i^{(L+1)})$ ; (2) For binary output vectors  $Y$ , we use a product of Bernoulli distributions,  $p(\mathbf{y} | \mathbf{z}_L) = \prod_i p(y_i | \mathbf{z}_L)$  where  $p(Y_i = 1 | \mathbf{z}_L) = \sigma(W_i^{(L+1)} \mathbf{z}_L + \mathbf{b}_i^{(L+1)})$ ; (3) For real-valued output vectors  $Y$ , we use Gaussian distribution,  $p(Y | \mathbf{z}_L) = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu} = W^{(L+1)} \mathbf{z}_L + \mathbf{b}^{(L+1)}, \boldsymbol{\sigma}^2)$ . The conditional distribution  $p(\mathbf{y} | \mathbf{x})$  from the model is computed using the Bayes' rule and the Markov assumption (Equation 5.2) in **PIBs**<sup>1</sup>:

$$p(\mathbf{y} | \mathbf{x}) = \int p(\mathbf{y}, \mathbf{z} | \mathbf{x}) d\mathbf{z} = \int p(\mathbf{y} | \mathbf{z}) p(\mathbf{z} | \mathbf{x}) d\mathbf{z} = \int \prod_{l=1}^{L+1} p(\mathbf{z}_l | \mathbf{z}_{l-1}) d\mathbf{z} \quad (5.5)$$

<sup>1</sup>Here we use integral  $\int$  even for discrete-valued variables instead of sum  $\Sigma$  for denotation simplicity.



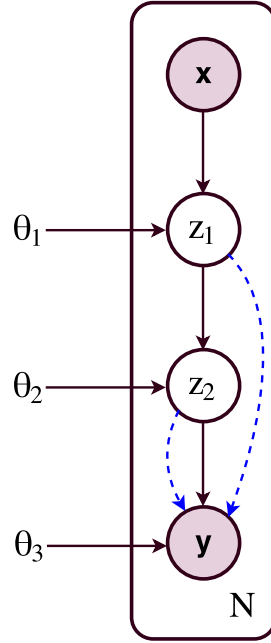


Figure 5.1: A directed graphical representation of a **PIB** of two bottlenecks. The neural network parameters  $\theta = (\theta_1, \theta_2, \theta_3)$ . The dashed blue arrows do not denote variable dependencies but the relevance decoders for each bottleneck. The relevance decoder  $p_{true}(\mathbf{y}|\mathbf{z}_i)$ , which is uniquely determined given the encoder  $p_{\theta}(\mathbf{z}_i|\mathbf{x})$  and the joint distribution  $p_D(\mathbf{x}, \mathbf{y})$ , is intractable. We use  $p_{\theta}(\mathbf{y}|\mathbf{z}_i)$  as a variational approximation to each intractable relevance decoder  $p_{true}(\mathbf{y}|\mathbf{z}_i)$ .

where  $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L)$  is the entire sequence of hidden layers in the neural network. Note that for a given joint distribution  $p_D(\mathbf{x}, \mathbf{y})$ , the relevance decoder  $p_{true}(\mathbf{y}|\mathbf{z}_l)$  is uniquely determined if an encoding function  $p(\mathbf{z}_l|\mathbf{x})$  is defined. Specifically, the relevance decoder is determined as follows:

$$p_{true}(\mathbf{y}|\mathbf{z}_l) = \int p_D(\mathbf{x}, \mathbf{y}) \frac{p(\mathbf{z}_l|\mathbf{x})}{p(\mathbf{z}_l)} d\mathbf{x} \quad (5.6)$$

It is also important to note that many stochastic neural networks have been proposed (e.g., [29], [30], [31], [32], [33]). However, our motivation for this stochasticity is that it enables sampling of bottleneck variables given the data variables  $(X, Y)$ . The generated bottleneck samples are then used to estimate mutual information. Thus, our framework does not depend on a specific stochastic model. Furthermore, deterministicity makes estimation of mutual information harder. In deterministic neural networks, we only have one sample of hidden variables given one data point. Thus, estimating mutual information for hidden variables in this case is as hard as estimating mutual information for the data variables themselves.

## 5.2.2 Learning Principle

Since the neural network is a lossy representation of the original data space, a learning principle should make this loss in a beneficial manner. Specifically in **PIBs**, we propose to jointly compress the network's intermediate spaces and preserve relevant information simultaneously at all layers of the network. For

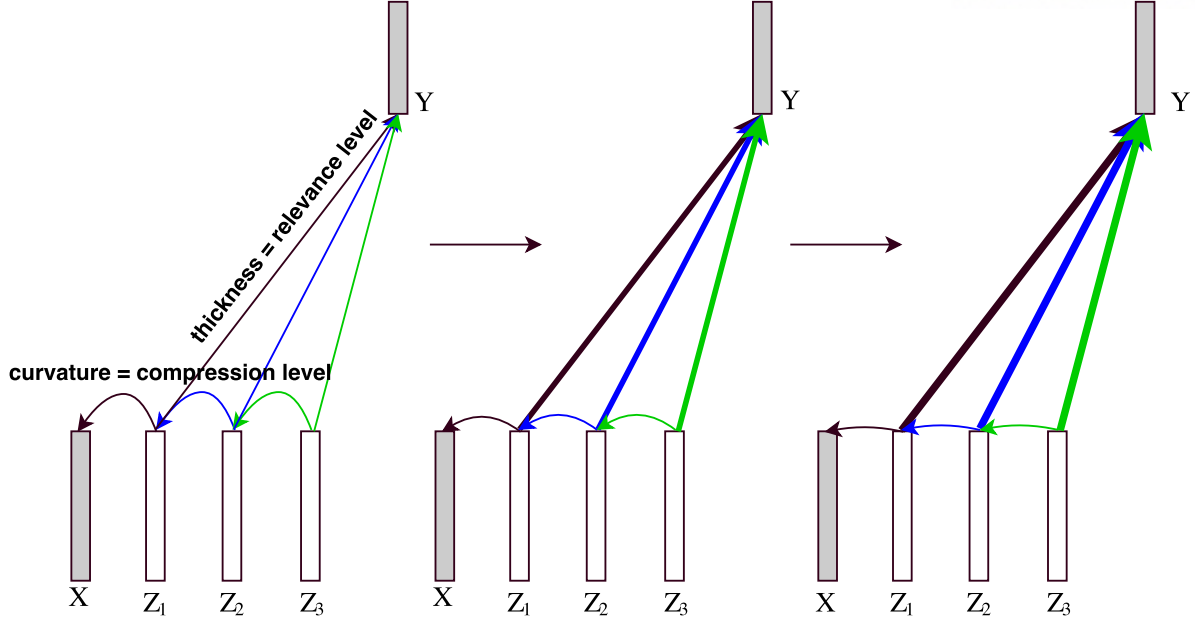


Figure 5.2: Minimizing  $J_{PIB}$  can be intuitively interpreted as tightening the information knots of a neural network architecture. Here a curvature of the curve connecting two consecutive layers represents compression while the thickness of the string connecting  $Z_l$  to  $Y$  indicates relevance level.

the  $l^{\text{th}}$ -level bottleneck  $Z_l$ , the compression is defined as the mutual information between  $Z_l$  and the previous-level bottleneck  $Z_{l-1}$  while the relevance is specified as its mutual information with the target variable  $Y$ . We explicitly define the learning objective for **PIB** as:

$$\mathcal{L}_{PIB}(Z) := \mathcal{L}_{PIB}(\boldsymbol{\theta}) := \sum_{l=0}^L [\beta_l^{-1} I(Z_l, Z_{l-1}) - I(Z_l, Y)] \quad (5.7)$$

where the layer-specific Lagrangian multiplier  $\beta_l^{-1}$  controls the tradeoff between layer complexity and predictive power in each bottleneck, and the concept of compression and relevance is taken to the extreme when  $l = 0$  (with convention that  $I(Z_0, Z_{-1}) = I(X, \emptyset) = H(X) = \text{constant}$ ). Here we prefer to this extreme, i.e., the  $0^{\text{th}}$  level, as the *super* level. While the  $l^{\text{th}}$  level for  $1 \leq l \leq L$  indicates a specific hidden layer  $l$ , the super level represents the entire neural network as a whole.

Specially, the PIB objective can be consider as a joint version of the theoretical analysis in [15]. A special notice is that the relevance terms in our PIB objective is equally weighted across different layers. A possible extension to our PIB objective is to weight the information terms of different layers but this is out of the scope of this work for now.

**Definition 5.2.1.** (The super level) The  $0^{\text{th}}$  level in a neural network is referred to as the *super* level.

Optimizing **PIBs** now becomes the minimization of  $\mathcal{L}_{PIB}(Z)$  which attempts to decrease  $I(Z_l, Z_{l-1})$  and increase  $I(Z_l, Y)$  simultaneously. The decrease of  $I(Z_l, Z_{l-1})$  makes the representation at the  $l^{\text{th}}$ -level more compressed while the increase of  $I(Z_l, Y)$  promotes the preservation of relevant information in  $Z_l$

about  $Y$ . An intuitive meaning of minimizing  $J_{PIB}$  is presented in Figure 5.2 as tightening the “information knots” of a neural network architecture at all levels (including the super level). That is, reducing the PIB objective is intuitively as gradually tightening the information with the hidden layers guided by the relevant information with regard to the target variable. In optimization’s aspect, the minimization of  $\mathcal{L}_{PIB}$  is much harder than the minimization of  $\mathcal{L}_{IB}$  since  $\mathcal{L}_{PIB}$  involves dependent terms that even the self-consistent equations of the IB framework are not applicable to this case. Furthermore,  $\mathcal{L}_{PIB}$  is intractable since the bottleneck spaces are usually high-dimensional and the relevance encoders  $p_{true}(\mathbf{y}|\mathbf{z}_l)$  (computed by Equation 5.6) are intractable. In the following section, we present our approximation to  $\mathcal{L}_{PIB}$  which fully utilizes the existing architecture without resorting to any model that is not part of the considered neural network. This approximation then leads to an effective gradient-based training of PIBs.

## 5.3 Approximate learning

Here, we present our approximations to the relevance and the compression terms in the PIB objective  $\mathcal{L}_{PIB}$ . We use variational methods for the relevance terms while simply rely on Monte Carlo sampling for estimating the compression terms. Key derivations and important propositions that accompany the approximations are also presented.

### 5.3.1 Approximate Relevance

Since the relevance decoder  $p_{true}(\mathbf{y}|\mathbf{z}_l)$  (Equation 5.6) is intractable, we cannot compute mutual information  $I(Y, Z_l)$  exactly. Instead we use a *variational* relevance decoder  $p_v(\mathbf{y}|\mathbf{z}_l)$  to approximate the intractable relevance decoder. In this variational approximation, we 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. Specifically, we firstly decompose the mutual information into the difference of two entropies:

$$I(Z_l, Y) = H(Y) - H(Y|Z_l) \quad (5.8)$$

where  $H(Y) = \text{constant}$  can be ignored in the minimization of  $\mathcal{L}(Z)$ , and

$$H(Y|Z_l) = - \int p(\mathbf{y}, \mathbf{z}_l) \log p_{true}(\mathbf{y}|\mathbf{z}_l) d\mathbf{y}d\mathbf{z}_l \quad (5.9)$$

$$= - \int p(\mathbf{z}_l) p_{true}(\mathbf{y}|\mathbf{z}_l) \log p_{true}(\mathbf{y}|\mathbf{z}_l) d\mathbf{y}d\mathbf{z}_l \quad (5.10)$$

$$= - \int p(\mathbf{z}_l) p_{true}(\mathbf{y}|\mathbf{z}_l) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{y}d\mathbf{z}_l - \int p(\mathbf{z}_l) D_{KL}[p_{true}(\mathbf{y}|\mathbf{z}_l) || p_v(\mathbf{y}|\mathbf{z}_l)] d\mathbf{z}_l \quad (5.11)$$

$$\leq - \int p(\mathbf{z}_l) p_{true}(\mathbf{y}|\mathbf{z}_l) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{y}d\mathbf{z}_l \quad (5.12)$$

$$= - \int p(\mathbf{y}, \mathbf{z}_l) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{y}d\mathbf{z}_l \quad (5.13)$$

$$= - \int p(\mathbf{x}, \mathbf{y}, \mathbf{z}_l) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{y}d\mathbf{z}_l d\mathbf{x} \quad (5.14)$$

$$= - \int p_D(\mathbf{x}, \mathbf{y}) p(\mathbf{z}_l|\mathbf{x}, \mathbf{y}) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{y}d\mathbf{z}_l d\mathbf{x} \quad (5.15)$$

$$= - \int p_D(\mathbf{x}, \mathbf{y}) p(\mathbf{z}_l|\mathbf{x}) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{z}_l d\mathbf{x} d\mathbf{y} \quad (\text{due to the Markov assumption 5.2}) \quad (5.16)$$

$$= - \mathbb{E}_{p_D(\mathbf{x}, \mathbf{y})} [\mathbb{E}_{p(\mathbf{z}_l|\mathbf{x})} [\log p_v(\mathbf{y}|\mathbf{z}_l)]] =: \tilde{H}(Y|Z_l) \quad (5.17)$$

where the equality holds **iff**  $p_{true}(\mathbf{y}|\mathbf{z}_l) = p_v(\mathbf{y}|\mathbf{z}_l), \forall \mathbf{y} \in \mathcal{Y}, \mathbf{z}_l \in \mathcal{Z}_l$ . In **PIBs**, we propose using the higher-level part of the existing network architecture at each layer to define the variational relevance encoder for that layer, i.e.,  $p_v(\mathbf{y}|\mathbf{z}_l) = p(\mathbf{y}|\mathbf{z}_l)$  where  $p(\mathbf{y}|\mathbf{z}_l)$  is determined by the network architecture. In this case, we have:

$$p_v(\mathbf{y}|\mathbf{z}_l) = p(\mathbf{y}|\mathbf{z}_l) = \int \prod_{i=l}^{L+1} p(\mathbf{z}_{i+1}|\mathbf{z}_i) d\mathbf{z}_L \dots d\mathbf{z}_{L+1} = \mathbb{E}_{p(\mathbf{z}_L|\mathbf{z}_l)} [p(\mathbf{y}|\mathbf{z}_L)] \quad (5.18)$$

In other words, the intractable relevance decoders  $p_{true}(\mathbf{y}|\mathbf{z}_l)$  is approximated by the distributions  $p(\mathbf{y}|\mathbf{z}_l)$  which is defined by the network architecture. Reducing  $\tilde{H}(Y|Z_l)$  has a meaning as reducing an upper bound of  $H(Y|Z_l)$  in hope that  $H(Y|Z_l)$  is reduced as well. More importantly,  $\tilde{H}(Y|Z_l)$  involves only the neural network's parameters thus helps optimizing the neural network explicitly using the **IB** mechanism. We will refer to  $\tilde{H}(Y|Z_l)$  as the *variational conditional relevance* for the  $l^{\text{th}}$ -level bottleneck variable  $Z_l$  for the rest of this work.

**Definition 5.3.1. (Variational Conditional Relevance (VCR))** The variational conditional relevance for  $l^{\text{th}}$ -level bottleneck variable  $Z_l$  is defined as  $\tilde{H}(Y|Z_l)$ .

What follow are two important results which indicate that the relevance terms in our objective is closely and mutually related to the concept of the **MLE** principle.

**Proposition 5.3.1.** The **VCR** at the super level (i.e.,  $l = 0$ ) equals the **negative log-likelihood (NLL)** function.

**Proposition 5.3.2.** The **VCR** at the highest-level bottleneck variable  $Z_L$  equals the **VCR** for the entire compositional bottleneck variable  $Z = (Z_1, Z_2, \dots, Z_L)$  which is an upper bound on the **NLL**. That is,

$$\tilde{H}(Y|Z_L) = \tilde{H}(Y|Z) \geq -\mathbb{E}_{p_D(\mathbf{x}, \mathbf{y})} [\log p(\mathbf{y}|\mathbf{x})] \quad (5.19)$$

While Proposition 5.3.1 is a direct result of Equation 5.18, Proposition 5.3.2 holds due to Jensen's inequality (its detail derivation in Appendix I.A).

In **PIB**'s terms, the **MLE** principle can be interpreted as a statistic mechanism that attempts to increase the **VCR** of the network as a whole. In contrast, the **PIB** objective takes into account the **VCR** at every level of the network, not just the entire network as in the **MLE** principle. In the other direction, the **VCR** can also be interpreted in terms of the **MLE** principle as well. It follows from Equation 5.17 and 5.18 that the **VCR** for layer  $l$  (including  $l = 0$ ) is the **NLL** function of  $p(\mathbf{y}|z_l)$ . Therefore, increasing the relevance components of  $J_{PIB}$  is equivalent to performing the **MLE** principle for every layer level instead of the only super level as in the standard **MLE**. Another sound interpretation is that our **PIB** framework encourages forwarding *explicit* information from all layer levels for better exploitation during learning while the **MLE** principle performs an *implicit* information forwarding by using only information from the super level. Finally, the **VCR** for a multivariate  $\mathbf{y}$  can be decomposed into the sum of that for each component of  $\mathbf{y}$  (see Appendix I.C).

### 5.3.2 Approximate Compression

The compression terms in  $\mathcal{L}_{PIB}$  involve computing mutual information between two consecutive bottlenecks. For simplicity, we present the derivation of  $I(Z_1, Z_0)$  only<sup>2</sup>. We decompose the mutual information as follows:

$$I(Z_1, Z_0) = H(Z_1) - H(Z_1|Z_0), \quad (5.20)$$

which consists of the entropy and conditional entropy term. The conditional entropy can be further rewritten as:

$$H(Z_1|Z_0) = \int p(\mathbf{z}_0) H(Z_1|Z_0 = \mathbf{z}_0) d\mathbf{z}_0 = \int p(\mathbf{z}_0) \sum_{i=1}^{N_1} H(Z_{1,i}|Z_0 = \mathbf{z}_0) d\mathbf{z}_0 \quad (5.21)$$

$$= \mathbb{E}_{p(\mathbf{z}_0)} \left[ \sum_{i=1}^{N_1} H(Z_{1,i}|Z_0 = \mathbf{z}_0) \right] \quad (5.22)$$

where  $Z_1 = (Z_{1,i})_{i=1}^{N_1}$  and  $H(Z_{1,i}|Z_0 = \mathbf{z}_0) = -q \log q - (1-q) \log(1-q)$  where  $q = p(Z_{1,i} = 1|Z_0 = \mathbf{z}_0)$ . The entropy term  $H(Z_1)$  however remains exponential in the dimensionality of  $z_1$  since  $p(\mathbf{z}_1)$  does not

<sup>2</sup>The extension at the other levels is straightforward from the derivation of  $I(Z_1, Z_0)$ .

have a closed-form representation. Here we propose resorting to empirical samples of  $\mathbf{z}_1$  generated by Monte Carlo sampling to estimate the entropy:

$$H(Z_1) = -\mathbb{E}_{p(\mathbf{z}_1)}[\log p(\mathbf{z}_1)] \approx -\frac{1}{M} \sum_{k=1}^M \log p(\mathbf{z}_1^{(k)}) =: \hat{H}_{MLE}(Z_1) \quad (5.23)$$

where

$$\mathbf{z}_1^{(k)} \sim p(\mathbf{z}_1) = \mathbb{E}_{p(\mathbf{z}_0)}[p(\mathbf{z}_1|\mathbf{z}_0)] \quad (5.24)$$

Even though  $p(\mathbf{z}_1)$  does not have a closed-form, sampling from  $p(\mathbf{z}_1)$  is made easy with Equation 5.24. This estimator is also known as the maximum likelihood estimator or ‘plug-in’ estimator ([34]). The larger number of samples  $M$  guarantees the better plug-in entropy by the following bias bound ([35])

$$|\mathbb{E}[\hat{H}_{MLE}(Z_1)] - H(Z_1)| \leq \log \left( 1 + \frac{|\mathcal{Z}_1| - 1}{M} \right) \quad (5.25)$$

where  $|\mathcal{Z}_1|$  denotes the cardinality of the space of  $Z_1$ . In practice,  $\log p(\mathbf{z}_1)$  may be numerically unstable for large cardinality  $|\mathcal{Z}_1|$ . In the large space of  $Z_1$ , the probability of a single point  $p(\mathbf{z}_1)$  may become very small that  $\log p(\mathbf{z}_1)$  become numerically unstable. To overcome this problem, we propose an upper bound on the entropy using Jensen’s inequality:

$$\log p(\mathbf{z}_1) = \log \mathbb{E}_{p(\mathbf{z}_0)}[p(\mathbf{z}_1|\mathbf{z}_0)] \geq \mathbb{E}_{p(\mathbf{z}_0)}[\log p(\mathbf{z}_1|\mathbf{z}_0)] \quad (5.26)$$

Thus,

$$H(Z_1) \leq -\mathbb{E}_{p(\mathbf{z}_1)}[\mathbb{E}_{p(\mathbf{z}_0)}[\log p(\mathbf{z}_1|\mathbf{z}_0)]] := \tilde{H}(Z_1) \quad (5.27)$$

The upper bound  $\tilde{H}(Z_1)$  is numerically stable because the conditional distribution  $p(\mathbf{z}_1|\mathbf{z}_0)$  is factorized into  $\prod_i p(z_{1,i}|\mathbf{z}_0)$ , therefore,  $\log p(\mathbf{z}_1|\mathbf{z}_0) = \sum_i \log p(z_{1,i}|\mathbf{z}_0)$  which is more stable. The upper bound  $\tilde{H}(Z_1)$  can then be estimated using Monte Carlo sampling for  $\mathbf{z}_0$  and  $\mathbf{z}_1$ .

Note that both approximate relevance and approximate compression involves integral or sum over exponential number of terms and there is no simplification. In practice, we further approximate the approximate relevance and approximate compression using Monte Carlo sampling since drawing  $\mathbf{z}_l$  from  $\mathbf{x}$  is easy in stochastic neural networks.

### 5.3.3 Approximate Gradients via Binary Bottlenecks

Discrete-valued variables in PIBs make standard back-propagation not straightforward. Fortunately, one can estimate the gradient in this case. For this problem, [31] used a Generalized EM algorithm while [36] proposed to resort to reinforcement learning. However, these estimators have high variance. In this work, we use the gradient estimator inspired by [32] for binary bottlenecks because it has low variance despite of being biased. Specifically, a bottleneck  $\mathbf{z} = (z_1, z_2, \dots, z_n)$  can be rewritten as being continuous by  $z_i = \sigma(a_i) + \varepsilon_i$  where

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

---

**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:  $\boldsymbol{\theta} \leftarrow$  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 Equations 5.17 and 5.27 to approximate  $\tilde{\mathcal{L}}_{PIB}(\boldsymbol{\theta})$
  - 8:    $\mathbf{g} \leftarrow \frac{\partial}{\partial \boldsymbol{\theta}} \tilde{\mathcal{L}}_{PIB}(\boldsymbol{\theta})$  using Raiko estimator
  - 9:    $\boldsymbol{\theta} \leftarrow$  Update parameters using the approximate gradients  $\mathbf{g}$  and SGD
  - 10: until convergence of parameters  $\boldsymbol{\theta}$
  - 11: **Output:**  $\boldsymbol{\theta}$
  - 12: **end procedure**
- 

The bottleneck component  $z_i$  defined as above still gets value of either 0 or 1 but it is decomposed into the sum of a deterministic term and a noise term. The gradient is then propagated only through the deterministic term and ignored in the noise term. A detail of gradient-based training of **PIB** is presented in Algorithm 1. A nice property of Algorithm 1 is that at each iteration we can perform Monte Carlo sampling for one single pass and use these samples to estimate the approximate relevance and compression for all levels. This property however comes with a tradeoff that it requires exponentially more samples for higher-level layers. A possible solution for this limitation is a new sampling mechanism in which a fixed number of samples are used for estimating the information terms regardless of layer levels. The details of this solution is however out of the scope of this thesis's current version.

# Chapter VI

## Experiments

This chapter presents some empirical results of our proposed **PIB** framework. Here we used the same architectures for **PIBs** and **Stochastic Feed-forward Neural Networks (SFNNs)** (e.g., [31]) and trained them on the MNIST dataset ([20]) for image classification, odd-even decision problem and multi-modal learning. Here, a **SFNN** simply prefers to feed-forward neural network models following the **MLE** principle for learning model parameters. Each hidden layer in **SFNNs** is also considered as a stochastic variable. The aforementioned tasks are to evaluate **PIBs**, as compared to **SFNNs**, in terms of generalization, learning dynamics, and capability of modeling complicated output structures, respectively. All models are implemented using Theano framework ([37]).

### 6.1 MNIST Classification

In this experiment, we compare **PIBs** with **SFNNs** and deterministic neural networks in the classification task. For comparisons, we trained **PIBs** and five additional models. The first model (Model A) is a deterministic neural network. In Model D, we used the weight trained in Model A to perform stochastic prediction at test time. Model E is **SFNN** and Model B is Model C with deterministic prediction during test phase. Model C uses the weighted trained in **PIB** but we report deterministic prediction instead of stochastic prediction for test performance.

	<b>Model</b>	<b>Mean (%)</b>	<b>Std dev.</b>
deterministic	deterministic (A)	1.73	-
	SFNN as deterministic (B)	1.88	-
	<b>PIB as deterministic (C)</b>	<b>1.46</b>	-
stochastic	deterministic as stochastic (D)	2.30	0.07
	SFNN (E)	1.94	0.036
	<b>PIB</b>	<b>1.47</b>	<b>0.034</b>

Table 6.1: The MNIST classification results of various models.

The MNIST dataset ([38]) contains a standard split of 60000, and 10000 examples of handwritten digit images for training and test, respectively in which each image is grayscale of size  $28 \times 28$  pixels. We used the last 10000 images of the training set as a holdout set for tuning hyper-parameters. The best configuration chosen from the holdout set is used to retrain the models from scratch in the full training set. The result in the test set is then reported (for stochastic prediction, we report mean and standard deviation). We scaled the images to  $[0, 1]$  and do not perform any other data augmentation. These base configurations are applied to all six models we use in this experiment.



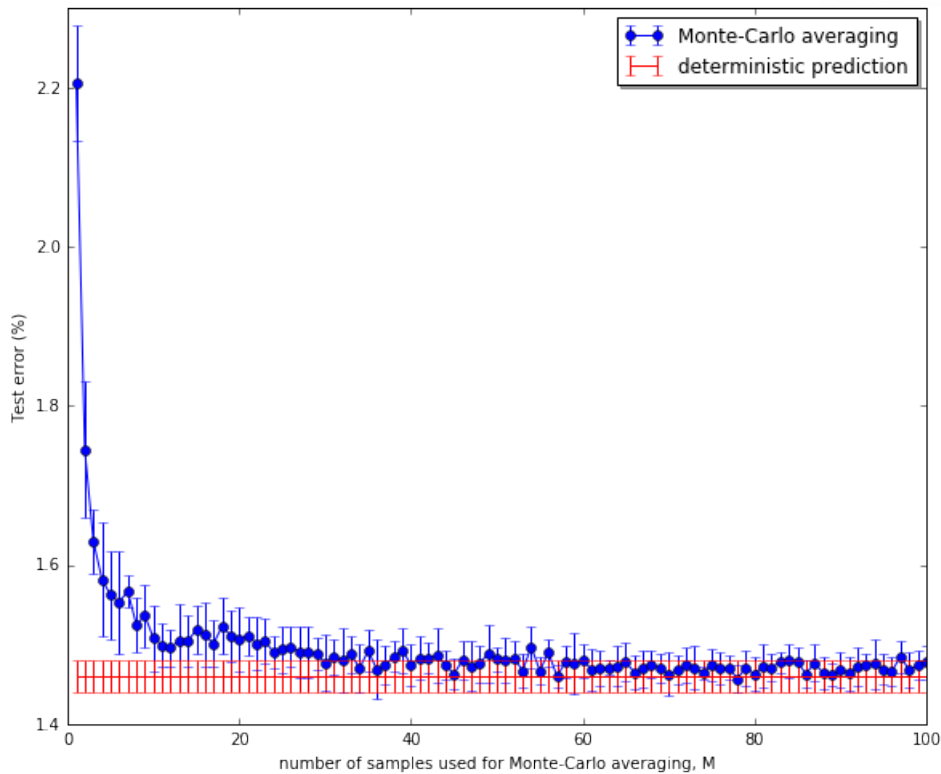


Figure 6.1: A comparison of Monte-Carlo averaging and deterministic prediction of PIB.

The base architecture is a fully-connected, sigmoid activation neural network with two hidden layers and 512 units per layer. Weights are initialized using Xavier initialization ([39]). Models were optimized with stochastic gradient descent with a constant learning rate of 0.1 and a batch size of 8. For stochastic sampling, we generate  $M = 16$  samples per point during training and  $M = 32$  samples per point during testing. For stochastic prediction, we run the prediction 10 times and report its mean and deviation standard. For PIBs, we set  $\beta_l = \beta, \forall 1 \leq l \leq L$ . We tuned  $\beta$  from  $\{0\} \cup \{10^{-i} : 1 \leq i \leq 7\}$ , and found  $\beta^{-1} = 10^{-4}$  works best.

Table 6.1 provides the results in the MNIST classification error in the test set for PIB and the comparative models (A), (B), (C), (D), and (E). As can be seen from the table, PIB and Model C gives nearly the same performance which outperform deterministic neural networks and SFNNs, and their stochastic and deterministic version.

It is interesting to empirically see that the deterministic version of PIB at test time (Model C) gives a slightly better result than PIB. This also empirically holds for the case of SFNN. To investigate more in this, we compute the test error for various values of the number of samples used for Monte-Carlo averaging,  $M$  (Figure 6.1). As we can see from the figure, the Monte-Carlo averaging of PIB obtains its good approximation around  $M = 30$  and the deterministic prediction roughly places a lower bound on the Monte-Carlo averaging at test time. Additionally, we visualize the first layer’s learned filters under different frameworks. In a standard deterministic neural network and SFNN, all layers in the network are modified in a collaborative manner to reduce the likelihood function as a whole at each iteration.

In **PIB**, on the other hand, each layer contributes to the relevance level of the entire network and the layer itself. Therefore, it is expected that a layer in an **PIB** captures more relevant information about the target variable. To observe this effect, we look at the first-level features learned by deterministic neural network, **SFNN** and **PIB** in Figure 6.2. The figure shows that **PIB** shows sharper features at many units that deterministic neural network and **SFNN** cannot learn.

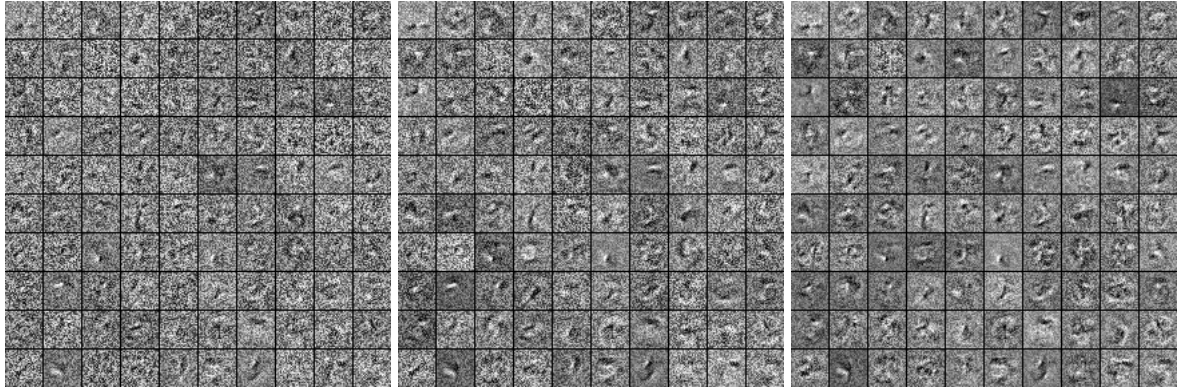


Figure 6.2: The learned weights of the first layer in MNIST classification for various models: deterministic neural networks (left), SFNN (middle), and PIB (right).

## 6.2 Learning dynamics

One way to visualize the learning dynamic of each layer of a neural network is to plot the layers in the information plane ([1], [27]). The information plane is an information-theoretic plane that characterizes any representation  $Z = Z(X)$  in terms of  $(I(Z, Y), I(Z, X))$  given the joint distribution  $I(X, Y)$ . The plane has  $I(Z, X)$  and  $I(Z, Y)$  as its horizontal axis and its vertical axis, respectively. In the general **IB** framework, each value of  $\beta$  specifies a unique point of  $Z$  in the information plane. As  $\beta$  varies from 0 to  $\infty$ ,  $Z$  traces a concave curve, known as information curve for representation  $Z$ , with a slope of  $\beta^{-1}$ . The information-theoretic goal of learning a representation  $Z = Z(X)$  is therefore to push  $Z$  as closer to its corresponding optimal point in the information curve as possible. For multi-layered neural networks, each hidden layer  $Z_l$  is a representation that can also be quantified in the information plane.

In this experiment, we considered an odd-even decision problem in the MNIST dataset in which the task is to determine if the digit in an image is odd or even. We used the same neural network architecture of 784-10-10-10-1 for **PIB** and **SFNN** and trained them with **Stochastic Gradient Descent (SGD)** with constant learning rate of 0.01 in the first 50000 training samples. For **PIB**, we use  $\beta_l^{-1} = \beta^{-1} = 10^{-4}$ . Since the network architecture is small, we can compute mutual information  $I_x := I(Z_i, X)$  and  $I_y := I(Z_i, Y)$  precisely and plot them over training epochs.

As indicated by Figure 6.3, 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

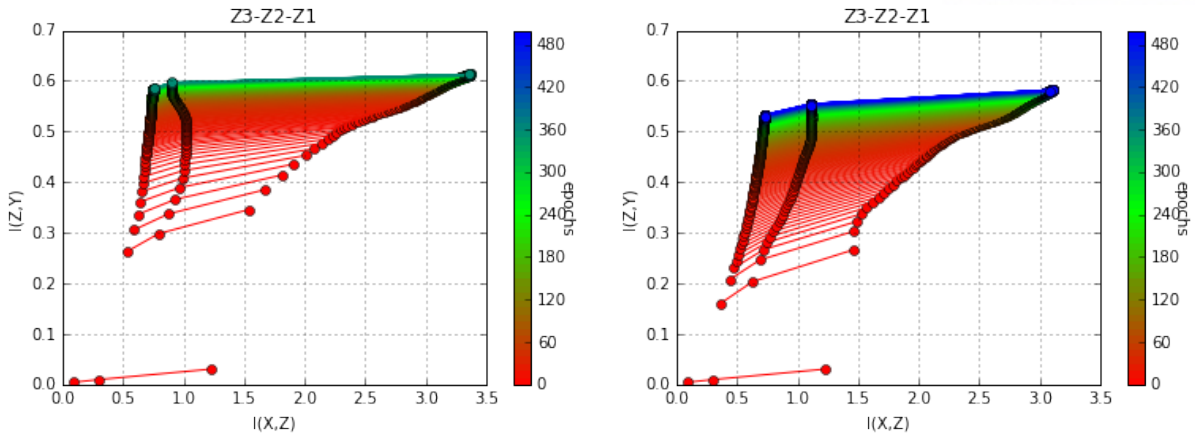


Figure 6.3: The learning dynamic of PIB (left) and SFNN (right) in a decision problem are presented in the information plane (the  $\log$  function is in the natural base  $e$ ). Each point represents a hidden layer while the color indicate epochs. Because of the Markov assumption (Equation 5.2), we have  $H(X) \geq I(Z_i, X) \geq I(Z_{i+1}, X)$  and  $I(X, Y) \geq I(Z_i, Y) \geq I(Z_{i+1}, Y)$ .

information (it reaches higher  $I(Z, Y)$  but in lesser number of epochs) while keeps the layers concise at higher epochs. The SFNN, on the other hand, encodes information in a way that matches the model distribution to the empirical data distribution. As a result, it may encode irrelevant information that hurts the generalization.

We also evaluate the learning dynamics of PIB in terms of classification errors in Figure 6.4. Both the training error curve and the validation error curve of PIB lie below its corresponding curve of SFNN. It indicates that our PIB framework exploits the neural network’s representation faster. We hypothesize that this faster convergence property is due to our explicit information encoding and compression in the PIB objective. At each iteration, while the SFNN is trying to match the model distribution to the empirical distribution, the PIB framework encourages a compressed yet informative representation at every level of the neural network. As a result, the neural network under the PIB framework somehow captures better the regularities in the data distribution that improve generalization.

### 6.3 Multi-modal learning

As PIB and SFNN are stochastic neural networks, they can model structured output space in which a one-to-many mapping is required. A binary stochastic variable  $\mathbf{z}_l$  of dimensionality  $n_l$  can take on  $2^{n_l}$  different states each of which would give a different  $\mathbf{y}$ . This is the reason why the conditional distribution  $p(\mathbf{y}|\mathbf{x})$  in stochastic neural networks is multi-modal.

In this experiment, we followed [32] and predicted the lower half of the MNIST digits using the upper half as inputs. We used the same neural network architecture of 392-512-512-392 for PIB and SFNN and trained them with SGD with constant learning rate of 0.01. We trained the models in the full training set of 60000 images and tested in the test set. For PIB, we also used  $\beta_l^{-1} = \beta^{-1} = 10^{-4}$ . The visualization in Figure 6.5 indicates that PIB models the structured output space better and faster (using

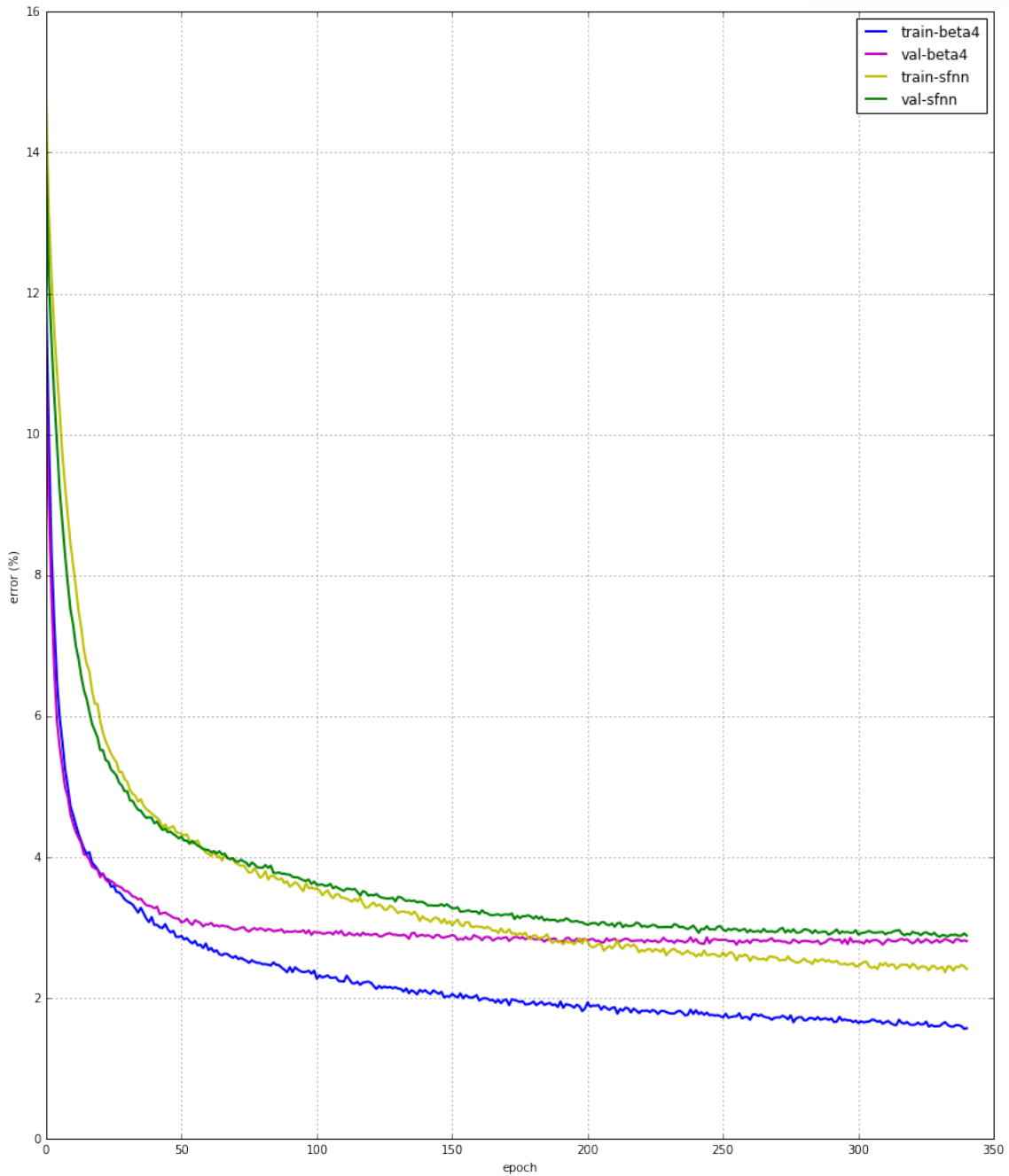


Figure 6.4: Classification errors during training and validation of PIB ( $\beta^{-1} = 10^{-4}$ ) and SFNN.

lesser number of epochs) than SFNN. The samples generated by PIB is totally recognizable while the samples generated by SFNN shows some discontinuity (e.g., digit 2,4,5,7) and confusion (e.g., digit 3 confuses with number 8, digit 5 is unrecognizable or confuses with number 6, digit 8 and 9 are unrecognizable).

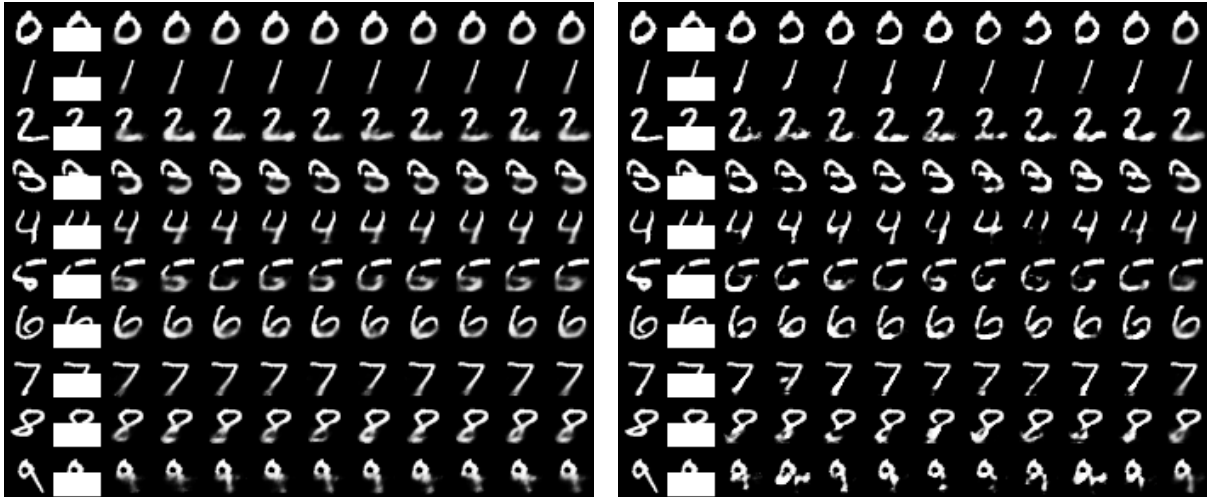


Figure 6.5: Samples drawn from the prediction of the lower half of the MNIST test data digits based on the upper half for PIB (left, after 60 epochs) and SFNN (right, after 200 epochs). The leftmost column is the original MNIST test digit followed by the masked out digits and nine samples. The rightmost 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.

## 6.4 Additional Experiment: Structure Analysis

It is also interesting to analyze how expanding <sup>1</sup> network architecture affects its learning and performance. The expansion can either in vertical direction (Figure 6.6, 6.7 and 6.8), i.e., changing the number of units within a layer, or horizontal direction, i.e., changing the number of hidden layers. For vertical expansion, it can be interpreted as changing the cardinality of the code space but in an effective way: exponentially. Increasing a code space's cardinality means allow more room for encoding more information. To extract relevant information from the original data space, the code space needs to have enough room for such relevant information. For this experiment, we use the same network architecture of 784-10-10-10-1 as a base one and decrease the number of units within a layer by 2. As we shrink the second layer, it gets closer to the third layer in the information plane. This can be explained as follows. Since the code space of the second layer gets smaller, it is easier for the third layer to encode almost all of information from the second layer's code space. In case of PIB, they even get closer as we explicitly reduce the mutual information between the second and the third layer.

<sup>1</sup>Here we use "expansion" to mean both expanding and shrinking.



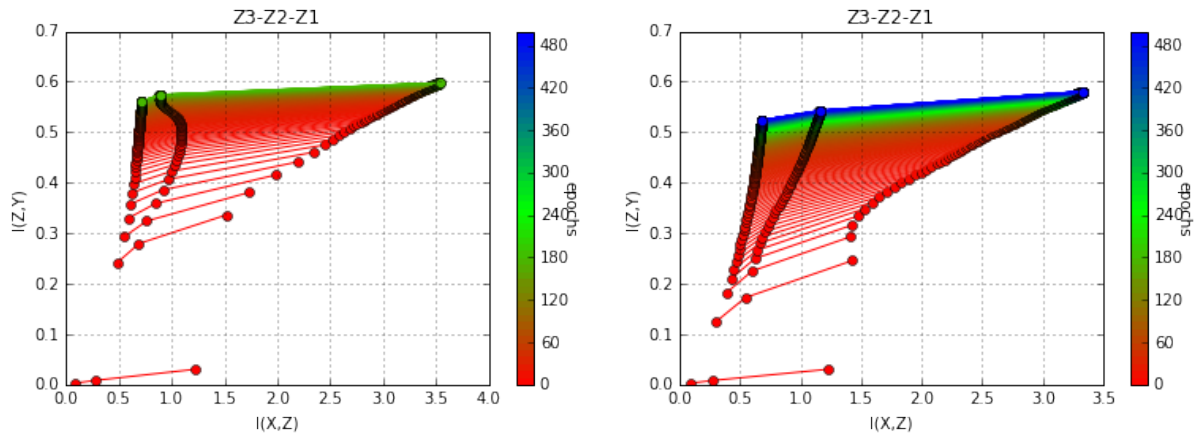


Figure 6.6: The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-8-10-1.

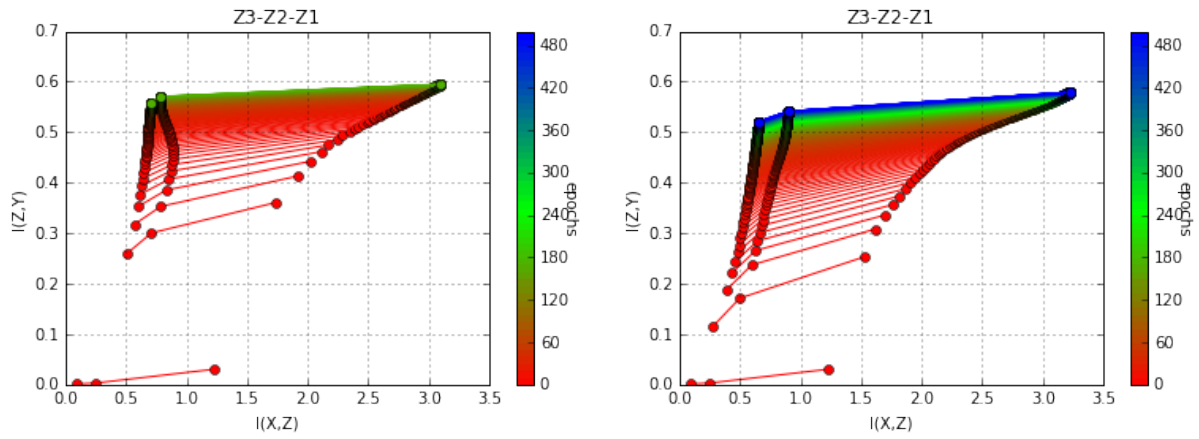


Figure 6.7: The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-6-10-1.

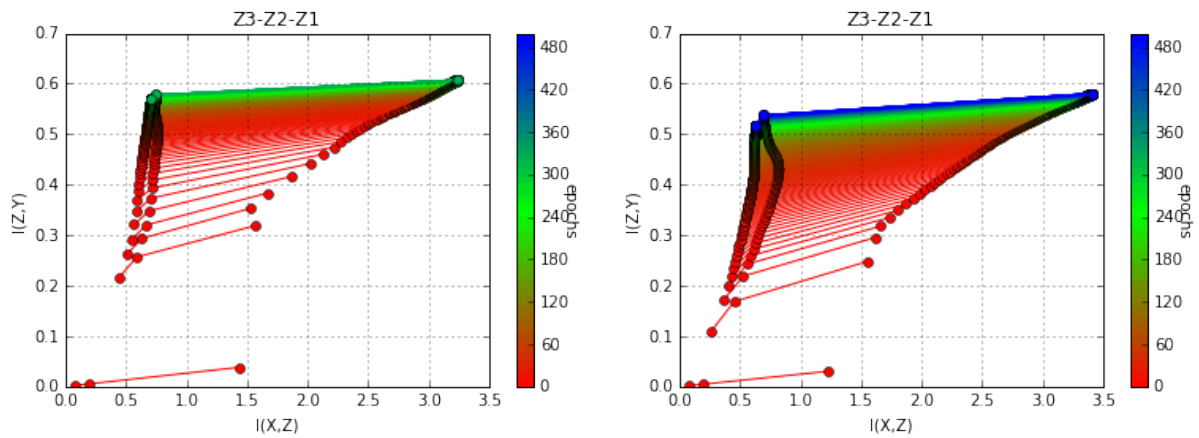


Figure 6.8: The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-4-10-1.

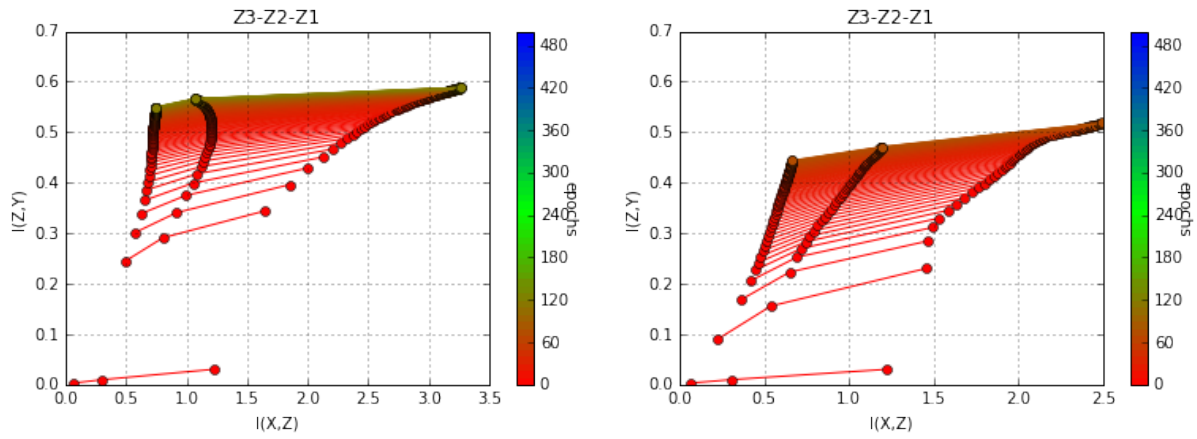


Figure 6.9: The learning dynamic of PIB (left) and MLE (right) in a decision problem in 10-10-8.

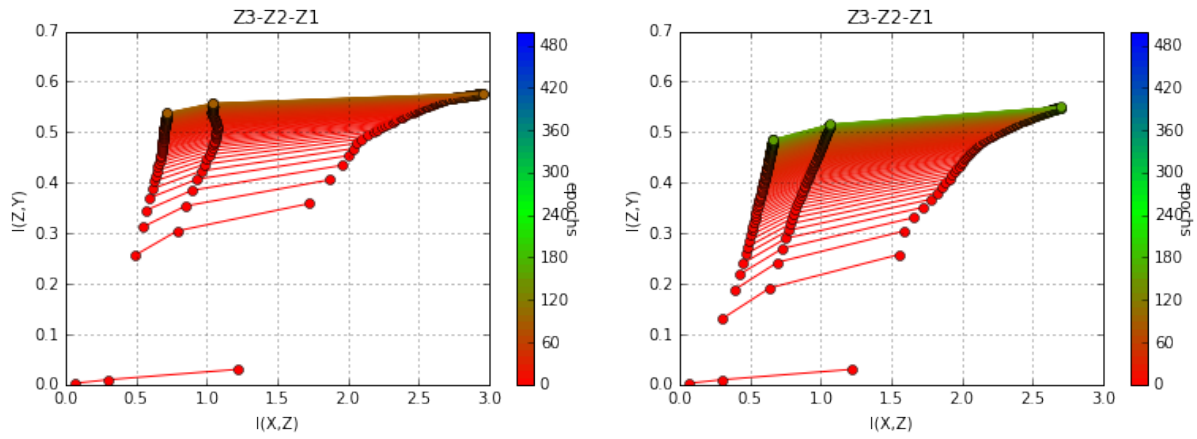


Figure 6.10: The learning dynamic of PIB (left) and MLE (right) in a decision problem in 10-10-6.

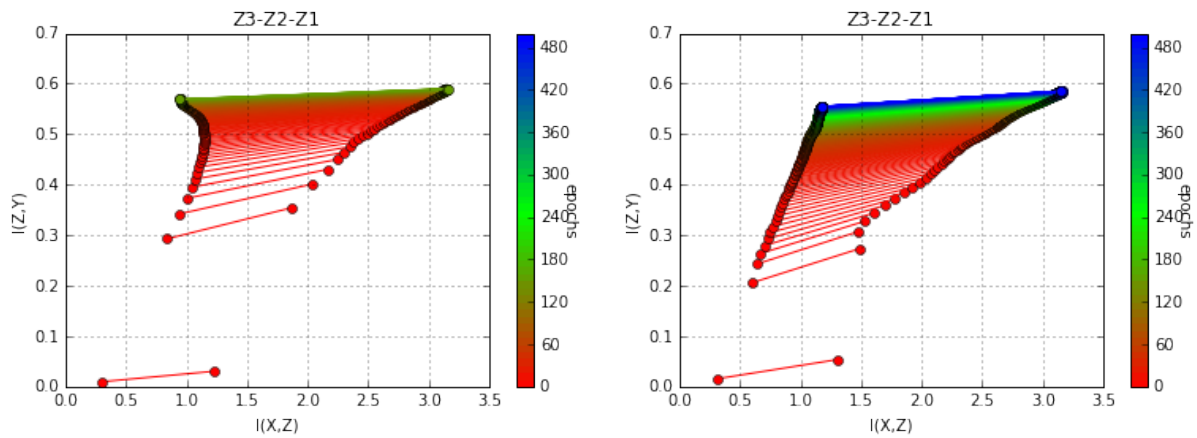


Figure 6.11: The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-10-1.

# Chapter VII

## Conclusion

In this chapter, we summarize our contributions and discuss limitations in our proposed algorithm. Followed that, we discuss our future work towards extending our framework to larger neural network architecture and developing an analysis tool for architecture diagnosis.

### 7.1 Summary

Here we summarize our contributions and achievements. In this thesis, we

- provided arguments about inefficiency of **MLE** principle for learning neural networks and encouraged a re-thinking of a new learning principle that is specifically tailored for neural networks;
- introduced a principled interpretation of multi-layered architecture of neural networks and an information-theoretic learning framework, **PIB**, to better exploit a neural network's representation by explicitly considering representation complexity and predictive power for every layer;
- proposed an approximation that fully utilizes all parameters in a neural network and does not resort to any extra models, followed by an efficient gradient-based algorithm, the first algorithm that learns all parameters using Information Bottleneck principle;
- supported the effectiveness and robustness of our PIB with the qualitative empirical results.

### 7.2 Discussion

We address here some limitations in our current work and some potential future directions that we can extend our ideas.

#### 7.2.1 Limitation

- The sampling mechanism in our proposed gradient-based algorithm 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.



### 7.2.2 Future work

- Since we used generated samples to estimate mutual information, we can potentially extend the learning framework to larger and more complicated neural network architectures. This work is our first step toward exploiting expressive power of large neural networks using information-theoretic perspective that is not yet fully utilized.
- Our framework incorporated information theory framework to a neural network architecture and learned all its parameters according to the complexity-predictiveness tradeoff for each layer; therefore it holds a great potential of analyzing neural network architectures (e.g, detecting which neuron is redundant in terms of the amount of information it preserves so that we can decide if it is beneficial to prune the neuron).

For further discussion, please catch me at <http://thanhnguyentang.com> or [thanhnguyen2792@gmail.com](mailto:thanhnguyen2792@gmail.com).

## Acknowledgements

I would like to express my gratitude for:

- My parents, Tan Nguyen and Tung Truong, for their unconditional love, trust and support for my exciting career journey;
- My supervisor, Jaesik Choi, for many fruitful discussions and valuable advice towards the completion of this work as well as about personal and professional life;
- My colleagues from Statistical Artificial Intelligence Laboratory (SAIL), for the helpful comments and discussions, especially Dr. Kallol Roy, Rafael de Lima and Anh Tong;
- My wonderful friends in Korea and Vietnam for their encourage, nice manner and endless laughs;

and all research fundings that are supporting my study in UNIST.

## References

- [1] N. Tishby, F. C. Pereira, and W. Bialek, “The information bottleneck method,” *In Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, 1999.
- [2] T. M. Cover and J. A. Thomas, *Elements of information theory*. Wiley Series in Telecommunications and Signal Processing, 2006.
- [3] Andrew Ng, “Multi-Layer Neural Network,” <http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/>.
- [4] B. Efron and T. Hastie, *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*, 1st ed. New York, NY, USA: Cambridge University Press, 2016.
- [5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, 2012, pp. 1106–1114. [Online]. Available: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>
- [6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 2015, pp. 1–9. [Online]. Available: <https://doi.org/10.1109/CVPR.2015.7298594>
- [7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.
- [8] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. [Online]. Available: <https://doi.org/10.1038/nature16961>
- [10] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2670313>
- [11] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine*

- Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 448–456. [Online]. Available: <http://jmlr.org/proceedings/papers/v37/ioffe15.html>
- [12] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [13] S. Kullback and R. Leibler, “On information and sufficiency,” *Annals of Mathematical Statistics*, vol. 22 (1), p. 79–86, 1951.
- [14] N. Slonim and Y. Weiss, “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]*, 2002, pp. 335–342. [Online]. Available: <http://papers.nips.cc/paper/2214-maximum-likelihood-and-the-information-bottleneck>
- [15] N. Tishby and N. Zaslavsky, “Deep learning and the information bottleneck principle,” pp. 1–5, 2015. [Online]. Available: <http://dx.doi.org/10.1109/ITW.2015.7133169>
- [16] G. Chechik, A. Globerson, N. Tishby, and Y. Weiss, “Information bottleneck for gaussian variables,” *Journal of Machine Learning Research*, vol. 6, pp. 165–188, 2005. [Online]. Available: <http://www.jmlr.org/papers/v6/chechik05a.html>
- [17] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, “Deep variational information bottleneck,” *CoRR*, vol. abs/1612.00410, 2016. [Online]. Available: <http://arxiv.org/abs/1612.00410>
- [18] C. E. Shannon, “A mathematical theory of communication,” *The Bell Systems Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.
- [19] K. Hornik, M. B. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8)
- [20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [21] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009. [Online]. Available: <http://dx.doi.org/10.1109/TPAMI.2008.137>
- [22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Neurocomputing: Foundations of research,” J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Learning Representations by Back-propagating Errors, pp. 696–699. [Online]. Available: <http://dl.acm.org/citation.cfm?id=65669.104451>
- [23] Andrew Ng, “Backpropagation Algorithm,” [http://deeplearning.stanford.edu/wiki/index.php/Backpropagation\\_Algorithm](http://deeplearning.stanford.edu/wiki/index.php/Backpropagation_Algorithm) (April 2013).

- [24] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016. [Online]. Available: <http://www.deeplearningbook.org/>
- [25] P. Grünwald, “A tutorial introduction to the minimum description length principle,” *CoRR*, vol. math.ST/0406077, 2004. [Online]. Available: <http://arxiv.org/abs/math.ST/0406077>
- [26] O. Shamir, S. Sabato, and N. Tishby, “Learning and generalization with the information bottleneck,” *Theor. Comput. Sci.*, vol. 411, no. 29-30, pp. 2696–2711, 2010. [Online]. Available: <https://doi.org/10.1016/j.tcs.2010.04.006>
- [27] N. Slonim, “Information bottleneck theory and applications,” *PhD thesis, Hebrew University of Jerusalem*, 2003.
- [28] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778. [Online]. Available: <https://doi.org/10.1109/CVPR.2016.90>
- [29] R. M. Neal, “Learning stochastic feedforward networks,” 1990.
- [30] —, “Connectionist learning of belief networks,” 1992, pp. 71–113.
- [31] Y. Tang and R. Salakhutdinov, “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.*, 2013, pp. 530–538. [Online]. Available: <http://papers.nips.cc/paper/5026-learning-stochastic-feedforward-neural-networks>
- [32] T. Raiko, M. Berglund, G. Alain, and L. Dinh, “Techniques for learning binary stochastic feedforward neural networks,” *CoRR*, vol. abs/1406.2989, 2014. [Online]. Available: <http://arxiv.org/abs/1406.2989>
- [33] Y. N. Dauphin and D. Grangier, “Predicting distributions with linearizing belief networks,” *ICLR*, 2016.
- [34] A. Antos and I. Kontoyiannis, “Convergence properties of functional estimates for discrete distributions,” 2001, p. 163–193.
- [35] L. Paninski, “Estimation of entropy and mutual information,” *Neural Computation*, vol. 15, no. 6, pp. 1191–1253, 2003. [Online]. Available: <https://doi.org/10.1162/089976603321780272>
- [36] Y. Bengio, N. Léonard, and A. C. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” *CoRR*, vol. abs/1308.3432, 2013. [Online]. Available: <http://arxiv.org/abs/1308.3432>

- [37] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. Bleicher Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. de Brébisson, O. Breuleux, P.-L. Carrier, K. Cho, J. Chorowski, P. Christiano, T. Cooijmans, M.-A. Côté, M. Côté, A. Courville, Y. N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. Ebrahimi Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. Goodfellow, M. Graham, C. Gulcehre, P. Hamel, I. Harlouchet, J.-P. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov, V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrancois, S. Lemieux, N. Léonard, Z. Lin, J. A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P.-A. Manzagol, O. Mastropietro, R. T. McGibbon, R. Memisevic, B. van Merriënboer, V. Michalski, M. Mirza, A. Orlandi, C. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. Schlüter, J. Schulman, G. Schwartz, I. V. Serban, D. Serdyuk, S. Shabanian, E. Simon, S. Spieckermann, S. R. Subramanyam, J. Sygnowski, J. Tanguay, G. van Tulder, J. Turian, S. Urban, P. Vincent, F. Visin, H. de Vries, D. Warde-Farley, D. J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, and Y. Zhang, “Theano: A Python framework for fast computation of mathematical expressions,” *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: <http://arxiv.org/abs/1605.02688>
- [38] B. L. e. B. Y. . H. P. LeCun, Yann, “Gradient-based learning applied to document recognition,” 1998, p. 2278–2324.
- [39] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, 2010, pp. 249–256. [Online]. Available: <http://www.jmlr.org/proceedings/papers/v9/glorot10a.html>

## Appendix I

### A. Proof of the Propositions

**Proof of the Proposition 2:** It follows from the Markov chain assumption (5.2) that  $p(\mathbf{y}|\mathbf{z}) = p(\mathbf{y}|\mathbf{z}_L, \mathbf{z}_{L-1}, \dots, \mathbf{z}_1) = p(\mathbf{y}|\mathbf{z}_L)$  and from Jensen's inequality that

$$\int p(\mathbf{z}|\mathbf{x}) \log p(\mathbf{y}|\mathbf{z}) d\mathbf{z} \leq \log \left( \int p(\mathbf{z}|\mathbf{x}) p(\mathbf{y}|\mathbf{z}) d\mathbf{z} \right) = \log p(\mathbf{y}|\mathbf{x})$$

Hence, the variational compositional relevance  $\tilde{H}(Y|Z)$  equals the variational relevance for the last bottleneck and is an upper bound on the negative log-likelihood as well (Q.E.D).

### B. MLE as distribution matching

The purpose of the **MLE** principle can be interpreted as matching the model distribution to the empirical data distribution using the **KL** divergence as a measure of their discrepancy. Rigorously, given a set of samples  $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  i.i.d. drawn from some underlying data distribution  $p_D(\mathbf{x})$ , a parametric model  $p_{model}(\mathbf{x}; \boldsymbol{\theta})$  attempts to map any data sample  $\mathbf{x}$  to a real number that estimates the true probability  $p_D(\mathbf{x})$ . The **MLE** principle maximizes the likelihood function under the empirical data distribution. This in turn can be interpreted as matching the model distribution  $p_{model}$  with the data distribution  $p_D$  by minimizing their **KL** divergence to find the maximum likelihood (point) estimator for  $\boldsymbol{\theta}$ :

$$\boldsymbol{\theta}_{ML} = \arg \max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim p_D(\mathbf{x})} [\log p_{model}(\mathbf{x}; \boldsymbol{\theta})] \quad (1)$$

$$= \arg \min_{\boldsymbol{\theta}} [-\mathbb{E}_{\mathbf{x} \sim p_D(\mathbf{x})} [\log p_{model}(\mathbf{x}; \boldsymbol{\theta})] + \mathbb{E}_{\mathbf{x} \sim p_D(\mathbf{x})} [\log p_D(\mathbf{x}; \boldsymbol{\theta})]] \quad (2)$$

$$= \arg \min_{\boldsymbol{\theta}} D_{KL} [p_D(\mathbf{x}) || p_{model}(\mathbf{x}; \boldsymbol{\theta})] \quad (3)$$

$$\approx \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^N \log p_{model}(\mathbf{x}_i; \boldsymbol{\theta}) \quad (4)$$

where expression (4) is an empirical estimation of expression (1) for N datapoints.

### C. Variational relevance for multivariate target variable

The **VCR** at level  $l$  (defined by (5.17), (5.18)) for a multivariate variable  $\mathbf{y}$  can be decomposed into the variational conditional relevances for each of its components. Indeed, without loss of generality, we assume bivariate target variable  $\mathbf{y} = (y_1, y_2)$ . It follows from the fact that the neurons within a layer are

independent given some previous layer that we have

$$\tilde{H}(Y|Z_l) = -\mathbb{E}_{p_D(\mathbf{x}, y_1, y_2)} [\mathbb{E}_{p(\mathbf{z}_l|\mathbf{x})} [\log p(y_1, y_2|\mathbf{z}_l)]] \quad (5)$$

$$= -\mathbb{E}_{p_D(\mathbf{x}) p_D(y_1, y_2|\mathbf{x})} [\mathbb{E}_{p(\mathbf{z}_l|\mathbf{x})} [\log p(y_1|\mathbf{z}_l) + \log p(y_2|\mathbf{z}_l)]] \quad (6)$$

$$= \sum_i -\mathbb{E}_{p_D(\mathbf{x}) p_D(y_i|\mathbf{x})} [\mathbb{E}_{p(\mathbf{z}_l|\mathbf{x})} [\log p(y_i|\mathbf{z}_l)]] \quad (7)$$

$$= \sum_i \tilde{H}(Y_i|Z_l) \quad (8)$$