Layer-wise Learning of Stochastic Neural Networks with Information Bottleneck

Thanh T. Nguyen
School of Computer Science and Engineering
Ulsan National Institute of Science and Technology
Ulsan, Republic of Korea 44919
thanhnt@unist.ac.kr

Jaesik Choi
School of Computer Science and Engineering
Ulsan National Institute of Science and Technology
Ulsan, Republic of Korea 44919
jaesik@unist.ac.kr

Abstract

Deep neural networks (DNNs) offer flexible modeling capability for various important machine learning problems. Given the same neural modeling capability, the success of DNNs is attributed to how effectively we could learn the networks. Currently, the maximum likelihood estimate (MLE) principle has been a de-facto standard for learning DNNs. However, the MLE principle is not explicitly tailored to the hierarchical structure of DNNs. In this work, we propose the Parametric Information Bottleneck (PIB) framework as a fully information-theoretic learning principle of DNNs. Motivated by the Information Bottleneck principle, our framework efficiently induces relevant information under compression constraint into each layer of DNNs via multi-objective learning. Consequently, PIB generalizes the MLE principle in DNNs, indeed empirically exploits the neural representations better than MLE and a partially information-theoretic treatment, and offers better generalization and adversarial robustness on MNIST and CIFAR10.

1 Introduction

Deep neural networks (DNNs) have demonstrated state-of-the-art performances in several important machine learning tasks including image recognition [Krizhevsky et al. (2012)], natural language translation [Cho et al. (2014); Bahdanau et al. (2014)] and game playing [Silver et al. (2016)]. Behind the practical success of DNNs are various revolutionary techniques such as data-specific design of network architecture (e.g., convolutional neural network architecture), regularization techniques (e.g., early stopping, weight decay, dropout [Srivastava et al. (2014)], and batch normalization [Ioffe and Szegedy (2015)]), and optimization methods [Kingma and Ba (2014)]. For learning DNNs, the maximum likelihood estimate (MLE) principle (in its various forms such as maximum log-likelihood or Kullback-Leibler divergence) has generally become a de-facto standard. The MLE principle maximizes the likelihood of the model for observing the entire training data. This principle is, however, generic and not specially tailored to hierarchy-structured models like neural networks. Particularly, MLE treats the entire neural network as a collective whole without considering an explicit contribution of its hidden layers to model learning. As a result, the information contained within the hidden structure may not be adequately modified to capture the data regularities reflecting a target variable. Thus, a reasonable question to ask is whether the MLE principle effectively...
and sufficiently exploits a neural network’s representative power and whether there is any better alternative?

The Information Bottleneck (IB) principle [Tishby et al. (1999)] is an alternative principle which extracts relevant information about a target variable $Y$ from an input variable $X$ via a bottleneck variable $Z$. In detail, the IB framework constructs a bottleneck variable $Z = Z(X)$ that is a compressed version of $X$ but preserves as much relevant information in $X$ about $Y$ as possible. The compression of the representation $Z$ is quantized by $I(Z; X)$, the mutual information of $Z$ and $X$. The relevance in $Z$, the amount of information $Z$ contains about $Y$, is specified by $I(Z; Y)$. An optimal representation $Z$ satisfying a certain compression-relevance trade-off constraint is then determined via minimization of the following Lagrangian $L_{IB}[p(z|x)] = I(Z; X) - \beta I(Z; Y)$, where $\beta$ is a positive Lagrangian multiplier that controls the trade-off. Although an exact solution to this minimization problem has been found [Tishby et al. (1999)], the solution is non-parametric, implicit and non-analytical for most practical cases of interest. Thus, it is still not clear how the IB principle can be applied to many conventional DNNs in practice.

In this work, we present a fully information-theoretic learning of DNNs which solely relies on the information relayed back and forth via each layer to construct the learning process. The main principle to construct such a learning process in this paper is to utilize the IB trade-off between compression and relevance occurring at each layer. We propose to interpret the learning of (multi-layered) neural networks as layer-wise multi-objective information trade-offs in which each layer attempts to obtain its own information optimality in terms of compression and relevance. We show that the simultaneous layer-wise information optimality is unfortunately impossible in the context of DNNs; thus propose compromised optimality. In addition, the information quantity - mutual information - is highly intractable in DNNs. We, therefore, propose simple but effective approximations to mutual information using variational methods and existing network architectures.

As a result, we are thereafter able to interpret the MLE principle in DNNs as a special case of our framework applied to a specific so-called super layer while our learning principle generalizes the MLE principle to every layer under compression constraints. In practice, our fully information-theoretic framework PIB, compared to the MLE principle and a partially information-theoretic treatment, offers a superior, or at least comparative, performance in terms of generalization, and adversarial robustness on MNIST and CIFAR10. We also show that our framework does empirically indeed exploit the neural representations better in terms of information.

This work is organized as follows. We first review related literature in Section 2. Section 3 introduces our layer-wise multi-objective Information Bottleneck Principle to stochastic neural networks (presented and explained next) followed by variational bounds on mutual information and compromised optimality algorithms. Section 4 demonstrates a case study that we apply our PIB to binary stochastic neural networks. Finally, Section 5 presents the empirical results of our framework on MNIST and CIFAR10, in comparison with MLE and a partially information-theoretic treatment.

2 Related Work

One can generalize the MLE principle in DNNs with Bayesian treatments of DNNs [Neal (1995); MacKay (1992); Dayan et al. (1995)] in which the network weights are assigned a prior distribution. This approach equips DNNs with an ability to reason about their uncertainty and takes good advantage of the well-studied tools in probability theory. As a result, this idea achieved interpretability and state-of-the-art performance on many tasks [Gal (2016)]. The challenges are still the scalability to big data and how one approximates an intractable posterior in the Bayesian treatment. In our work, we take a different perspective on DNNs using Information Theory and thereby reason about the learning in terms of information contained in the neural representations. In addition, our framework is both interpretable and scalable with efficient mutual information and gradient-based algorithms.

[Tishby et al. (1999)] originally proposed the IB framework which offers a principled way of extracting the relevant information in one variable $X$ about another variable $Y$. An exact solution to the IB problem is given via highly-nonlinear self-consistent equations and can be found with the iterative Blahut Arimoto algorithm if the variables are discrete. However, the algorithm is non-parametric and not applicable to DNNs. In practice, the IB problem has been solved efficiently in the following three cases only: (1) $X$, $Y$ and $Z$ are all discrete [Tishby et al. (1999)]; (2) $X$, $Y$ and $Z$ are mutually joint Gaussian [Chechik et al. (2005)]; (3) or ($X$, $Y$, $Z$) has meta-Gaussian distributions [Rey and Roth]
where $Z$ is a bottleneck variable. In addition, the IB principle has been proven to be mathematically equivalent to the MLE principle for the multinomial mixture model for the clustering problem when the input distribution $X$ is uniform or has a large sample size Slonim (2003). It is, however, not clear how the IB principle is related to the MLE principle in the context of DNNs.

Some works have recently attempted to apply the IB principle to learn DNNs. For example, Tishby and Zaslavsky (2015) proposes the use of the 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, Tishby and Zaslavsky (2015) establishes an information-theoretic learning principle for DNNs. In principle, one can optimize a neural network by pushing up the network and all its hidden layers to the IB optimal limit in one layer after another. Although the analysis offers a new perspective on optimality in neural networks, it proposes a general analysis of optimality rather than practical optimization criteria. Furthermore, estimating the mutual information between the variables transformed by network layers and the data variables poses several computational challenges which are not addressed in this work. A small change in a multi-layered neural network could greatly modify the entropy of the input variables; it is difficult to analytically capture such modifications. Strouse and Schwab (2016); Chalk et al. (2016); Alemi et al. (2017) has overcome the intractability of mutual information in DNNs by using variational approximations. This approach however considers one single bottleneck and parameterizes the encoder $p(z|x; \theta)$ using an entire neural network. Therefore, their proposed framework still treats an entire neural network as a whole rather than explicitly optimizing it layer-wise.

3 Parametric Information Bottleneck

3.1 Stochastic Neural Network

Definition 1 (Random Variables in Neural Networks). Consider a neural network with $L$ hidden layers with no feedback or skip connection, we view the input layer $X$, the output of the $i$th hidden layer $Z_i$, and the network output layer $\hat{Y}$ as random variables (RVs). Without any feedback or skip connection, $Y, X, Z_i, Z_{i+1}$, and $\hat{Y}$ form a Markov chain in that order, denoted as:

$$Y \to X \to Z_i \to Z_{i+1} \to \hat{Y}$$ (1)
In addition, we also denote $n_l$ as the dimension of $Z_l$, use subscript $D$ to refer to the data distribution, write $X \perp Y$ (respectively, $X \not\perp Y$) to indicate $X$ and $Y$ are independent (respectively, not independent) and abuse the notation integral, e.g., $\int f(z)dz$, regardless of whether the variable $z$ is real-valued or discrete-valued.

The role of the neural network is, therefore, reduced to transforming from one RV to another via the Markov chain $X \rightarrow Z_l \rightarrow Z_{l+1} \rightarrow \hat{Y}$ where $\hat{Y}$ is used as a surrogate for $Y$. We call the transition distribution $p(Z_l|X)$ from $X$ to $Z_l$ an encoder as it encodes the data $X$ into the representation $Z_l$. For each encoder $p(Z_l|X)$, there is a unique corresponding decoder, namely relevance decoder, that decodes the relevant information in $X$ about $Y$ from representation $Z_l$:

$$p(y|z_l) = \int p_D(x,y) \frac{p(z_l|x)}{p(z_l)} \, dx$$  \hspace{1cm} (2)

Note that our view on RVs in a neural network is different from that in the Bayesian treatment of neural networks [Neal (1995), MacKay (1992), Dayan et al. (1995)] where the network weights are RVs. Furthermore, the learning of the neural network totally depends on these transformations whose outputs $\{Z_l : 1 \leq l \leq L\}$ reflect their quality for the learning.

**Definition 2 (Stochastic Neural Networks).** A neural network is said to be stochastic if there exists $l \in \{1, ..., L\}$ s.t. the mapping from $Z_l$ to $X$ is stochastic. A neural network is said to be completely stochastic if $\forall 1 \leq l \leq L$, the mapping from $Z_l$ to $X$ is stochastic.

It follows from the Markov chain in Equation (1) that inference in (either completely or partially) stochastic neural networks can be done as:

$$p(y|x) = \int p(y,z|x)dz = \int p(y|z)p(z|x)dz$$

$$= \int \prod_{l=1}^{L+1} p(z_l|z_{l-1})dz$$  \hspace{1cm} (3)

where $z = (z_1, ..., z_L)$, $Z_0 := X$ and $Z_{L+1} := \hat{Y}$.

In this work, we consider completely stochastic neural networks but refer to them as stochastic neural networks for simplicity. The reason that we do not consider any deterministic mapping is that a deterministic mapping can make an estimation of mutual information $I(Z_l;X)$ as hard as that of $H(X)$ which is unknown in most applications. For example, consider $Z_1 = g(X)$ where $g(.)$ is a deterministic function, then the mutual information becomes the entropy of the deterministic function of the data variable:

$$I(Z_1;X) = H(Z_1) - H(Z_1|X) = H(Z_1) = H(g(X)).$$

### 3.2 Multi-Objective Information Bottleneck

In the perspective that a neural network is a data-processing system that transforms one RV to another RV, we propose that each of the neural representations $Z_l$ should ideally be maximally compressed while preserving much of the relevant information in $X$ about $Y$. This notion of compression and relevance can be quantified by mutual information $I(Z_l;X)$ and $I(Z_l;Y)$, respectively. Motivated by the (non-parametric) Information Bottleneck principle [Tishby et al. (1999)], we propose the Parametric Information Bottleneck (PIB) which interprets the learning of a neural network as the following multi-objective (MO) layer-wise optimizations:

$$p^*(Z_l|X) = \arg\min_{p(Z_l|X): I(Z_l;X) \geq D_l} I(Z_l;X)$$  \hspace{1cm} (4)

where $D_l$ is some positive number $\forall 1 \leq l \leq L$. Note that $p(Z_{l+1}|X)$ partly depends on $p(Z_l|X)$. Equivalently, by introducing Lagrange multipliers $\beta_l > 0$ for the constrained relevance, we find optimal encoders $p(Z_l|X)$ for all $1 \leq l \leq L$ by minimizing the functionals:

$$L_l[p(z_l|x)] := I(Z_l;X) - \beta_l I(Z_l;Y).$$  \hspace{1cm} (5)

\(^1\)If the mapping from $X$ to $Z_l$ is deterministic, then $p(Z_l|X)$ is simply a delta function.
An important question to ask here is therefore whether there exists a single solution that simultaneously optimizes $L$ objective functionals in Equation\(^{(5)}\); otherwise, the objective functions are (said to be) conflicting. Theorem\(^{[1]}\) provides an answer to the question.

**Theorem 1** (Conflicting Information Optimality). Given $Y \rightarrow X \rightarrow Z_1 \rightarrow Z_2, Z_2 \not\perp Z_1, \beta_1 > 0$, and $\beta_2 > 0$, then $L_1$ and $L_2$ defined by Equation\(^{(2)}\) are conflicting, i.e., there does not exist a single solution that minimizes $L_1$ and $L_2$ simultaneously.

**Proof of Theorem**\(^{[1]}\). Here we briefly sketch a proof for Theorem\(^{[1]}\) which uses the contradiction technique and following two lemmas.

**Lemma 1.** $p^*(z_1|x)$ is a critical point\(^{[4]}\) of $L_1$ if and only if:

$$\frac{\partial L_1'}{\partial p^*(z_1|x)} = 0$$  \hspace{1cm} (6)

where

$$L_1' := L_1 - \int_x \lambda_1(x) \int_{z_1} p(z_1|x)dz_1dx,$$  \hspace{1cm} (7)

$$\lambda_1(x) := p(x) \left[ \log \mathbb{E}_{z_1} e^{-\beta d(x, z_1)} + \beta I(x, Y) \right],$$  \hspace{1cm} (8)

$$d(x, z_1) := D_{KL}[p(Y|x)||p(Y|z_1)].$$  \hspace{1cm} (9)

**Proof of Lemma**\(^{[7]}\). This result can be readily induced from the proof of the solutions to the IB problem presented in \cite{shiby2019} (details in Appendix).

**Lemma 2.** Given $Y \rightarrow X \rightarrow Z_1 \rightarrow Z_2$, we have

$$I(Z_2; X) = I(Z_1; X) - I(Z_1; X|Z_2)$$  \hspace{1cm} (10)

$$I(Z_2; Y) = I(Z_1; Y) - I(Z_1; Y|Z_2)$$  \hspace{1cm} (11)

Consequently,

$$L_2 = L_1 + (\beta_1 - \beta_2)I(Z_1; Y) - L_{1|Z_2}$$  \hspace{1cm} (12)

where

$$L_{1|Z_2} := I(Z_1; X|Z_2) - \beta_2 I(Z_1; Y|Z_2).$$  \hspace{1cm} (13)

**Proof of Lemma**\(^{[2]}\). $I(X; Z_1; Z_2) = I(X; Z_2) + I(X; Z_1|Z_2) = I(X; Z_1) + I(X; Z_2|Z_1)$, $I(X; Z_2|Z_1) = 0$ since $X \not\perp Z_2|Z_1$. \hspace{1cm} $\square$

Assume, by contradiction, that there exists a solution that minimizes both $L_1$ and $L_2$ simultaneously, i.e., $\exists p^*(z_1|x), p^*(z_2|z_1)$ s.t. $L_1$ has a minimum at $p^*(z_1|x)$ and $L_2$ has a minimum at $(p^*(z_1|x), p^*(z_2|z_1))$ or at $p^*(z_2|x)$ := $\int_{z_1} p^*(z_2|z_1) p^*(z_1|x)dz_1$. Thus, there exists Lagrange multipliers $\lambda_2(x)$ for the normalization of the conditional distribution $p(z_1|x)$ s.t. $L_2' := L_2 - \int_x \lambda_2(x) \int_{z_1} p(z_1|x)dz_1dx$ has a zero partial derivative at $p^*(z_1|x)$. We now rewrite Equation\(^{(12)}\) as:

$$L_2' = L_1' + (\beta_1 - \beta_2)I(Z_1; Y) - L_{1|Z_2}$$  \hspace{1cm} (14)

where

$$L_{1|Z_2} := L_{1|Z_2} - \int_x (\lambda_2(x) - \lambda_1(x)) \int_{z_1} p(z_1|x)dz_1dx,$$

and $\lambda_1(x)$ is defined in Equation\(^{(5)}\). Taking the derivative w.r.t $p^*(z_1|x)$ both sides of Equation\(^{(14)}\), noting that $p^*(z_1|x)$ is a critical point of both $L_1'$ and $L_2'$, we have:

$$\frac{\partial}{\partial p^*(z_1|x)} (\beta_1 - \beta_2)I(Z_1; Y) - L_{1|Z_2}' = 0$$  \hspace{1cm} (15)

\(^{[4]}\)A critical point of a continuous function is a point at which the function achieves its (either local or global) minimum or maximum.
However, $L'_{1,|Z_2}$ depends on $Z_2$ if $Z_2 \not\sim Z_1$; thus setting the LHS of Equation (15) to zero leads $p^*(z_1|x)$ to depend on $Z_2$ as well. This contradiction shows that the initial assumption is wrong, thus implying Theorem 1.

There are two main challenges in the MO optimizations in Equation (5): (a) Given Theorem 1, we cannot optimize each of $L$ objectives in Equation (5) simultaneously, i.e., these objectives need to compromise on their optimality to some degree; (b) the mutual information $I(Z_l;X)$ and $I(Z_l;Y)$ are intractable in neural networks. Thus, we focus on deriving efficient variational bounds on mutual information to deal with (b) and propose two simple compromised strategies to handle (a).

### 3.3 Bounds on Mutual Information

Here we propose simple but efficient variational bounds on compression and relevance for each layer. These bounds can then be efficiently optimized with gradient-based algorithms.

#### 3.3.1 Approximate Relevance

The relevance $I(Z_l;Y)$ is intractable due to the intractable relevance decoder $p(y|z_l)$ in Equation (2). It follows from Jensen’s inequality that:

$$H(Y|Z_l) = -\int p(y|z_l)p(z_l) \log p(y|z_l) dy dz_l$$

$$\leq -\int p(y|z_l)p(z_l) \log p_o(y|z_l) dy dz_l$$

$$= -E_{(x,y)\sim D} E_{z_l|x} \log p_o(y|z_l) =: \hat{H}(Y|Z_l)$$

where $p_o(y|z_l)$ is any probability distribution. Note that $I(Z_l;Y) = H(Y) - H(Y|Z_l)$ where $H(Y) = \text{constant}$ which can be ignored in the minimization of $L_l$. Specifically in PIB, we propose to use the network architecture connecting $Z_l$ to $\hat{Y}$ to define the variational relevance decoder for layer $l$, i.e., $p_o(y|z_l) = p(\hat{y}|z_l)$ where $p(\hat{y}|z_l)$ is determined by the network architecture:

$$p_o(y|z_l) := p(\hat{y}|z_l) = \int \prod_{i=l}^L p(z_{i+1}|z_i) dz_L ... dz_1$$

$$= E_{z_L|z_l} [p(\hat{y}|z_L)].$$

(17)

For the rest of this work, we refer to $\hat{H}(Y|Z_l)$ with $p_o(y|z_l) = p(\hat{y}|z_l)$ as the variational conditional relevance (VCR) of the $l^{th}$ layer. Theorem 2 addresses the relation between VCR and the MLE principle.

**Theorem 2** (Information on the extreme layers). The VCR of the lowest-level (so-called super) layer (i.e., $l = 0$) is the negative log-likelihood (NLL) function of the neural network, i.e.,

$$\hat{H}(Y|Z_0) = -E_{(x,y)\sim D} [\log p(\hat{y}|x)].$$

(18)

Similarly, the VCR of the highest-level layer (i.e., $l = L$) equals that of the compositional layer $Z = (Z_1, Z_2, ..., Z_L)$, a composite of all hidden layers; in addition, their VCR is an upper bound on the NLL:

$$\hat{H}(Y|Z_L) = \hat{H}(Y|Z) \geq -E_{(x,y)\sim D} [\log p(\hat{y}|x)].$$

(19)

**Proof of Theorem 2** Using the definition of VCR, the Markov Chain assumption, and Jensen’s inequality (details in Appendix).
3.3.2 Approximate Compression

While \( p(z_l|z_{l-1}) \) in DNNs has an analytical form, \( p(z_l|x) \) for \( l > 1 \) generally does not as it is a mixture of \( p(z_l|x) \). We thus propose to avoid directly estimating \( I(Z_l; X) \) by instead resorting to its upper bound \( I(Z_l; Z_{l-1}) \) as its surrogate in the optimization. However, \( I(Z_l; Z_{l-1}) \) is still intractable as it has the intractable distribution \( p(z_l) \). We then approximate \( I(Z_l; Z_{l-1}) \) using a mean-field (factorized) variational distribution \( r(z_l) = \prod_{i=1}^{m_l} r(z_{l,i}) \):

\[
I(Z_l; X) \leq I(Z_l; Z_{l-1}) \\
= \int p(z_l|z_{l-1})p(z_{l-1}) \log \frac{p(z_l|z_{l-1})}{p(z_l)} dz_l dz_{l-1} \\
\leq \int p(z_l|z_{l-1})p(z_{l-1}) \log \frac{p(z_l|z_{l-1})}{r(z_l)} dz_l dz_{l-1} \\
= \mathbb{E}_{z_{l-1}} \sum_{i=1}^{n_l} D_{KL} [p(z_{l,i}|z_{l-1})||r(z_{l,i})] \\
= : \hat{I}(Z_l; Z_{l-1}) \tag{20}
\]

The factorized variational distribution \( r(z_l) \) does not only enable feasible computation but also promotes independence among the dimensions of each vector point \( z_l \). This independence-favored property encourages each dimension to explain for a different underlying causal factor of the observed data, therefore manifests the concept of distributed representation [Bengio et al. (2013)]. In PIB, the mean-field distribution \( r(z_l) \) is made learnable.

3.4 Compromised Information Optimality

Since we could not simultaneously achieve the optimality in all layers in the MO optimizations in Equation (14), we propose two simple compromised strategies, namely JointPIB and GreedyPIB. JointPIB (Details at Algorithm 1) is a weighted sum of layer-wise PIB objectives:

\[
\mathcal{L}_{\text{joint}} := \sum_{l=0}^{L} \eta_l \hat{L}_l \tag{21}
\]

where \( \gamma_l > 0 \). The main idea of JointPIB is to simultaneously optimize all encoders and variational relevance decoders. Even though each layer might not achieve its individual optimality, their joint optimality encourages a joint compromise. On the other hand, GreedyPIB applies PIB progressively in a greedy manner. In other words, GreedyPIB tries to obtain the conditional optimality of a current layer which is conditioned on the achieved conditional optimality of the previous layers.

**Algorithm 1 Optimize SNN with JointPIB**

1: procedure JointPIB
2: \hspace{2em} Input: \( S_0 \leftarrow (x_i, y_i)_{i=1}^N \sim p_D(x, y) \)
3: \hspace{2em} Initialization: \( \theta \)
4: \hspace{2em} while not converged do
5: \hspace{3em} for \( i = 1 \) to \( L \) do
6: \hspace{4em} \( S_i \leftarrow (z_{l(i)})_{k=1}^M \sim p(z_l|z_{l-1}) \forall z_{l-1} \in S_{l-1} \)
7: \hspace{3em} end for
8: \hspace{2em} \( \mathcal{L}_{\text{joint}}(\theta) \leftarrow \text{Eqs (16), (17), (20), and \{S_i\}^L_{i=0}} \)
9: \hspace{2em} \( g \leftarrow \frac{\partial}{\partial \theta} \mathcal{L}_{\text{PIB}}(\theta) \)
10: \hspace{2em} \( \theta \leftarrow \theta - \nu g \)
11: \hspace{2em} Output: \( \theta \)
12: end while
13: end procedure
| Model       | Classification (Error %) | Adv. Robustness (%)                  |
|-----------|--------------------------|-------------------------------------|
| DET       | 1.73                     | 00.00 00.00                         |
| VIB       | 1.45                     | 83.70 93.10                         |
| SFNN      | 1.44                     | 83.00 95.20                         |
| GreedyPIB | 1.54                     | 83.21 94.30                         |
| JointPIB  | 1.36                     | 84.16 96.00                         |

Table 1: The performance of PIB for classification and adversarial robustness on MNIST and CIFAR10 in comparison with MLE and a partially information-theoretic treatment VIB.

4 An Application to Neural Networks

To analyze our PIB framework, we apply it to a simple network architecture: binary stochastic feed-forward (fully-connected) neural networks (SFNN) and leave other more complicated architectures for future work. In binary SFNN, we use sigmoid as its activation function: $p(z_l|z_{l-1}) = \sigma(W_{l-1}z_{l-1} + b_{l-1})$ where $\sigma(.)$ is the (element-wise) sigmoid function, $W_{l-1}$ is the network weights connecting layer $l-1$ to layer $l$, $b_{l-1}$ is a bias vector and $Z_l \in \{0, 1\}^n$.

It has been not clear so far as to how the gradient is computed in stochastic neural network at line 10 of Algorithm 1. The sampling operation in stochastic neural networks precludes the backpropagation in a computation graph. It becomes even more challenging with binary stochastic neural networks as it is not well-defined to compute gradients w.r.t. discrete-valued variables. Fortunately, we can find approximate gradients which has been proved to be efficient in practice:\footnote{To back-propagate through the sampling of real-valued variables, we can use re-parameterization trick\cite{Kingma2013}.}

REINFORCE estimator\cite{Williams1992, Bengio2013}, straight-through estimator\cite{Hinton2013}, the generalized EM algorithm\cite{Tang2013}, and Raiko (biased) estimator\cite{Raiko2015}. Especially, we found the Raiko gradient estimator works best in our specific setting thus deployed it in this application. In the Raiko estimator, the gradient of a bottleneck particle $z_{l,i} \sim p(z_{l,i}|z_{l-1}) = \sigma(a^{(i)}_l)$ is propagated only through the deterministic term $\sigma(a^{(i)}_l)$:

$$\frac{\partial z_{l,i}}{\partial \theta} \approx \frac{\partial \sigma(a^{(i)}_l)}{\partial \theta}.$$

5 Experiments

In this section, we evaluated our PIB framework using GreedyPIB and JointPIB algorithms on MNIST\cite{LeCun1998} and CIFAR10\cite{Krizhevsky2009} for classification, learning dynamics and robustness against adversarial attacks. The MNIST dataset consists of $28 \times 28$ pixel greyscale images.
of handwritten digits 0-9, with 60,000 training and 10,000 test examples. The CIFAR10 dataset consists of 60,000 (50,000 for train and 10,000 for test) 32x32 colour images in 10 classes, with 6,000 images per class.

5.1 Image classification

To evaluate the generalization capability of the layer-wise informativeness in PIB, we compared JointPIB and GreedyPIB with other three comparative models which used the same network architecture without any explicit regularizer: (1) Standard deterministic neural network (DET) which simply treats each hidden layer as deterministic; (2) Stochastic Feed-forward Neural Network (SFNN) Raiko et al. (2015) which is a binary stochastic neural network as in PIB but is trained with the MLE principle; (3) Variational Information Bottleneck (VIB) Alemi et al. (2017) which employs the entire deterministic network as an encoder, adds an extra stochastic layer as a out-of-network bottleneck variable, and is then trained with the IB principle on that single bottleneck layer. The base network architecture in this experiment had two hidden layers with 512 sigmoid-activated neurons per each layer.

Adopted from the common practice, we used the last 10,000 images of the training set as a validation (holdout) set for tuning hyper-parameters. We then retrained models from scratch in the full training set with the best validated configuration. We trained each of the five models with the same set of 5 different initializations and reported the average results over the set. For the stochastic models (all except DET), we drew $M = 32$ samples per stochastic layer during both training and inference, and performed inference 10 times at test time to report the mean of classification errors for MNIST and classification accuracy for CIFAR10. For JointPIB and GreedyPIB, we set $\gamma_l = 1$ (in JointPIB only) and $\beta_l = \beta, \forall 1 \leq l \leq L$, tuned $\beta$ on a linear log scale $\beta \in \{10^{-i} : 1 \leq i \leq 10\}$. We found $\beta = 10^{-4}$ worked best for both models. For VIB, we found that $\beta = 10^{-3}$ and $\beta = 10^{-4}$ worked best on MNIST and CIFAR10, respectively. We trained the models on MNIST with Adadelta optimization Zeiler (2012) and on CIFAR10 with Adagrad optimization Duchi et al. (2011) (except for VIB we used Adam optimization Kingma and Ba 2014) as we found that they worked best in the validation set.
The results are shown in Table 1. We see that we can outperform the other models with the explicit informativeness induced into each layer either by JointPIB on MNIST and GreedyPIB on CIFAR10.

5.2 Robustness against adversarial attacks

Neural networks are prone to adversarial attacks which disturb the input pixels by small amounts imperceptible to humans (Szegedy et al., 2013; Nguyen et al., 2015). Adversarial attacks generally fall into two categories: untargeted and targeted attacks. An untargeted adversarial attack $A$ maps the target model $M$ and an input image $\mathbf{x}$ into an adversarially perturbed image $\mathbf{x'}$: $A: (M, \mathbf{x}) \rightarrow \mathbf{x'}$, and is considered successful if it can fool the model $M(\mathbf{x}) \neq M(\mathbf{x'})$. A targeted attack, on the other hand, has an additional target label $l$: $A: (M, \mathbf{x}, l) \rightarrow \mathbf{x'}$, and is considered successful if $M(\mathbf{x'}) = l \neq M(\mathbf{x})$.

To evaluate PIB’s capability to equip a neural network with robustness against adversarial attacks, we performed adversarial attacks to the neural networks trained by MLE and PIB, and resorted to the accuracy on adversarially perturbed versions of the test set to rank a model’s robustness. In addition, we use the $L_2$ attack method for both targeted and untargeted attacks from Carlini and Wagner (2017), which has shown to be most effective attack algorithm with smaller perturbations. Specifically, we attacked the same four comparative models described from the previous experiment on the first 1,000 samples of the MNIST test set. For the targeted attacks, we targeted each image into the other 9 labels other than the true label of the image.

The results are also shown in Table 1. We see that the deterministic model DET is totally fooled by the attacks. It is known that stochasticity in neural networks improves adversarial robustness which is consistent in our experiment as SFNN is significantly more adversarially robust than DET. VIB has compatible adversarial robustness with SFNN even if VIB has “less stochasticity” than SFNN (VIB has one stochastic layer while all hidden layers of SFNN are stochastic). This is because VIB performance is compensated with IB principle for its stochastic layer. Finally, our JointPIB is shown to be more adversarially robust than the other models.
5.3 Learning dynamics

To demonstrate PIB’s capability to explicitly alter the layer-wise information, we visualize the compression and relevance of each layer over the course of training of SFNN, JointPIB and GreedyPIB. To simplify our analysis, we considered a binary decision problem where \( X \) is 12 binary inputs making up \( 2^{12} = 4096 \) equally likely input patterns and \( Y \) is a binary variable equally distributed among 4096 input patterns [Shwartz-Ziv and Tishby (2017)]. The base neural network architecture had 4 hidden layers with widths: 10-8-6-4 neurons. Since the network architecture is small, we could precisely compute the (true) compression \( I_x := I(Z_i; X) \) and (true) relevance \( I_y := I(Z_i; Y) \) over training epochs. We fixed \( \beta_l = \beta = 10^{-4} \) for both JointPIB and GreedyPIB, trained five different randomly initialized neural networks for each comparative model with SGD up to 20,000 epochs on 80% of the data, and averaged the mutual information.

Figure 2, Figure 3, and Figure 4-7 provide a visualization of the learning dynamics of SFNN, JointPIB, and GreedyPIB over different layers, respectively, on the information plane \((I_x, I_y)\). We can observe a common trend in the learning dynamics offered by both MLE (in SFNN model) and PIB framework (in JointPIB and GreedyPIB). Both principles allow the network to gradually encode more information about \( X \) and the relevant information about \( Y \) into the hidden layers at the beginning as \( I(Z_i; X) \) and \( I(Z_i; Y) \) both increase. Especially, compression does occur in SFNN which is consistent with the result reported in Shwartz-Ziv and Tishby (2017) that compression phase is an inherit property of DNNs trained with SGD even though the MLE principle is not explicitly designed to do so.

What distinguishes PIB from MLE is the maximum level of relevance at each layer and the number of epochs to encode the same level of relevance. Firstly, JointPIB and GreedyPIB at \( l = 1 \) needs only about 4.68% and 17.95%, respectively, of the training epochs to achieve at least the same level of relevance in all layers of SFNN at the final epoch. Recall that in GreedyPIB at \( l = 1 \) the PIB principle is applied to the first hidden layer only. Secondly, MLE is unable to encode the network layers to reach the maximum level of relevance enabled by PIB (We also trained SFNN up to 100,000 epochs and observed that the level of relevance of each layer degrades before ever reaching the value of 0.8 bits.). While it is not clear in this experiment what compression in PIB offers in comparison with MLE (as MLE is observed to possess compression capability as well), the compression constraints do help within the PIB framework itself by keeping the layer representation from shifting to the the right (in the information plane) during the encoding of relevant information (e.g., the layer representation at the final epoch gradually shifts to the left (i.e., more compressed) while not degrading the relevance over the greedy training from layer 1 to layer 4 as presented in Figure 4-7).

6 Conclusion

In this work, we introduce the Parametric Information Bottleneck (PIB) as an alternative to the MLE principle for learning DNNs. PIB offers a principled yet efficient way of inducing compression and
relevance into each layer of DNNs. While PIB can be considered as a fully parameterized version of the non-parametric IB principle to DNNs, it also generalizes the MLE principle to every layer levels under compression constraints. The competitive performance of PIB on MNIST and CIFAR10 suggests the effectiveness of PIB for exploiting the neural representations.

References

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2017). Deep variational information bottleneck. In Proceedings of the International Conference on Learning Representations, ICLR.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. ArXiv e-prints, 1409.0473.

Bengio, Y., Courville, A. C., and Vincent, P. (2013a). Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell., 35(8):1798–1828.

Bengio, Y., Léonard, N., and Courville, A. C. (2013b). Estimating or propagating gradients through stochastic neurons for conditional computation. CoRR, abs/1308.3432.

Carlini, N. and Wagner, D. A. (2017). Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017, pages 39–57. IEEE Computer Society.

Chalk, M., Marre, O., and Tkacik, G. (2016). Relevant sparse codes with variational information bottleneck. In Advances in Neural Information Processing Systems, pages 1957–1965.

Chechik, G., Globerson, A., Tishby, N., and Weiss, Y. (2005). Information bottleneck for gaussian variables. Journal of Machine Learning Research, 6:165–188.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In EMNLP, pages 1724–1734.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The helmholtz machine. Neural Computation, 7(5):889–904.

Duchi, J. C., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12:2121–2159.

Gal, Y. (2016). Uncertainty in Deep Learning. PhD thesis, University of Cambridge.

Hinton, G. (2016). Lecture 9.3 — using noise as a regularizer [neural networks for machine learning].

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, ICML, pages 448–456.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. ArXiv e-prints, 1412.6980.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. CoRR, abs/1312.6114.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Master’s thesis.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Proceedings of the Annual Conference on Neural Information Processing Systems, NIPS, pages 1106–1114.

LeCun, Yann, B. L. e. B. Y., H. P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324.

MacKay, D. J. C. (1992). A practical bayesian framework for backpropagation networks. Neural Computation, 4(3):448–472.
Neal, R. M. (1995). *Bayesian Learning for Neural Networks*. PhD thesis, Toronto, Ont., Canada. AAINN02676.

Nguyen, A. M., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 427–436. IEEE Computer Society.

Raiko, T., Berglund, M., Alain, G., and Dinh, L. (2015). Techniques for learning binary stochastic feedforward neural networks. In *Proceedings of the International Conference on Learning Representations, ICLR*.

Rey, M. and Roth, V. (2012). Meta-gaussian information bottleneck. In *Proceedings of the Annual Conference on Neural Information Processing Systems, NIPS*, pages 1925–1933.

Shwartz-Ziv, R. and Tishby, N. (2017). Opening the black box of deep neural networks via information. *CoRR*, abs/1703.00810.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Slonim, N. (2003). Information bottleneck theory and applications. *PhD thesis, Hebrew University of Jerusalem*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Strouse, D. and Schwab, D. J. (2016). The deterministic information bottleneck. In Ihler, A. T. and Janzing, D., editors, *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI 2016, June 25-29, 2016, New York City, NY, USA*. AUAI Press.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2013). Intriguing properties of neural networks. *CoRR*, abs/1312.6199.

Tang, Y. and Salakhutdinov, R. (2013). Learning stochastic feedforward neural networks. pages 530–538.

Tishby, N., Pereira, F. C., and Bialek, W. (1999). The information bottleneck method. In *Proceedings of the Annual Allerton Conference on Communication, Control and Computing*.

Tishby, N. and Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. In *Proceedings of the IEEE Information Theory Workshop, ITW*, pages 1–5.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.
Appendix

Proof of Lemma 1

Proof of Lemma 1. Here we only briefly introduce the main derivations from the solution to the IB problem in (Tishby, Pereira, and Bialek 1999) that are necessary for the conclusion of Lemma 1. For even more details, please refer to (Tishby, Pereira, and Bialek 1999).

To find the critical points of \( L_1[p(\mathbf{z}_1|\mathbf{x})] := I(Z_1; X) - \beta_1 I(Z_1; Y) \), we introduce Lagrangian multipliers \( \lambda_1(\mathbf{x}) \) for the normalization of the encoder \( p(\mathbf{z}_1|\mathbf{x}) \) for each \( \mathbf{x} \). The Lagrangian then becomes:

\[
L'_1 = I(Z_1; X) - \beta_1 I(Z_1; Y) - \int \lambda_1(\mathbf{x}) \int p(\mathbf{z}_1|\mathbf{x}) d\mathbf{z}_1 d\mathbf{x}
\]

Taking the derivatives w.r.t. \( p(\mathbf{z}_1|\mathbf{x}) \) for given \( \mathbf{z}_1 \) and \( \mathbf{x} \), we have that \( \frac{\partial L'_1}{\partial p(\mathbf{z}_1|\mathbf{x})} \) becomes:

\[
p(\mathbf{x}) \left[ \log \frac{p(\mathbf{z}_1|\mathbf{x})}{p(\mathbf{z}_1)} + \beta_1 \sum_y p(y|\mathbf{x}) \log \frac{p(y|x)}{p(y|\mathbf{z}_1)} + \beta_1 \lambda_1(\mathbf{x}) \right]
\]

where

\[
\hat{\lambda}_1(\mathbf{x}) := \frac{\lambda_1(\mathbf{x})}{p(\mathbf{x})} - \beta_1 I(\mathbf{x}; Y)
\]  \hspace{1cm} (22)

By setting \( \frac{\partial L'_1}{\partial p(\mathbf{z}_1|\mathbf{x})} \) to zero, we obtain the following form for the encoder:

\[
p(\mathbf{z}_1|\mathbf{x}) = \frac{p(\mathbf{z}_1)}{Z(\mathbf{x}, \beta_1)} e^{-\beta d(\mathbf{x}, \mathbf{z}_1)}
\]  \hspace{1cm} (23)

where \( Z(\mathbf{x}, \beta_1) \) is the normalization partition function for \( p(\mathbf{z}_1|\mathbf{x}) \):

\[
e^{\beta \hat{\lambda}_1(\mathbf{x})} = Z(\mathbf{x}, \beta_1) = \mathbb{E}_{\mathbf{z}_1} e^{-\beta d(\mathbf{x}, \mathbf{z}_1)}
\]  \hspace{1cm} (24)

It follows from Equation (22) and (24) that

\[
\lambda_1(\mathbf{x}) := p(\mathbf{x}) \left[ \log \mathbb{E}_{\mathbf{z}_1} e^{-\beta d(\mathbf{x}, \mathbf{z}_1)} + \beta I(\mathbf{x}, Y) \right], \hspace{1cm} (25)
\]

implying Lemma 1.

\( \square \)

Proof of Theorem 2

Proof of Theorem 2. The first claim of Theorem 2 immediately follows from the definition of VCR:

\[
\hat{H}(Y|Z_0) = -\mathbb{E}_{(\mathbf{x}, \mathbf{y}) D} \mathbb{E}_{\mathbf{z}_0|\mathbf{x}} \log p(\tilde{y}|\mathbf{z}_0)
\]
\[
= -\mathbb{E}_{(\mathbf{x}, \mathbf{y}) D} \log p(\tilde{y} | \mathbf{x}). \hspace{1cm} (26)
\]

For the second claim of Theorem 2, it follows from the Markov assumption in Equation (1) and from Jensen’s inequality, respectively, that:

\[
p(\tilde{y}|\mathbf{z}) = p(\tilde{y}|\mathbf{z}_L, \mathbf{z}_{L-1}, ..., \mathbf{z}_1) = p(\tilde{y}|\mathbf{z}_L)
\]  \hspace{1cm} (27)

\[
\int p(\mathbf{z}|\mathbf{x}) \log p(\tilde{y}|\mathbf{z}) d\mathbf{z} \leq \log \left( \int p(\mathbf{z}|\mathbf{x}) p(\tilde{y}|\mathbf{z}) d\mathbf{z} \right)
\]
\[
\leq \log p(\tilde{y} | \mathbf{x}) \hspace{1cm} (28)
\]

Thus, we have

\[
-\mathbb{E}_{(\mathbf{x}, \mathbf{y}) D} \log p(\tilde{y} | \mathbf{x}) \leq -\mathbb{E}_{(\mathbf{x}, \mathbf{y}) D} \int p(\mathbf{z}|\mathbf{x}) \log p(\tilde{y}|\mathbf{z}) d\mathbf{z}
\]
\[
= -\mathbb{E}_{(\mathbf{x}, \mathbf{y}) D} \mathbb{E}_{\mathbf{z}|\mathbf{x}} \log p(\tilde{y}|\mathbf{z})
\]
\[
= \hat{H}(Y|Z)
\]
\[
= -\mathbb{E}_{(\mathbf{x}, \mathbf{y}) D} \mathbb{E}_{\mathbf{z}_L|\mathbf{z}_{L-1}, ..., \mathbf{z}_1} \log p(\tilde{y}|\mathbf{z}_L)
\]
\[
= -\mathbb{E}_{(\mathbf{x}, \mathbf{y}) D} \mathbb{E}_{\mathbf{z}_L | \mathbf{z}} \log p(\tilde{y}|\mathbf{z}_L)
\]
\[
= \hat{H}(Y|Z_L), \hspace{1cm} (29)
\]

implying Theorem 2.

\( \square \)
VCR decomposition for a multivariate target variable

We will prove that the VCR of level \( l \) for a multivariate variable \( \mathbf{y} \) can be decomposed as the sum of the VCRs of each of its vector elements. Indeed, consider \( \mathbf{y} = (y_1, ..., y_n) \). It follows from the fact that the neurons within a layer are conditionally independent given the previous layer that we have:

\[
\tilde{H}(\mathbf{Y}|Z_l) = -\mathbb{E}_{(x,y_1, ..., y_n, z_l)} \log p(\hat{y}_1, ..., \hat{y}_n|z_l)
\]

\[
= -\mathbb{E}_x \mathbb{E}_{y_1, ..., y_n} \mathbb{E}_{z_l} \sum_{i=1}^n \log p(\hat{y}_i|z_l)
\]

\[
= \sum_{i=1}^n \tilde{H}(Y_i|Z_l)
\]

implying the claim about the decomposibility of VCR for a multivariate target variable.

Derivation for the relevance decoder

Derivation for the relevance decoder.

\[
p(y|z_l) = \frac{p(y, z_l)}{p(z_l)} = \int \frac{p(x, y, z_l)}{p(z_l)} dx
\]

\[
= \int \frac{p(x, y)p(z_l|x, y)}{p(z_l)} dx
\]

\[
= \int \frac{p(x, y)p(z_l|x)}{p(z_l)} dx (\text{due to } Z_l \perp Y|X),
\]

implying Equation (2) in the main material.

Derivation for a variational upper bound on mutual information

Proof. We give a more detailed derivation for Equation (16) in the main material:

\[
H(\mathbf{Y}|Z_l) = -\int p(z_l)p(y|z_l) \log p(y|z_l) dy dz_l
\]

\[
= -\int p(z_l)p(y|z_l) \log p_e(y|z_l) dy dz_l
\]

\[
- \int p(z_l)D_{KL}[p(y|z_l)||p_e(y|z_l)] dz_l
\]

\[
\leq -\int p(z_l)p(y|z_l) \log p_e(y|z_l) dy dz_l
\]

\[
= -\int p(y, z_l) \log p_e(y|z_l) dy dz_l
\]

\[
= -\int p(x, y, z_l) \log p_e(y|z_l) dy dz_l dx
\]

\[
= -\int p_D(x, y)p(z_l|x, y) \log p_e(y|z_l) dy dz_l dx
\]

\[
= -\int p_D(x, y)p(z_l|x) \log p_e(y|z_l) dz_l dx dy
\]

\[
= -\mathbb{E}_{p_D(x,y)} \left[ \mathbb{E}_{p(z_l|x)} [\log p_e(y|z_l)] \right] =: \tilde{H}(\mathbf{Y}|Z_l)
\]

\[
\square
\]
MLE as distribution matching

MLE as distribution matching. The purpose of the MLE principle can be interpreted as matching the model probability function to the empirical data probability function using the Kullback-Leibler (KL) divergence as a measure of their discrepancy. Indeed, given a set of samples \( X = \{x_1, x_2, \ldots, x_N\} \) i.i.d. drawn from some underlying data probability function \( p_D(x) \), a parametric model \( p_{\text{model}}(x; \theta) \) attempts to map any data sample \( x \) to a real number that estimates the true probability \( p_D(x) \). The MLE principle maximizes the likelihood function on the empirical data. This in turn can be interpreted as matching the model probability function \( p_{\text{model}} \) with the data probability function \( p_D \) by minimizing their KL divergence to find the maximum likelihood (point) estimator for \( \theta \):

\[
\theta_{ML} = \arg \max_{\theta} \mathbb{E}_{x \sim p_D(x)} \left[ \log p_{\text{model}}(x; \theta) \right]
= \arg \min_{\theta} \left\{ -\mathbb{E}_{x \sim p_D(x)} \left[ \log p_{\text{model}}(x; \theta) \right] \right\} + \mathbb{E}_{x \sim p_D(x)} \left[ \log p_D(x; \theta) \right]
= \arg \min_{\theta} D_{KL} \left[ p_D(x) \parallel p_{\text{model}}(x; \theta) \right]
\approx \arg \max_{\theta} \sum_{i=1}^{N} \log p_{\text{model}}(x_i; \theta)
\]

(31)

(32)

where Expression (32) is an empirical estimation of Expression (31) for \( N \) datapoints.