PARAMETRIC INFORMATION BOTTLENECK TO OPTIMIZE STOCHASTIC NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, 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) framework (Tishby et al. (1999)) 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, as compared to the maximum likelihood estimate (MLE) principle, PIBs: (i) improve the generalization of neural networks in classification tasks, (ii) push the representation of neural networks closer to the optimal information-theoretical representation in a faster manner.

1 INTRODUCTION

Deep neural networks (DNNs) have demonstrated competitive performance in several learning tasks including image recognition (e.g., [Krizhevsky et al. (2012), Szegedy et al. (2015]), natural language translation (e.g., [Cho et al. (2014), Bahdanau et al. (2014)]) and game playing (e.g., [Silver et al. (2016)]). Specially 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 (Srivastava et al. (2014)), and batch normalization (Ioffe & Szegedy (2015))), and optimizations (e.g., [Kingma & Ba (2014)]). 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 that 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.

The Information Bottleneck (IB) framework (Tishby et al. (1999)) 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:

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

(1)

where $\beta$ is the positive Lagrangian multiplier that controls the trade-off between the complexity of the representation, $I(Z, X)$, and the amount of relevant information in $Z$, $I(Z, Y)$. The exact solution to the minimization problem above is found (Tishby et al. (1999)) with the implicit self-
consistent equations:

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

(2)

where \(Z(x; \beta)\) is the normalization function, and \(D_{KL}[.||.]\) is the Kullback-Leibler (KL) divergence (Kullback & Leibler (1951)). Unfortunately, the self-consistent equations are highly non-linear and still too abstract to be used for many practical applications. 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 \(p_{data}\) as possible (e.g., see Appendix I.B). The MLE principle treats the neural network model \(p(x; \theta)\) as a whole without explicitly considering the contribution of its internal structures (e.g., hidden layers and hidden neurons). 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 are proved to be mathematically equivalent to the IB principle for multinomial mixture model for clustering problem when the input distribution \(X\) is uniform or has a large sample size (Slonim & Weiss (2002)). However in general they are not obviously related.

In this work, 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 (Tishby & Zaslavsky (2015)), we name this method as Parametric Information Bottleneck (PIB). Because the generalized IB objective is intractable, we approximate it using variational methods and Monte Carlo estimation. We propose to re-use the existing neural network architecture as variational decoders for each hidden layers. The approximate generalized IB objective in turn presents itself 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.

2 RELATED WORK

Originally, the general IB framework is proposed in Tishby et al. (1999). 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 (Tishby et al. (1999)); or (2) \(X, Y\) and \(Z\) are mutually joint Gaussian (Chechik et al. (2005)) where \(Z\) is a bottleneck variable.

Recently, the IB principle is applied to DNNs Tishby & Zaslavsky (2015). 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 rather proposes an general analysis of optimality 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 author 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.

The recent work Alemi et al. (2016) also uses 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 \( p(z|x; \theta) \) by an entire neural network. The encoder maps the input variable \( x \) to a single bottleneck variable \( z \) that is not a part of the considered neural network architecture. Therefore, their approach still learns a neural network as a whole rather than actually optimizing it in parts. Furthermore, the work imposes a prior distribution in the code space. In our work, the distribution of a code space is simply transformed by an encoder from another space.

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 IB principle as in [Alemi et al., 2016] 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.

3 Parametric Information Bottleneck

This section presents an information-theoretic perspective of neural networks and then defines our PIB framework. This perspective paves a way for the soundness of constraining the compression-relevance trade-off into a neural network.

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 \( l^{th} \) 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 \( p_u(x), p_u(y) \) and \( p(z_l) \) where \( p_u(\cdot) \) 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 \( p(z_l|z_{l-1}) \). 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.

3.1 Neural Networks as Sequential Quantization

An encoder \( p(z|x) \) introduces a soft partitioning of the space \( \mathcal{X} \) into a new space \( \mathcal{Z} \) whose probability measure is determined as \( p(z) = \int p(z|x)p_u(x)dx \). 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 \( p(z|x) \). 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 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 \( Z_l \). These code spaces are lossy representations of the data space as it follows from the data-processing inequality (DPI) ([Cover & Thomas, 2006]) that

\[
H(X) \geq I(X, Z_l) \geq I(X, Z_{l+1}) \tag{3}
\]

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

\[
Y \rightarrow X \rightarrow Z_l \rightarrow Z_{l+1} \tag{4}
\]

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.
Figure 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 \( p_{true}(y|z_i) \), which is uniquely determined given the encoder \( p_{\theta}(z_i|x) \) and the joint distribution \( p(x, y) \), is intractable. We use \( p_{\theta}(y|z_i) \) as a variational approximation to each intractable relevance decoder \( p_{true}(y|z_i) \).

3.2 PIB Model

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 (LeCun et al. (1998)) and ResNet (He et al. (2016)).

3.2.1 Stochasticity

In this paper, we focus on binary bottlenecks in which the encoder \( p(z_l|z_{l-1}) \) is defined as

\[
p(z_l|z_{l-1}) = \prod_{i=1}^{n_l} p(z_{l,i}|z_{l-1})
\]

where

\[
p(z_{l,i} = 1|z_{l-1}) = \sigma(a_{l,i}^{(l)}) = \sigma(W_{l,i}^{(l)}z_{l-1} + b_{l,i}^{(l)}),
\]

\( \sigma(.) \) is the sigmoid function, and \( W^{(l)} \) is the weights connecting the \( l \)th layer to the \((l + 1)\)th layer. Depending on the structure of the target space \( Y \), we can use an appropriate model for output distributions as follows: (1) For classification, we model the output distribution with softmax function, \( p(Y = i|z_L) = \text{softmax}(W^{(L+1)}z_L + b^{(L+1)}) \); (2) For binary output vectors \( Y \), we use a product of Bernoulli distributions, \( p(y|z_L) = \prod_i p(y_i|z_L) \) where \( p(Y_i = 1|z_L) = \sigma(W^{(L+1)}z_L + b^{(L+1)}) \); (3) For real-valued output vectors \( Y \), we use Gaussian distribution,
p(Y|zl) = N(y; μ = W(L+1)zl + b(L+1), σ^2). The conditional distribution p(y|x) from the model is computed using the Bayes’ rule and the Markov assumption (Equation 4) in PIBs

\[ p(y|x) = \int p(y,z|x)dz = \int p(y|z)p(z|x)dz = \prod_{l=1}^{L+1} p(z_l|z_{l-1})dz \]

(7)

where z = (z_1, z_2, ..., z_L) is the entire sequence of hidden layers in the neural network. Note that for a given joint distribution p_u(x, y) when a bottleneck Z_l is defined, so the encoding function p(z_l|x), the corresponding relevance decoder p_{true}(y|z_l) is uniquely determined by

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

(8)

It is also important to note that many stochastic neural networks have been proposed (e.g., Neal (1990), Neal (1992), Tang & Salakhutdinov (2013), Raiko et al. (2014), Dauphin & Grangier (2016)). However, our motivation for this stochasticity is that it enables bottleneck sampling 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. For 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.

3.2.2 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 the lth-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:

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

(9)

where the layer-specific Lagrangian multiplier \( \beta_l^{-1} \) controls the tradeoff between relevance and compression in each bottleneck, and the concept of compression and relevance is taken to the extreme when \( l = 0 \) (with convention that \( I(X_0, Z_{-1}) = I(X, \emptyset) = H(X) \) = constant).

Optimizing PIBs now becomes the minimization of \( L_{PIB}(Z) \) 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 lth-level more compressed while the increase of \( I(Z_l, Y) \) promotes the preservation of relevant information in \( Z_l \) about \( Y \). In optimization’s aspect, the minimization of \( L_{PIB} \) is much harder than the minimization of \( L_{IB} \) since \( L_{PIB} \) involves dependent terms that even the self-consistent equations of the IB framework are not applicable to this case. Furthermore, \( L_{PIB} \) is intractable since the bottleneck spaces are usually high-dimensional and the relevance encoders \( p_{true}(y|z_l) \) (computed by Equation 8) are intractable. In the following section, we present our approximation to \( L_{PIB} \) which fully utilizes the existing architecture without resorting to any model that is not part of the considered neural network. The approximation then leads to effective gradient-based training of PIBs.

3.3 APPROXIMATE LEARNING

Here, we present our approximations to the relevance and the compression terms in the PIB objective \( L_{PIB} \).

3.3.1 APPROXIMATE RELEVANCE

Since the relevance decoder \( p_{true}(y|z_l) \) (Equation 8) is intractable, we use a variational relevance decoder \( p_{v}(y|z_l) \) to approximate it. Firstly, we decompose the mutual information into a difference

---

\[ \int \text{ even for discrete-valued variables instead of sum } \sum \text{ for denotation simplicity.} \]
of two entropies:

\[ I(Z_l, Y) = H(Y) - H(Y | Z_l) \]  

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

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

\[
= - \int p_D(\mathbf{x}, y) p(z_l | \mathbf{x}) \log p_{\text{true}}(y | z_l) \, dz_l \, dy
\]

\[
\leq - \int p_D(\mathbf{x}, y) p(z_l | \mathbf{x}) \log p_{v}(y | z_l) \, dz_l \, dy
\]

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

where the equality in Equation\(^{(12)}\) holds due to the Markov assumption (Equation\(^{(4)}\)). In PIBs, we propose to use the higher-level part of the existing network architecture at each layer to define the variational relevance encoder for that layer, i.e., \( p_v(y | z_l) = p(y | z_l) \) where \( p(y | z_l) \) is determined by the network architecture. In this case, we have:

\[
p_v(y | z_l) = p(y | z_l) = \int \prod_{i=l}^{L+1} p(z_{i+1} | z_i) \, dz_L ... dz_{l+1} = \mathbb{E}_{p(z_L | z_l)} \left[ p(y | z_L) \right]
\]

We will refer to \( \hat{H}(Y | Z_l) \) as the variational conditional relevance for the \( l \)th-level bottleneck variable \( Z_l \) for the rest of this work. In the following, we present 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 1:** The variational conditional relevance for the zero-level bottleneck \( Z_0 = X \) is the negative log-likelihood of the PIB.

**Proposition 2:** The variational conditional relevance for the highest-level bottleneck variable \( Z_L \) equals the variational relevance for the whole compositional bottleneck variable \( Z \) which is an upper bound on the negative log-likelihood. That is,

\[
\hat{H}(Y | Z_L) = \hat{H}(Y | Z) \geq - \mathbb{E}_{p_D(\mathbf{x}, y)} \left[ \log p(y | \mathbf{x}) \right]
\]

While the Proposition 1 is a direct result of Equation\(^{(15)}\) the Proposition 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 increasing the variational relevance of the network as a whole while the PIB objective takes into account the relevance at every level of the network. The variational relevance can also be interpreted in terms of the MLE principle as follows. Maximizing the variational relevance at level \( l \) is the maximum of the log-likelihood of \( p(y | z_l) \). Furthermore, the variational conditional relevance for a multivariate \( y \) can be decomposed into the sum of that for each component of \( y \) (see Appendix I.C).

### 3.3.2 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\(^3\) For the compression, we decompose the mutual information as follows:

\[
I(Z_1, Z_0) = H(Z_1) - H(Z_1 | Z_0)
\]

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

\[
H(Z_1 | Z_0) = \int p(z_0) H(Z_1 | Z_0 = z_0) \, dz_0 = \int p(z_0) \sum_{i=1}^{N_1} H(Z_{1,i} | Z_0 = z_0) \, dz_0
\]

\[
= \mathbb{E}_{p(z_0)} \left[ \sum_{i=1}^{N_1} H(Z_{1,i} | Z_0 = z_0) \right]
\]

\(^3\)The extension at the other levels is straightforward from the derivation of \( I(Z_1, Z_0) \).
Algorithm 1 Minibatch version of training PIB, we use $M = 16$ for training (and $M = 32$ for testing).

1: procedure GRAD-PIB
2: Input: Labeled training dataset $S_D$
3: $\theta \leftarrow$ Initialize parameters
4: repeat:
5: \hspace{1em} $(x_i, y_i)_{i=1}^{N} \leftarrow$ Random minibatch of $N$ samples drawn from $S_D$
6: \hspace{1em} Generate $M$ samples of $z_i$ per each sample of $x_{i-1}$ for $1 \leq i \leq L$
7: \hspace{1em} Use the generated samples above and Equations 14 and 23 to approximate $\hat{\mathcal{L}}_{PIB}(\theta)$
8: \hspace{1em} $g \leftarrow \frac{\partial}{\partial \theta} \hat{\mathcal{L}}_{PIB}(\theta)$ using Raiko estimator
9: \hspace{1em} $\theta \leftarrow$ Update parameters using the approximate gradients $g$ and SGD
10: until convergence of parameters $\theta$
11: Output: $\theta$
12: end procedure

\[ H(Z_1) = -E_{p(z_1)}[\log p(z_1)] = \frac{1}{M} \sum_{k=1}^{M} \log p(z_1^{(k)}) =: \hat{H}_{MLE}(Z_1) \tag{20} \]

where $z_1^{(k)} \sim p(z_1) = E_{p(z_0)}[p(z_1|z_0)]$. This estimator is also known as the maximum likelihood estimator or ‘plug-in’ estimator (Antos & Kontoyiannis [2001]). The larger number of samples $M$ guarantees the better plug-in entropy by the following bias bound (Paninski [2003])

\[ |E[H_{MLE}(Z_1)] - H(Z_1)| \leq \log(1 + \frac{|Z_1| - 1}{M}) \tag{21} \]

where $|Z_1|$ denotes the cardinality of the space of $Z_1$. In practice, $\log p(z_1)$ may be numerically unstable for large cardinality $|Z_1|$. In the large space of $Z_1$, the probability of a single point $p(z_1)$ may becomes very small that $\log p(z_1)$ become numerically unstable. To overcome this problem, we propose an upper bound on the entropy using Jensen’s inequality:

\[ \log p(z_1) = \log E_{p(z_0)}[p(z_1|z_0)] \geq E_{p(z_0)}[\log p(z_1|z_0)] \tag{22} \]

Thus,

\[ H(Z_1) \leq -E_{p(z_1)}[E_{p(z_0)}[\log p(z_1|z_0)]] := \tilde{H}(Z_1) \tag{23} \]

The upper bound $\tilde{H}(Z_1)$ is numerically stable because the conditional distribution $p(z_1|z_0)$ is factorized into $\prod_i p(z_i|z_0)$, therefore, $\log p(z_1|z_0) = \sum_i \log p(z_i|z_0)$ which is more stable. The upper bound $H(Z_1)$ can then be estimated using Monte Carlo sampling for $z_0$ and $z_1$.

3.3.3 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. Tang & Salakhutdinov [2013] used a Generalized EM algorithm while Bengio et al. [2013] proposed to resort to reinforcement learning. However, these estimators have high variance. In this work, we use the gradient estimator inspired by Raiko et al. [2014] for binary bottlenecks because it has low variance despite of being biased. Specifically, a bottleneck $z = (z_1, z_2, ..., z_n)$ can be rewritten as being continuous by $z_i = \sigma(a_i) + \epsilon_i$ where

\[ \epsilon_i = \begin{cases} 1 - \sigma(a_i) & \text{with probability } \sigma(a_i) \\ -\sigma(a_i) & \text{with probability } 1 - \sigma(a_i) \end{cases} \]

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].
4  EXPERIMENTS

We used the same architectures for PIBs and Stochastic Feed-forward Neural Networks (SFNNs) (e.g., [Tang & Salakhutdinov (2013)]) and trained them on the MNIST dataset (LeCun et al. (1998)) 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 (Al-Rfou et al. (2016)).

4.1 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean (%)</th>
<th>Std dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deterministic (A)</td>
<td>1.73</td>
<td>-</td>
</tr>
<tr>
<td>SFNN as deterministic (B)</td>
<td>1.88</td>
<td>-</td>
</tr>
<tr>
<td>PIB as deterministic (C)</td>
<td><strong>1.46</strong></td>
<td>-</td>
</tr>
<tr>
<td>stochastic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deterministic as stochastic (D)</td>
<td>2.30</td>
<td>0.07</td>
</tr>
<tr>
<td>SFNN (E)</td>
<td>1.94</td>
<td>0.036</td>
</tr>
<tr>
<td>PIB</td>
<td><strong>1.47</strong></td>
<td><strong>0.034</strong></td>
</tr>
</tbody>
</table>

Table 1: The MNIST classification results of various models.

Figure 2: A comparison of Monte-Carlo averaging and deterministic prediction of PIB.

The MNIST dataset (LeCun (1998)) 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 × 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.

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 (Glorot & Bengio (2010)). 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 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 2). 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. For visualization of learned filters of PIB, see Appendix II.A.

4.2 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 (Tishby et al. (1999), Slonim (2003)). 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
Figure 4: 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.

architecture of 784-10-10-10-1 for PIB and SFNN and trained them with 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 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 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.

For additional visualization, an empirical architecture analysis of PIB and SFNN is presented in Appendix II.B.

4.3 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 $Z_{i}$ of dimensionality $n_{i}$ can take on $2^{n_{i}}$ different states each of which would give a different $Y$. This is the reason why the conditional distribution $p(Y|X)$ in stochastic neural networks is multi-modal.

In this experiment, we followed Raiko et al. (2014) 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 4 indicates that PIB models the structured output space better and faster (using 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).
5 Conclusion

In this paper we introduced an information-theoretic learning framework to better exploit a neural network’s representation. We have also proposed an approximation that fully utilizes all parameters in a neural network and does not resort to any extra models. Our learning framework offers a principled way of interpreting and learning all layers of neural networks and encourages a more informative yet compressed representation, which is supported by qualitative empirical results. One limitation is that we consider here fully-connected feed-forward architecture with binary hidden layers. 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.

References


Under review as a conference paper at ICLR 2018


APPENDIX I

A. PROOF OF THE PREPOSITIONS

Proof of the Preposition 2: It follows from the Markov chain assumption (4) that \( p(y|z) = p(y|z_L, z_{L-1}, ..., z_1) = p(y|z_L) \) and from Jensen’s inequality that

\[
\int p(z|x) \log p(y|z) dz \leq \log \left( \int p(z|x)p(y|z) dz \right) = \log p(y|x)
\]

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. This in turn can be interpreted as matching the model distribution \( p_{\text{model}}(x; \theta) \) with the data distribution \( p_D(x) \) by minimizing their KL divergence under the empirical data distribution. Rigorously, given a set of samples \( X = \{x_1, x_2, ..., x_N\} \) i.i.d. drawn from some underlying data distribution \( p_D(x) \), 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 under the empirical data distribution. This in turn can be interpreted as matching the model distribution \( p_{\text{model}} \) with the data distribution \( p_D \) by minimizing their KL divergence to find the maximum likelihood (point) estimator for \( \theta \):

\[
\theta_{\text{ML}} = \arg \max_\theta \mathbb{E}_{x \sim p_D(x)} [\log p_{\text{model}}(x; \theta)]
\]

(24)

\[
= \arg \min_\theta [-\mathbb{E}_{x \sim p_D(x)} [\log p_{\text{model}}(x; \theta)] + \mathbb{E}_{x \sim p_D(x)} [\log p_D(x; \theta)]]
\]

(25)

\[
= \arg \min_\theta D_{\text{KL}}[p_D(x) \parallel p_{\text{model}}(x; \theta)]
\]

(26)

\[
\approx \arg \max_\theta \sum_{i=1}^N \log p_{\text{model}}(x_i; \theta)
\]

(27)

where expression (27) is an empirical estimation of expression (24) for \( N \) datapoints.

C. VARIATIONAL RELEVANCE FOR MULTIVARIATE TARGET VARIABLE

The variational conditional relevance at level \( l \) (defined by (14), (15)) for a multivariate variable \( y \) can be decomposed into the variational conditional relevances for each of its components. Indeed, without loss of generality, we assume bivariate target variable \( y = (y_1, y_2) \). It follows from the fact that the neurons within a layer are independent given some previous layer that we have

\[
\tilde{H}(Y_i|Z_i) = -\mathbb{E}_{p_D(y_1, y_2|x)} \left[ \mathbb{E}_{p(z_i|x)} [\log p(y_1, y_2|z_i)] \right]
\]

(28)

\[
= -\mathbb{E}_{p_D(y_1|x)p_D(y_2|x)} \left[ \mathbb{E}_{p(z_i|x)} [\log p(y_1|z_i) + \log p(y_2|z_i)] \right]
\]

(29)

\[
= \sum_i -\mathbb{E}_{p_D(y_1|x)p_D(y_2|x)} \left[ \mathbb{E}_{p(z_i|x)} [\log p(y_i|z_i)] \right]
\]

(30)

\[
= \sum_i \tilde{H}(Y_i|Z_i)
\]

(31)
APPENDIX II

A. LEARNED FILTERS IN PIB

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 a 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 5. The figure shows that PIB shows sharper features at many units that deterministic neural network and SFNN cannot learn.

Figure 5: The learned weights of the first layer in MNIST classification for various models: deterministic neural networks (left), SFNN (middle), and PIB (right).

B. ARCHITECTURE ANALYSIS

It is also interesting to analyze how expanding network architecture affects its learning and performance. The expansion can either in vertical direction (Figure 6, 7 and 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.

\[^{3}\text{Here we use “expansion” to mean both expanding and shrinking.}\]
Figure 6: The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-8-10-1.

Figure 7: The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-6-10-1.

Figure 8: The learning dynamic of PIB (left) and SFNN (right) in architecture of 784-10-4-10-1.