A Probabilistic Representation for Deep Learning: Delving into The Information Bottleneck Principle

Anonymous Author(s)
Affiliation
Address
email

Abstract

The Information Bottleneck (IB) principle has recently attracted great attention to explaining Deep Neural Networks (DNNs), and the key is to accurately estimate the mutual information between a hidden layer and dataset. However, some unsettled limitations weaken the validity of the IB explanation for DNNs. To address these limitations and fully explain deep learning in an information theoretic fashion, we propose a probabilistic representation for deep learning that allows the framework to estimate the mutual information, more accurately than existing non-parametric models, and also quantify how the components of a hidden layer affect the mutual information. Leveraging the probabilistic representation, we take into account the back-propagation training and derive two novel Markov chains to characterize the information flow in DNNs. We show that different hidden layers achieve different IB trade-offs depending on the architecture and the position of the layers in DNNs, whereas a DNN satisfies the IB principle no matter the architecture of the DNN.

1 Introduction

Deep learning [21] has already achieved great success in numerous applications. Deep Neural Networks (DNNs), however, are still commonly viewed as 'black boxes' [32]. Considerable efforts have been devoted to explaining the internal mechanism of DNNs from various perspectives, such as mathematics [5, 14], statistics [16, 23, 28], computer vision [43, 25], *etc.* Recently, the Information Bottleneck (IB) principle has attracted attention in opening the 'black boxes' of DNNs [35, 38].

Given a joint distribution $P(X, Y)$, the IB principle posits a random variable $T = f(X)$ obeying the Markov chain $Y \to X \to T$ and optimizes T by the IB Lagrangian [37, 36]

$$\min_{P(T|X)} I(X;T) - \beta I(Y;T), \tag{1}$$

where $f(\cdot)$ is an arbitrary function, $I(\cdot;\cdot)$ denotes mutual information, and the Lagrange multiplier $\beta > 0$ controls the IB trade-off between compressing the input X and preserving the information of the label Y. In the seminal work [35], Tishby *et al.* manifest the IB trade-off in every layer of DNNs $= \{\boldsymbol{x}; \boldsymbol{t}_1; \cdots; \boldsymbol{t}_I; \hat{\boldsymbol{y}}\}$ via studying $I(X;T_i)$ and $I(Y;T_i)$, where T_i is the random variable of the ith hidden layer $\boldsymbol{t}_i$. Especially, the authors ascribe DNN generalization to the compression [34].

In the context of deterministic DNNs, recent works reveal some limitations of the IB principle for explaining DNNs. Amjad *et al.* argue that the IB principle becomes an ill-posed optimization problem due to $I(X;T_i) = \infty$ [1], and Kolchinsky *et al.* demonstrate that not every layer of DNNs satisfies a strict IB trade-off, *i.e.*, different layers only differ in $I(X;T_i)$ but $I(Y;T_i)$ keeps consistent in all layers [17]. In addition, Saxe *et al.* experimentally show that the compression does not occur in DNNs with non-saturating activation functions, *e.g.*, the popular ReLU function [33], and Goldfeld *et al.* doubt the causality between the generalization of DNNs and the compression [11, 7]. These unsettled limitations greatly weakens the validity of the IB explanations for DNNs.

The key to examining the IB principle in DNNs is the accurate estimation of the mutual information. However, regarding DNNs as deterministic models hinders us from specifying the random variable T_i and the distribution $P(T_i)$, thus it is difficult to accurately estimate $I(X;T_i)$ and $I(Y;T_i)$. More specifically, in the absence of a clear definition of T_i, simply assuming the activations of $\boldsymbol{t}_i$ as the *i.i.d.* samples of T_i induces T_i being a continuous random variable and $I(X;T_i) = \infty$ in deterministic DNNs (see Appendix C in [33]). The complicated architecture of DNNs makes it challenging to specify $P(T_i)$. Therefore, most previous works have to indirectly estimate $P(T_i)$ via non-parametric models [40], such as the empirical distribution [35], Kernel Density Estimation (KDE) [33], and Gaussian convolution [11]. However, we experimentally confirm that classical non-parametric models derives poor mutual information estimation [29, 26] in DNNs, and one reason is because activations do not satisfy the *i.i.d.* prerequisite of non-parametric models (see Appendix G). In summary, the limitations mainly stem from the lack of an explicit probabilistic representation for deep learning.

The IB principle only formulates the information flow in DNNs $= \{\boldsymbol{x}, \boldsymbol{t}_1, \cdots, \boldsymbol{t}_I, \hat{\boldsymbol{y}}\}$ after training, and the corresponding Markov chain (see Fig. 1 in [35])

$$Y \to X \to T_1 \cdots \to T_I \to \hat{Y} \tag{2}$$

indicates that the information of Y transfers to T_i in the forward direction and T_i receives the information of Y only via X. However, training DNNs by the back-propagation [30] implies that the information of Y transfers to T_i in the backward direction during training and retains information in T_i after training. Notably, Zhang *et al*. show that a DNN can fit labels well even using Gaussian noise as input to train the DNN [44], which implies that T_i can directly receive the information of Y. Hence, the IB principle does not comprehensively characterize the information flow in DNNs.

To address the above limitations and comprehensively explain DNNs in an information theoretic fashion, we introduce the probability space $(\Omega_{T_i}, \mathcal{F}, P_{T_i})$ [6] for the ith hidden layer $\boldsymbol{t}_i$ in DNNs. Compared to previous works, the probability space $(\Omega_{T_i}, \mathcal{F}, P_{T_i})$ enables us to: (i) accurately estimate $I(X;T_i)$ and $I(Y;T_i)$ via specifying T_i and $P(T_i)$, and (ii) quantify the effect of the architecture of $\boldsymbol{t}_i$ and the back-propagation on $I(X;T_i)$ and $I(Y;T_i)$ via explicitly modeling all the ingredients of $\boldsymbol{t}_i$, such as the activation function and the weights in a probabilistic way. To the best of our knowledge, this is the first time the probability space of a hidden layer in DNNs is as defined.

Leveraging $(\Omega_{T_i}, \mathcal{F}, P_{T_i})$, we derive information theoretic explanations for DNNs as follows:

- Two Markov chains[1] characterize the information flow in DNNs $= \{\boldsymbol{x}, \boldsymbol{t}_1, \cdots, \boldsymbol{t}_I, \hat{\boldsymbol{y}}\}$

$$\begin{aligned} \bar{X} \to & T_1 \to \cdots \to T_I \to \hat{Y} \\ & T_1 \leftarrow \cdots \leftarrow T_I \leftarrow \hat{Y} \leftarrow Y. \end{aligned} \tag{3}$$

- Different hidden layers manifest different IB trade-offs depending on the architecture and the position of hidden layers in DNNs.
- A DNN satisfies the IB principle no matter the architecture of the DNN.

Preliminaries. $P(X,Y) = P(X)P(Y|X)$ is an unknown joint distribution between X and Y. A dataset $\boldsymbol{\mathcal{D}} = \{(\boldsymbol{x}^j, y^j) | \boldsymbol{x}^j \in \mathbb{R}^M, y^j \in \mathbb{Z}\}_{j=1}^J$ consists of J *i.i.d.* samples generated from $P(X,Y)$ with finite L labels, *i.e.*, $y^j \in \{1, \cdots, L\}$. In the context of supervised learning, we focus on feedfworad fully connected DNNs $= \{\boldsymbol{x}, \boldsymbol{t}_1, \cdots, \boldsymbol{t}_I, \hat{\boldsymbol{y}}\}$, *i.e.*, Multi-Layer Perceptions (MLPs) [8] for the image classification task. Without loss of generality, we use the MLP $= \{\boldsymbol{x}, \boldsymbol{t}_1, \boldsymbol{t}_2, \hat{\boldsymbol{y}}\}$ with the cross-entropy loss ℓ_{CE} for most theoretical derivations. In addition, $H(\cdot)$ denotes entropy.

In the MLP, $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ have N and K neurons, respectively, and $\boldsymbol{t}_1 = \{t_{1n} = \sigma_1[\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle]\}_{n=1}^N$, where $\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle = \sum_{m=1}^M \omega_{mn}^{(1)} \cdot x_m + b_{1n}$ is the nth dot-product given the weight $\omega_{mn}^{(1)}$ and the bias b_{1n}, and $\sigma_1(\cdot)$ denotes an activation function, *e.g.*, ReLU. Similarly, $\boldsymbol{t_2} = \{t_{2k} = \sigma_2[\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle]\}_{k=1}^K$, where $\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle = \sum_{n=1}^N \omega_{nk}^{(2)} \cdot t_{1n} + b_{2k}$. The output layer $\hat{\boldsymbol{y}}$ is softmax with L nodes

$$\hat{\boldsymbol{y}} = \{\hat{y}_l = \frac{1}{Z_Y} \exp[\langle \boldsymbol{\omega}_l^{(3)}, \boldsymbol{t}_2 \rangle] = \frac{1}{Z_Y} \exp[g_l(\boldsymbol{t_2}(\boldsymbol{t_1}(\boldsymbol{x})))]\}_{l=1}^L, \tag{4}$$

where $\langle \boldsymbol{\omega}_l^{(3)}, \boldsymbol{t}_2 \rangle = \sum_{k=1}^K \omega_{kl}^{(3)} \cdot t_{2k} + b_{yl}$ and $Z_Y = \sum_{l=1}^L \exp[\langle \boldsymbol{\omega}_l^{(3)}, \boldsymbol{t}_2 \rangle]$ is the partition function.

[1] In which the virtual random variable $\bar{X}$ has all the information of X except Y, namely $H(\bar{X}) = H(X|Y)$.

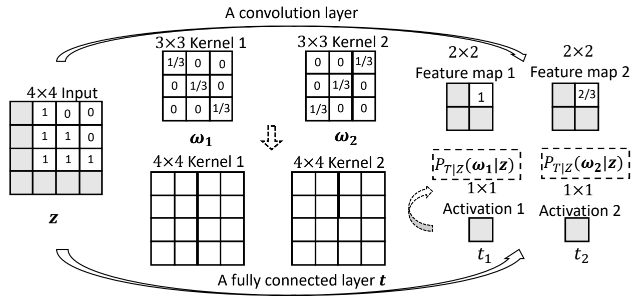

Figure 1: Given a 4×4 input $\boldsymbol{z}$, a fully connected layer $\boldsymbol{t}$ is equivalent to a convolution layer with 4×4 convolution kernels. The definition of convolution (Chapter 9.1 in [12]) implies that the 4×4 weights $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ define two global features, and the two activations t_1, t_2 indicate the cross-correlation between $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2$ and $\boldsymbol{z}$, respectively. $P_{T|Z}(\boldsymbol{\omega}_1|\boldsymbol{z})$ and $P_{T|Z}(\boldsymbol{\omega}_2|\boldsymbol{z})$ measure the probability of $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ being recognized as the feature with the largest cross-correlation to $\boldsymbol{z}$, respectively.

2 A probabilistic representation for deep learning

To accurately estimate $I(X;T_i)$ and $I(Y;T_i)$, in this section, we specify the probability space [6] for a fully connected layer and derive the probabilistic explanations of the entire MLP.

It is known that a convolution kernel (namely the weights of convolution) defines a local feature, and a convolution operation derives a feature map to measure the cross-correlation between the local feature and input in a receptive field (Chapter 9.1 in [12]). Notably, a fully connected layer is equivalent to a convolution layer with the kernel size having the same dimension as input. Thus the weights of a neuron can be viewed as a global feature, and a fully connected layer with multiple neurons derives activations to measure the cross-correlation between the multiple global features and the input. The cross-correlation explanation for a fully connected layer is visualized in Figure 1.

Assuming that a fully connected layer $\boldsymbol{t}$ consists of N neurons $\{t_n = \sigma[\langle \boldsymbol{\omega}_n, \boldsymbol{z}\rangle]\}_{n=1}^N$, where $\boldsymbol{z} \in \mathbb{R}^M$ is the input of $\boldsymbol{t}$, $\langle \boldsymbol{\omega}_n, \boldsymbol{z}\rangle = \sum_{m=1}^M \omega_{mn} \cdot z_m + b_n$ is the dot-product between $\boldsymbol{z}$ and $\boldsymbol{\omega}_n$, and $\sigma(\cdot)$ is an activation function. Based on the cross-correlation explanation, the behavior of $\boldsymbol{t}$ is to measure the cross-correlations between $\boldsymbol{z}$ and the N possible features defined by the the weights $\{\boldsymbol{\omega}_n\}_{n=1}^N$. In the context of pattern recognition [39], we define a virtual random process or 'experiment' as $\boldsymbol{t}$ recognizing one of the patterns/features with the largest cross-correlation to $\boldsymbol{z}$ from the N possible features. The experiment characterizes the behavior of $\boldsymbol{t}$ (*i.e.*, before recognizing the features with the largest cross-correlation, $\boldsymbol{t}$ must measure the cross-correlations between $\boldsymbol{z}$ and all the N possible features) while meets the requirement of the 'experiment' definition (*i.e.*, only one outcome will occur on each trial of the experiment [6]). The probability space $(\Omega_T, \mathcal{F}, P_T)$ is defined as follows:

Definition 1. $(\Omega_T, \mathcal{F}, P_T)$ consists of three components: the sample space Ω_T has N possible outcomes (features) $\{\boldsymbol{\omega}_n = \{\omega_{mn}\}_{m=1}^M\}_{n=1}^N$ defined by the weights[2] of the N neurons; the event space $\mathcal{F}$ is the σ-algebra; and the probability measure P_T is a Gibbs distribution [22] to quantify the probability of $\boldsymbol{\omega}_n$ being recognized as the feature with the largest cross-correlation to $\boldsymbol{z}$.

Taking into account the randomness of $\boldsymbol{z}$, the conditional distribution $P_{T|Z}$ is formulated as

$$P_{T|Z}(\boldsymbol{\omega}_n|\boldsymbol{z}) = \frac{1}{Z_T}\exp(t_n) = \frac{1}{Z_T}\exp[\sigma(\langle \boldsymbol{\omega}_n, \boldsymbol{z}\rangle)], \tag{5}$$

where Z is the random variable of $\boldsymbol{z}$ and $Z_T = \sum_{n=1}^N \exp(t_n)$ is the partition function.

$(\Omega_T, \mathcal{F}, P_T)$ clearly explains all the ingredients of $\boldsymbol{t}$ in a probabilistic fashion. The nth neuron defines a global feature by the weights $\boldsymbol{w}_n$ and the activation $t_n = \sigma(\langle \boldsymbol{\omega}_n, \boldsymbol{z}\rangle)$ measures the cross-correlation between $\boldsymbol{w}_n$ and $\boldsymbol{z}$. The Gibbs distribution $P_{T|Z}$ indicates that if $\boldsymbol{w}_n$ has the higher activation, *i.e.*, the larger cross-correlation to $\boldsymbol{z}$, it has the larger probability being recognized as the feature with largest cross-correlation to $\boldsymbol{z}$. For instance, if $\boldsymbol{z} \in \mathbb{R}^{16}$ and $\boldsymbol{t}$ includes $N = 2$ neurons, then $\Omega_T = \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2\}$ defines two possible outcomes (features), where $\boldsymbol{\omega}_n = \{\omega_{mn}\}_{m=1}^{16}$. $\mathcal{F} = \{\emptyset, \{\boldsymbol{\omega}_1\}, \{\boldsymbol{\omega}_2\}, \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2\}\}$ means that neither, one, or both of the features are recognized by $\boldsymbol{t}$ given $\boldsymbol{z}$, respectively. $P_{T|Z}(\boldsymbol{\omega}_1|\boldsymbol{z})$ and $P_{T|Z}(\boldsymbol{\omega}_2|\boldsymbol{z})$ are the probability of $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ being recognized as the feature with the largest cross-correlation to $\boldsymbol{z}$, respectively.

[2] We do not take into account the scalar value b_n for defining Ω_T, as it not affects the feature defined by $\boldsymbol{\omega}_n$.

$(\Omega_T, \mathcal{F}, P_T)$ explains the representation ability of deep learning. Compared to Restricted Boltzmann Machines (RBMs) [31] simply using binary units to indicate features being recognized or not given input, the Gibbs distribution[3] $P_{T|Z}(\boldsymbol{\omega}_n|\boldsymbol{z})$ measures the probability of $\boldsymbol{\omega}_n$ being recognized with the largest cross-correlation to $\boldsymbol{z}$, *i.e.*, it characterizes the relation between features and input more accurately. Moreover, Equation 5 shows that $t_n = \sigma(\langle \boldsymbol{\omega}_n, \boldsymbol{z} \rangle)$ is the negative energy function [22] of the Gibbs distribution, thus $P_{T|Z}(\boldsymbol{\omega}_n|\boldsymbol{z})$ can be derived as long as $\sigma(\langle \boldsymbol{\omega}_n, \boldsymbol{z} \rangle)$ are known because the energy function is the sufficient statistics [2] of the Gibbs distribution. That enables subsequent hidden layers to generate high-level features of input via directly processing the activations $\{t_n\}_{n=1}^N$, thus deep learning can form a hierarchical structure to represent much complex features.

$(\Omega_T, \mathcal{F}, P_T)$ answers a fundamental question: which component of a hidden layer contains the information of the layer? Since $\boldsymbol{\omega}_n$ defines Ω_T, the weights contain all the information of a layer. In particular, since the activation $t_n = \sigma(\langle \boldsymbol{\omega}_n, \boldsymbol{z} \rangle)$ is a function of $\boldsymbol{\omega}_n$, the data processing inequality [4] indicates that the information of t_n is no more than the information of $\boldsymbol{\omega}_n$. Simulations in Section 4.2 demonstrate that if activations do not correctly characterize the cross-correlation between weights and input, activations contain less information than weights do.

Based on $(\Omega_T, \mathcal{F}, P_T)$, we define the random variable T as follows:

Definition 2. Given the fully connected layer $\boldsymbol{t}$, we define the random variable $T : \Omega_T \to E_T$ as

$$T(\boldsymbol{\omega}_n) \triangleq n, \tag{6}$$

where the measurable space $E_T = \{1, \cdots, N\}$.

Since Ω_T is composed of finite N possible outcomes, T is a discrete random variable. Notably, the one-to-one correspondence between $\boldsymbol{\omega}_n$ and n indicates

$$P_{T|Z}(\boldsymbol{\omega}_n|\boldsymbol{z}) = P_{T|Z}(n|\boldsymbol{z}). \tag{7}$$

If not considering the back-propagation training, the weights (namely Ω_{T_i}) of each layer are fixed. Thus T_{i+1} entirely depends on T_i and the MLP $= \{\boldsymbol{x}; \boldsymbol{t}_1; \boldsymbol{t}_2; \hat{\boldsymbol{y}}\}$ forms a Markov chain

$$X \to T_1 \to T_2 \to \hat{Y}. \tag{8}$$

Based on the corresponding joint distribution $P(\hat{Y}, T_2, T_1|X) = P(T_1|X)P(T_2|T_1)P(\hat{Y}|T_2)$ and Definition 2, we derive a probabilistic explanation for the entire MLP, which is summarized in Theorem 1. The detailed derivation is presented in Appendix B.

Theorem 1. The MLP $= \{\boldsymbol{x}; \boldsymbol{t}_1; \boldsymbol{t}_2; \hat{\boldsymbol{y}}\}$ formulates a conditional Gibbs distribution

$$P_{\hat{Y}|X}(l|\boldsymbol{x}) = \sum_{k=1}^{K}\sum_{n=1}^{N} P(\hat{Y} = l, T_2 = k, T_1 = n|X = \boldsymbol{x}) = \frac{1}{Z_{\text{MLP}}(\boldsymbol{x})}\exp[g_l(\boldsymbol{t_2}(\boldsymbol{t_1}(\boldsymbol{x})))], \tag{9}$$

where $Z_{\text{MLP}}(\boldsymbol{x}) = \sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{n=1}^{N} P_{\hat{Y},T_2,T_1|X}(l, k, n|x)$ is the partition function.

Since $P_{\hat{Y}|X}(l|\boldsymbol{x})$ exactly equals the output $\hat{y}_l$ of the MLP, namely Equation (4), we conclude that the entire architecture of the MLP forms a family of Gibbs distribution $P_{\hat{Y}|X}(l|\boldsymbol{x})$. In general, the back-propagation updates a weight ω based on the gradient of ℓ_{CE} with respect to ω,

$$\omega(s+1) = \omega(s) - \alpha \cdot \frac{\partial \ell_{\text{CE}}}{\partial \omega(s)} = \omega(s) - \alpha \cdot \frac{\partial \text{KL}[P(Y|X)||P(\hat{Y}|X)]}{\partial \omega(s)}, \tag{10}$$

where s is the index of training iteration, α is the training rate, and $\text{KL}[\cdot||\cdot]$ is the KL-divergence.

Figure 2 summarizes the probabilistic explanation for deep learning based on the MLP. In general, a single learning iteration, an epoch, consists of two phases: training and inference (after training). During inference, the MLP bridges X and $\hat{Y}$ via multiple intermediate features Ω_{T_1}, Ω_{T_2}, and $\Omega_{\hat{Y}}$ defined by weights, and formulates the statistical relation between $\hat{Y}$ and X as a family of conditional Gibbs distribution $P(\hat{Y}|X)$. During training, the back-propagation updates weights to learn optimal intermediate features for searching an optimal $P(\hat{Y}|X)$ to accurately approximate $P(Y|X)$.

[3]Recent works about Gibbs explanations for a hidden layer are discussed in Appendix A.

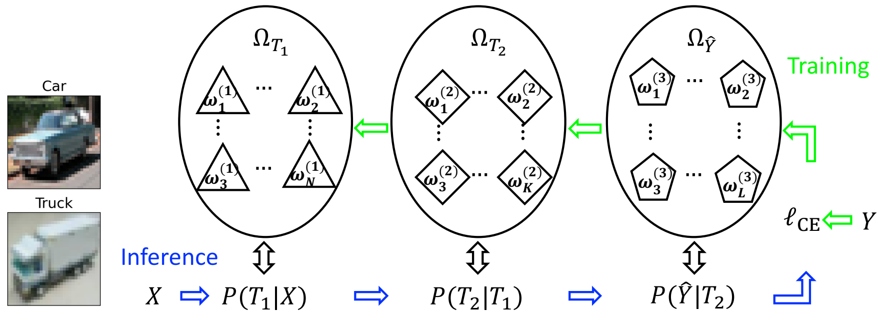

Figure 2: The visualization of the probabilistic explanation for deep learning based on the MLP.

3 The information theoretic explanations for deep learning

To address the limitations of existing IB explanations, this section proposes some novel information theoretic explanations for DNNs based on the proposed probabilistic representation.

Proposition 1. The mutual information between a fully connected layer and dataset is finite.

$$I(X;T) < \infty. \tag{11}$$

Proof: Definition 2 shows $E_T = \{1, \cdots N\}$. Thus T is a discrete random variable and $H(T) < \infty$, thereby $I(X;T) \leq H(T) < \infty$.

Proposition 1 circumvents the infinite mutual information problem. In the absence of a clear definition $T : \Omega_T \to E_T$, most previous works [33, 3, 1] simply viewing the activation t_n as the sample of T, namely $t_n \in E_T = \mathbb{R}$, implies T being continuous and gives rise to the infinite mutual information problem in deterministic DNNs. However, $(\Omega_T, \mathcal{F}, P_T)$ indicates that t_n actually is a variable measuring the cross-correlation between $\boldsymbol{w}_n$ and $\boldsymbol{z}$ rather than the sample of T, namely $t_n \notin E_T$.

Theorem 2. The information of Y flows into the MLP in the backward direction during training

$$T_1 \leftarrow T_2 \leftarrow \hat{Y} \leftarrow Y. \tag{12}$$

Proof: First, since Ω_T is defined by ω in $(\Omega_T, \mathcal{F}, P_T)$ and Equation (10) shows that $\omega(s+1)$ is determined by all the previous gradients $\{\frac{\partial \ell_{CE}}{\partial \omega(s)}\}_{s=1}^{S}$, and $\omega(0)$ is randomly initialized and α is a constant, we can derive that Ω_T is determined by $\frac{\partial \ell_{CE}}{\partial \omega}$. Second, based on the back-propagation, the relation between gradients in two adjacent layers in the MLP $= \{\boldsymbol{x}; \boldsymbol{t}_1; \boldsymbol{t}_2; \hat{\boldsymbol{y}}\}$ is formulated as

$$\frac{\partial \ell_{CE}^{\star}}{\partial \omega_{kl}^{(3)}} = [P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})] \cdot t_{2k},$$

$$\frac{\partial \ell^{\odot}_{CE}}{\partial \omega^{(2)}_{nk}} = \sum_{l=1}^{L} \frac{\partial \ell^{\star}_{CE}}{\partial \omega^{(3)}_{kl}} \cdot \omega^{(3)}_{kl} \cdot \frac{\sigma_2'(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle)}{t_{2k}} \cdot t_{1n}, \quad \frac{\partial \ell^{\diamond}_{CE}}{\partial \omega^{(1)}_{mn}} = \sum_{k=1}^{K} \frac{\partial \ell^{\odot}_{CE}}{\partial \omega^{(2)}_{nk}} \cdot \omega^{(2)}_{nk} \cdot \frac{\sigma_1'(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle)}{t_{1n}} \cdot x_m. \tag{13}$$

Equation 13 shows that $\frac{\partial \ell_{CE}}{\partial \omega^{(3)}}$ is a function of $P_{Y|X}(l|\boldsymbol{x})$ and $\frac{\partial \ell_{CE}}{\partial \omega^{(i)}}$ is a function of $\frac{\partial \ell_{CE}}{\partial \omega^{(i+1)}}$, where $\omega^{(3)}$ denotes the weight of $\hat{\boldsymbol{y}}$. The two points above enable us to derive that Ω_{T_i} is a function of $\Omega_{T_{i+1}}$ and $\Omega_{\hat{Y}}$ is a function of $P(Y|X)$. Based on Definition 2, we can further derive that T_i is a function of T_{i+1} and $\hat{Y}$ is a function of Y, *i.e.*, $T_1 \leftarrow T_2 \leftarrow \hat{Y} \leftarrow Y$. (See the detailed proof in Appendix C).

Theorem 2 is consistent with the prevailing explanation for deep learning. LeCunn *et al*. show that deep learning exploits the hierarchical property of signals [21], *i.e.*, the layers farther from output learn lower-level features, such as edges, whereas the layers closer to output assemble lower-level features into the higher-level features corresponding to labels (see Figure 2 in [43]). Notably, since lower-level features commonly exist in signals with different labels (*e.g.*, lower-level features, such as the edges of the vehicle frame and the circular contour of wheels, exist in both the car and the truck classes in the CIFAR-10 dataset [18] in Figure 2), lower-level features do not contain much information of labels. Therefore, the layers farther from output do not have much information of labels, which is consistent with the Markov chain $T_1 \leftarrow T_2 \leftarrow \hat{Y} \leftarrow Y$.

Since all the information of Y stems from X (*i.e.*, $H(Y) = I(X;Y)$ proven in Appendix D), Theorem 2 implies that partial information of X flows into the MLP in the backward direction during training. Equation (2) shows the information of X flowing into the MLP in the forward direction during inference. Overall, the information of X flows in the backward and forward directions during training and inference, respectively. As a result, the Markov chain, Equation (2), proposed by recent works could not fully characterize the information flow of X in the MLP in each epoch. In other words, $I(X;T_i)$ is not necessarily greater than $I(X;T_{i+1})$ in the MLP in each epoch.

Equation (2) shows that T_i receives the information of Y via X during inference. Theorem 2 shows that T_i also directly receives information of Y during training, because the back-propagation updates weights (*i.e.*, Ω_{T_i}) based on the label Y. Thus Equation (2) cannot fully characterize the information flow of Y in the MLP in each epoch, when we take into account the back-propagation training.

To fully characterize the information flow in the MLP in each epoch, we introduce Corollary 1.

Corollary 1. The information flow in the MLP can be characterized by two Markov chains as

$$\begin{aligned} \bar{X} \rightarrow & T_1 \rightarrow T_2 \rightarrow \hat{Y} \\ & T_1 \leftarrow T_2 \leftarrow \hat{Y} \leftarrow Y. \end{aligned} \tag{14}$$

The virtual random variable $\bar{X}$ contains all the information of X except Y, *i.e.*, $H(\bar{X}) = H(X|Y)$.

Proof of the first Markov chain: Since $\bar{X}$ does not have any information of Y, it can only flow into the MLP in the forward direction during inference. Again since $\bar{X}$ does not have any information of Y, the information flow of Y during training will not affect the information flow of $\bar{X}$. Therefore, $\bar{X} \rightarrow T_1 \rightarrow T_2 \rightarrow \hat{Y}$ characterizes the information flow of $\bar{X}$ in both training and inference phases.

Proof of the second Markov chain: Since the weights are fixed after training, the sample space and the distribution of hidden layers are fixed after training. Therefore, the information of Y transferred into hidden layers during training will retain there after training (*i.e.*, during inference). In addition, Definition 1 indicates that a fully connected layer $\boldsymbol{t} = \{t_n = \sigma(\langle \boldsymbol{\omega}_n, \boldsymbol{z} \rangle)\}_{n=1}^N$ measures the cross-correlation between $\boldsymbol{\omega}_n$ and $\boldsymbol{z}$ during inference, thus $\{\boldsymbol{\omega}_n\}_{n=1}^N$ can be viewed as a representation of Z. As a result, even though Z has all the information of Y, the information of Y that $\boldsymbol{t}$ can learn from Z is determined by how much information of Y the representation $\{\boldsymbol{\omega}_n\}_{n=1}^N$ has. Overall, the information flow of Y during inference will be the same as that during training. Based on Theorem 2, we conclude that $T_1 \leftarrow T_2 \leftarrow \hat{Y} \leftarrow Y$ characterizes the information flow of Y in the MLP in both training and inference phases. Detailed derivations and explanations are presented in Appendix E.

To quantify how much information of X and Y is learned by the MLP, we introduce Corollary 2.

Corollary 2. The mutual information between dataset and the entire MLP can be expressed as

$$\begin{aligned} I(X;T_{\text{MLP}}) &= I(\bar{X};T_1) + I(Y;\hat{Y}) \\ I(Y;T_{\text{MLP}}) &= I(Y;\hat{Y}) \end{aligned} \tag{15}$$

where T_{MLP} denotes a random variable corresponding to the entire architecture of the MLP.

Proof: Since $H(Y) = I(X;Y)$ (Appendix D), $H(X) = H(\bar{X}) + I(X;Y) = H(\bar{X}) + H(Y)$. Hence, Corollary 2 can be derived by Corollary 1 and the chain rule. The proof is in Appendix F.

4 Simulations

In this section, we propose a mutual information estimator based on $(\Omega_T, \mathcal{F}, P_T)$ and demonstrate the probabilistic representation and information theoretic explanations for deep learning on a synthetic dataset with known entropy. Additional experiments on benchmark datasets are in Appendix H.

4.1 Setup

Mutual information estimator. Based on the definition of mutual information, we have

$$I(X;T_i) = H(T_i) - H(T_i|X). \tag{16}$$

Previous works simply estimate $I(X;T_i) = H(T_i)$, because T_i is assumed to be entirely dependent on X in the Markov chain, Equation (2), thereby $H(T_i|X) = 0$. However, Corollary 1 shows that T_i depends on both X and Y if taking into account the training phase, thereby $H(T_i|X) \neq 0$.

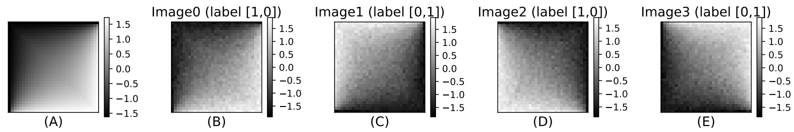

Figure 3: (A) the deterministic image $\hat{\boldsymbol{x}}$. Image0 is generated by adding $\mathcal{N}(\mu, \sigma^2)$ without rotation, Image1 is generated by rotating $\hat{\boldsymbol{x}}$ along the secondary diagonal direction and adding $\mathcal{N}(\mu, \sigma^2)$, Image2 and Image are generated by rotating $\hat{\boldsymbol{x}}$ along the vertical and horizontal directions, respectively, and adding $\mathcal{N}(\mu, \sigma^2)$.

Table 1: The number of neurons(nodes) and the activation function in the layers of the MLPs

	$\boldsymbol{x}$	$\boldsymbol{t}_1$	$\boldsymbol{t}_2$	$\hat{\boldsymbol{y}}$	$\sigma(\cdot)$
MLP1	1024 (32×32)	8	6	2	$\text{ReLU}(z) = \max(0, z)$
MLP2	1024 (32×32)	8	6	2	$\text{Tanh}(z) = (e^z - e^{-z})/(e^z + e^{-z})$
MLP3	1024 (32×32)	2	6	2	ReLU

To accurately estimate $I(X;T_i)$, we need to specify $P(T_i|X)$ and $P(T_i)$. Based on $(\Omega_{T_i}, \mathcal{F}, P_{T_i})$, we formulate $P_{T_i|X}(n|\boldsymbol{x}^j)$ of the three fully connected layers in the MLP as

$$P_{T_1|X}(n|\boldsymbol{x}^j) = \tfrac{1}{Z_{F_1}}\exp[\sigma_1(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}^j \rangle)], \;\; P_{T_2|X}(k|\boldsymbol{x}^j) = \tfrac{1}{Z_{F_2}}\exp[\sigma_2(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1(\boldsymbol{x}^j) \rangle)], \\ P_{\hat{Y}|X}(l|\boldsymbol{x}^j) = \tfrac{1}{Z_{F_Y}}\exp[\langle \boldsymbol{\omega}_l^{(3)}, \boldsymbol{t}_2(\boldsymbol{t}_1(\boldsymbol{x}^j)) \rangle]. \tag{17}$$

To derive the marginal distribution $P(T_i)$, we sum the joint distribution $P(T_i, X)$ over $\boldsymbol{x} \in \mathcal{X}$,

$$P(T_i = n) = \textstyle\sum_{\boldsymbol{x} \in \mathcal{X}} P_X(\boldsymbol{x}) P_{T_i|X}(n|\boldsymbol{x}) \approx \sum_{\boldsymbol{x}^j \in \mathcal{D}} P_X(\boldsymbol{x}^j) P_{T_i|X}(n|\boldsymbol{x}^j) = \frac{1}{J}\sum_{\boldsymbol{x}^j \in \mathcal{D}} P_{T_i|X}(n|\boldsymbol{x}^j), \tag{18}$$

where $P_X(\boldsymbol{x}^j)$ is estimated by the empirical distribution $1/J$ given $\mathcal{D}$. Finally, we can derive $I(X;T_i)$ by Equation 16, 17, and 18. Similarly, based on the definition of mutual information, we have

$$I(Y;T_i) = H(T_i) - H(T_i|Y). \tag{19}$$

To estimate $H(T_i|Y)$, we reformulate $P(T_i|Y)$ as

$$P_{T_i|Y}(n|l) = \textstyle\sum_{\boldsymbol{x} \in \mathcal{X}} P_{T_i|X}(n|\boldsymbol{x}) P_{X|Y}(\boldsymbol{x}|l) \approx \frac{1}{N(l)} \sum_{\boldsymbol{x}^j \in \mathcal{D}, y^j = l} P_{T_i|\boldsymbol{X}}(n|\boldsymbol{x}^j), \tag{20}$$

where $P_{X|Y}(\boldsymbol{x}^j|l)$ is estimated by the empirical distribution $1/N(l)$ and $N(l)$ denotes the number of samples with the label l in $\mathcal{D}$. Finally, we can derive $I(Y;T_i)$ by Equation 18, 19, and 20.

Synthetic dataset. The dataset consists of 512 gray-scale 32×32 images, which are evenly generated by rotating a deterministic image $\hat{\boldsymbol{x}}$ in four different orientations and adding Gaussian noise with expectation $\mu = \mathbb{E}(\hat{\boldsymbol{x}})$ and variance $\sigma^2 = 1$, namely $\boldsymbol{x} = r(\hat{\boldsymbol{x}}) + \mathcal{N}(\mu, \sigma^2)$, where $r(\cdot)$ denotes the rotation method shown in Figure 3. The reason for adding Gaussian noise is to avoid DNNs directly memorizing the deterministic image. In addition, the binary labels [1,0] and [0,1] evenly divide the synthetic dataset into two classes. As a result, the synthetic dataset has (approximately) 2 bits information and the labels have 1 bit information. Compared to popular benchmark dataset with unknown features and entropy, *e.g.*, MNIST [19] and Fashion-MNIST [41], the features and the entropy of the synthetic dataset are clear and known, which enables us to examine the probabilistic representation and the mutual information estimator.

Neural Networks. We train three MLPs, namely MLP1, MLP2 and MLP3, on the synthetic dataset by a variant of Stochastic Gradient Descent (SGD) method, namely Adam [15], over 1000 epochs with the learning rate $\alpha = 0.03$. Table 1 summarizes the architecture of the three MLPs.

4.2 Validating the probability space and the mutual information estimator

We demonstrate the sample space Ω_T by visualizing the weights[4] of the eight neurons in $\boldsymbol{t}_1$, *i.e.*, $\boldsymbol{\omega}_n^{(1)} = \{\omega_{mn}^{(1)}\}_{m=1}^{1024}$, in 5 different epochs (*i.e.*, 0,1,4,128,1000) in Figure 4 (Left). As training continues, we observe that $\boldsymbol{\omega}_n^{(1)}$ quickly learns all the spatial features of the synthetic dataset. For instance, $\boldsymbol{\omega}_2^{(1)}$ has low magnitude at top-left positions and high magnitude at bottom-right positions, which correctly characterizes the spatial feature of Image0. Similarly, $\boldsymbol{\omega}_3^{(1)}$, $\boldsymbol{\omega}_4^{(1)}$, and $\boldsymbol{\omega}_5^{(1)}$ correctly characterize the spatial feature of Image1, Image2, and Image3 in Figure 3, respectively.

[4]We only show the learned weights in MLP1 because we observe that the learned weights in MLP1 and MLP2 are very similar, though they use different activation functions.

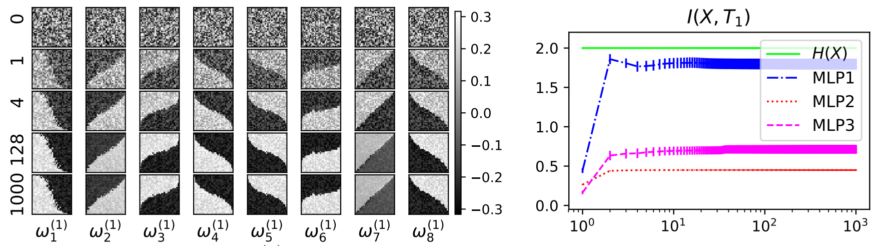

Figure 4: (Left) The eight features $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^{8}$ learned by the weights of the eight neurons in 5 different epochs (*i.e.*, 0,1,4,128,1000), where $\boldsymbol{\omega}_n^{(1)} = \{\omega_{mn}^{(1)}\}_{m=1}^{1024}$ are reshaped into 32×32 to show the spatial structure. (Right) The variation of $I(X;T_1)$ in the MLP1, MLP2, and MLP3 during 1000 epochs.

Table 2: The Gibbs probability $P_{F_1|X}(\boldsymbol{\omega}_n^{(1)}|\text{Image0})$ in MLP1 and MLP2 in the 1000 epoch

	$\boldsymbol{\omega}_1^{(1)}$	$\boldsymbol{\omega}_2^{(1)}$	$\boldsymbol{\omega}_3^{(1)}$	$\boldsymbol{\omega}_4^{(1)}$	$\boldsymbol{\omega}_5^{(1)}$	$\boldsymbol{\omega}_6^{(1)}$	$\boldsymbol{\omega}_7^{(1)}$	$\boldsymbol{\omega}_8^{(1)}$
$\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle$	-63.6	**208.8**	-181.6	45.1	-55.6	157.5	-210.0	-30.1
$f_{1n}^{\text{ReLU}}(\boldsymbol{x})$	0.0	**208.8**	0.0	45.1	0.0	157.5	0.0	0.0
$\exp[f_{1n}^{\text{ReLU}}(\boldsymbol{x})]$	1.0	**4.79e+90**	1.0	3.86e+19	1.0	2.51e+68	1.0	1.0
$P_{T_1\|X}^{\text{ReLU}}$	0.0	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0
$f_{1n}^{\text{Tanh}}(\boldsymbol{x})$	-1.0	**1.0**	-1.0	**1.0**	-1.0	**1.0**	-1.0	-1.0
$\exp[f_{1n}^{\text{Tanh}}(\boldsymbol{x})]$	0.36	**2.71**	0.36	**2.71**	0.36	**2.71**	0.36	0.36
$P_{T_1\|X}^{\text{Tanh}}$	0.037	**0.272**	0.037	**0.272**	0.037	**0.272**	0.037	0.037

$f_{1n}^{\text{Tanh}}(\boldsymbol{x}) = \sigma^{\text{Tanh}}(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle)$ and $f_{1n}^{\text{ReLU}}(\boldsymbol{x}) = \sigma^{\text{ReLU}}(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle)$ are the activations given the same $\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle$.

We demonstrate that $P(T_1|X)$ correctly measures the probability of $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^{8}$ being recognized the feature with the largest cross-correlation to $\boldsymbol{x}$ in Table 2. For instance, $\boldsymbol{\omega}_2^{(1)}$ correctly characterizes the feature of Image0 and has the largest cross-correlation $\langle \boldsymbol{\omega}_2^{(1)}, \boldsymbol{x} \rangle = 190.8$, thus it has the largest probability $P_{T_1|X}^{\text{ReLU}}(\boldsymbol{\omega}_2^{(1)}|\text{Image0}) = 1.0$ being recognized as the feature with largest cross-correlation to Image0. In contrast, since $\boldsymbol{\omega}_7^{(1)}$ incorrectly characterizes the feature of Image0 and has the lowest cross-correlation $\langle \boldsymbol{\omega}_7^{(1)}, \boldsymbol{x} \rangle = -210.0$, so it has the lowest probability $P_{T_1|X}^{\text{ReLU}}(\boldsymbol{\omega}_7^{(1)}|\text{Image0}) = 0.0$ being recognized as the feature with largest cross-correlation to Image0.

We observe that an activation function (abbr. ACT) plays an important role in the distribution. Specifically, ReLU, a non-saturating (unbounded) ACT [10], preserves the positive cross-correlations while resets all the negative ones as zero. $P_{T_1|X}^{\text{ReLU}}(\boldsymbol{\omega}_2^{(1)}|\text{Image0}) = 1.0$ shows that ReLU derives the correct probability of $\omega_2^{(1)}$ being recognized as the feature with largest cross-correlation. In contrast, though $\boldsymbol{\omega}_2^{(1)}$ has stronger cross-correlation to Image0 than $\boldsymbol{\omega}_4^{(1)}$, *i.e.*, $\langle \boldsymbol{\omega}_2^{(1)}, \boldsymbol{x} \rangle > \langle \boldsymbol{\omega}_4^{(1)}, \boldsymbol{x} \rangle$, Tanh, a saturating (bounded) ACT, derives $f_{12}^{\text{Tanh}}(\boldsymbol{x}) = f_{14}^{\text{Tanh}}(\boldsymbol{x}) = 1.0$, and makes $\boldsymbol{\omega}_4^{(1)}$ to incorrectly have the same probability 0.272 to $\boldsymbol{\omega}_2^{(1)}$ being recognized as the feature with the largest cross-correlation to Image0, *i.e.*, Tanh hinders $\boldsymbol{t}_1$ from correctly recognizing the features of input. The simulations for validating the probability space based on other synthetic images are presented in Appendix G.

To validate the mutual information estimator, we follow recent works [35, 33] to train the three MLPs with 50 different random initialization and study the average mutual information. Figure 4 (Right) shows that $I(X;T_1)$ quickly increases to 1.81 and keeps stable in the MLP1, *i.e.*, $\boldsymbol{t}_1$ learns most information of the dataset as $H(X) = 2.0$. Notably, the result is consistent with the variation of the weights in Figure 4 (Left), which shows that the weights correctly characterize the features of the dataset and keeps stable after the fourth epoch. As a comparison, we observe that $I(X;T_1)$ keeps stable at 0.44 in the MLP2, which confirms the statement that Tanh hinders $\boldsymbol{t}_1$ from correctly recognizing the features of input. In addition, Figure 4 (Right) shows that $I(X;T_1) \approx 0.79$ in MLP3 is smaller than $I(X;T_1) \approx 1.81$ in MLP1, which is consistent with Definition 1, *i.e.*, a layer with fewer neurons would represent fewer possible features, thus it contains less information.

In summary, we demonstrate the probability space $(\Omega_T, \mathcal{F}, P_T)$ and show that if an ACT cannot preserve the cross-correlation between weights(features) and input, it would distort the distribution of a layer, thereby affecting the mutual information between the layer and data/labels. In addition, we show that the proposed mutual information estimator outperforms the existing non-parametric models, *e.g.*, empirical distribution [35] and KDE [33], based on the synthetic dataset. Especially, activations do not satisfy the *i.i.d.* prerequisite of non-parametric models is an important reason for non-parametric models deriving inaccurate mutual information in DNNs. Due to limited space, the experimental comparison and study of non-parametric models are presented in Appendix G.

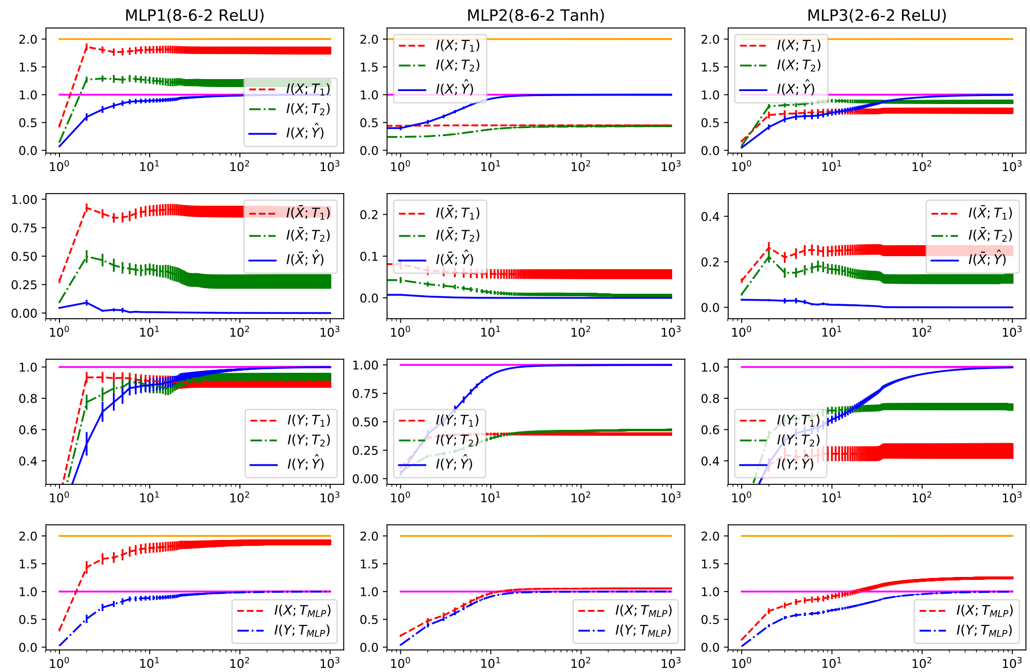

Figure 5: All the x-axis index training epochs. In each column, the first three figures show $I(X;T_i)$, $I(\bar{X};T_i)$, and $I(Y;T_i)$ respectively. The forth figure shows $I(X;T_{\text{MLP}})$ and $I(Y;T_{\text{MLP}})$ in a MLP. The pink line denotes $H(Y) = 1.0$ and the orange line denotes $H(X) = 2.0$.

4.3 Validating the information theoretic explanations for DNNs

In Figure 5, we observe $I(X;T_i) \leq I(X;\hat{Y})$ in MLP2 and MLP3, which confirms that the Markov chain proposed by previous works, Equation (2), cannot fully explain the information flow in MLPs, if taking into account the back-propagation training. As a comparison, the second and third row show $I(\bar{X};T_1) \geq I(\bar{X};T_2) \geq I(\bar{X};\hat{Y})$ and $I(Y;T_1) \leq I(Y;T_2) \geq I(Y;\hat{Y})$ in all the three MLPs, which validates that Corollary 1, *i.e.*, Equation (14) characterizes the information flow in MLPs.

Figure 5 demonstrates that different hidden layers achieve different IB trade-offs depending on the architecture and the position of the layers in MLPs. In terms of architecture, $I(Y;T_1) > 0.8$ and $I(\bar{X};T_1) > 0.75$ in MLP1 indicate that $\boldsymbol{t}_1$, with ReLU, achieves a good prediction without much compression, whereas $I(Y;T_1) < 0.5$ and $I(\bar{X};T_1) < 0.1$ in MLP2 show that $\boldsymbol{t}_1$, with Tanh, achieves a different IB trade-off. In addition, $I(Y;T_1) \approx 0.45$ and $I(\bar{X};T_1) \approx 0.25$ in MLP3 show the effect of neuron numbers on the IB trade-off. In terms of position, $I(Y;\hat{Y}) = 1$ and $I(\bar{X};\hat{Y}) = 0$ in MLP1 means that $\hat{\boldsymbol{y}}$ has a different IB trade-off to $\boldsymbol{t}_1$ in MLP1.

We demonstrate that a MLP satisfies the IB principle no matter what the architecture of the MLP is. Figure 5 visualizes $I(X;T_{\text{MLP}})$ and $I(Y;T_{\text{MLP}})$ based on Corollary 2. It shows that all of three MLPs satisfy the IB principle, namely $I(X;T_{\text{MLP}}) < H(X) = 2$ and $I(Y;T_{\text{MLP}}) = H(Y) = 1$, though they have different architectures. Importantly, in contrast to previous work [33] claiming that the compression not exists in DNNs with non-saturating ACT, such as ReLU, Figure 5 clearly shows that the compression exists in all the MLPs, no matter the activation function of MLPs.

We further demonstrate the information theoretic explanations for DNNs on the benchmark MNIST and Fashion-MNIST datasets. The experiments are presented in Appendix H.

5 Conclusion and future work

In this work, we (1) specify the probability space for a hidden layer for (2) accurately estimating the mutual information and (3) clearly explaining how the components of the layer affect the mutual information. We take into account the back-propagation training and derive two novel Markov chains to characterize the information flow in DNNs. Furthermore, we demonstrate that a DNN satisfies the IB principle no matter the architecture of the DNN. In contrast, different hidden layers show different IB trade-offs depending on the architecture and the position of the layers in DNNs. A potential direction is to study the generalization of DNNs based on the probabilistic representation.

References

[1] Rana Ali Amjad and Bernhard Claus Geiger. Learning representations for neural network-based classification using the information bottleneck principle. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[2] George Casella and Roger L Berger. *Statistical inference*. Cengage Learning, 2021.

[3] Ivan Chelombiev, Conor Houghton, and Cian O'Donnell. Adaptive estimators show information compression in deep neural networks. In *International Conference on Learning Representations*, 2019.

[4] Thomas Cover and Joy Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, New Jersy, 2006.

[5] Balázs Csanád Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7, 2001.

[6] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[7] Marylou Gabrié, Andre Manoel, Clément Luneau, Nicolas Macris, Florent Krzakala, Lenka Zdeborová, et al. Entropy and mutual information in models of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1821–1831, 2018.

[8] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[9] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions. on Pattern Analysis and Machine Intelligence*, pages 721–741, June 1984.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[11] Ziv Goldfeld, Ewout Van Den Berg, Kristjan Greenewald, Igor Melnyk, Nam Nguyen, Brian Kingsbury, and Yury Polyanskiy. Estimating information flow in deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2299–2308, 2019.

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[13] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

[14] Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks. In *Conference on Learning Theory*, pages 2306–2327. PMLR, 2020.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

[17] Artemy Kolchinsky, Brendan D Tracey, and Steven Van Kuyk. Caveats for information bottleneck in deterministic scenarios. *arXiv preprint arXiv:1808.07593*, 2018.

[18] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technique Report*, 2009.

[19] Y. LeCun, BE. Boser, and JS. Denker. Handwritten digit recognition with a back-propagation network. In *NeurIPS*, pages 396–494, 1990.

[20] Y. LeCun, L. Bottou, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 11:2278–2324, 1998.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[22] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. *A tutorial on energy-based learning*. MIT Press, 2006.

[23] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. In *ICLR*, 2018.

[24] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

[25] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[26] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pages 875–884. PMLR, 2020.

[27] Pankaj Mehta and David J. Schwab. An exact mapping between the variational renormalization group and deep learning. *arXiv preprint arXiv:1410.3831*, 2014.

[28] Julian D Olden and Donald A Jackson. Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological modelling*, 154(1-2):135–150, 2002.

[29] Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

[30] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, October 1986.

[31] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR, 2009.

[32] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

[33] Andrew Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Tracey, and David Cox. On the information bottleneck theory of deep learning. In *International Conference on Representation Learning*, 2018.

[34] Ohad Shamir, Sivan Sabato, and Naftali Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711, 2010.

[35] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[36] Noam Slonim. *The information bottleneck: Theory and applications*. PhD thesis, Citeseer, 2002.

[37] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[38] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.

[39] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[40] Larry Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

[41] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[42] Sho Yaida. Non-gaussian processes and neural networks at finite widths. *arXiv preprint arXiv:1910.00019*, 2019.

[43] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[44] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] see Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] see the URL in Appendix G
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Section 4.1, Appendix G, and Appendix H
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see Section 4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see Appendix G
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] see Appendix H
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Recent works about Gibbs explanations for a hidden layer

As a fundamental probabilistic graphic model, the Gibbs distribution (a.k.a., Boltzmann distribution, the energy based model, or the renormalization group) formulates the dependence within X by associating an energy $E(\boldsymbol{x};\boldsymbol{\theta})$ to each dependence structure [9].

$$P(X;\boldsymbol{\theta},\beta) = \frac{1}{Z(\boldsymbol{\theta},\beta)}\exp[-\beta E(\boldsymbol{x};\boldsymbol{\theta})], \tag{21}$$

where $E(\boldsymbol{x};\boldsymbol{\theta})$ is the energy function, $\boldsymbol{\theta}$ denote the parameters of $E(\boldsymbol{x};\boldsymbol{\theta})$, β is the inverse temperature constant. Since β can be absorbed into $\boldsymbol{\theta}$, $P(X;\boldsymbol{\theta},\beta)$ can be simplified as

$$P(X;\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\exp[-E(\boldsymbol{x};\boldsymbol{\theta})], \tag{22}$$

where the partition function[5] is defined as

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}\in\mathcal{X}}\exp[-E(\boldsymbol{x};\boldsymbol{\theta})]. \tag{23}$$

The Gibbs distribution has three appealing properties. First, the deterministic energy function $E(\boldsymbol{x};\boldsymbol{\theta})$ is a sufficient statistics of $P(X;\boldsymbol{\theta})$. The property allows us to explain a deterministic function, e.g., a hidden layer, in a probabilistic way. Second, a Gibbs distribution can be easily reformulated as various probabilistic models via redefining $E(\boldsymbol{x};\boldsymbol{\theta})$, which allows us to clarify the complicated architecture of a hidden layer. For example, if the energy function is defined as the summation of multiple functions, namely $E(\boldsymbol{x};\boldsymbol{\theta}) = -\sum_k f_k(\boldsymbol{x};\boldsymbol{\theta}_k)$, the Gibbs distribution would be the Product of Experts (PoE) model, *i.e.*, $P(\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\prod_k F_k$, where $F_k = \exp[-f_k(\boldsymbol{x};\boldsymbol{\theta}_k)]$ and $Z(\boldsymbol{\theta}) = \prod_k Z(\boldsymbol{\theta}_k)$ [13]. Third, the energy minimization is a typical optimization for $\boldsymbol{\theta}$, namely $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} E(\boldsymbol{x};\boldsymbol{\theta})$ [22], which allows us to explain the back-propagation training, as the energy minimization can be implemented by the gradient descent algorithm as long as $E(\boldsymbol{x};\boldsymbol{\theta})$ is differentiable.

A well-known Gibbs distribution model in machine learning is the Restricted Boltzmann Machines (RBMs) [31, 27]. Though Yaida indirectly proves the distribution of a fully connected layer as a Gibbs distribution [42], and Lin *et al.* clarify certain advantages of DNNs based on the Gibbs distribution [24], there is few work to extend the Gibbs explanation to complicated hidden layers, e.g., fully connected layers and convolutional layers.

B The marginal distribution of the MLP

Since the entire architecture of the MLP $= \{\boldsymbol{x},\boldsymbol{t}_1,\boldsymbol{t}_2,\hat{\boldsymbol{y}}\}$ corresponds to a joint distribution

$$P(\hat{Y},T_2,T_1|X) = P(\hat{Y}|T_2)P(T_2|T_1)P(T_1|X), \tag{24}$$

the marginal distribution $P(\hat{Y}|X)$ can be formulated as

$$\begin{aligned} P_{\hat{Y}|X}(l|\boldsymbol{x}) &= \sum_{k=1}^{K}\sum_{n=1}^{N} P(\hat{Y}=l,T_2=k,T_1=n|X=\boldsymbol{x}) \\ &= \sum_{k=1}^{K}\sum_{n=1}^{N} P_{\hat{Y}|T_2}(l|k)P_{T_2|T_1}(k|n)P_{T_1|X}(n|\boldsymbol{x}). \end{aligned} \tag{25}$$

Based on the definition of the Gibbs probability measure (Equation 5), we have

$$P_{T_1|X}(n|x) = \frac{1}{Z_{T_1}}\exp(t_{1n}) = \frac{1}{Z_{T_1}}\exp[\sigma_1(\langle\boldsymbol{\omega}_n^{(1)},\boldsymbol{x}\rangle)], \tag{26}$$

where $\langle\boldsymbol{\omega}_n^{(1)},\boldsymbol{x}\rangle = \sum_{m=1}^{M}\omega_{mn}^{(1)}\cdot x_m + b_{1n}$. Similarly, we have

$$P_{T_2|T_1}(k|n) = \frac{1}{Z_{T_2}}\exp(t_{2k}) = \frac{1}{Z_{T_2}}\exp[\sigma_2(\langle\boldsymbol{\omega}_k^{(2)},\boldsymbol{t}_1\rangle)], \tag{27}$$

where $\langle\boldsymbol{\omega}_k^{(2)},\boldsymbol{t}_1\rangle = \sum_{n=1}^{N}\omega_{nk}^{(2)}\cdot t_{1n} + b_{2k}$. Thus we have

$$\begin{aligned} &\sum_{n=1}^{N} P_{T_2|T_1}(k|n)P_{T_1|X}(n|x) \\ =&\frac{1}{Z_{T_2}}\frac{1}{Z_{T_1}}\sum_{n=1}^{N}\exp[\sigma_2(\langle\boldsymbol{\omega}_k^{(2)},\boldsymbol{t}_1\rangle)]\exp[\sigma_1(\langle\boldsymbol{\omega}_n^{(1)},\boldsymbol{x}\rangle)]. \end{aligned} \tag{28}$$

[5] We only consider the discrete case in the paper.

Since $\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle = \sum_{n=1}^{N} \omega_{nk}^{(2)} \cdot t_{1n} + b_{2k}$ is a constant with respect to n, we have

$$\begin{aligned} &\sum_{n=1}^{N} P_{T_2|T_1}(k|n) P_{T_1|X}(n|x) \\ =&\frac{1}{Z_{T_2}} \frac{1}{Z_{T_1}} \exp[\sigma_2(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle)] \sum_{n=1}^{N} \exp[\sigma_1(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle)]. \end{aligned} \tag{29}$$

In addition, $\sum_{n=1}^{N} \exp[\sigma_1(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle)] = Z_{T_1}$, thus we have

$$\sum_{n=1}^{N} P_{T_2|T_1}(k|n) P_{T_1|X}(n|\boldsymbol{x}) = \frac{1}{Z_{T_2}} \exp[\sigma_2(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle)]. \tag{30}$$

Therefore, we can simplify $P_{\hat{Y}|X}(l|\boldsymbol{x})$ as

$$\begin{aligned} P_{\hat{Y}|X}(l|\boldsymbol{x}) &= \sum_{k=1}^{K} P_{\hat{Y}|T_2}(l|k) \sum_{n=1}^{N} P_{T_2|T_1}(k|n) P_{T_1|X}(t|\boldsymbol{x}) \\ &= \sum_{k=1}^{K} P_{\hat{Y}|T_2}(l|k) \frac{1}{Z_{T_2}} \exp[\sigma_2(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle)]. \end{aligned} \tag{31}$$

Since $P_{\hat{Y}|T_2}(l|k) = \frac{1}{Z_{\hat{Y}}} \exp[\sigma_3(\langle \boldsymbol{\omega}_l^{(3)}, \boldsymbol{t}_2 \rangle)]$ and $\langle \boldsymbol{\omega}_l^{(3)}, \boldsymbol{t}_2 \rangle = \sum_{k=1}^{K} \omega_{lk}^{(3)} f_{2k} + b_{yl}$ is a constant with respect to k, we can derive

$$P_{\hat{Y}|X}(l|\boldsymbol{x}) = P_{\hat{Y}|T_2}(l|k) \sum_{k=1}^{K} \frac{1}{Z_{T_2}} \exp[\sigma_2(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle)]. \tag{32}$$

Since $Z_{T_2} = \sum_{k=1}^{K} \exp[\sigma_2(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle)]$ is also constant to k,

$$\begin{aligned} P_{\hat{Y}|X}(l|\boldsymbol{x}) &= P_{\hat{Y}|T_2}(l|k) \frac{1}{Z_{T_2}} \sum_{k=1}^{K} \exp[\sigma_2(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle)]. \\ &= P_{\hat{Y}|T_2}(l|k) = \frac{1}{Z_{F_Y}} \exp[\langle \boldsymbol{\omega}_l^{(3)}, \boldsymbol{t}_2 \rangle]. \end{aligned} \tag{33}$$

In addition, since $\boldsymbol{t}_2 = \{t_{2k}\}_{k=1}^{K} = \{\sigma_2(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle)\}_{k=1}^{K}$, we can extend $P_{\hat{Y}|X}(l|\boldsymbol{x})$ as

$$\begin{aligned} P_{\hat{Y}|X}(l|\boldsymbol{x}) &= P_{\hat{Y}|F_2}(l|k) = \frac{1}{Z_{F_Y}} \exp[\langle \boldsymbol{\omega}_l^{(3)}, \boldsymbol{t}_2 \rangle] \\ &= \frac{1}{Z_{\hat{Y}}} \exp[\langle \boldsymbol{\omega}_l^{(3)}, \begin{pmatrix} \sigma_2(\langle \boldsymbol{\omega}_1^{(2)}, \boldsymbol{t}_1 \rangle) \\ \vdots \\ \sigma_2(\langle \boldsymbol{\omega}_K^{(2)}, \boldsymbol{t}_1 \rangle) \end{pmatrix} \rangle]. \end{aligned} \tag{34}$$

Since $\boldsymbol{t}_1 = \{t_{1n}\}_{n=1}^{N} = \{\sigma_1(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle)\}_{n=1}^{N}$, we can further extend $P_{\hat{Y}|X}(l|\boldsymbol{x})$ as

$$\begin{aligned} P_{\hat{Y}|X}(l|\boldsymbol{x}) &= \frac{1}{Z_{\hat{Y}}} \exp[\langle \boldsymbol{\omega}_l^{(3)}, \begin{pmatrix} \sigma_2(\langle \boldsymbol{\omega}_1^{(2)}, \begin{pmatrix} \sigma_1(\langle \boldsymbol{\omega}_1^{(1)}, \boldsymbol{x} \rangle) \\ \vdots \\ \sigma_1(\langle \boldsymbol{\omega}_N^{(1)}, \boldsymbol{x} \rangle) \end{pmatrix} \rangle) \\ \vdots \\ \sigma_2(\langle \boldsymbol{\omega}_K^{(2)}, \begin{pmatrix} \sigma_1(\langle \boldsymbol{\omega}_1^{(1)}, \boldsymbol{x} \rangle) \\ \vdots \\ \sigma_1(\langle \boldsymbol{\omega}_N^{(1)}, \boldsymbol{x} \rangle) \end{pmatrix} \rangle) \end{pmatrix} \rangle] \\ &= \frac{1}{Z_{\text{MLP}}(\boldsymbol{x})} \exp[g_l(\boldsymbol{t}_2(\boldsymbol{t}_1(\boldsymbol{x})))]. \end{aligned} \tag{35}$$

Overall, we prove $P_{\hat{Y}|X}(l|\boldsymbol{x})$ as the Gibbs distribution expressed as

$$P_{\hat{Y}|X}(l|\boldsymbol{x}) = \frac{1}{Z_{\text{MLP}}(\boldsymbol{x})} \exp[g_l(\boldsymbol{t}_2(\boldsymbol{t}_1(\boldsymbol{x})))]. \tag{36}$$

where $E_l(x) = -g_l(\boldsymbol{t}_2(\boldsymbol{t}_1(\boldsymbol{x})))$ is the energy function and the partition function

$$\begin{aligned} Z_{\text{MLP}}(\boldsymbol{x}) &= \sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{t=1}^{T} P(\hat{Y}, T_2, T_1 | X = \boldsymbol{x}) \\ &= \sum_{l=1}^{L} \exp[g_l(\boldsymbol{t}_2(\boldsymbol{t}_1(\boldsymbol{x})))]. \end{aligned} \tag{37}$$

C The proof of Theorem 2

Based on the definition of the cross entropy, ℓ_{CE} can be formulated as

$$\ell_{\text{CE}} = -\sum_{l=1}^{L} P_{Y|X}(l|\boldsymbol{x}) \log P_{\hat{Y}|X}(l|\boldsymbol{x}). \tag{38}$$

where $P_{\hat{Y}|X}(l|\boldsymbol{x})$ is the output of the MLP, and $P_{Y|X}(l|\boldsymbol{x})$ is the one-hot probability of $\boldsymbol{x}$ given the label y, *i.e.*,

$$P_{Y|X}(l|\boldsymbol{x}) = \begin{cases} 1 & \text{for} \quad l = y \\ 0 & \text{for} \quad l \neq y \end{cases} \tag{39}$$

The derivative of ℓ_{CE} with respect to $P_{\hat{Y}|X}(l|\boldsymbol{x})$ is

$$\frac{\partial \ell_{\text{CE}}}{\partial P_{\hat{Y}|X}(l|\boldsymbol{x})} = -\frac{P_{Y|X}(l|\boldsymbol{x})}{P_{\hat{Y}|X}(l|\boldsymbol{x})}. \tag{40}$$

Since $P_{\hat{Y}|X}(l|\boldsymbol{x})$ can be expressed as

$$P_{\hat{Y}|X}(l|\boldsymbol{x}) = \frac{1}{Z_{\text{MLP}}(\boldsymbol{x})} \exp[g_l(\boldsymbol{t}_2\boldsymbol{t}_1(\boldsymbol{x}))], \tag{41}$$

the derivative of $P_{\hat{Y}|X}(z|\boldsymbol{x})$ with respect to $g_l(\boldsymbol{t}_2\boldsymbol{t}_1(\boldsymbol{x}))$ is

$$\frac{\partial P_{\hat{Y}|X}(z|\boldsymbol{x})}{\partial g_l} = \frac{\frac{1}{Z_{\text{MLP}}}\exp(g_z)}{\partial g_l} = \begin{cases} P_{\hat{Y}|X}(l|\boldsymbol{x}) \cdot [1 - P_{\hat{Y}|X}(l|\boldsymbol{x})] & \text{for} \quad z = l \\ -P_{\hat{Y}|X}(l|\boldsymbol{x}) \cdot P_{\hat{Y}|X}(z|\boldsymbol{x}) & \text{for} \quad z \neq l \end{cases}. \tag{42}$$

Overall, the derivative of ℓ_{CE} with respect to g_l can be expressed as

$$\begin{aligned} \frac{\partial \ell_{\text{CE}}}{\partial g_l} &= \sum_{z=1}^{L} \frac{\partial \ell_{\text{CE}}}{\partial P_{\hat{Y}|X}(z|\boldsymbol{x})} \frac{\partial P_{\hat{Y}|X}(z|\boldsymbol{x})}{\partial g_l} \\ &= -P_{Y|X}(l|\boldsymbol{x})(1 - P_{\hat{Y}|X}(l|\boldsymbol{x}) + \sum_{z \neq l} P_{Y|X}(z|\boldsymbol{x}) P_{\hat{Y}|X}(l|\boldsymbol{x}) \\ &= P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x}). \end{aligned} \tag{43}$$

Since $g_l = \langle \boldsymbol{\omega}_l^{(3)}, \boldsymbol{t}_2 \rangle = \sum_{k=1}^{K} \omega_{kl}^{(3)} \cdot t_{2k} + b_{yl}$, the derivative of ℓ_{CE} with respect to $\omega_{kl}^{(3)}$ can be expressed as

$$\frac{\partial \ell_{\text{CE}}}{\partial \omega_{kl}^{(3)}} = \frac{\partial \ell_{\text{CE}}}{\partial g_l} \frac{\partial g_l}{\partial \omega_{kl}^{(3)}} = [P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})] t_{2k}. \tag{44}$$

Similarly, the derivative of ℓ_{CE} with respect to $\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle$ can be expressed as

$$\begin{aligned} \frac{\partial \ell_{\text{CE}}}{\partial \langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle} &= \sum_{l=1}^{L} \frac{\partial \ell_{\text{CE}}}{\partial g_l} \frac{\partial g_l}{\partial t_{2k}} \frac{\partial t_{2k}}{\partial \langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle} \\ &= \sum_{l=1}^{L} [P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})] \omega_{kl}^{(3)} \sigma_2'(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle). \end{aligned} \tag{45}$$

Since $\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle = \sum_{n=1}^{N} \omega_{nk}^{(2)} \cdot t_{1n} + b_{2k}$, the derivative of ℓ with respect to $\omega_{nk}^{(2)}$ can be expressed as

$$\begin{aligned} \frac{\partial \ell_{\text{CE}}}{\partial \omega_{nk}^{(2)}} &= \frac{\partial \ell_{\text{CE}}}{\partial \langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle} \frac{\partial \langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle}{\partial \omega_{nk}^{(2)}} \\ &= \sum_{l=1}^{L} [P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})] \omega_{kl}^{(3)} \sigma_2'(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1 \rangle) t_{1n} \end{aligned} \tag{46}$$

Similarly, the derivative of ℓ_{CE} with respect to $\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle$ can be expressed as

$$\begin{aligned}\frac{\partial \ell_{\text{CE}}}{\partial \langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle} &= \sum_{k=1}^{K} \frac{\partial \ell_{\text{CE}}}{\partial \langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1\rangle} \frac{\partial \langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1\rangle}{\partial t_{1n}} \frac{\partial t_{1n}}{\partial \langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle} \\ &= \sum_{k=1}^{K}\sum_{l=1}^{L}[P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})]\omega_{kl}^{(3)}\sigma_2'(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1\rangle)\omega_{nk}^{(2)}\sigma_1'(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle).\end{aligned} \tag{47}$$

Since $\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle = \sum_{m=1}^{M} \omega_{mn}^{(1)} \cdot x_m + b_{1n}$, the derivative of ℓ_{CE} with respect to $\omega_{mn}^{(1)}$ can be expressed as

$$\begin{aligned}\frac{\partial \ell_{\text{CE}}}{\partial \omega_{mn}^{(1)}} &= \frac{\partial \ell_{\text{CE}}}{\partial \langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle} \frac{\partial \langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle}{\partial \omega_{mn}^{(1)}} \\ &= \sum_{k=1}^{K}\sum_{l=1}^{L}[P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})]\omega_{kl}^{(3)}\sigma_2'(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1\rangle)\omega_{nk}^{(2)}\sigma_1'(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle)x_m.\end{aligned} \tag{48}$$

Overall, the derivative of ℓ_{CE} with respect to the weight in each layer is summarized as

$$\begin{aligned}\frac{\partial \ell_{\text{CE}}}{\partial \omega_{kl}^{(3)}} &= [P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})]t_{2k} \\ \frac{\partial \ell_{\text{CE}}}{\partial \omega_{nk}^{(2)}} &= \sum_{l=1}^{L}[P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})]\omega_{kl}^{(3)}\sigma_2'(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1\rangle)t_{1n} \\ \frac{\partial \ell_{\text{CE}}}{\partial \omega_{mn}^{(1)}} &= \sum_{k=1}^{K}\sum_{l=1}^{L}[P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})]\omega_{kl}^{(3)}\sigma_2'(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1\rangle)\omega_{nk}^{(2)}\sigma_1'(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle)x_m.\end{aligned} \tag{49}$$

Based on the above three equations, we can reformulate the derivatives as

$$\begin{aligned}\frac{\partial \ell_{\text{CE}}^{\star}}{\partial \omega_{kl}^{(3)}} &= [P_{\hat{Y}|X}(l|\boldsymbol{x}) - P_{Y|X}(l|\boldsymbol{x})] \cdot t_{2k}, \\ \frac{\partial \ell_{\text{CE}}^{\odot}}{\partial \omega_{nk}^{(2)}} &= \sum_{l=1}^{L} \frac{\partial \ell_{\text{CE}}^{\star}}{\partial \omega_{kl}^{(3)}} \cdot \omega_{kl}^{(3)} \cdot \frac{\sigma_2'(\langle \boldsymbol{\omega}_k^{(2)}, \boldsymbol{t}_1\rangle)}{t_{2k}} \cdot t_{1n} \\ \frac{\partial \ell_{\text{CE}}^{\diamond}}{\partial \omega_{mn}^{(1)}} &= \sum_{k=1}^{K} \frac{\partial \ell_{\text{CE}}^{\odot}}{\partial \omega_{nk}^{(2)}} \cdot \omega_{nk}^{(2)} \cdot \frac{\sigma_1'(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x}\rangle)}{t_{1n}} \cdot x_m.\end{aligned} \tag{50}$$

The above three equations indicates that $\frac{\partial \ell_{\text{CE}}^{\star}}{\partial \omega_{kl}^{(3)}}$ is a function of $P_{Y|X}(l|\boldsymbol{x})$, $\frac{\partial \ell_{\text{CE}}^{\odot}}{\partial \omega_{nk}^{(2)}}$ is a function of $\frac{\partial \ell_{\text{CE}}^{\star}}{\partial \omega_{kl}^{(3)}}$, and $\frac{\partial \ell_{\text{CE}}^{\diamond}}{\partial \omega_{mn}^{(1)}}$ is a function of $\frac{\partial \ell_{\text{CE}}^{\odot}}{\partial \omega_{nk}^{(2)}}$. In addition, the back-propagation algorithm shows that

$$\begin{aligned}\omega_{mn}^{(1)}(s+1) &= \omega_{mn}^{(1)}(s) - \alpha \frac{\partial \ell_{\text{CE}}}{\partial \omega_{mn}^{(1)}(s)} \\ \omega_{nk}^{(2)}(s+1) &= \omega_{nk}^{(2)}(s) - \alpha \frac{\partial \ell_{\text{CE}}}{\partial \omega_{nk}^{(2)}(s)} \\ \omega_{kl}^{(3)}(s+1) &= \omega_{kl}^{(3)}(s) - \alpha \frac{\partial \ell_{\text{CE}}}{\partial \omega_{kl}^{(3)}(s)}\end{aligned} \tag{51}$$

where α is the learning rate and s denotes the index of the sth learning iteration. Therefore, $\omega(s+1)$ is determined by all the previous gradients $\{\frac{\partial \ell_{\text{CE}}}{\partial \omega(s)}\}_{s=1}^{S}$ as $\omega(0)$ is randomly initialized and α is a constant.

Definition 1 indicates that the weights define the sample space Ω_{T_i}, thus we can derive that the gradients $\frac{\partial \ell_{\text{CE}}}{\partial \omega^{(i)}}$ determine Ω_{T_i}. As a result, Ω_{T_i} is a function of $\Omega_{T_{i+1}}$ and $\Omega_{\hat{Y}}$ is a function of $P(Y|X)$. Based on Definition 2, we can further derive that T_i is a function of T_{i+1} and $\hat{Y}$ is a function of Y, *i.e.*, $T_1 \leftarrow T_2 \leftarrow \hat{Y} \leftarrow Y$.

D The proof of $H(Y) = I(X;Y)$

Given a training sample $\boldsymbol{x}^j$ and the corresponding label y^j, the target distribution $P_{Y|X}(y^j|\boldsymbol{x}^j)$ is commonly formulated as the one-hot format, *i.e.*,

$$P_{Y|X}(l|\boldsymbol{x}^j) = \begin{cases} 1 & \text{for} \quad l = y^j \\ 0 & \text{for} \quad l \neq y^j \end{cases} \tag{52}$$

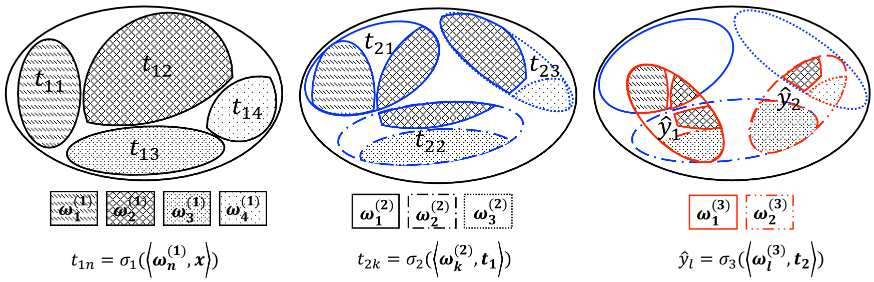

Figure 6: The graphical explanation for Corollary 1 based on the MLP $= \{\boldsymbol{x}, \boldsymbol{t}_1, \boldsymbol{t}_2, \hat{\boldsymbol{y}}\}$. The largest oval represents of the input $\boldsymbol{x}$, and each small shape indicates the representation capacity of a single feature. For example, if t_{12} is the largest activation, then the feature $\boldsymbol{\omega}_2^{(1)}$ has the largest cross-correlation to $\boldsymbol{x}$, *i.e.*, $\boldsymbol{\omega}_2^{(1)}$ has largest representation capacity. Therefore, $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^4$ can be viewed as a representation of $\boldsymbol{x}$, and the representation capacity of $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^4$ is measured by $\{t_{1n}\}_{n=1}^N$, which is visualized by the left figure. The blue ovals indicates the representation capacity of the three features $\{\boldsymbol{\omega}_k^{(2)}\}_{k=1}^3$ generated by combining the four features $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^4$. The two red ovals indicates the representation capacity of the two features $\{\boldsymbol{\omega}_l^{(3)}\}_{l=1}^2$ generated by combining the three features $\{\boldsymbol{\omega}_k^{(2)}\}_{k=1}^3$.

As a result, the conditional entropy $H(Y|X)$ can be formulated as

$$H(Y|X) = -\sum_{(\boldsymbol{x}^j, y^j) \in \mathcal{D}} P_{X,Y}(\boldsymbol{x}^j, y^j) \log P_{Y|X}(y^j|\boldsymbol{x}^j) = 0. \tag{53}$$

Therefore, we can derive $H(Y) = I(X;Y)$ because $H(Y) = H(Y|X) + I(X;Y)$.

E The detailed derivations and explanations for Corollary 1

Definition 1 indicates that $\boldsymbol{t}_1 = \{t_{1n} = \sigma_1(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle)\}_{n=1}^N$ defines N features of $\boldsymbol{x}$, namely $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^N$, thus $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^N$ can be viewed as a representation of $\boldsymbol{x}$. In addition, $\{t_{1n}\}_{n=1}^N$ measures the cross-correlation between $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^N$ and $\boldsymbol{x}$, (*i.e.*, if $\boldsymbol{\omega}_n^{(1)}$ describes $\boldsymbol{x}$ more accurately and comprehensively, then t_{1n} is larger.), thus $\{t_{1n}\}_{n=1}^N$ quantifies the representation capacity of $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^N$. For example, in Figure 6 (Left), $\boldsymbol{t}_1$ defines 4 features to describe $\boldsymbol{x}$ and t_{12} is the largest activation, thus $\boldsymbol{\omega}_2^{(1)}$ has the largest representation capacity of $\boldsymbol{x}$.

During inference, the second hidden layer $\boldsymbol{t}_2$ will process $\{t_{1n}\}_{n=1}^N$, and $t_{2k} = \sigma_2(\langle \boldsymbol{w}_k^{(2)}, \boldsymbol{t}_1 \rangle)$ can be explained to generating a new feature via combining all the features $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^N$, *i.e.*,

$$\{\omega_{1k}^{(2)} \otimes \boldsymbol{\omega}_1^{(1)}, \cdots, \omega_{Nk}^{(2)} \otimes \boldsymbol{\omega}_N^{(1)}\}. \tag{54}$$

Since the new feature is the linear combination of $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^N$, it can be simply noted as

$$\{\omega_{1k}^{(2)}, \cdots, \omega_{Nk}^{(2)}\} = \boldsymbol{\omega}_k^{(2)}, \tag{55}$$

and the representation capacity of the new feature $\boldsymbol{\omega}_k^{(2)}$ is

$$t_{2k} = \omega_{1k}^{(2)} \cdot t_{11} + \cdots + \omega_{Nk}^{(2)} \cdot t_{1N} \tag{56}$$

For example, if $N = 4$ and $K = 3$, the representation capacity of the three new features is visualized by Figure 6 (Middle). Similarly, $\hat{\boldsymbol{y}}$ generates L new features via combining all the features $\{\boldsymbol{\omega}_k^{(2)}\}_{k=1}^K$.

$$\{\omega_{1l}^{(3)} \otimes \boldsymbol{\omega}_1^{(2)}, \cdots, \omega_{Kl}^{(3)} \otimes \boldsymbol{\omega}_K^{(2)}\}. \tag{57}$$

Since the new feature is the linear combination of $\{\boldsymbol{\omega}_k^{(2)}\}_{k=1}^K$, it can be simply noted as

$$\{\omega_{1l}^{(3)}, \cdots, \omega_{Kl}^{(3)}\} = \boldsymbol{\omega}_l^{(3)}, \tag{58}$$

and the representation capacity of the new feature $\boldsymbol{\omega}_l^{(3)}$ is

$$\hat{y}_l = \omega_{1l}^{(3)} \cdot t_{21} + \cdots + \omega_{Kl}^{(3)} \cdot t_{2K} \tag{59}$$

For example, if $L = 2$, the representation capacity of the two new features is visualized by Figure 6 (Right).

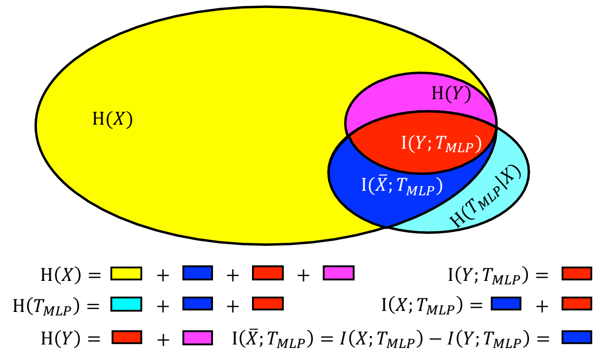

Figure 7: The Venn diagram of $H(X)$, $H(Y)$, and $I(X; T_{\text{MLP}})$.

Overall, the inference phase is a procedure of feature combination, *i.e.*, $\boldsymbol{\omega}_l^{(3)}$ is a combination of $\{\boldsymbol{\omega}_k^{(2)}\}_{k=1}^K$, and $\boldsymbol{\omega}_k^{(2)}$ is a combination of $\{\boldsymbol{\omega}_n^{(1)}\}_{n=1}^N$. Theorem 2 proves that the layer closer to output has more information of labels, *i.e.*, $T_1 \leftarrow T_2 \leftarrow \hat{Y} \leftarrow Y$, during training. Since the weights are fixed after training, the sample space and the distribution of hidden layers are fixed after training. Therefore, the information of Y transferred into hidden layers during training will retain there after training (*i.e.*, during inference), i.e., $T_1 \leftarrow T_2 \leftarrow \hat{Y} \leftarrow Y$ characterizes the information flow of Y in the MLP in both training and inference phases.

For example, Figure 6 (Right) shows that the representation capacity of $\boldsymbol{\omega}_1^{(3)}$ is the weighted combination of t_{11}, t_{12}, and t_{13}, and the representation capacity of $\boldsymbol{\omega}_2^{(3)}$ is the weighted combination of t_{12}, t_{13}, and t_{14}. Therefore, $\boldsymbol{\omega}_2^{(1)}$ and $\boldsymbol{\omega}_3^{(1)}$ exist in both classes, *i.e.*, the low-level features in $\boldsymbol{t}_1$ do not represent too much information of the labels, though we combine low-level features to generate high-level features for representing labels.

F The proof of Corollary 2

Based on the property of mutual information, we have

$$\begin{aligned} H(X) &= H(X|Y) + I(X;Y) \\ &= H(X|Y) + H(Y) \quad \text{(Appendix D)} \\ &= H(\bar{X}) + H(Y) \end{aligned} \tag{60}$$

where $\bar{X}$ is the virtual random variable containing all the information of X except Y, namely $H(\bar{X}) = H(X|Y)$.

Therefore, $I(X; T_{\text{MLP}})$ can be reformulated as

$$I(X; T_{\text{MLP}}) = I(\bar{X}; T_{\text{MLP}}) + I(Y; T_{\text{MLP}}). \tag{61}$$

The Venn diagram of $H(X)$, $H(Y)$, and $I(X; T_{\text{MLP}})$ are visualized in Figure 7. Corollary 1 indicates that all the information of $\bar{X}$ and Y learned by a MLP retains in T_1 and $\hat{Y}$, respectively. Therefore, we can derive

$$\begin{aligned} I(X; T_{\text{MLP}}) &= I(\bar{X}; T_1) + I(Y; \hat{Y}) \\ I(Y; T_{\text{MLP}}) &= I(Y; \hat{Y}) \end{aligned} \tag{62}$$

G Studying non-parametric models for mutual information estimation

In this section, we use the synthetic dataset to show that non-parametric models are sensitive to hyper-parameters for mutual information estimation. In addition, we show that the proposed mutual information estimator derives more accurate mutual information estimation than non-parametric models. Furthermore, we demonstrate that one reason for non-parametric models deriving poor mutual information estimation is because activations do not satisfy the *i.i.d.* prerequisite of non-parametric models. The experiment codes are available online[6].

G.1 Non-parametric models are sensitive to hyper-parameters

To show non-parametric models being sensitive to hyper-parameters, we choose two commonly used non-parametric models, namely the empirical distribution [35] and KDE [33], to measure the information flow in MLP1 and MLP2 defined in Table 1 on the synthetic dataset.

[6] `https://github.com/Dlib-NeurIPS/Deep-Learning-Information-Theory`

Table 3: The hyper-parameters of empirical distributions and KDE

bs	0.001	0.01	0.1	1.0	2.0	4.0	6.0	8.0
σ_n^2	0.01	0.05	0.1	1.0	2.0	4.0	8.0	16.0

The empirical distribution is defined as

$$P(T=n) = \frac{1}{J}\mathbb{1}(\boldsymbol{t}, \boldsymbol{l}_n, \boldsymbol{r}_n) \tag{63}$$

where J is the number of samples, n denotes the nth bin, $\boldsymbol{t}$ denotes an activation vector, $\boldsymbol{l}_n$ and $\boldsymbol{r}_n$ are the left and right boundary vectors, respectively. The indicator function $\mathbb{1}(\boldsymbol{t}, \boldsymbol{l}_n, \boldsymbol{r}_n)$ is defined as

$$\mathbb{1}(\boldsymbol{t}, \boldsymbol{l}_n, \boldsymbol{r}_n) = \begin{cases} 1 & \text{for} \quad \boldsymbol{l}_n \leq \boldsymbol{t} < \boldsymbol{r}_n \\ 0 & \text{otherwise} \end{cases} \tag{64}$$

Given a specific range, the hyper-parameter of the empirical distribution is the bin size, namely $bs = |\boldsymbol{r}_n - \boldsymbol{l}_n|$. Based on the empirical distribution, Tishby *et al.* estimate $I(X;T_i)$ and $I(Y;T_i)$ (see Section 3.2 in [35]).

To estimate $I(X;T_i)$ and $I(Y;T_i)$ via KDE, Saxe *et al.* assume that the empirical distribution of input samples is the true distribution and the distribution of a hidden layer is a mixture of Gaussian. In addition, Saxe *et al.* regard a hidden layer as a deterministic function of input samples, thus the Gaussian noise $\mathcal{N}(0, \sigma_n^2)$ is added into activations to avoid infinite mutual information, and $I(X;T_i)$ is estimated as

$$I(X;T_i) \leq -\frac{1}{J}\sum_j \log \frac{1}{J}\sum_{j'} \exp(-\frac{\|\boldsymbol{t}_j^{(i)} - \boldsymbol{t}_{j'}^{(i)}\|_2^2}{2\sigma_n^2}) \tag{65}$$

where J is the number of samples, $\boldsymbol{t}_j^{(i)}$ denote the activations vector of the ith hidden layer in response to the input sample $\boldsymbol{x}^j$ (see Appendix B.1 in [33]). Therefore, the hyper-parameter of KDE is the noise variance σ_n^2.

Leveraging the same training method in Section 4.1, we achieves 100% training accuracy in MLP1 and MLP2 on the synthetic dataset. We specify 8 different values for each hyper-parameter, namely bs and σ_n^2, in Table 3, and use the empirical distribution and KDE to estimate $I(X;T_i)$ during training MLP1 and MLP2.

Figure 8 and 9 show that the empirical distribution is sensitive to the hyper-parameter, namely the bin size bs. Figure 8 shows that $I(X;T_i)$ in different hidden layers of MLP1 converges to 1.5 as bs increases from 0.001 to 8.0. Figure 9 shows that $I(X;T_1)$, $I(X;T_2)$, and $I(X;\hat{Y})$ in MLP2 converge to 1.2, 0.8, and 0.7, respectively, as bs increases from 0.001 to 8.0. Notably, since the synthetic dataset only has 2 bits information, $I(X;T_i)$ must be smaller than $H(X) = 2$ bits. However, we observe that if $bs < 1.0$, the empirical distribution derives $I(X;T_i) > 2.0$ in both MLP1 and MLP2, thus the empirical distribution cannot correctly estimate $I(X;T_i)$ in MLP1 and MLP2 on the synthetic dataset when $bs < 1.0$.

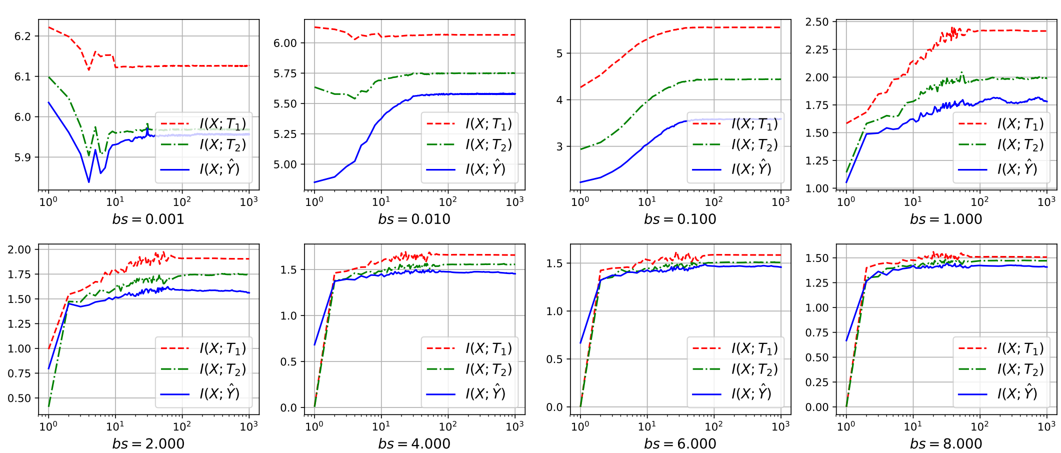

Figure 8: The estimation of $I(X;T_i)$ in MLP1 on the synthetic dataset via the empirical distribution with 8 different bs. All the x-axis index training epochs.

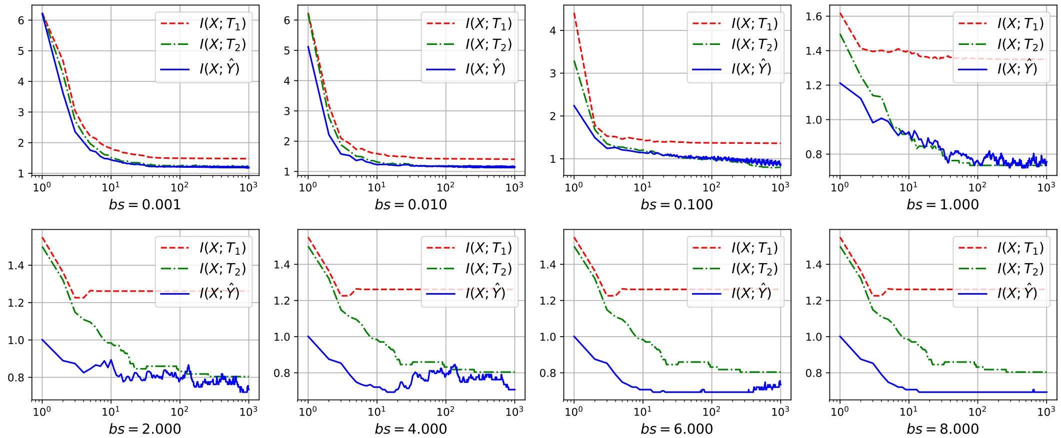

Figure 9: The estimation of $I(X;T_i)$ in MLP2 on the synthetic dataset via the empirical distribution with 8 different bs. All the x-axis index training epochs.

Similarly, Figure 10 and 11 show that KDE is also sensitive to the hyper-parameter, namely the noise variance σ_n^2. Figure 10 shows that $I(X;T_i)$ in different hidden layers of MLP1 converges to 2.0 as σ_n^2 increases from 0.01 to 16.0. Figure 11 shows that KDE derives different $I(X;T_1)$ and $I(X;T_2)$ in MLP2 given different σ_n^2, except $I(X;\hat{Y})$ converges to 1.0, as bs increases from 0.01 to 16.0. Again, since the synthetic dataset only has 2 bits information, $I(X;T_i)$ must be smaller than $H(X)=2$. However, we also observe that KDE derives $I(X;T_i) > 2.0$ when $\sigma_n^2 < 1.0$. Overall, different σ_n^2 make KDE to derive different mutual information estimations for $I(X;T_i)$ in MLP1 and MLP2 on the synthetic dataset, especially KDE does not correctly estimate $I(X;T_i)$ in MLP1 and MLP2 when $\sigma_n^2 < 1.0$.

In summary, the two non-parametric models are sensitive to hyper-parameters for mutual information estimation. Especially, since the entropy of the synthetic dataset is known, we can determine which hyper-parameter is appropriate to estimate the mutual information. However, if the entropy of dataset is unknown, it is very difficult to choose an appropriate hyper-parameter for non-parametric models to estimate the mutual information.

G.2 Comparison to non-parametric models on the synthetic dataset

In this section, we compare the proposed mutual information estimator to the empirical distribution and KDE in MLP1 and MLP2 on the synthetic datset, and demonstrate that the proposed mutual information estimator derives more accurate mutual information estimation than non-parametric models. Based on Appendix G.1, we choose $bs = 2.0$ and $\sigma_n^2 = 2.0$ as the optimal hyper-parameters for the empirical distribution and KDE to estimate $I(X;T_i)$ and $I(Y;T_i)$. All the training methods are the same as Section 4.1.

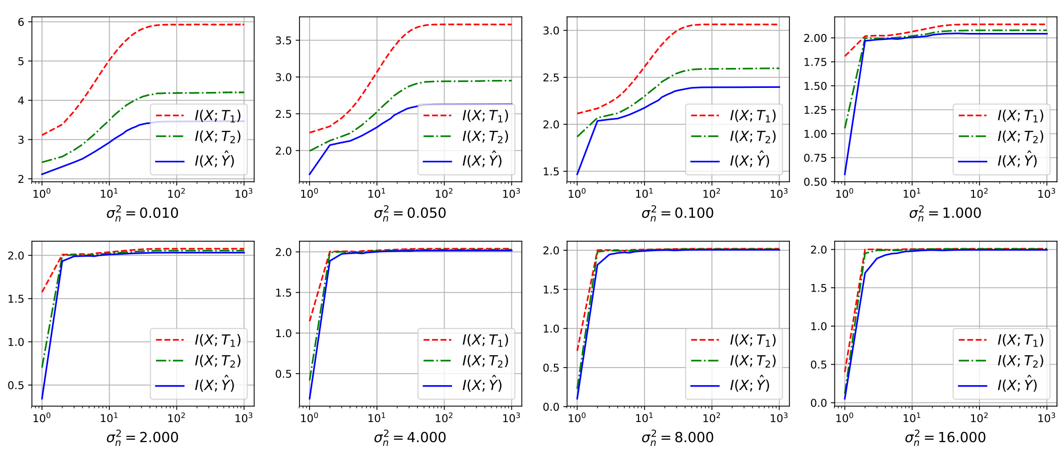

Figure 10: The estimation of $I(X;T_i)$ in MLP1 on the synthetic dataset by KDE with 8 different σ_n^2. All the x-axis index training epochs.

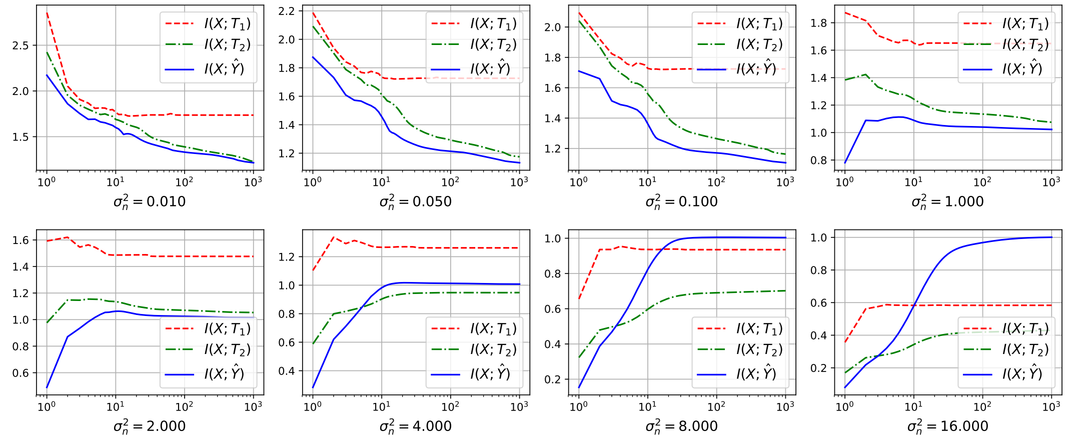

Figure 11: The estimation of $I(X;T_i)$ in MLP2 on the synthetic dataset by KDE with 8 different σ_n^2. All the x-axis index training epochs.

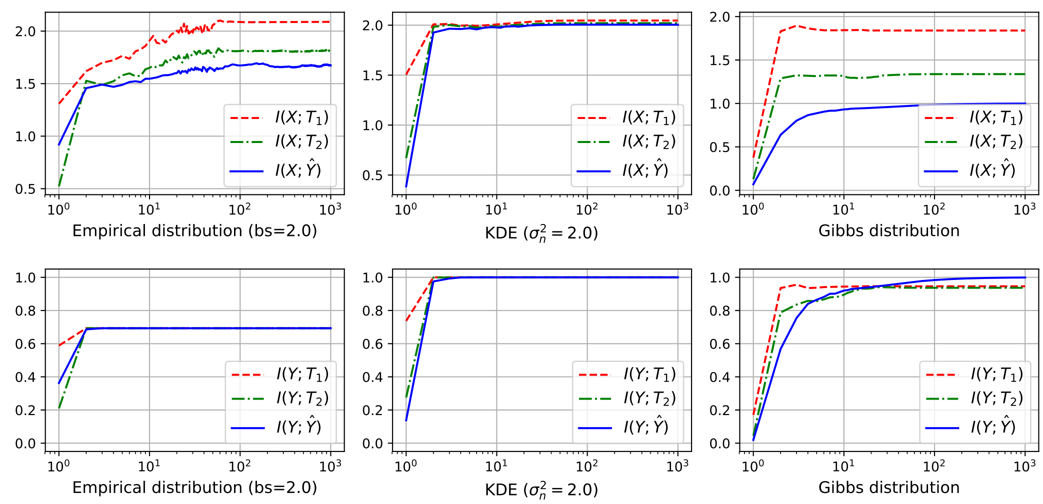

Figure 12: The estimation of $I(X;T_i)$ and $I(Y;T_i)$ in MLP1 based on the three mutual information estimators. All the x-axis index training epochs.

Figure 12 shows the estimation of $I(X;T_i)$ and $I(Y;T_i)$ in MLP1 derived by the three methods, namely the empirical distribution ($bs = 2.0$), KDE ($\sigma_n^2 = 2.0$), and the Gibbs distribution. Since $\hat{\boldsymbol{y}}$ only has two nodes, the maximal information of X that $\hat{\boldsymbol{y}}$ can have is 1 bit, i.e., $I(X;\hat{Y}) \leq 1$, based on Definition 1. However, we observe that the empirical distribution derives $I(X;\hat{Y}) > 1.5$ and KDE derives $I(X;\hat{Y}) = 2.0$, thus the empirical distribution and KDE do not accurately estimate $I(X;\hat{Y})$. In addition, since MLP1 correctly predicts all the labels of synthetic images, it should have all the information of the labels. However, we observe that the empirical distribution estimates $I(Y;T_i) = 0.7$ bits, which contradicts the fact. As a comparison, the proposed method based on Gibbs distribution accurately estimate the information flow in MLP1.

Figure 13 shows the estimation of $I(X;T_i)$ and $I(Y;T_i)$ in MLP2 derived by the three methods. As shown in Figure 4, MLP2 quickly learns all the features of the synthetic dataset, thus $I(X;T_i)$ should have an increasing trend as training epochs increases. However, $I(X;T_i)$ estimated by the empirical distribution shows a decreasing trend, which contradicts the variation of the weights shown in Figure 4. Therefore, the empirical distribution does not accurately estimate $I(X;T_i)$ in MLP2. In addition, Section 4.2 shows that Tanh hinders $\boldsymbol{t}_1$ from correctly recognizing the features of input, thus $\boldsymbol{t}_1$ in MLP2 does not contain too much information of X, *i.e.*, $I(X;T_1)$ is small. However, KDE estimates $I(X;T_1) > 1.5$, *i.e.*, $\boldsymbol{t}_1$ in MLP2 has most information of X. Therefore, KDE does not correctly measures the effect of activation functions on the mutual information.

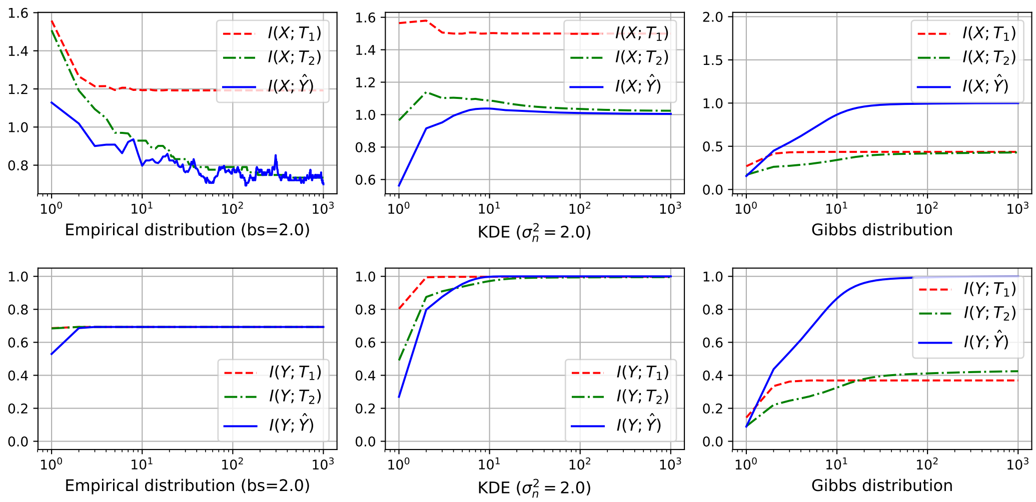

Figure 13: The estimation of $I(X;T_i)$ and $I(Y;T_i)$ in MLP2 based on the three mutual information estimators. All the x-axis index training epochs.

G.3 Activations do not satisfy the i.i.d. prerequisite of non-parametric models

In this section, we demonstrate that one reason for non-parametric models deriving poor mutual information estimation is because activations do not satisfy the *i.i.d.* prerequisite of non-parametric models.

Given an input $\boldsymbol{x} \in \mathbb{R}^M$, we define the corresponding multivariate random variable as $X = [X_1, \cdots, X_M]$, where X_m is the scalar-valued random variable of x_m. In the context of frequentist probability, all the parameters of MLPs are viewed as constants, thus the random variable of $\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle = \sum_{m=1}^{M} \omega_{mn}^{(1)} \cdot x_m + b_{1n}$ is defined as $G_{1n} = \sum_{m=1}^{M} \omega_{mn}^{(1)} X_m + b_{1n}$, and the random variable of the activation $t_{1n} = \sigma_1(\langle \boldsymbol{\omega}_n^{(1)}, \boldsymbol{x} \rangle)$ is defined as $T_{1n} = \sigma_1(G_{1n})$. Therefore, the multivariate random variable of $\boldsymbol{t}_1 = [t_{11}, \cdots, t_{1N}]$ can be defined as $T_1 = [T_{11}, \cdots, T_{1N}]$. Similarly, we define the multivariate random variable of $\boldsymbol{t}_2$ as $T_2 = [T_{21}, \cdots, T_{2K}]$ and the multivariate random variable of $\hat{\boldsymbol{y}}$ as $\hat{Y} = [\hat{Y}_1, \cdots, \hat{Y}_L]$.

Samples being *i.i.d.* is the prerequisite of applying non-parametric models, e.g. the empirical distribution and KDE, to model the true distribution of a random variable [40]. In the context of MLPs, most previous works regard the activations of a layer as the samples of the random variable of the layer, and use non-parametric models to simulate the distribution of the layer. As a result, activations must be *i.i.d.* samples.

Since the necessary condition for samples being *i.i.d.* is the samples being uncorrelated, we can use the sample correlation to examine if activations being *i.i.d.*. More specifically, given two *i.i.d.* input samples $\boldsymbol{x}^j$ and $\boldsymbol{x}^{j}{'}$, the two activation vectors of the ith hidden layers are $\boldsymbol{t}_i^j$ and $\boldsymbol{t}_i^{j'}$. If $\boldsymbol{t}_i^j$ and $\boldsymbol{t}_i^{j'}$ are *i.i.d.* samples of T_i, the sample correlation $R(\boldsymbol{t}_i^j, \boldsymbol{t}_i^{j'})$ must be zero, namely

$$R(\boldsymbol{t}_i^j, \boldsymbol{t}_i^{j'}) = \frac{\sum_{n=1}^{N} (t_{in}^j - \bar{t}_i^j)(t_{in}^{j'} - \bar{t}_i^{j'})}{\sqrt{\sum_{n=1}^{N} (t_{in}^j - \bar{t}_i^j)^2 \sum_{n=1}^{N} (t_{in}^{j'} - \bar{t}_i^{j'})^2}} = 0, \tag{66}$$

where $\bar{t}_i^j = \frac{1}{N} \sum_{n=1}^{N} t_{in}^j$, and N is the number of neurons in $\boldsymbol{t}_i$.

To study the sample correlation between activations given different samples, we use the Adam to train a MLP on the MNIST dataset [20] over 200 epochs with the learning rate $\alpha = 0.0005$. Since the dimension of each image is 28×28, the number of the input nodes is $M = 784$. In addition, $\boldsymbol{t}_1$, $\boldsymbol{t}_2$, and $\hat{\boldsymbol{y}}$ have $N = 96$, $K = 32$, and $L = 10$ neurons/nodes, respectively. All the activation functions are Tanh.

After training, we derive $R(\boldsymbol{t}_i^j, \boldsymbol{t}_i^{j'})$ on 5000 training samples $\{\boldsymbol{x}^j\}_{j=1}^{5000}$ and show the result in Figure 14. In particular, we rearrange the order of $\{\boldsymbol{x}^j\}_{j=1}^{5000}$ such that images with the same label have consecutive index, i.e., images with the label l has the index $[l \times 500, (l+1) \times 500)$, thus we can easily check the sample correlation between activations with the same label. Figure 14 shows that the sample correlation between activations with the same label becomes larger as the layer is closer to the output. In other words, activations are not *i.i.d.*. Therefore, it is invalid to apply non-parametric models to model the true distribution of all the layers of the MLP, because activations do not satisfy the *i.i.d.* prerequisite of non-parametric models.

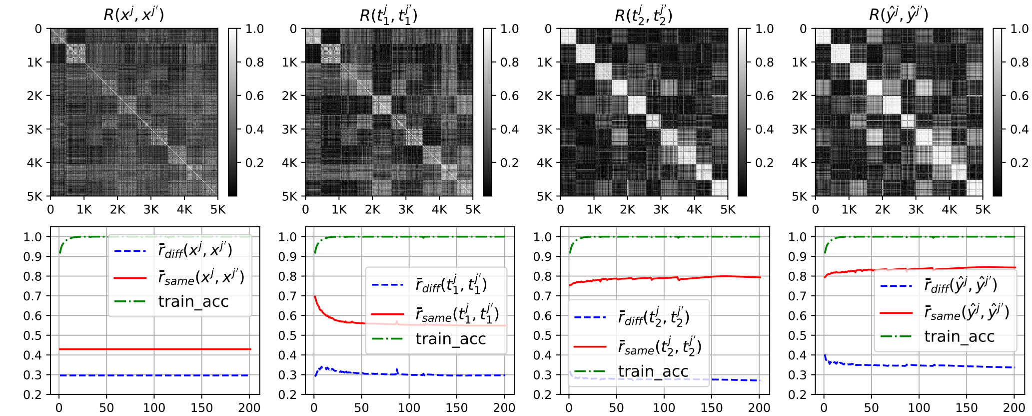

Figure 14: The first row shows that sample correlation between different samples/activations in each layer of the MLP after training. The second row shows the variation of the average sample correlation between different activations with different labels and with the same labels in each layer during training.

More specifically, Figure 14 shows that the sample correlation between each pair of training samples $\{\boldsymbol{x}^j\}_{j=1}^{5000}$ is very small, thus *i.i.d.* can be viewed as a valid assumption for the input samples $\{\boldsymbol{x}^j\}_{j=1}^{5000}$. However, we observe an ascending trend for the sample correlation between different activations with the same label as the layer is closer to the output. For instance, the pixels at the top-left corner of $R(\boldsymbol{t}_i^j, \boldsymbol{t}_i^{j'})$ becomes lighter as the layer is closer to the output, i.e., the sample correlation between the activations with the label 0 becomes larger.

In addition, the second row of Figure 14 also show the ascending trend, i.e., $\bar{r}_{\text{same}}(\boldsymbol{t}_1^j, \boldsymbol{t}_1^{j'})$, $\bar{r}_{\text{same}}(\boldsymbol{t}_2^j, \boldsymbol{t}_2^{j'})$, and $\bar{r}_{\text{same}}(\hat{\boldsymbol{y}}^j, \hat{\boldsymbol{y}}^{j'})$ converge to 0.55, 0.79, and 0.84, respectively, where $\bar{r}_{\text{same}}(\boldsymbol{t}_i^j, \boldsymbol{t}_i^{j'})$ denotes the average sample correlation of $\{\boldsymbol{t}_i^j\}_{j=1}^{5000}$ with the same label in the ith hidden layer.

As a comparison, Figure 14 shows that the sample correlation of activations with different labels being relatively stable in different layers, because $\bar{r}_{\text{diff}}(\boldsymbol{t}_1^j, \boldsymbol{t}_1^{j'})$, $\bar{r}_{\text{diff}}(\boldsymbol{t}_2^j, \boldsymbol{t}_2^{j'})$, and $\bar{r}_{\text{diff}}(\hat{\boldsymbol{y}}^j, \hat{\boldsymbol{y}}^{j'})$ converge to 0.29, 0.27, and 0.33, respectively, where $\bar{r}_{\text{diff}}(\boldsymbol{t}_i^j, \boldsymbol{t}_i^{j'})$ denotes the average sample correlation of $\{\boldsymbol{t}_i^j\}_{j=1}^{5000}$ with different labels.

In summary, the sample correlation of activations with the same label becomes larger as the layer is closer to the output, thus activations being *i.i.d.* is not valid for all the layers of the MLP. As a result, non-parametric models, e.g., the empirical distribution and KDE, cannot correctly simulate the true distribution of all the layers, thus they are invalid for estimating the mutual information between each layer and dataset.

H Experiments on benchmark dataset

To further demonstrate the information theoretic explanations for DNNs, we design more complicated neural networks and conduct experiments on the bechmark MNIST and Fashion-MNIST (abbr. FMNIST) dataset. The experiment codes are also available online[7].

H.1 Experiments on the MNIST dataset

We design three MLPs, namely MLP4, MLP5, and MLP6, and summarize the architectures of the three MLPs in Table 4. We train the three MLPs on the MNIST dataset by Adam [15] over 500 epochs with the learning rate $\alpha = 0.0005$. Based on the mutual information estimator proposed in Section 4.1, we measure the information flow in the three MLPs during 500 training epochs.

In Figure 15, we observe that the information flow of X in the three MLPs does not satisfy the Markov chain, namely Equation (2), proposed by previous works, *i.e.*, we further confirm that Equation (2) does not fully characterize the information flow of X, especially when taking into account of the back-propagation training.

Moreover, the second and the third row of Figure 15 show $I(\bar{X}; T_1) \geq I(\bar{X}; T_2) \geq I(\bar{X}; \hat{Y})$ and $I(Y; T_1) \leq I(Y; T_2) \geq I(Y; \hat{Y})$ in all the three MLPs, which further validate that Corollary 1, *i.e.*, Equation (14), correctly characterizes the information flow in MLPs.

The last row of Figure 15 shows that $I(X; T_{\text{MLP}}) > H(Y)$ and $I(Y; T_{\text{MLP}}) = H(Y)$ for most epochs in all the three MLPs. Though $H(X)$ is unknown for the MNIST dataset, we still can conclude that the three MLPs form three compressed representations of the data while preserve all the information of the labels. Hence, Figure 15 further confirms that a MLP satisfies the IB principle no matter what the architecture of the MLP is.

[7] `https://github.com/Dlib-NeurIPS/Deep-Learning-Information-Theory`

Table 4: The number of neurons(nodes) and the activation function in MLP4 - MLP6

	$\boldsymbol{x}$	$\boldsymbol{t}_1$	$\boldsymbol{t}_2$	$\hat{\boldsymbol{y}}$	$\sigma(\cdot)$
MLP4	784 (28×28)	96	32	10	$\text{ReLU}(z) = \max(0, z)$
MLP5	784 (28×28)	96	32	10	$\text{Tanh}(z) = (e^z - e^{-z})/(e^z + e^{-z})$
MLP6	784 (28×28)	32	96	10	ReLU

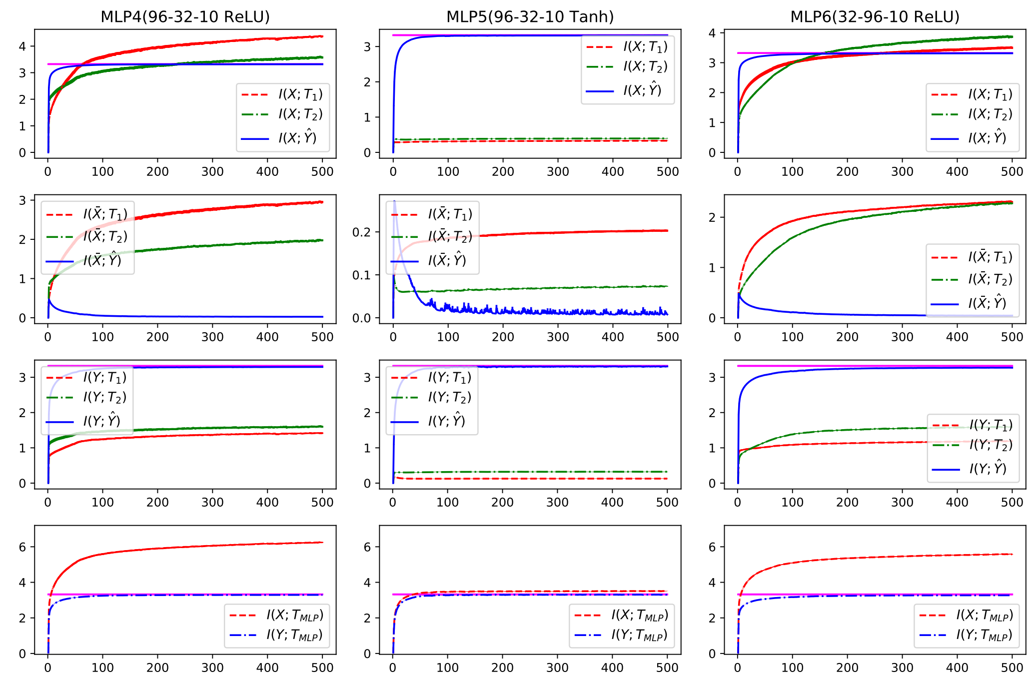

Figure 15: The information flow in MLP4, MLP5, and MLP6 on the MNIST dataset. All the x-axis index training epochs. In each column, the first three figures show $I(X;T_i)$, $I(\bar{X};T_i)$, and $I(Y;T_i)$ respectively. The forth figure shows $I(X;T_{\text{MLP}})$ and $I(Y;T_{\text{MLP}})$ in a MLP. The pink line denotes $H(Y) = \log_2 10$.

H.2 Experiments on the Fashion-MNIST dataset

We design three MLPs, namely MLP7, MLP8, and MLP9, and summarize the architectures of the three MLPs in Table 5. Compared to the MLPs on the MNIST dataset, the three MLPs has one more hidden layer and each hidden layer has more neurons, *i.e.*, the MLPs are more complicated. Similarly, we train the three MLPs by Adam [15] over 500 epochs with the learning rate $\alpha = 0.0005$. Based on the mutual information estimator proposed in Section 4.1, we measure the information flow in the three MLPs during 500 training epochs.

Figure 16 shows similar results as Section 4.3 and Section H.1, thus it further confirms the information theoretic explanations for DNNs.

Table 5: The number of neurons(nodes) and the activation function in MLP7 - MLP9

	$\boldsymbol{x}$	$\boldsymbol{t}_1$	$\boldsymbol{t}_2$	$\boldsymbol{t}_3$	$\hat{\boldsymbol{y}}$	$\sigma(\cdot)$
MLP7	784 (28×28)	256	128	96	10	$\text{ReLU}(z) = \max(0, z)$
MLP8	784 (28×28)	256	128	96	10	$\text{Tanh}(z) = (e^z - e^{-z})/(e^z + e^{-z})$
MLP9	784 (28×28)	96	128	256	10	ReLU

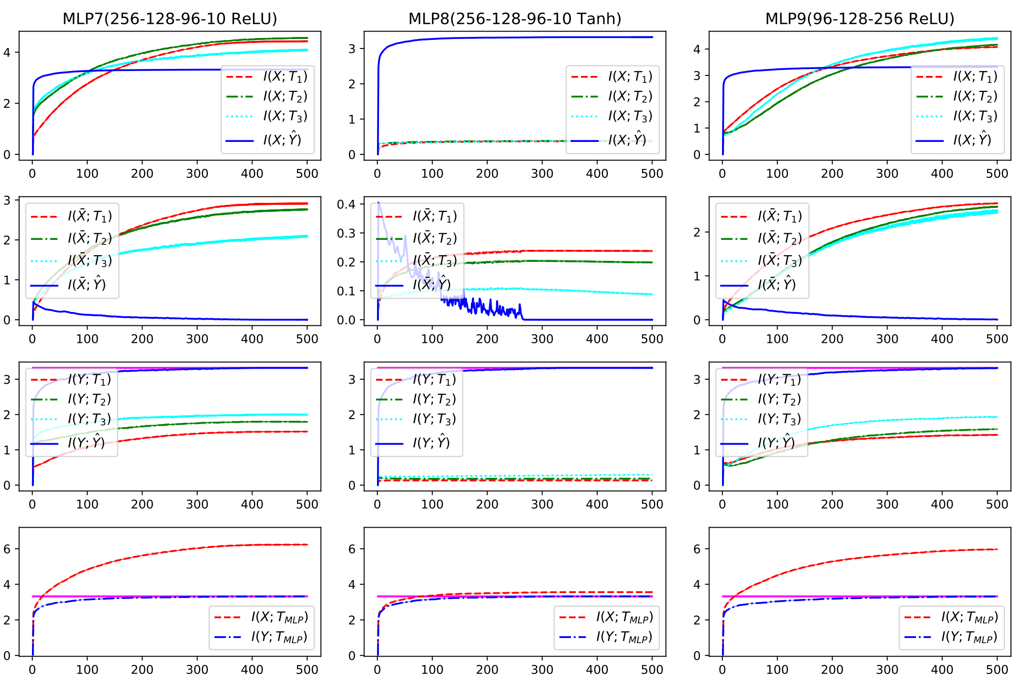

Figure 16: The information flow in MLP7, MLP8, and MLP9 on the MNIST dataset. All the x-axis index training epochs. In each column, the first three figures show $I(X;T_i)$, $I(\bar{X};T_i)$, and $I(Y;T_i)$ respectively. The forth figure shows $I(X;T_{\text{MLP}})$ and $I(Y;T_{\text{MLP}})$ in a MLP. The pink line denotes $H(Y) = \log_2 10$.