Evaluating Capability of Deep Neural Networks for Image Classification via Information Plane

Hao Cheng[0000−0001−8864−7818], Dongze Lian[0000−0002−4947−0316], Shenghua Gao[0000−0003−1626−2040], and Yanlin Geng**[0000−0002−4451−7242]

Shanghaitech University
{chenghao,liandz,gaoshh,gengyl}@shanghaitech.edu.cn

Abstract. Inspired by the pioneering work of information bottleneck principle for Deep Neural Networks (DNNs) analysis, we design an information plane based framework to evaluate the capability of DNNs for image classification tasks, which not only helps understand the capability of DNNs, but also helps us choose a neural network which leads to higher classification accuracy more efficiently. Further, with experiments, the relationship among the model accuracy, $I(X;T)$ and $I(T;Y)$ are analyzed, where $I(X;T)$ and $I(T;Y)$ are the mutual information of DNN's output T with input X and label Y. We also show the information plane is more informative than loss curve and apply mutual information to infer the model's capability for recognizing objects of each class. Our studies would facilitate a better understanding of DNNs.

Keywords: Information Bottleneck, Mutual Information, Neural Networks, Image Classification

1 Introduction

Deep Neural Networks (DNNs) have demonstrated their successes in many computer vision and natural language processing tasks [1–5], but the theoretical reasons that contribute to the successes of DNNs haven't been fully unveiled. Recently, information theory has shown its preponderance for DNNs understanding. Specifically, Tishby and Zaslavsky [6] note that layered neural networks can be represented as a Markov chain and analyze the neural network via the information bottleneck. Schwartz-Ziv and Tishby [7] calculate the mutual information $I(X;T)$, $I(T;Y)$ for each hidden layer, where X is the input data, Y is the label and T is the hidden layer output, respectively. Then they demonstrate the effectiveness of the visualization of neural networks. These works inspire us to leverage mutual information to evaluate the capability of DNNs.

Fig. 1 depicts the evolution of the mutual information along with the training epochs in the information plane [7]. As can be seen, the green point, which is referred to as the **transition point**, in each mutual information path separates the learning process into two distinct phases: the 'fitting phase', which takes a

** Corresponding Author

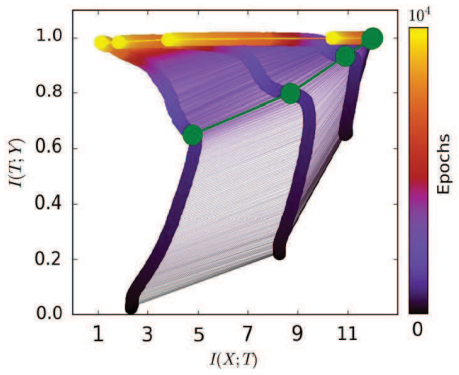

Fig. 1. This figure is adapted from [7]. The mutual information path is calculated based on a fully connected neural network. X is a 12-dimensional binary input and Y has 2 classes. Each hidden layer first reaches the green point (transition point), then converges at the yellow point. The leftmost path corresponds to the last hidden layer and the rightmost path corresponds to the first hidden layer. (best viewed in color)

few hundred epochs, and the layers' information on the label, namely $I(T;Y)$, increases; the subsequent 'compression phase', which takes most of the training time and the layers' information on the input, i.e. $I(X;T)$, decreases (this means the layers remove irrelevant information until convergence).

The evolution of $I(X;T)$ and $I(T;Y)$ explains how DNNs work. However, the models used in [6, 7] are some simple fully connected neural networks. In real applications, Convolutional Neural Networks (CNNs) are commonly used in computer vision. Pushing these works [6, 7] forward, in this paper, we design an information plane based framework to study the capability of some classical CNN structures for image classification, including AlexNet [2], VGG [8]. The contributions of our work can be summarized as follows:

- Our work unveils that $I(X;T)$ also contributes to the training accuracy and the correlation grows stronger as the network gets deeper. We perform experiments to validate this claim.
- An evaluation framework based on the information plane is proposed. The framework is more 'informative' than the loss curve and would facilitate a better understanding of DNNs.
- We show that mutual information can be used to infer the DNN's capability of recognizing objects of each class in the image classification task.

2 Related Work

The most related topic is the information bottleneck (IB) principle [9]. IB provides a technique for extracting information in some *input* random variable that is relevant for predicting some different *output* random variable. [10] extends the

original IB method to obtain continuous representations that preserve relevant information, rather than discrete clusters, for the special case of multivariate Gaussian variables. [11] introduces an alternative formulation called the deterministic IB (DIB), which replaces mutual information with entropy and better captures the notion which features are relevant. [12] theoretically analyzes the IB method and its relation to learning algorithms and minimal sufficient statistics. [13] shows that K-means and deterministic annealing algorithms for geometric clustering can be derived from a more general IB approach.

Recently, we have seen some applications of IB in deep learning. [14] presents a variational approximation to the IB method. This variational approach can parameterize the IB model using a neural network and leverage the reparameterization trick for efficient training. [15] proposes a method that allows IB to be used in more general domains, such as discrete or continuous inputs and outputs, nonlinear encoding and decoding maps. [16] proposes a Parametric IB (PIB) framework to jointly optimize the compression and relevance of all layers in stochastic neural networks for better exploiting the networks' representation capabilities. [17] introduces the Information Dropout method, which generalizes the dropout method in deep learning, rooted in information theoretic principles that automatically adapts to the data, and can better exploit architectures with limited capacity.

[6, 7], which are most relevant to our work, visualize the mutual information of hidden layers and the input/output of a neural network in the information plane to understand the optimization process and the internal organization of DNNs. While in this paper, different from these works which study DNNs with fully connected layers, we propose to study the behavior of more commonly used CNNs in image classification.

3 Mutual Information and Deep Neural Networks

In this section, we first revisit the definition of mutual information and its properties relevant to DNNs analysis, then we interpret the representation learning in DNNs with mutual information and show how to calculate mutual information in DNNs.

3.1 Mutual Information

Given two random variables X and Y with a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$, the mutual information between two variables, $I(X;Y)$, is defined as:

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{1}$$

The entropy of X, $H(X)$, can be defined using the mutual information:

$$H(X) = I(X;X) = -\sum_{x} p(x) \log p(x). \tag{2}$$

In general, the mutual information of two random variables is a measurement of the mutual dependence between the two variables. More specifically, it quantifies the amount of information obtained about one random variable, through the other one.

There are two properties (3)(4) of mutual information which are useful for analyzing DNNs:

- function transformation:

$$I(X;Y) = I(\psi(X);\phi(Y)) \tag{3}$$

  for any invertible functions ψ and ϕ.
- Markov chain. Suppose $X \to Y \to Z$ forms a Markov chain, then we have the data processing inequality:

$$I(X;Y) \geq I(X;Z). \tag{4}$$

3.2 Optimal Representation of Learning Process

In representation learning, we want our model to learn an efficient representation of the original data X without losing prediction capability of the label Y, which means we want to learn a minimal sufficient statistics of X with respect to Y. A minimal sufficient statistics $T(X)$ is the solution to the following optimization problem:

$$T(X) = \underset{S(X):I(S(X);Y)=I(X;Y)}{\arg\min} I(S(X);X) \tag{5}$$

So, from the minimal sufficient statistics perspective, the goal of DNNs is to make $I(X;S(X))$ as small as possible, which means the representation is efficient; while $I(S(X);Y)$ should be the same value of $I(X;Y)$ which means the information on Y is not lost. In practice, the explicit minimal sufficient statistics only exist for very special distributions. The actual learning process is a tradeoff between $I(X;S(X))$ and $I(S(X);Y)$, and it leads to the IB method [9]. IB can be seen as a special case of Rate Distortion theory and provides a framework to find approximate minimal sufficient statistics. The efficient representation is a tradeoff between the compression of X and the prediction ability of Y.

Let x be an input point, and t be the corresponding model's output, or the compressed representation of x. This representation is defined by the probabilistic mapping $p(t|x)$. The information bottleneck tradeoff is formulated by the following optimization problem:

$$\min_{p(t|x),Y\to X\to T} \{I(X;T) - \beta I(T;Y)\}. \tag{6}$$

The Lagrange multiplier β determines the level of relevant information captured by the representation T. So given a joint distribution $p(x,y)$ and the parameter β, minimizing (6) yields the optimal $I(X;T)$ and $I(T;Y)$ (see (31) in [9]).

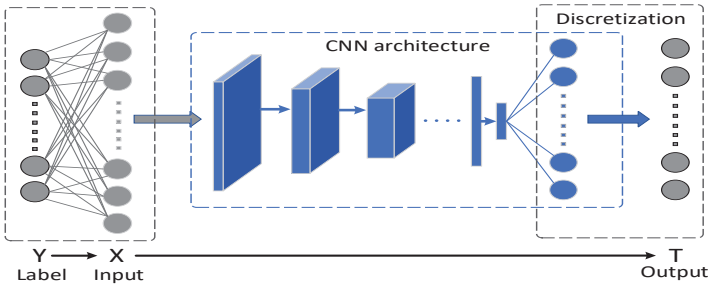

Fig. 2. This figure shows how we obtain T from the network for calculating $I(X;T)$ and $I(T;Y)$. $Y \to X \to T$ forms a Markov chain. The output of the last layer (blue circles) is the softmax probability.

3.3 Calculating Mutual Information in DNNs

From Section 3.2, we know $I(X;T)$ and $I(T;Y)$ are essential to evaluate the representation learning algorithms, including DNNs, but the calculation in DNNs is a difficult problem.

[7] uses the hyperbolic tangent function as the hidden layer's activation function, and bins the neuron's output activation into 30 equal intervals between -1 and 1. Then they use these discretized values for each t, to directly calculate the joint distributions $p(x,t)$ and $p(t,y)$ over the equally likely patterns of the input data for every hidden layer. But when the number of neurons in the hidden layer is large (it happens when we visualize CNN layers), $I(X;T)$ and $I(T;Y)$ barely change. The reason is that the sample space of T is huge even if we decrease the number of intervals, and the output of a particular input data x falls into one interval of t with high probability. Thus $p(x|t)$ and $p(y|t)$ are approximately deterministic, $I(X;T) \approx H(X)$ and $I(T;Y) \approx H(Y)$ from (1)(2). So this issue makes it hard to analyze universal neural networks. Luckily our goal is to evaluate different network structures, so we just need to visualize the last hidden layer since it directly reveals the relationship among the model output T, input X and label Y. Since the number of neurons of the last hidden layer in the DNNs for image classification task is precisely the number of classes of input data, our method is only subject to the number of classes.

Suppose there are C classes, the outputs of the last hidden layer are scores of different classes which are unbounded. We use the normalized exponential function to squash a C dimensional real vector z of arbitrary real values to a C dimensional vector $\sigma(z)$ of real values in the range $[0,1]$ that add up to 1. The function is given by

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{c=1}^{C} e^{z_c}} \quad \text{for } j = 1, \ldots, C, \tag{7}$$

which is exactly what the softmax function does in the neural network. We bin the neuron's output $\sigma(z)$ into 10 equal intervals between 0 and 1 and get our final

model output T. Then we can calculate $I(X;T)$ and $I(T;Y)$ for any network architecture. An advantage of this calculation is that the sample space of T is a bit smaller since we enforce the C dimensional vector $\sigma(z)$ add up to 1. This process is illustrated in Fig. 2.

4 Experiments

This section goes as follows: in Section 4.1, we analyze the relationship among the model accuracy, $I(X;T)$ and $I(T;Y)$; in Section 4.2, we propose a framework that can be used to evaluate DNNs; in Section 4.3, we show the evaluation framework is more informative than the loss curve when evaluating DNNs and how to use this framework to guide us on choosing networks efficiently; in Section 4.4, we show how to apply mutual information to infer the capability of a model for objects of each class in image classification tasks.

4.1 Relationship among Classification Accuracy, $I(X;T)$ and $I(T;Y)$ in DNNs

In addition to developing the theory of deep learning, it is also important to empirically validate it. In the original IB theory [12], X, Y and T represent the *training* input, *training* label and model output, respectively; and [12] states that $I(T;Y)$ explains the *training* accuracy, $I(X;T)$ serves as a regularization term that controls the generalization. Here we find that in DNNs, low $I(X;T)$ also contributes to the training accuracy. In particular, when $I(T_1;Y)$ and $I(T_2;Y)$ are equal, the model with smaller $I(X;T)$ has a larger probability to achieve higher training accuracy.

To validate the hypothesis that low $I(X;T)$ also contributes to the training accuracy, we train neural networks on the CIFAR-10 dataset to sample values of $I(X;T)$, $I(T;Y)$ and the training accuracy. During the training process, the sampling is performed at every fixed iteration steps. For the i-th sample, we use $I(X;T_i)$, $I(T_i;Y)$ and Acc_i to denote the mutual information values and the training accuracy, respectively. A direct way to examine the rightness of our hypothesis is to find pairs (i,j) which satisfy $I(T_i;Y) = I(T_j;Y)$, then check the relationship of $I(X;T)$ and the training accuracy.

Since $I(T;Y)$ is a real number, it's hard to find a pair of samples who have the same value of $I(T;Y)$. Instead, we examine the hypothesis by checking inversions. An inversion is a pair of samples (i,j) which satisfy $I(T_i;Y) < I(T_j;Y)$ and $Acc_i > Acc_j$. Among all these inversion pairs, we calculate the percentage of pairs that satisfy $I(X;T_i) < I(X;T_j)$. This percentage is a proper indicator of the rightness of our hypothesis since if the percentage is near 0.5, then $I(X;T)$ almost has no relation to the training accuracy. Otherwise, if the percentage is high, then low $I(X;T)$ also contributes to the training accuracy. In our experiments, we set different training conditions to train neural networks. The percentages are listed in Table 1.

network structure	training method	percentages with 600 samples
CNN-9	SGD	**0.865**
	BGD	**0.821**
Linear Network	SGD	**0.755**
	BGD	**0.594**

Table 1. This table records the percentages with 600 samples for DNNs with different network structures and training methods on the training set. The percentage converges when we include 600 samples. CNN-9 is a deep convolutional neural network with 9 convolutional layers. Linear network is a feedforward network whose activation function is the identity function. SGD is short for Stochastic Gradient Descent, and BGD for Batch Gradient Descent. For computational limitation, we include 10000 training samples when performing BGD. Also BGD and SGD use the same training set.

The results in Table 1 show that $I(X;T)$ also contributes to training accuracy since the percentages are over 0.5. Different network structures may end up with different percentages. Also SGD has higher percentage than BGD. We want to emphasize that the percentages may have a little deviation from the ground truth since the mutual information in DNNs was calculated approximately by binning. This is crucial especially when mutual information values do not vary too much. We believe the accurate mutual information will make our hypothesis more convincing. Table 1 can be further interpreted as follows:

First, notice that $I(T;Y)$ is not a monotonic function of the training accuracy. For example, suppose we have C classes in the dataset, and $\mathcal{C}_i$ denotes the i-th class. Consider two cases: In the first case, $T = \sigma(Y)$ where σ is an identity mapping which means T always predicts the true class. In the second case $T = \varphi(Y)$ where φ is a shift mapping which means if the true class is $\mathcal{C}_i$, the prediction of T is $\mathcal{C}_{i+1}$. In both two cases, since σ and φ are invertible functions, from (3), we have $I(T;Y) = I(\sigma(Y);Y) = I(\varphi(Y);Y) = H(Y)$. But in case 1, the training accuracy is 1, whereas in case 2 it is 0.

Second, unlike linear networks, the loss function of CNNs is highly non-convex. By using SGD or BGD to train neural networks, the training loss respect to all the training data does not decrease all the time during the training process which indicates the network sometimes is learning in the wrong direction. Since SGD only uses a mini-batch of samples for each iteration, the loss curve becomes more unstable. Only in the linear network (the loss function is convex) trained by BGD, with a proper learning rate, the training loss always decreases during the training process, which means the model always makes T closer to the true label Y (the model is stablest in this case). So $I(T;Y)$ can fully explain the training accuracy and $I(X;T)$ may not contribute to training accuracy very much.

Third, [18] defines that a learning algorithm is stable if its output does not depend too much on any individual training example. So when $I(T_1;Y)$ and $I(T_2;Y)$ are equal, the model with low $I(X;T)$ has large stability, which may lead to a high training accuracy.

We also find that when trained by SGD, the percentages increase as more convolutional layers are considered, which can be seen from the columns in Table 2. This interesting phenomenon may reveal some inherent properties of CNNs which we would further explore in our future work.

network structure	CNN-2	CNN-4	CNN-9	CNN-16 (VGG)
percentage with 600 samples	0.56	0.68	0.87	0.96

Table 2. This table records the percentages with 600 samples for DNNs with different network structures on the training set. CNN-i is a deep convolutional neural network with i convolutional layers.

We also validate our hypothesis on the *validation* data where X and Y now represent the *validation* input and *validation* label, respectively. The percentages in Table 3 also show that low $I(X;T)$ contributes to validation accuracy. This result will be useful in the next subsection for evaluating DNNs.

number of samples	100	200	300	400	500	600
percentage	0.905	0.921	0.912	0.924	0.924	**0.924**

Table 3. The percentages with numbers of samples on the validation set. The network is VGG-16 trained by SGD.

4.2 Evaluating DNNs in the Information Plane

Evaluating the capability of DNNs during the training process is important because it would help us understand the training phase better. Section 3.2 shows that an optimal representation (a minimal sufficient statistics of X with respect to Y) is a tradeoff between $I(X;T)$ and $I(T;Y)$. We validate the hypothesis in Section 4.1 that, in DNNs trained by SGD, not only $I(T;Y)$ but also $I(X;T)$ is a measurement of validation accuracy where X and Y represent the *validation* input and *validation* label, respectively. So we use $\frac{\Delta I(T;Y)}{\Delta I(X;T)}$ (the slope of the curve) to represent the model's learning capability **at each moment** in the information plane.

Fig. 1 shows two learning phases of the training process. The model begins to generalize in the second compression phase, and the first fitting phase takes very little time compared to the compression phase. So we just use $\frac{\Delta I(T;Y)}{\Delta I(X;T)}$ in the second compression phase to evaluate the model's capability of generalization. We expect that a good model has small (negative) $\frac{\Delta I(T;Y)}{\Delta I(X;T)}$ at the second phase. While for the first fitting phase, $I(T;Y)$ and $I(X;T)$ grow simultaneously (in

order to fit the label, the model needs to remember X at first). So we use $I(T;Y)$ instead of $\frac{\Delta I(T;Y)}{\Delta I(X;T)}$ to represent the model's capability of fitting the label. Based on the discussion above, we propose our evaluation framework in Fig. 3.

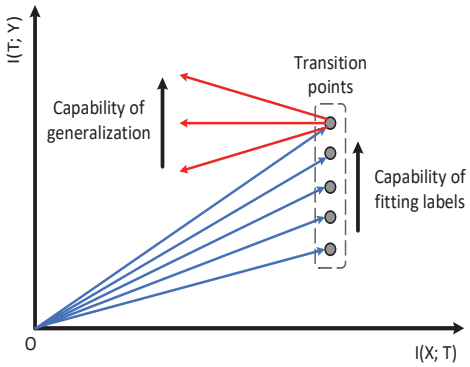

Fig. 3. Evaluation framework based on $I(X;T)$ and $I(T;Y)$. The height of transition point ($I(T;Y)$) represents the model's capability of fitting the label. The slope after transition point ($\frac{\Delta I(T;Y)}{\Delta I(X;T)}$) represents the model's capability of generalization.

We are interested in how different neural networks behave under the framework we propose in Fig. 3. So we run different network structures on MNIST and CIFAR-10 dataset (see Fig. 4). Notice that in this and the subsequent experiments, X and Y represent the *validation* input and *validation* label respectively. Mutual information curves are smoothed for better visualization since smoothing doesn't change the trend of the curve. Also, DNNs are just trained once until convergence without data augmentation or retraining since we want to compare networks in an equal way. We also record the mutual information, training epochs, model validation accuracy at the transition point and convergence point in Table 4. Fig. 4 and Table 4 show some interesting phenomenons.

- Convolutional neural networks (CNNs) may have lower capabilities of fitting the label than fully connected networks (FCs) in the first fitting phase by comparing $I(T;Y)$ at the transition point (The reason may attribute to the large number of parameters of FCs), but CNNs have stronger capabilities of generalization (smaller $\frac{\Delta I(T;Y)}{\Delta I(X;T)}$) in the compression phase which lead to higher final validation accuracies.
- Some models may not have second compression phase. For MNIST, all models have exactly two learning phases, but for CIFAR-10, the models with fewer layers don't show second compression phase (see CNN-2, CNN-4, and FC-3 for CIFAR-10 in Fig. 4). It reveals that when the dataset is harder to classify, neural networks with fewer layers can not generalize well.
- For CIFAR-10, $I(X;T)$ and $I(T;Y)$ of FC-6 and FC-9 both drop down in the second phase indicating that increasing layers in FCs may lead to overfitting.

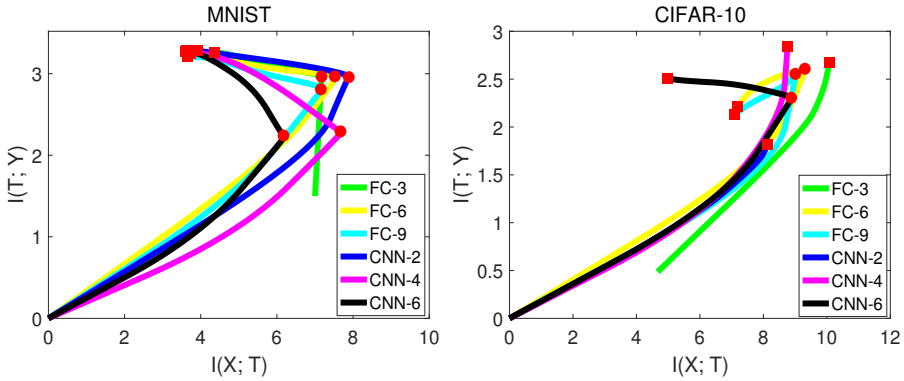

Fig. 4. The figures depict mutual information paths with training epochs in the information plane. The left and right figures represent MNIST and CIFAR-10, respectively. Both datasets are trained by fully connected neural networks and convolutional neural networks. FC-i denotes a fully connected neural network which has i layers including the input and output layers. CNN-i denotes a convolutional neural network which has i convolutional layers.

dataset	model	transition point				convergence point			
		$I(T;Y)$	$I(X;T)$	epochs	accuracy	$I(T;Y)$	$I(X;T)$	epochs	accuracy
MNIST	FC-3	2.96	7.183	1	0.836	3.259	4.358	51	0.983
	FC-6	2.962	7.532	1	0.846	3.249	3.746	56	0.988
	FC-9	2.803	7.166	1	0.774	3.214	3.647	54	0.988
	CNN-2	2.952	7.898	1	0.75	3.282	3.916	50	0.99
	CNN-4	2.286	7.683	1	0.451	3.284	3.621	53	0.994
	CNN-6	2.236	6.184	1	0.515	3.275	3.592	54	0.994
CIFAR-10	FC-3	2.671	10.085	65	0.534	2.671	10.085	65	0.534
	FC-6	2.604	9.321	20	0.537	2.218	7.197	66	0.575
	FC-9	2.55	9.02	21	0.555	2.218	7.197	66	0.56
	CNN-2	1.816	8.133	63	0.451	1.816	8.133	63	0.451
	CNN-4	2.840	8.761	67	0.705	2.840	8.761	67	0.705
	CNN-6	2.301	8.891	5	0.52	2.472	4.862	66	0.781

Table 4. The table records $I(T;Y)$, $I(X;T)$, training epochs and validation accuracy of every network at the transition point and convergence point. For FC-3, CNN-2 and CNN-4 on CIFAR-10, the values on the transition point and convergence point are the same since they don't show the compression phase.

This evaluation framework allows us to visualize any CNN or FC in the information plane. In the next subsection, we will show this framework is more informative than the loss curve when evaluating neural networks.

4.3 Informativeness and Guidance of Information Plane

Usually, for a particular problem, the network structure is determined based on the exhausting search of different DNNs on the validation set which is time-consuming. Next, we will show our evaluation framework is more informative than the loss curve and would facilitate the model selection of DNNs.

Specifically, by comparing the number of training epochs at the transition point and convergence point, we can find that most of the training time is spent on the compression phase, as shown in Table 4. So we can visualize the information plane during training the network, and stop training once the model has crossed the transition point for several epochs. The height of the transition point ($I(T;Y)$) represents the model's capability of fitting the label. The slope ($\frac{\Delta I(T;Y)}{\Delta I(X;T)}$) after transition point represents the model's capability of generalization. These two indicators will give us a general prediction about the model's quality. Fig. 5 shows the mutual information paths of different network structures on the CIFAR-10 dataset. Table 5 records the model validation accuracy and 'percentages' defined in Section 4.1.

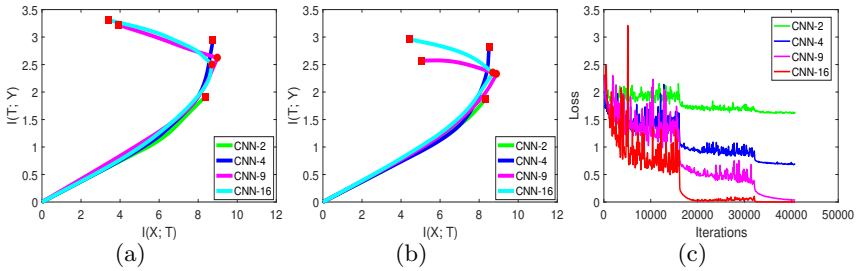

Fig. 5. (a) Mutual information path of each model with SGD optimization on the training set of CIFAR-10. (b) Mutual information path of each model on the validation set. (c) Training loss of each model with training iterations.

network structure	CNN-2	CNN-4	CNN-9	CNN-16 (VGG)
percentage of 600 samples	0.56	0.68	0.87	0.96
final acc on validation set	0.45	0.70	0.77	0.89

Table 5. The percentages of each network are from Table 2.

From Fig. 5 (c), we can see the loss of each model continues to decrease with training iterations. While in the information plane, each model behaves differently. In Fig. 5 (a) and Fig. 5 (b), the models with few layers do not have

clear second stage in the mutual information paths. Actually, we can visualize the information path of each model on the validation set to help us evaluate or select model efficiently. From Fig. 5 (b), compared with CNN-9, the slope of information path of CNN-16 in the second stage is smaller (negative), which represents better generalization capability. The validation accuracy of each model in Table 5 is consistent with our analysis. Thus, the information plane is more 'informative' than loss curve when evaluating the DNN model. Since the first stage only takes little time compared to the second stage, we can choose a better model quickly given different model architectures by visualizing the information plane on the validation set.

It is worth noticing that our prediction may not always be true, since the mutual information path may have a larger slope change in the future. So it's a trade-off between training time and confidence of our prediction. The longer time we train the network, the more confident prediction about the model we can make. But it is still an efficient way to guide us on choosing neural network structure for a given task.

Fig. 5 (a) (b) and Table 5 also show that when CNNs have fewer layers, the information plane does not clearly show the second phase, and the percentages are low. Whereas for CNN-9 and CNN-16, the information plane clearly show the second phase and the percentages are high. This experiment shows that $I(X;T)$ contributes to training accuracy mostly at the second stage of information paths. One possible reason is that the model begins to 'compress' the information of the training set and learns to generalize (extract common features from each mini-batch) at the second stage. From the percentages, this process happens even when $I(T;Y)$'s remain the same. The correlation between accuracy and $I(X;T)$ grows stronger when the number of layers of DNN increases, since DNN with more layers has better generalization capability. We can view $I(X;T)$ and $I(T;Y)$ as: $I(T;Y)$ determines how much the knowledge T has about the label Y, and $I(X;T)$ determines how easy this knowledge can be learned by the network.

4.4 Evaluating DNN's Capability of Recognizing Objects from Different Classes

Furthermore, we also evaluate the model's capability of recognizing objects from each class for the image classification task. The information plane provides a method in an informative way. Suppose there are C classes in the dataset, $\mathcal{C}_i$ denotes the i-th class. To test the model's capability of recognizing $\mathcal{C}_i$ from the data, we can label other classes in the validation data as one class, thus label Y changes from $\mathbf{R}^C$ to $\mathbf{R}^2$. When calculating the mutual information, we make label Y balanced so that $H(Y)$ is equal to 1. Then $I(X;T)$ and $I(T;Y)$ can be calculated directly given a neural network. Note that the structure of the neural network does not change. The output T is still $\mathbf{R}^C$. We only alter the way how to test the data. Repeating this process for C times and the model's capability of recognizing each class can be visualized in the information plane. This method is similar to one-vs-all classifications [19]. It measures the model's capability of recognizing the true class from all the data.

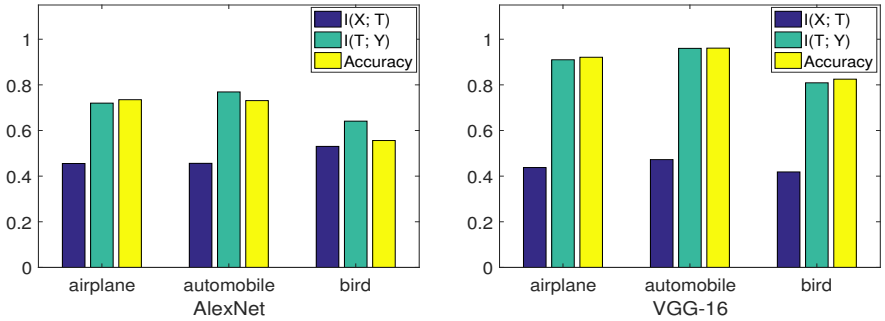

Fig. 6. Models' capabilities of recognizing objects from each class on the CIFAR-10 dataset. Models are well trained AlexNet and VGG-16. For each class, we show its $I(X;T)$, $I(T;Y)$ and validation accuracy. The validation accuracy of each class is the percentage of how many samples are correctly predicted out of all samples belonging to that class. Note that since $I(T;Y)$ is bounded by $H(Y)$ which is 1, the accuracy is also bounded by 1. To facilitate the visualization, we divide $I(X;T)$ by its upper-bound $H(X)$ so that $I(X;T)$, $I(T;Y)$ and the validation accuracy have the same magnitude.

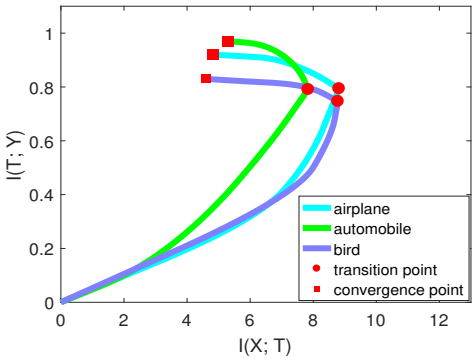

Fig. 7. Mutual information paths of different classes on CIFAR-10 dataset during the training phase for VGG-16.

For better visualization, we select the first 3 classes (airplane, automobile, bird) on CIFAR-10. Fig. 7 shows how network recognizes objects from each class during the training stage in the information plane. Fig. 6 compares different networks' recognizing capabilities for each class at the end of the training.

As shown in Fig. 7, Automobile has almost the same $I(T;Y)$ as airplane at the transition point, but automobile has smaller slop after that point. So we conclude that VGG-16 model has higher classification accuracy on automobile than airplane. For airplane and bird, model has almost equally generalization capabilities, but the capability of fitting the label of airplane is better than that of bird. So we conclude model has better classification accuracy on airplane than

bird. The final classification accuracies for these three classes are 0.921, 0.961 and 0.825 which is consistent with our analysis.

Fig. 6 shows that VGG-16 has stronger recognizing capability than AlexNet on each class. For each model, we can still use $I(X;T)$ and $I(T;Y)$ to compare each class. Like in AlexNet, after comparing $I(X;T)$ and $I(T;Y)$ of automobile and bird, we can conclude model has more recognizing capability on automobile rather than bird since automobile has a higher $I(T;Y)$ and lower $I(X;T)$.

Of course, 'model accuracy' can still be used to evaluate the model's recognizing capability for each class. But $I(X;T)$ and $I(T;Y)$ provide more information about the model's property in an informative way. Moreover, in some problems where the distribution of sample is unbalanced, we can use the information plane to test how many samples we need to train a neural network with balanced classification capability for each class.

5 Discussion

In this paper, we apply mutual information to evaluate the capability of DNNs for image classification tasks. We explore the relationship among model accuracy, $I(X;T)$ and $I(T;Y)$ in DNNs through extensive experiments. The results show that $I(X;T)$ also contributes to accuracy. We propose a general framework that can be used to evaluate DNNs in the information plane. This framework is more informative than the loss curve and can guide us on choosing network structures. We also apply mutual information to validate the network's recognizing capability for each class in the image classification tasks.

The datasets we use in the paper are MNIST and CIFAR. the difficulty of validating IB on large dataset like Imagenet is that Imagenet has 1000 classes. The sample space of T is huge and we can not calculate $I(X;T)$ and $I(T;Y)$ accurately by binning. Estimating accurate mutual information in high dimension space is still an open problem. Some future works can be done to develop more efficient ways to calculate mutual information and further explore the relationship between accuracy and $I(X;T)$ for understanding neural networks better.

Acknowledgements

This project is supported by NSFC (No. 61601288 and No. 61502304).

References

1. Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (icassp), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.
2. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
3. William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.
4. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
5. Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.
6. Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE, 2015.
7. Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.
8. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
9. Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.
10. Gal Chechik, Amir Globerson, Naftali Tishby, and Yair Weiss. Information bottleneck for gaussian variables. *Journal of Machine Learning Research*, 6(Jan):165–188, 2005.
11. DJ Strouse and David J Schwab. The deterministic information bottleneck. *Neural computation*, 29(6):1611–1630, 2017.
12. Ohad Shamir, Sivan Sabato, and Naftali Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711, 2010.
13. Susanne Still, William Bialek, and Léon Bottou. Geometric clustering using the information bottleneck method. In *Advances in Neural Information Processing systems*, pages 1165–1172, 2004.
14. Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.
15. Artemy Kolchinsky, Brendan D Tracey, and David H Wolpert. Nonlinear information bottleneck. *arXiv preprint arXiv:1705.02436*, 2017.
16. Jaesik Choi Thann T. Nguyen. Layer-wise learning of stochastic neural networks with information bottleneck. *arXiv preprint arXiv:1712.01272*, 2018.
17. Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
18. Maxim Raginsky, Alexander Rakhlin, Matthew Tsao, Yihong Wu, and Aolin Xu. Information-theoretic analysis of stability and bias of learning algorithms. In *Information Theory Workshop (ITW), 2016 IEEE*, pages 26–30. IEEE, 2016.
19. Christopher M Bishop. *Pattern Recognition and Machine Learning*. springer, 2006.