UNDERSTANDING, ANALYZING, AND OPTIMIZING THE COMPLEXITY OF DEEP MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper aims to evaluate and analyze the complexity of feature transformations encoded in DNNs. We propose metrics to measure three types of complexity of transformations based on the information theory. We further discover and prove the strong correlation between the complexity and the disentanglement of transformations. Based on the proposed metrics, we analyze two typical phenomena of the change of the transformation complexity during the training process, and explore the ceiling of a DNN's complexity. The proposed metrics can also be used as a loss to learn a DNN with the minimum complexity, which also controls the significance of over-fitting of the DNN. Comprehensive comparative studies have provided new perspectives to understand the DNN. *We will release the code when the paper is accepted.*

1 INTRODUCTION

Understanding the black-box of deep neural networks (DNNs) has attracted increasing attention in recent years. Previous studies usually interpret DNNs by either explaining DNNs visually/semantically (Lundberg & Lee, 2017; Ribeiro et al., 2016), or analyzing the feature representation capacity of a DNN (Higgins et al., 2017; Achille & Soatto, 2018a;b; Fort et al., 2019).

In this paper, we focus on the complexity of feature transformations encoded in the DNN. Based on the information theory, we define three metrics to measure the complexity of feature transformations in DNNs. The transformation complexity reveals new insights into the feature disentanglement, over-fitting, and the representation power of a DNN.

The first metric is defined as the entropy of the gating states of nonlinear operations. Specifically, let us consider the task (X, Y). Given a DNN, the complexity of transformations represents the diversity of transformations that map each input $x \in X$ to the corresponding $y \in Y$ through all images. From this perspective, the simplest model is a linear transformation $y = f(w^{\mathrm{T}}x + b) = w^{\mathrm{T}}x + b$, where the nonlinear operation $f(\cdot)$ is by-passed as an all-passing gating layer. In this way, the entropy of its gating state is zero. The complexity is quantified as the entropy of gating states of all gating layers (*e.g.* ReLU, max-Pooling, and Dropout layers). The DNN usually generates various gating states in gating layers for different inputs, thereby applying different transformations to these inputs. Let the binary vector $\sigma_l = [\sigma_l^1, \sigma_l^2, \ldots, \sigma_l^d] \in \{0, 1\}^d$ denote gating states of whether the feature in each dimension can pass through the gating layer. $\Sigma_l = \{\sigma_l\}$ denotes the set of gating states σ_l among all samples X[1]. In this way, the entropy $H(\Sigma)$ among all inputs measures the complexity of transformations in all the L gating layers, where $\Sigma = [\Sigma_1, \Sigma_2, \ldots, \Sigma_L]$.

Based on the entropy $H(\Sigma)$, we further propose $I(X; \Sigma)$ and $I(X; \Sigma; Y)$ as two additional metrics to measure the complexity of transformations. The mutual information $I(X; \Sigma)$ measures the complexity of transformations that are caused by the input. The mutual information $I(X; \Sigma; Y)$ represents the complexity of transformations that are caused by the input and are directly used for the inference. For example, in the task of object classification, not all transformations in the DNN are category-specific. $I(X; \Sigma; Y)$ reflects category-specific components of transformations. Notice that $I(X; \Sigma) \neq H(\Sigma)$ in some cases. For example, the sampling operation in the Variational AutoEncoder (VAE) and the dropout operation both introduce additional transformations that are not caused by the input X, which makes $I(X; \Sigma) \neq H(\Sigma)$.

[1]Please see Appendix I for details.

First, we prove that the complexity decreases through the layerwise propagation. In other words, deep features usually require simpler transformations to conduct inference than shallow features. Then, we use the proposed complexity metrics to diagnose DNNs. We summarize the change of the complexity during the training process into two types. In traditional DNNs without skip-connections, the complexity usually decreases first, and increases later. This indicates that DNNs may drop noisy features in early stages, and learn useful information later. Whereas, in residual networks, the transformation complexity increases monotonously during the learning process.

In particular, in this study, we conduct the following explorations based on the complexity metrics.

(1) **Disentanglement: we prove the strong correlation between the complexity and the disentanglement of transformations.** Let us consider DNNs with similar activation rates in a certain layer. If the complexity of the transformation encoded in a specific layer is larger, then gating states of different feature dimensions tend to be more independent with each other.

(2) **Minimum complexity: we use the complexity as a loss to learn a DNN with the minimum complexity.** A DNN usually learns over-complex transformations *w.r.t.* the task. Thus, we propose a complexity loss, which penalizes unnecessary transformations to learn a DNN with the minimum complexity.

(3) **Over-fitting: reducing the transformation complexity alleviates the over-fitting problem.** Given the DNN learned using the complexity loss, we find that the gap between the training loss and the testing loss of a DNN decreases, when we reduce the complexity of transformations.

(4) **Maximum complexity: we explore the ceiling of a DNN's complexity.** The complexity of a DNN does not always increase when we keep adding more gating layers into the DNN. In contrast, the traditional stacked DNN with deeper architecture may encode transformations of much lower complexity in some cases. Whereas, the transformation complexity of residual networks is saturated instead of decreasing when we used more gating layers. **Besides, the complexity of transformations does not increase monotonously along with the increase of the complexity of tasks.** In contrast, if the task complexity exceeds a certain limit, the complexity of transformations encoded in the DNN will decrease along with the increase of the task complexity.

Relationship to the information bottleneck (IB). The transformation complexity $I(X;\Sigma)$ has essential difference from the $I(X;Z)$ term in the IB (Shwartz-Ziv & Tishby, 2017), where Z denotes the feature of an intermediate layer in DNNs. In the Markov process $X \rightarrow Z \rightarrow Y$, given the feature Z, X and Y are conditional independent. In comparison, the gating state Σ does not contain all information of Z, which makes the transformation complexity $I(X;\Sigma)$ essentially different from $I(X;Z)$ in mathematics. Please see Appendix B for more discussions.

Contributions of the study can be summarized as follows. (1) We define three metrics to evaluate the complexity of transformations in DNNs. (2) We prove the strong correlation between the complexity and the disentanglement of transformations. (3) We further use the transformation complexity as a loss to learn a minimum-complexity DNN, which also reduces over-fitting. (4) Comparative studies reveal the ceiling of a DNN's complexity.

2 Related Work

In this section, we limit our discussion within the literature of understanding representations of DNNs. In general, previous studies can be roughly summarized into the following two types.

• The first type is the semantic explanations for DNNs. Some studies directly visualized knowledge representations encoded in the DNN (Zeiler & Fergus, 2014; Mahendran & Vedaldi, 2015; Yosinski et al., 2015; Dosovitskiy & Brox, 2016; Simonyan et al., 2017). Other methods estimated the pixel-wise attribution to the network output (Zhou et al., 2015; Selvaraju et al., 2017; Fong & Vedaldi, 2017; Kindermans et al., 2017; Zhou et al., 2016). The LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017) extracted important input units that directly contributed to the output. Some visual explanations reveal certain insightful properties of DNNs. Fong & Vedaldi (2017) analyzed how multiple filters jointly represented a certain semantic concept. In contrast, in this paper, we propose to investigate the representation capacity from the perspective of transformation complexity encoded in DNNs.

• The second type is to analyze the feature representation capacity of DNNs. The stiffness (Fort et al., 2019) was proposed to diagnose the generalization capacity of a DNN. Xu (2018) applied the Fourier analysis to explain the generalization capacity of DNNs. The CLEVER score (Weng et al., 2018) was used to estimate the robustness of DNNs. Wolchover (2017); Shwartz-Ziv & Tishby (2017) proposed the information bottleneck theory and used mutual information to quantify the information encoded in DNNs. Xu & Raginsky (2017); Achille & Soatto (2018b); Goldfeld et al. (2019) further extended the information theory to constrain the feature representation to learn more disentangled features. Chen et al. (2018) selected instance-wise features based on mutual information to interpret DNNs. Kornblith et al. (2019) used the canonical correlation analysis (CCA) to compare feature representations from the perspective of similarity.

Unlike previous studies, we focus on the complexity of feature transformations encoded in DNNs. Previous methods on the complexity of DNNs can be summarized as follows:

• Computational complexity and difficulty of learning a DNN: Some studies focus on the computational complexity required to learn a DNN with the certain accuracy. Blum & Rivest (1989) proved that learning a one-layer network with a sign activation function was NP-hard. Livni et al. (2014) further discussed other activation functions. Boob et al. (2018); Manurangsi & Reichman (2018) proved learning a two-layer ReLU network was also NP-hard. Arora et al. (2016) showed that it was possible to learn a ReLU network with one hidden layer in polynomial time, when the dimension of input was constant.

• Architectural complexity and representation complexity: Raghu et al. (2017) proved that as the depth of the DNN increased, the maximal complexity of features grew exponentially. Pascanu et al. (2013); Zhang et al. (2016) proposed three metrics to measure the architectural complexity of recurrent neural networks. To estimate the maximal representation capacity, Liang et al. (2017); Cortes et al. (2017) applied the Rademacher complexity. Kalimeris et al. (2019) analyzed the complexity of features in a DNN by comparing those features with the feature learned by a linear classifier.

Unlike analyzing the complexity of the DNN based on its architecture, we aim to measure the complexity of feature transformations learned by the DNN. We define three types of transformation complexity, which provide new perspectives to understand the DNN.

3 TRANSFORMATION COMPLEXITY

In this section, we define three types of complexity of transformations in DNNs based on the information theory. Given the input $x \in X$ and the target label $y \in Y$, the DNN is learned to map x to y. Layerwise transformations of mapping x to y can be roughly represented by

$$y = f(W_{L+1}\boldsymbol{\sigma_l}(W_l \dots (W_2\boldsymbol{\sigma}_1(W_1 x + b_1) + b_2) \cdots + b_l) + b_{L+1}) \tag{1}$$

where f denotes the optional layer on the top, *e.g.* the softmax layer. W_l and b_l denote the weight and the bias of the l-th linear layer. Let $\sigma_l = [\sigma_l^1, \sigma_l^2, \dots, \sigma_l^d] \in \{0,1\}^d$ denote gating states of the l-th gating layer. $\boldsymbol{\sigma}_l = diag(\sigma_l^1, \sigma_l^2, \dots, \sigma_l^d)$ is a diagonal matrix with σ_l as its main diagonal. Gating layers include the ReLU, max-Pooling, and Dropout layer. Take the ReLU layer as an example[2]. If the i-th dimension of the input feature is larger than 0, then we have $\sigma_l^i = 1$; otherwise, $\sigma_l^i = 0$. Let $\Sigma_l = \{\sigma_l\}$ denote the set of gating states of the l-th gating layer. Given a certain input x, $\sigma = [\sigma_1, \sigma_2, \dots, \sigma_L]$ represents concatenated and vectorized gating states of all gating layers. Accordingly, $\Sigma = [\Sigma_1, \Sigma_2, \dots, \Sigma_L]$ denotes the set of gating states of all L gating layers over all samples[1]. We focus on gating layers in the DNN and define the following metrics to measure three types of complexity of transformations from x to y.

• $H(\Sigma)$: the entropy of gating states among all inputs. $H(\Sigma)$ measures the complexity of transformations that are encoded in gating layers. A larger $H(\Sigma)$ indicates the DNN learns more complex transformations. The complexity $H(\Sigma)$ can be decomposed as $H(\Sigma) = H(\Sigma_1) + H(\Sigma_2|\Sigma_1) + \cdots + H(\Sigma_L|\Sigma_1, \dots, \Sigma_{L-1})$.

• $I(X;\Sigma)$: the complexity of transformations that are caused by the input. If the DNN does not introduce additional complexity that is not caused by the input, then Σ is determined by X. *I.e.* $H(\Sigma|X) = 0$ and $I(X;\Sigma) = H(\Sigma) - H(\Sigma|X) = H(\Sigma)$.

[2]Please see Appendix A for more details about other types of gating layers.

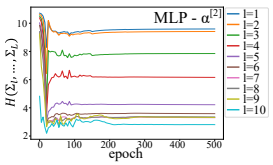


Figure 1: The complexity of transforming the feature T_l to the output, *i.e.* $H(\Sigma_l, \ldots, \Sigma_L)$.

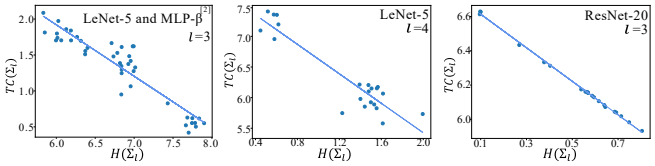


Figure 2: The negative correlation between the complexity of transformations $H(\Sigma_l)$ and the entanglement $TC(\Sigma_l)$ of DNNs with similar activation rates.

• $I(X;\Sigma;Y)$: the complexity of transformations that are caused by the input and used for the inference, which is defined as $I(X;\Sigma;Y) = I(X;Y) - I(X;Y|\Sigma)$. $I(X;Y|\Sigma) = I(X|\Sigma) - I(X|\Sigma,Y)$ measures the mutual information between X and Y that is irrelevant to gating layers.

Proposition 1. *(Proof in Appendix D.1) If the DNN does not introduce additional information that is not contained by the input X (*e.g. *there are no operations of randomly sampling or dropout throughout the DNN), we have $I(X;\Sigma;Y) \geq 0$.*

Proposition 2. Decrease of the complexity through layerwise propagation. *(Proof in Appendix D.2) Let $t_l = \boldsymbol{\sigma}_l(W_l \ldots \boldsymbol{\sigma}_1(W_1 x + b_1) \cdots + b_l) \in T_l$ denote the output feature of the l-th gating layer. Then, the complexity decreases as we use the feature of higher layers.* I.e. *$H(\Sigma_l, \ldots, \Sigma_L) \geq H(\Sigma_{l+1}, \ldots, \Sigma_L)$, $I(T_{l-1}; \{\Sigma_l, \ldots, \Sigma_L\}) \geq I(T_l; \{\Sigma_{l+1}, \ldots, \Sigma_L\})$, $I(T_{l-1}; \{\Sigma_l, \ldots, \Sigma_L\}; Y) \geq I(T_l; \{\Sigma_{l+1}, \ldots, \Sigma_L\}; Y)$, where $\Sigma_{l+1}, \ldots, \Sigma_L$ denote the gating states of the $(l+1)$-th,$\ldots$, L-th gating layer.*

In Proposition 2, we focus on the complexity of transforming the intermediate-layer feature T_l to the output Y. It shows that the transformation complexity decreases through the layerwise propagation. This proposition can also be proved by the data processing inequality (DPI (Cover, 1999)).

Verification of the decrease of the complexity through layerwise propagation. Figure 1 shows the change of the complexity of transforming the l-th layer feature T_l to the output, *i.e.* $H(\Sigma_l, \ldots, \Sigma_{10}), l = 1, \ldots, 10$ encoded in the DNN during the training process, which verified the decrease of the complexity through layerwise propagation.

Quantification of the complexity. We use the KDE (Kolchinsky & Tracey, 2017; Kolchinsky et al., 2019; Saxe et al., 2019) to estimate the complexity $H(\Sigma)$, $I(X;\Sigma)$ and $I(X;\Sigma;Y)$, which is widely used in recent years. $H(\Sigma_l)$ is usually quantified using the following upper bound.

$$H(\Sigma_l) \leq -\frac{1}{n}\sum_{j=1}^{n} \log \frac{1}{n}\sum_{k=1}^{n} \exp\left(-\frac{1}{2}\frac{\|\sigma_{l,j} - \sigma_{l,k}\|_2^2}{\sigma_0^2}\right) \tag{2}$$

where n denotes the number of training samples. $\sigma_{l,j}$ and $\sigma_{l,k}$ denote the vectorized gating states of the l-th gating layer for the j-th sample and the k-th sample, respectively. σ_0^2 is quantified as $\sigma_0^2 = \alpha \cdot Var(\Sigma_l)$, where $Var(\Sigma_l) = \mathbb{E}_x[\|\sigma_l - \mu\|^2], \mu = \mathbb{E}_x[\sigma_l]$. $Var(\Sigma_l)$ measures the variance of gating states of the l-th gating layer. α is a positive constant. The above equation can also be used to quantify $H(\Sigma)$, when we simply replace Σ_l with Σ.

If the DNN does not introduce additional complexity besides the input X, we have $H(\Sigma_l|X) = 0$, $I(X;\Sigma_l) = H(\Sigma_l) - H(\Sigma_l|X) = H(\Sigma_l)$. If the DNN introduces additional complexity (*e.g.* using the sampling operation in VAE, or the dropout operation), then $I(X;\Sigma_l)$ can be quantified as follows.

$$I(X;\Sigma_l) \leq -\frac{1}{n}\sum_{j=1}^{n} \log \frac{1}{n}\sum_{k=1}^{n} \exp\left(-\frac{1}{2}\frac{\|\hat{\sigma}_{l,j} - \hat{\sigma}_{l,k}\|_2^2}{\sigma_0^2}\right) \tag{3}$$

where $\hat{\sigma}_{l,j}$ and $\hat{\sigma}_{l,k}$ represent the vectorized gating states when sampling operations are removed (in this way, we can use the method of measuring $H(\Sigma_l)$ to quantify $I(X;\Sigma_l)$).

Similarly, the complexity $I(X;\Sigma_l;Y)$ can be estimated by its upper bound:

$$\begin{aligned} I(X;\Sigma_l;Y) \leq & - I(\Sigma_l;Y|X) - \frac{1}{n}\sum_{j=1}^{n} \log \frac{1}{n}\sum_{k=1}^{n} \exp\left(-\frac{1}{2}\frac{\|\sigma_{l,j} - \sigma_{l,k}\|_2^2}{\sigma_0^2}\right) \\ & - \sum_{m=1}^{M} p_m \left[-\frac{1}{n_m}\sum_{j,Y_j=m} \log \frac{1}{n_m}\sum_{k,Y_k=m} \exp\left(-\frac{1}{2}\frac{\|\sigma_{l,j} - \sigma_{l,k}\|_2^2}{\sigma_0^2}\right) \right] \end{aligned} \tag{4}$$

For the task of multi-category classification, M denotes the number of categories. n_m is the number of training samples belonging to the m-th category, and $p_m = n_m/M$. If the DNN does not introduce

additional complexity besides the input, we have $I(\Sigma_l; Y|X) = 0$. Equations (2)(3)(4) are given in (Kolchinsky & Tracey, 2017; Kolchinsky et al., 2019; Saxe et al., 2019).

4 ANALYSIS OF DNNS BASED ON THE TRANSFORMATION COMPLEXITY

4.1 STRONG CORRELATION BETWEEN THE COMPLEXITY AND THE DISENTANGLEMENT

Although intuitively, the disentanglement of gating states seems not related to the complexity, in this section, we prove a clear correlation between these two terms. Let us consider the complexity of the transformation encoded in a single gating layer, *e.g.* $H(\Sigma_l)$, $I(T_{l-1}; \Sigma_l)$ and $I(T_{l-1}; \Sigma_l; Y)$ for the l-th gating layer. For gating states of the l-th gating layer Σ_l, we analyze whether gating states σ_l^i in different dimensions are related to each other. The entanglement of transformations $TC(\Sigma_l)$ measures the dependence between gating states of different dimensions, which is given in (Achille & Soatto, 2018a; Ver Steeg & Galstyan, 2015) as the multi-variate mutual information.

$$TC(\Sigma_l) = KL(p(\sigma_l)||\textstyle\prod_i p(\sigma_l^i)) \tag{5}$$

where $p(\sigma_l^i)$ denotes the marginal distribution of the i-th element in σ_l. Assume that $p(\sigma_l^i) \sim \text{Bernoulli}(\alpha_l^i)$, where α_l^i is the activation rate of the i-th dimension of the l-th gating layer. In particular, $TC(\Sigma_l)$ is zero if and only if all elements of σ_l are independent with each other. In this case, we say Σ_l is disentangled. ***I.e.*** $TC(\Sigma_l)$ measures the entanglement. Further, we prove

$$H(\Sigma_l) + TC(\Sigma_l) = C_l, \quad C_l = -\mathbb{E}_{\sigma_l}[\log \textstyle\prod_i p(\sigma_l^i)] \tag{6}$$

Please see Appendix D.3 for the proof. Let us consider DNNs with similar activation rates α_l^i. Because $p(\sigma_l^i)$ follows the Bernoulli distribution with the activation rate α_l^i, for DNNs with similar activation rates α_l^i, they share similar values of C_l. In this way, there is a negative correlation between the complexity $H(\Sigma_l)$ and the entanglement $TC(\Sigma_l)$. In other words, for a specific layer, if the complexity of the transformation is larger, then gating states of different dimensions tend to be more independent with each other. We extend the similar conclusion in Eq. (6), *i.e.* the negative correlation between the complexity and the entanglement, to $I(X; \Sigma_l)$ and $I(X; \Sigma_l; Y)$ as follows.

$$\begin{aligned} I(X;\Sigma_l) + TC(\Sigma_l) &= C_l - H(\Sigma_l|X) \\ I(X;\Sigma_l;Y) + \underbrace{(TC(\Sigma_l) - TC(\Sigma_l|Y))}_{\text{multi-variate mutual information used to infer } Y} &= C_l - C_{l|Y} - (H(\Sigma_l|X) - H(\Sigma_l|X,Y)) \end{aligned} \tag{7}$$

where $C_l \triangleq -\mathbb{E}_{\sigma_l, y}[\log \prod_i p(\sigma_l^i|y)]$. If the DNN does not introduce additional information besides X and Y, then $H(\Sigma_l|X) = H(\Sigma_l|X,Y) = 0$. Just like the case in Eq. (6), for DNNs with similar activation rates, we can also roughly consider that these DNNs share similar values of $C_{l|Y}$. Please see Appendix E for more detailed discussions. Thus, these DNNs share similar values of $C_l - H(\Sigma_l|X)$, $C_{l|Y} - H(\Sigma_l|X,Y)$. ***I.e.*** the negative correlation between the complexity and the entanglement of transformations still holds true for $I(X; \Sigma_l)$ and $I(X; \Sigma_l; Y)$.

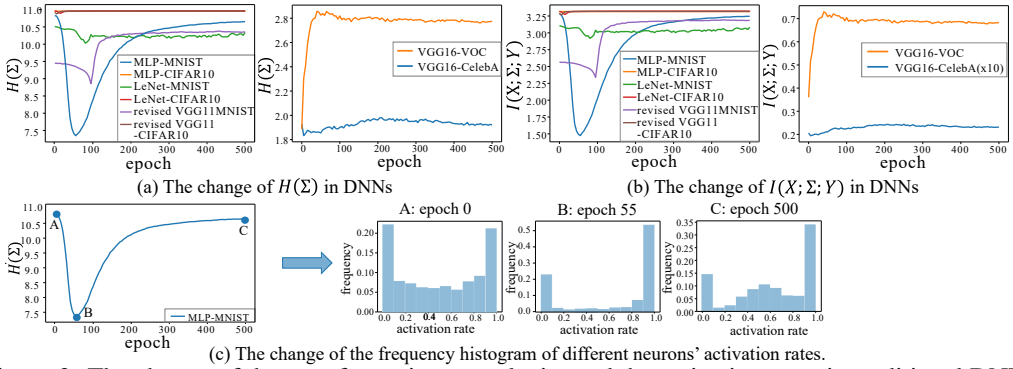
Verification of the strong correlation between the complexity and the disentanglement (Eq. (6)). We learned 21 LeNet-5 models and 21 MLP-β[3] models with different initialized parameters on the MNIST dataset. These models shared similar activation rates on each dimension in the l-th gating layer (we used $l = 3, 4$). Thus, we quantified $H(\Sigma_l)$ and $TC(\Sigma_l)$ (Please see Eq. (28) for the quantification of $TC(\Sigma_l)$) over the 42 models. We also conducted such an experiment on 21 ResNet-20 models with $l = 3$. Figure 2 shows the negative correlation between $H(\Sigma_l)$ and $TC(\Sigma_l)$, which was verified using different layers of DNNs with different architectures.

4.2 COMPARATIVE STUDIES

This section introduces several comparative studies and provides new insights about the representation capacity of DNNs.

The change of complexity during the learning of DNNs. Figure 3 shows the change of three types of transformation complexity encoded in traditional DNNs without skip-connections during

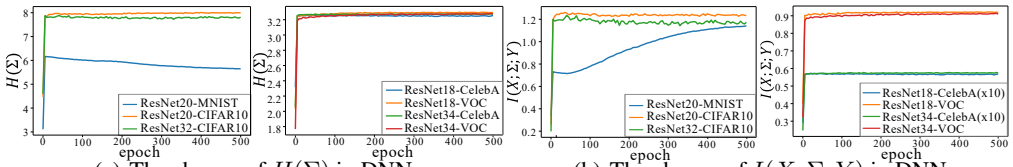
[3]Please see Section 4.4 for the network architecture and experimental settings.



(a) The change of $H(\Sigma)$ in DNNs (b) The change of $I(X;\Sigma;Y)$ in DNNs

(c) The change of the frequency histogram of different neurons' activation rates.

Figure 3: The change of the transformation complexity and the activation rates in traditional DNNs without skip-connections.



(a) The change of $H(\Sigma)$ in DNNs. (b) The change of $I(X;\Sigma;Y)$ in DNNs.

Figure 4: The change of the transformation complexity in residual networks.

the training process. Note that $H(\Sigma) = I(X;\Sigma)$. We found two phenomena in the change of complexity: **(1)** For most DNNs (like MLPs[3] and LeNets), both $H(\Sigma)$ and $I(X;\Sigma;Y)$ decreased first, and increased later. Figure 3 (c) shows the frequency histogram of different neurons' activation rates of gating states in Epoch 0, 55, and 500. In Epoch 0, gating states were usually randomly activated, which led to a large value of $H(\Sigma)$. The learning process gradually removed noisy activations, which reduced $H(\Sigma)$ and achieved the minimum complexity in Epoch 55. Then, the DNN mainly learned complex transformations to boost the performance, which made $H(\Sigma)$ begin to increase. This indicated that DNNs dropped noisy features in early stages of the training process, then learned useful information for the inference. **(2)** For other DNNs (like the VGG-16 trained on the Pascal VOC dataset), the complexity increased monotonously during the early stage of the training process, and saturated later.

Figure 4 shows the change of transformation complexity encoded in residual networks. **(3)** Both $H(\Sigma)$ and $I(X;\Sigma;Y)$ increased monotonously during the training process in most residual networks. This indicated that skip-connections reduced the influence of noisy features on the learning of DNNs.

In particular, in VAEs, $H(\Sigma) \neq H(X;\Sigma)$. We were given a VAE[3], in which both the encoder and the decoder had two FC layers. We added a classifier with two FC layers and two ReLU layers after the encoder. The VAE was trained on the MNIST dataset and the CIFAR-10 dataset, respectively. Figure 6 shows the complexity of transformations encoded in each gating layer of the classifier. **(4)** The difference between $H(\Sigma_l)$ and $I(X;\Sigma_l)$ gradually decreased during the training process. This indicated that the impact of inputs on the transformation complexity kept increasing. At the same time, the noisy features encoded in the DNN kept decreasing.

Maximum complexity: estimation of the ceiling of a DNN's complexity. Based on the CIFAR-10 dataset, we constructed a set of MLPs (termed *task MLPs*), each consisting of n ReLU layers and FC layers with the width of 1024. We used $n = 0, 1, \ldots, 31$. The task MLP first transformed images in the CIFAR-10 dataset into gray scale and took them as input. We learned another set of MLPs (termed *target MLPs*), and these target MLPs consisted of 6,12,18,24 ReLU layers and FC layers with the width of 1024. We conducted multiple experiments to train each target MLP to reconstruct the output of each task MLP with an MSE loss. In this way, since the task MLP contained n ReLU layers, we considered the complexity of using target MLPs to mimic task MLPs was Task-n.

Findings from stacked networks. Figure 5 (left) compares the complexity of transformations encoded in target MLPs (with the traditional stacked architecture) for different tasks. **(1)** For the task of low complexity, deep MLPs learned more complex transformations than shallow MLPs. **(2)** However, as the complexity of the task increased, the complexity of transformations encoded in shallow MLPs was usually higher than that encoded in deep MLPs.

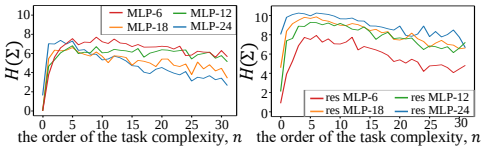


Figure 5: The change of the transformation complexity along with the increase of the task complexity n.

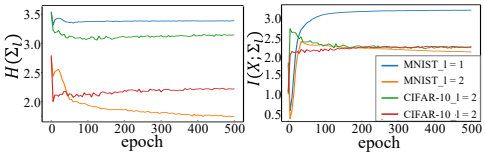


Figure 6: The change of $H(\Sigma_l)$ and $I(X;\Sigma_l)$ in VAEs learned on the MNIST dataset and the CIFAR-10 dataset.

Findings from residual networks. Besides above target MLPs, we further designed new target MLPs with skip-connections, which were termed residual MLPs. We added a skip-connection to each FC layer in each of above target MLPs. The complexity of transformations encoded in residual MLPs was shown in Figure 5 (right). **(1)** Deep residual MLPs always encoded higher transformation complexity than shallow residual MLPs. **(2)** We also found that when we gradually increased the complexity of the task to train target MLPs, the transformation complexity encoded in target MLPs increased along with the increase of the task complexity in the beginning. **(3)** However, when the task complexity exceeded a certain limit, the complexity of transformations saturated and started to decrease. This phenomenon indicated the limit of the representation capacity of DNNs.

4.3 Learning a DNN with the minimum complexity

Minimum complexity. A DNN may use over-complex transformations for prediction, *i.e.* the complexity $I(X;\Sigma;Y)$ does not always represent the real complexity of the task. In this section, we develop a method to avoid learning an over-complex DNN. The basic idea is to use the following loss to quantify and penalize the complexity of transformations during the training process.

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{complexity}} \tag{8}$$

The first term $\mathcal{L}_{\text{task}}$ denotes the task loss. *E.g.* if the task is object classification, then $\mathcal{L}_{\text{task}}$ can be the cross-entropy. The second term $\mathcal{L}_{\text{complexity}}$ penalizes the complexity of transformations encoded in the DNN. λ is a positive scalar. For simplicity, $\mathcal{L}_{\text{complexity}}$ is implemented as follows.

$$\mathcal{L}_{\text{complexity}} = \sum\nolimits_{l=1}^{L} H(\Sigma_l) = \sum\nolimits_{l=1}^{L} \{-\mathbb{E}_{\sigma_l}[\log p(\sigma_l)]\} \tag{9}$$

The exact value of $p(\sigma_l)$ is difficult to calculate, so we use an energy-based model (EBM (LeCun et al., 2006; Gao et al., 2018)) with parameter θ_f to approximate it, as follows.

$$p_{\theta_f}(\sigma_l) = \frac{1}{Z(\theta_f)} \exp[f(\sigma_l;\theta_f)] \cdot q(\sigma_l) \tag{10}$$

where $q(\sigma_l)$ is a prior distribution defined as $q(\sigma_l) = \prod_i q(\sigma_l^i)$ and $q(\sigma_l^i) \sim \text{Bernoulli}(\alpha_l^i)$. $f(\sigma_l;\theta_f)$ is implemented as a ConvNet with parameters θ_f (Gao et al., 2018). The constant $Z(\theta_f) = \int_{\sigma_l} q(\sigma_l)\exp[f(\sigma_l;\theta_f)]\text{d}\sigma_l$ is for normalization. For the EBM, the parameter θ_f is learned via the maximum likelihood estimation (MLE).

$$\hat{\theta}_f = \operatorname*{argmax}_{\theta_f} \frac{1}{n} \sum\nolimits_{j=1}^{n} \log p_{\theta_f}(\sigma_{l,j}) \tag{11}$$

where n denotes the number of samples. $\sigma_{l,j}$ is a vector, which represents gating states in the l-th gating layer for the j-th sample. We follow (Gao et al., 2018) to optimize the EBM parameters θ_f with Markov Chain Monte Carlo (MCMC).

Thus, the penalty term for the complexity can be rewritten as

$$\mathcal{L}_{\text{complexity}} = -\sum\nolimits_{l=1}^{L} \mathbb{E}_{\sigma_l}[\log p_{\hat{\theta}_f}(\sigma_l)] = \frac{1}{n} \sum\nolimits_{l=1}^{L} \sum\nolimits_{j=1}^{n} \log \left[\frac{1}{Z(\hat{\theta}_f)} \exp[f(\sigma_{l,j};\hat{\theta}_f)] \cdot q(\sigma_{l,j}) \right] \tag{12}$$

Note that the gating state $\sigma_{l,j}$ is not differentiable *w.r.t.* the network parameters. To this end, the ReLU operation can be approximated using the Swish function $\text{ReLU}(x) = x \odot \sigma_l = x \odot \text{sigmoid}(\beta x)$ (Ramachandran et al., 2017), where $\odot$ denotes the element-wise multiplication. This enables us to use $\mathcal{L}_{\text{complexity}}$ to learn network parameters. During the training process, the EBM and the original DNN are trained alternatively. For every batch of training data, we firstly train the EBM using Eq. (11). Then, we fix parameters in the EBM and optimize the original DNN with the loss $\mathcal{L} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{complexity}}$.

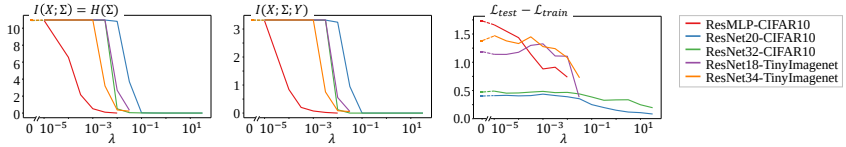


Figure 7: The complexity of transformations and the gap between the training loss and the testing loss of the learned minimum-complexity DNNs. The left-most point in each subfigure at $\lambda = 0$ refers to DNNs learned by only using the task loss.

The complexity loss reduced the complexity of transformations and the gap between the training loss and the testing loss. We added the complexity loss to the last four gating layers in each DNN to train the residual MLP[4], ResNet-20/32 (He et al., 2016a) based on the CIFAR-10 dataset, and train ResNet-18/34 based on the first ten classes in the Tiny ImageNet dataset. We repeatedly trained these DNNs with different values of λ. In particular, when $\lambda = 0$, DNNs were learned only with $\mathcal{L}_{\text{task}}$, which can be taken as baselines. The significance of over-fitting was quantified as the numerical difference between the training loss and the testing loss.

Figure 7 shows the complexity and the gap between the training loss and the testing loss of DNNs learned with different λ values. We found that $H(\Sigma_l)$ usually decreased along with the increase of λ. $I(X;\Sigma;Y)$ also decreased along with the increase of λ, which indicated that the complexity loss also effectively reduced the complexity $I(X;\Sigma;Y)$. We also found that the decrease of transformation complexity reduced the gap between the testing loss and the training loss. As Figure 8 shows, the complexity loss also decreased the testing loss in some cases.

4.4 Experimental settings

We conducted a set of comparative studies on the task of classification using the MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky et al., 2009), CelebA (Liu et al., 2015), Pascal VOC 2012 (Everingham et al., 2015), and Tiny ImageNet (Le & Yang, 2015) datasets. For the MNIST dataset and the CIFAR-10 dataset, we learned LeNet-5 (LeCun et al., 1998), the revised VGG-11 (Jastrzebski et al., 2017), the pre-activation version of ResNet-20/32 (He et al., 2016a;b), and the MLP. In particular, for the MNIST dataset, we learned three MLP models: *MLP-MNIST* contained 5 fully connected (FC) layers with the width of 784-1024-256-128-64-10, *MLP-*α contained 11 FC layers with the width of 784-1024-1024-512-512-256-256-128-128-64-16-10, and *MLP-*β contained 5 FC layers with the width of 784-1024-256-120-84-10[5]. For the CIFAR-10 dataset, the architecture of the MLP was set as 3072-1024-256-128-64-10 (termed *MLP-CIFAR10*). For the CelebA, Pascal VOC 2012, and Tiny ImageNet datasets, we learned VGG-16 (Simonyan et al., 2017) and the pre-activation version of ResNet-18/34. Considering the essential difference between convolutional layers and FC layers, we set $\alpha = 0.04$ for gating layers following each convolutional layer, and $\alpha = 0.01$ for gating layers following each FC layer. We also tested the effects of different α values to demonstrate the stability and trustworthiness of the complexity quantification in Appendix H. We used object images cropped by bounding boxes for both training and testing. Please see Appendix G for the classification accuracy of these DNNs. We analyzed the transformation complexity of ReLU layers.

5 Conclusion

In this paper, we have proposed three complexity measures for feature transformations encoded in DNNs. We further prove the decrease of the transformation complexity through layerwise propagation. We also prove the strong correlation between the complexity and the disentanglement of transformations. Based on the proposed metrics, we develop a generic method to learn a minimum-complexity DNN, which also reduces the significance of over-fitting of the DNN. Comparative studies reveal the ceiling of a DNN's complexity. Furthermore, we summarize the change of the transformation complexity during the training process into two typical cases. As a generic tool, the transformation complexity enables us to understand DNNs from new perspectives.

[4]The residual MLP has the similar architecture to the one in the *"Findings from residual networks"* paragraph in Section 4.2, with 10 FC layers and 9 ReLU layers. Both the input and features are 3072-d vectors.

[5]For comparison, we modified the architecture of MLP-MNIST and made the width of the last three FC layers be the same with the fully connected layers in the LeNet-5 network.

REFERENCES

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018a.

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40 (12):2897–2905, 2018b.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

Avrim Blum and Ronald L Rivest. Training a 3-node neural network is np-complete. In *Advances in neural information processing systems*, pp. 494–501, 1989.

Digvijay Boob, Santanu S Dey, and Guanghui Lan. Complexity of training relu neural network. *arXiv preprint arXiv:1809.10787*, 2018.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pp. 882–891, 2018.

Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 874–883. JMLR. org, 2017.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4829–4837, 2016.

Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437, 2017.

Stanislav Fort, Paweł Krzysztof Nowak, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019.

Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative convnets via multi-grid modeling and sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9155–9164, 2018.

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 *International Conference on Machine Learning*, pp. 2299–2308, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos J. Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. In *Advances in Neural Information Processing Systems 32*, pp. 3496–3506. Curran Associates, Inc., 2019.

Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

Artemy Kolchinsky and Brendan D Tracey. Estimating mixture entropy with pairwise distances. *Entropy*, 19(7):361, 2017.

Artemy Kolchinsky, Brendan D Tracey, and David H Wolpert. Nonlinear information bottleneck. *Entropy*, 21(12):1181, 2019.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. *arXiv preprint arXiv:1711.01530*, 2017.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in neural information processing systems*, pp. 855–863, 2014.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

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

Pasin Manurangsi and Daniel Reichman. The computational complexity of training relu (s). *arXiv preprint arXiv:1810.04207*, 2018.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2847–2854. JMLR. org, 2017.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144. ACM, 2016.

Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

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

K Simonyan, A Vedaldi, and A Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2017.

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

Greg Ver Steeg and Aram Galstyan. Maximally informative hierarchical representations of high-dimensional data. In *Artificial Intelligence and Statistics*, pp. 1004–1012, 2015.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.

Natalie Wolchover. New theory cracks open the black box of deep learning. *In Quanta Magazine*, 2017.

Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2524–2533, 2017.

Zhiqin John Xu. Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*, 2018.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

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

Saizheng Zhang, Yuhuai Wu, Tong Che, Zhouhan Lin, Roland Memisevic, Ruslan R Salakhutdinov, and Yoshua Bengio. Architectural complexity measures of recurrent neural networks. In *Advances in neural information processing systems*, pp. 1822–1830, 2016.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *In ICLR*, 2015.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

A GATING STATES IN GATING LAYERS

In this section, we mainly discuss gating states of different gating layers, which is mentioned in Section 3 of the paper. Let $h_l \triangleq W_l(\boldsymbol{\sigma}_{l-1}(W_{l-1}\dots(W_2\boldsymbol{\sigma}_1(W_1x+b_1)+b_2)\cdots+b_{l-1}))+b_l$ denote the input of the l-th gating layer. We consider the vectorized form of h_l. Given $h_l \in \mathbb{R}^d$, the formulation of $\boldsymbol{\sigma}_l$ in different gating layers is given as follows.

(1) ReLU layer. In this case, $\boldsymbol{\sigma}_l = diag(\sigma_l^1, \sigma_l^2, \dots, \sigma_l^d) \in \{0,1\}^d$, which is a diagonal matrix. If the i-th dimension of h_l is larger than 0, then we have $\sigma_l^i = 1$; otherwise, $\sigma_l^i = 0$.

(2) Dropout layer. In this case, $\boldsymbol{\sigma}_l = diag(\sigma_l^1, \sigma_l^2, \dots, \sigma_l^d) \in \{0,1\}^d$, which is a diagonal matrix. If the i-th dimension of h_l is not dropped, then we have $\sigma_l^i = 1$; otherwise, $\sigma_l^i = 0$.

(3) Max-Pooling layer. Since a pooling layer may change the size of the input, $\boldsymbol{\sigma}_l$ is not necessarily a square matrix. Let the output of the max-pooling layer $\boldsymbol{\sigma}_l h_l \in \mathbb{R}^{d'}$, *i.e.* the input h_l is divided into d' regions. In this case, we have $\boldsymbol{\sigma}_l \in \{0,1\}^{d' \times d}$. If $(h_l)_j$ is the largest element in the i-th region, then we have $(\boldsymbol{\sigma}_l)_{ij} = 1$; otherwise, $(\boldsymbol{\sigma}_l)_{ij} = 0$.

B THE DIFFERENCE BETWEEN THE TRANSFORMATION COMPLEXITY AND THE INFORMATION BOTTLENECK

Note that the transformation complexity $I(X;\Sigma)$ has essential difference from the $I(X;Z)$ term in the information bottleneck (Tishby et al., 2000; Wolchover, 2017; Shwartz-Ziv & Tishby, 2017), where Z denotes the feature of an intermediate layer in DNNs. The information bottleneck reflect the trade-off between $I(X;Z)$ and $I(Z;Y)$, which leads to the approximate minimal sufficient statistics. In the forward propagation, the feature Z contains all information encoded in the DNN, thereby forming a Markov process $X \to Z \to Y$. Thus, given the feature Z, X and Y are conditional independent, *i.e.* $I(X;Y|Z) = 0$. However, the gating state Σ does not contain all information of the feature Z. Let us take the ReLU layer for an instance. In ReLU layers, the gating state Σ only represents whether the elements in feature Z are positive. The information encoded in Σ cannot be directly used to conduct inference, which makes the transformation complexity $I(X;\Sigma)$ essentially different from $I(X;Z)$ in mathematics.

C DECREASING THE TESTING LOSS VIA THE COMPLEXITY LOSS

This section provides more discussions about the experiment in Section 4.3. Figure 7 has shown that the complexity loss enabled people to reduce the gap between the testing loss and the training loss. Sometimes, the complexity loss also significantly reduced the testing loss. We trained the residual MLP on the CIFAR-10 dataset and trained ResNet-34 on the first ten classes of the Tiny ImageNet dataset. Figure 8 shows that when we increased the weight λ of the complexity loss, the testing loss dropped significantly. This further demonstrated the effectiveness of the complexity loss.

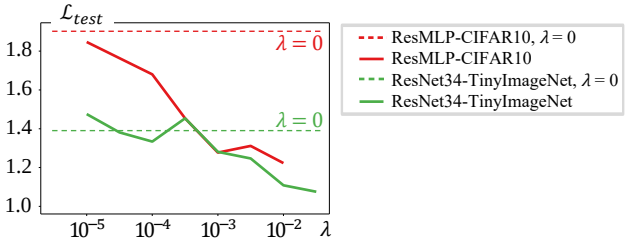


Figure 8: Decrease of the testing loss along with the increas of the weight of the complexity loss. The dashed lines show the baseline testing losses when DNNs were learned only using the task loss $\mathcal{L}_{\text{task}}$, *i.e.* $\lambda = 0$.

D PROOFS OF IMPORTANT CONCLUSIONS

This section gives detailed proof for some important conclusions in the paper.

D.1 NON-NEGATIVITY OF THE COMPLEXITY $I(X;\Sigma;Y)$

Proposition 1 *If the DNN does not introduce information besides the input X (*e.g. *there is no sampling operations or dropout operations throughout the DNN), we have $I(X;\Sigma;Y)\geq 0$.*

Proof. Recall that the mutual information is defined as

$$\begin{aligned} I(X;\Sigma;Y) &= I(X;Y) - I(X;Y|\Sigma) \\ &= (H(Y) - H(Y|X)) - (H(Y|\Sigma) - H(Y|X,\Sigma)) \\ &= (H(Y) - H(Y|\Sigma)) - (H(Y|X) - H(Y|X,\Sigma)) \end{aligned} \tag{13}$$

If the DNN does not introduce additional information besides X, which means that Σ is determined by X, then we have $H(Y|X) - H(Y|X,\Sigma) = I(\Sigma;Y|X) = 0$. Therefore,

$$I(X;\Sigma;Y) = H(Y) - H(Y|\Sigma) \geq 0 \tag{14}$$

D.2 DECREASE OF THE COMPLEXITY THROUGH LAYERWISE PROPAGATION

The complexity of transforming the intermediate-layer feature t_l to the output y decreases, when we use the feature of higher layers.

• $H(\Sigma_l,\ldots,\Sigma_L) \geq H(\Sigma_{l+1},\ldots,\Sigma_L)$

Proof.

$$\begin{aligned} & H(\Sigma_l,\ldots,\Sigma_L) - H(\Sigma_{l+1},\ldots,\Sigma_L) \\ =& H(\Sigma_l|\Sigma_{l+1},\ldots,\Sigma_L) \\ =& -\mathbb{E}_{\sigma_l,\ldots,\sigma_L}\left[\log p(\sigma_l|\sigma_{l+1},\ldots,\sigma_L)\right] \\ \geq& 0 \end{aligned} \tag{15}$$

• $I(T_{l-1};\{\Sigma_l,\ldots,\Sigma_L\}) \geq I(T_l;\{\Sigma_{l+1},\ldots,\Sigma_L\})$

Proof. If the DNN does not introduce additional information besides the input during the forward propagation, then $\Sigma_l,\Sigma_{l+1},\ldots,\Sigma_L$ are all determined by T_{l-1}, thereby $H(\Sigma_l,\ldots,\Sigma_L|T_{l-1}) = 0$. Therefore,

$$\begin{aligned} & I(T_{l-1};\{\Sigma_l,\ldots,\Sigma_L\}) - I(T_l;\{\Sigma_{l+1},\ldots,\Sigma_L\}) \\ =& (H(\Sigma_l,\ldots,\Sigma_L) - H(\Sigma_l,\ldots,\Sigma_L|T_{l-1})) - (H(\Sigma_{l+1},\ldots,\Sigma_L) - H(\Sigma_{l+1},\ldots,\Sigma_L|T_l)) \\ =& H(\Sigma_l,\ldots,\Sigma_L) - H(\Sigma_{l+1},\ldots,\Sigma_L) \\ \geq& 0 \end{aligned} \tag{16}$$

• $I(T_{l-1};\{\Sigma_l,\ldots,\Sigma_L\};Y) \geq I(T_l;\{\Sigma_{l+1},\ldots,\Sigma_L\};Y)$

Proof. According to Eq. (14), if there is no additional information besides the input throughout the DNN, then $I(T_{l-1};\{\Sigma_l,\ldots,\Sigma_L\};Y) = H(Y) - H(Y|\{\Sigma_l,\ldots,\Sigma_L\})$. We can obtain the following inequality:

$$\begin{aligned} & I(T_{l-1};\{\Sigma_l,\ldots,\Sigma_L\};Y) - I(T_l;\{\Sigma_{l+1},\ldots,\Sigma_L\};Y) \\ =& (H(Y) - H(Y|\{\Sigma_l,\ldots,\Sigma_L\})) - (H(Y) - H(Y|\{\Sigma_{l+1},\ldots,\Sigma_L\})) \\ =& H(Y|\{\Sigma_{l+1},\ldots,\Sigma_L\}) - H(Y|\{\Sigma_l,\ldots,\Sigma_L\}) \\ =& I(\Sigma_l;Y|\{\Sigma_{l+1},\ldots,\Sigma_L\}) \\ \geq& 0 \end{aligned} \tag{17}$$

D.3 STRONG CORRELATIONS BETWEEN THE COMPLEXITY AND THE DISENTANGLEMENT OF TRANSFORMATIONS

Some previous studies used the entanglement (the multi-variate mutual information) to analyze the information encoded in DNNs. Ver Steeg & Galstyan (2015) used $TC(X)$ to measure the correlation between different input samples. In comparison, in this paper, we apply $TC(\Sigma)$ to measure the independence between gating states of different dimensions. Intuitively, the disentanglement of gating states does not seem related to the complexity. Therefore, our contribution is to find out the strong correlation between the two factors which seem not related.

We consider the complexity of the transformation of the a single gating layer, *e.g.* $H(\Sigma_l)$, $I(X;\Sigma_l)$ and $I(X;\Sigma_l;Y)$ for the l-th gating layer.

• $H(\Sigma_l)$

Proof.

$$\begin{aligned} H(\Sigma_l) + TC(\Sigma_l) &= H(\Sigma_l) + KL(p(\sigma_l)||\prod_i p(\sigma_l^i)) \\ &= \mathbb{E}_{\sigma_l}\left[\log \frac{1}{p(\sigma_l)}\right] + \mathbb{E}_{\sigma_l}\left[\log \frac{p(\sigma_l)}{\prod_i p(\sigma_l^i)}\right] \\ &= -\mathbb{E}_{\sigma_l}\left[\log \prod_i p(\sigma_l^i)\right] \quad p(\sigma_l^i) \text{ does not depend on the input} \\ &= C_l \end{aligned} \tag{18}$$

Let us consider DNNs with similar activation rates α_l^i. Because $p(\sigma_l^i)$ follows the Bernoulli distribution with the activation rate α_l^i, for DNNs with similar activation rates α_l^i, they share similar values of C_l. It this case, there is a negative correlation between $H(\Sigma_l)$ and $TC(\Sigma_l)$.

•$I(X;\Sigma_l)$

Proof.

$$\begin{aligned} I(X;\Sigma_l) + TC(\Sigma_l) &= H(\Sigma_l) - H(\Sigma_l|X) + KL(p(\sigma_l)||\prod_i p(\sigma_l^i)) \\ &= C_l - H(\Sigma_l|X) \end{aligned} \tag{19}$$

If the DNN does not introduce additional information through the layerwise propagation, then the X determines Σ_l, *i.e.* $H(\Sigma_l|X) = 0$. Thus, for DNNs with similar values of C_l, there is a negative correlation between $I(X;\Sigma_l)$ and $TC(\Sigma_l)$.

•$I(X;\Sigma_l;Y)$

Proof. According to the definition of $I(X;\Sigma_l;Y)$, we have

$$I(X;\Sigma_l;Y) = I(X;\Sigma_l) - I(X;\Sigma_l|Y) \tag{20}$$

We have discussed the first term $I(X;\Sigma_l)$ above, so we focus on the second term $I(X;\Sigma_l|Y)$, which measures the complexity of transformations that are unrelated to the inference. Similarly, the entanglement of the inference-irrelevant transformations is represented by

$$TC(\Sigma_l|Y) = \mathbb{E}_y(KL(p(\sigma_l|y)||\prod_i p(\sigma_l^i|y))) \tag{21}$$

Then, we have

$$\begin{aligned} I(X;\Sigma_l|Y) + TC(\Sigma_l|Y) &= H(\Sigma_l|Y) - H(\Sigma_l|X,Y) + TC(\Sigma_l|Y) \\ &= \mathbb{E}_y \left[H(\Sigma_l|y) + KL(p(\sigma_l|y)||\prod_i p(\sigma_l^i|y)) \right] - H(\Sigma_l|X,Y) \\ &= \mathbb{E}_{\sigma_l,y} \left[\log \frac{1}{p(\sigma_l|y)} + \log \frac{p(\sigma_l|y)}{\prod_i p(\sigma_l^i|y)} \right] - H(\Sigma_l|X,Y) \\ &= -\mathbb{E}_{\sigma_l,y} \left[\log \prod_i p(\sigma_l^i|y) \right] - H(\Sigma_l|X,Y) \\ &= C_{l|Y} - H(\Sigma_l|X,Y) \end{aligned} \tag{22}$$

If there is no additional information beside the input in the DNN, then $H(\Sigma_l|X,Y) = 0$. Thus, we have

$$\begin{aligned} I(X;\Sigma_l) + TC(\Sigma_l) &= C_l - H(\Sigma_l|X) = C_l \\ I(X;\Sigma_l|Y) + TC(\Sigma_l|Y) &= C_{l|Y} - H(\Sigma_l|X,Y) = C_{l|Y} \end{aligned} \tag{23}$$

Therefore,

$$\begin{aligned} I(X;\Sigma_l;Y) &= I(X;\Sigma_l) - I(X;\Sigma_l|Y) \\ &= (C_l - TC(\Sigma_l)) - (C_{l|Y} - TC(\Sigma_l|Y)) \\ &= (C_l - C_{l|Y}) - \underbrace{(TC(\Sigma_l) - TC(\Sigma_l|Y))}_{\text{multi-variate mutual information used to infer } Y} \end{aligned} \tag{24}$$

where the difference between $TC(\Sigma_l)$ and $TC(\Sigma_l|Y)$ represents the entanglement of the transformations that are used to infer Y. For DNNs with similar activation rates, we can also roughly consider that these DNNs share similar values of C_l and $C_{l|Y}$. Thus, we can conclude that the higher complexity makes the DNN use more disentangled transformation for inference.

E About values of $C_{l|Y}$

In experiments, we found that in most cases, for DNNs with similar activation rates α_l^i in their corresponding layers, these DNNs usually shared similar values of $C_{l|Y}$. However, in some extreme cases, *e.g.* when the DNN was learned from very few training samples, or when the target layer was very close to the input layer or the output layer, values of $C_{l|Y}$ of these DNNs were different from those values of other DNNs.

F Quantification of $I(\hat{T};X)$, $I(\hat{T};Y)$, and $TC(\Sigma_l)$

This section introduces more details about the KDE approach in Section 3, which is used to quantify the complexity. The Kernel Density Estimation (KDE) approach was proposed to estimate the mutual information between the input X and the feature of an intermediate layer T in a DNN (Kolchinsky & Tracey, 2017; Kolchinsky et al., 2019). The KDE approach assumes that the intermediate-layer feature is distributed as a mixture of Gaussians. Since that T is a continuous variable, $H(T)$ can be negative. The KDE method transforms each feature point into a local Gaussian distribution to approximate the accurate feature distribution. Let $\hat{T} = T + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_0^2 I)$. Then, the distribution of $\hat{T}$ can be considered as a mixture of Gaussians, with a Gaussian centered on T. In this setting (Kolchinsky & Tracey, 2017; Kolchinsky et al., 2019; Saxe et al., 2019), an upper bound for the mutual information with the input is

$$I(\hat{T};X) \leq -\frac{1}{P} \sum_i \log \frac{1}{P} \sum_j \exp\left(-\frac{1}{2} \frac{||t_i - t_j||^2}{\sigma_0^2} \right) \tag{25}$$

where P is the number of training samples, and t_i denotes the intermediate-layer feature of the input sample i. Similarly, the upper bound for the mutual information *w.r.t* the output Y can be calculated

as

$$\begin{aligned} I(\hat{T};Y) =& H(\hat{T}) - H(\hat{T}|Y) \\ =& -\frac{1}{P}\sum_i \log \frac{1}{P}\sum_j \exp\left(-\frac{1}{2}\frac{||t_i - t_j||^2}{\sigma_0^2}\right) \\ & -\sum_{l=1}^{L} p_l \left[-\frac{1}{P}\sum_{i:Y_i=l} \log \frac{1}{P}\sum_{j:Y_j=l} \exp\left(-\frac{1}{2}\frac{||t_i - t_j||^2}{\sigma_0^2}\right)\right] \end{aligned} \tag{26}$$

where L is the number of categories. P_l denotes the number of samples belonging to the l-th category. $p_l = P_l/P$ denotes the probability of the category l.

The entanglement of transformations is formulated as

$$TC(\Sigma_l) = KL(p(\sigma_l)||\prod_i p(\sigma_l^i)) = \mathbb{E}_{\sigma_l}\left[\log \frac{p(\sigma_l)}{\prod_i p(\sigma_l^i)}\right], \tag{27}$$

where $p(\sigma_l^i)$ denotes the marginal distribution of the i-th element in σ_l. To enable fair comparisons between $I(\hat{T}, X)$ computed by the KDE method in Eq. (25) and $TC(\Sigma_l)$, we also apply the KDE method to approximate $TC(\Sigma_l)$. To this end, we synthesize a new distribution $p(\hat{\sigma}_l)$ to represent the distribution of $\prod_i p(\sigma_l^i)$. In $\hat{\sigma}_l$, $\hat{\sigma}_l^i$ in each dimension follows the Bernoulli distribution with the same activation rate α_l^i with the original σ_l^i. Gating states $\hat{\sigma}_l^i$ in different dimensions are independent with each other. In this way, $\prod_i p(\sigma_l^i)$ can be approximated by $p(\hat{\sigma}_l)$.

Inspired by (Kolchinsky & Tracey, 2017; Kolchinsky et al., 2019), $TC(\Sigma_l)$ is quantified as the following upper bound.

$$\begin{aligned} TC(\Sigma_l) &= \mathbb{E}_{\sigma_l}\left[\log \frac{p(\sigma_l)}{p(\hat{\sigma}_l)}\right] \\ &\leq \frac{1}{P}\sum_j \log \frac{\sum_k \exp\left(-\frac{1}{2}\frac{\|\sigma_{l,j} - \sigma_{l,k}\|_2^2}{\sigma_0^2}\right)}{\sum_k \exp\left(-\frac{1}{2}\frac{\|\sigma_{l,j} - \hat{\sigma}_{l,k}\|_2^2}{\sigma_0^2}\right)} \end{aligned} \tag{28}$$

where P denotes the number of samples. $\hat{\sigma}_{l,k}$ denotes the synthesized gating states, which have the same activation rates with the gating states of the sample k.

G THE CLASSIFICATION ACCURACY OF DNNS IN COMPARATIVE STUDIES

This section contains more details of DNNs in Section 4. We trained five types of DNNs on the MNIST dataset and the CIFAR-10 dataset, and trained three types of DNNs on the CelebA dataset and the Pascal VOC 2012 dataset. Table 1 reports the testing accuracy of the trained DNNs.

Table 1: The classification accuracy of DNNs on different datasets.

(a) On the MNIST and CIFAR-10 datasets.

	MLP	LeNet-5	ResNet-20	ResNet-32	revised VGG-11
MNIST	96.52%	97.41%	98.70%	98.24%	99.00%
CIFAR-10	52.52%	61.5%	81.75%	79.76%	84.53%

(b) On the CelebA and Pascal VOC 2012 datasets.

	ResNet-18	ResNet-34	VGG-16
CelebA	80.25%	80.91%	89.70%
Pascal VOC 2012	67.99%	64.27%	62.50%

H THE VALUE OF α IN THE KDE APPROACH

In this section, we discuss about the value of the hyper-parameter α used in the KDE approach. Note that the features of convolutional layers usually contain far more dimensions than features of fully-connected layers. Therefore, we set $\alpha = 0.04$ for gating layers following each convolutional layer,

and $\alpha = 0.01$ for gating layers following each FC layer. We also tested the effects of different α values to the quantification of the complexity. Figure 9 shows the complexity $I(X;\Sigma)$ and $I(\Sigma;Y)$ calculated by different values of α in MLP-α learned on the MNIST dataset. We found that the value of α did not affect the change of the complexity during the training process.

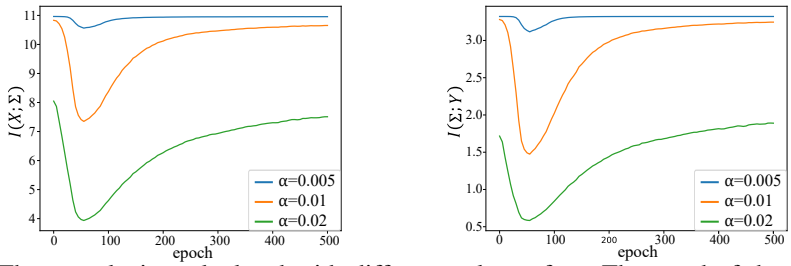


Figure 9: The complexity calculated with different values of α. The trend of the change of the complexity was consistent when we used different values of α, which enabled fair comparisons between different DNNs.

I EXPERIMENTAL DETAILS

Recall that $\Sigma_l = \{\sigma_l\}$ denotes the set of gating states σ_l among all samples X. In this paper, we randomly sample 2000 images from the training set of the each dataset for the calculation of the transformation complexity. Thus, X denotes the set of 2000 randomly sampled images that are used for analysis.

J LEARNING A MINIMUM-COMPLEXITY DNN

This section introduces more details about the learning of a minimum-complexity DNN in Section 4.3. In Section 4.3, the complexity loss is defined as

$$\mathcal{L}_{\text{complexity}} = \sum\nolimits_{l=1}^{L} H(\Sigma_l) = \sum\nolimits_{l=1}^{L} \{-\mathbb{E}_{\sigma_l}[\log p(\sigma_l)]\} \tag{29}$$

The exact value of $p(\sigma_l)$ is difficult to calculate. Thus, inspired by (Gao et al., 2018), we design an energy-based model (EBM) $p_{\theta_f}(\sigma_l)$ to approximate it, as follows.

$$\begin{aligned} p_{\theta_f}(\sigma_l) &= \frac{1}{Z(\theta_f)} \exp[f(\sigma_l;\theta_f)] \cdot q(\sigma_l) \\ Z(\theta_f) &= \mathbb{E}_q\left[\exp[f(\sigma_l;\theta_f)]\right] = \int_{\sigma_l} q(\sigma_l) \exp[f(\sigma_l;\theta_f)] \mathrm{d}\sigma_l \end{aligned} \tag{30}$$

where $q(\sigma_l)$ denotes the prior distribution, which is formulated as follows.

$$q(\sigma_l) = \prod_i q(\sigma_l^i), \quad q(\sigma_l^i) = \begin{cases} \hat{p} & \sigma_l^i = 1 \\ 1-\hat{p} & \sigma_l^i = 0 \end{cases} \tag{31}$$

If we write the EBM as $p_{\theta_f}(\sigma_l) = \frac{1}{Z(\theta_f)} \exp[-\mathcal{E}(\sigma_l)]$, then the energy function is as follows.

$$\mathcal{E}_{\theta_f}(\sigma_l) = -\log q(\sigma_l) - f(\sigma_l;\theta_f) \tag{32}$$

The EBM can be learned via the maximum likelihood estimation (MLE) with the following loss.

$$\hat{\theta}_f = \arg\max_{\theta_f} L(\theta_f) = \arg\max_{\theta_f} \frac{1}{n} \sum_{j=1}^{n} \log p_{\theta_f}(\sigma_{l,j}) \tag{33}$$

where n denotes the number of samples. $\sigma_{l,j}$ is a vector, which represents gating states in the l-th gating layer for the j-th sample.

The loss and gradient of θ_f can be calculated as follows.

$$L(\theta_f) = -\frac{1}{n}\sum_{j=1}^{n}\log p_{\theta_f}(\sigma_{l,j}) = -\frac{1}{n}\sum_{j=1}^{n}[f(\sigma_{l,j};\theta_f) + \log q(\sigma_{l,j})] + \log Z(\theta_f) \tag{34}$$

$$\frac{\partial L(\theta_f)}{\partial \theta_f} = \mathbb{E}_{\theta_f}\left[\frac{\partial}{\partial \theta_f} f(\sigma_l;\theta_f)\right] - \frac{1}{n}\sum_{j=1}^{n}\frac{\partial}{\partial \theta_f} f(\sigma_{l,j};\theta_f) \tag{35}$$

where $\frac{\partial}{\partial \theta_f}\log Z(\theta_f) = \mathbb{E}_{\theta_f}[\frac{\partial}{\partial \theta_f} f(\sigma_l;\theta_f)]$.

The first term $\mathbb{E}_{\theta_f}\left[\frac{\partial}{\partial \theta_f} f(\sigma_l;\theta_f)\right]$ in the above equation is analytically intractable and has to be approximated by MCMC, such as the Langevin dynamics.

$$\begin{aligned}\sigma_l^{\text{new}} &= \sigma_l - \frac{\Delta\tau}{2}\frac{\partial}{\partial \sigma_l}\mathcal{E}_{\theta_f}(\sigma_l) + \sqrt{\Delta\tau}\epsilon \\ &= \sigma_l + \frac{\Delta\tau}{2}\left[\frac{\partial f(\sigma_l;\theta_f)}{\partial \sigma_l} + \sum_{i=1}^{d}\frac{1}{q(\sigma_l^i)}\frac{\partial q(\sigma_l^i)}{\partial \sigma_l^i}\right] + \sqrt{\Delta\tau}\epsilon\end{aligned} \tag{36}$$

where $\epsilon \backsim N(\mathbf{0}, \mathbf{I})$ is a Gaussian white noise. $\Delta\tau$ denotes the size of the Langevin step.

Then, the Monte Carlo approximation to $\frac{\partial L(\theta_f)}{\partial \theta_f}$ is given as follows.

$$\begin{aligned}\frac{\partial L(\theta_f)}{\partial \theta_f} &\approx \frac{1}{n}\sum_{j=1}^{n}\frac{\partial}{\partial \theta_f} f(\widetilde{\sigma}_{l,j};\theta_f) - \frac{1}{n}\sum_{j=1}^{n}\frac{\partial}{\partial \theta_f} f(\sigma_{l,j};\theta_f) \\ &= \frac{\partial}{\partial \theta_f}\left[\frac{1}{n}\sum_{j=1}^{n}\mathcal{E}_{\theta_f}(\sigma_{l,j}) - \frac{1}{n}\sum_{j=1}^{n}\mathcal{E}_{\theta_f}(\widetilde{\sigma}_{l,j})\right]\end{aligned} \tag{37}$$

where $\widetilde{\sigma}_{l,j}$ is the sample synthesized via Langevin dynamics.

Thus, the loss for the learning of the DNN can be rewritten as follows.

$$\begin{aligned}\mathcal{L}_{\text{complexity}} &= -\sum_{l=1}^{L}\mathbb{E}_{\sigma_l}[\log p_{\hat{\theta}_f}(\sigma_l)] \\ &= \frac{1}{n}\sum_{l=1}^{L}\sum_{j=1}^{n}[\mathcal{E}_{\hat{\theta}_f}(\sigma_{l,j}) + \log Z(\hat{\theta}_f)] \\ &= -\frac{1}{n}\sum_{l=1}^{L}\sum_{j=1}^{n}[f(\sigma_{l,j};\hat{\theta}_f) + \log q(\sigma_{l,j}) - \log Z(\hat{\theta}_f)]\end{aligned} \tag{38}$$

Let θ_{DNN} denote parameters in the DNN. The gradient of θ_{DNN} can be calculated as follows.

$$\frac{\partial Loss}{\partial \theta_{\text{DNN}}} = -\frac{1}{n}\sum_{l=1}^{L}\sum_{j=1}^{n}\left\{\frac{\partial f(\sigma_l;\hat{\theta}_f)}{\partial \sigma_{l,j}} + \sum_i \frac{1}{q(\sigma_{l,j}^i)}\frac{\partial q(\sigma_{l,j}^i)}{\partial \sigma_{l,j}^i}\right\}\frac{\partial \sigma_{l,j}}{\partial \theta_{\text{DNN}}} \tag{39}$$

We consider $Z(\theta_f)$ as a constant in the computation of $\frac{\partial Loss}{\partial \hat{\theta}_{\text{DNN}}}$.

To enable the computation of $\frac{\partial \sigma_{l,j}}{\partial \theta_{DNN}}$ and $\frac{\partial q(\sigma_{l,j}^i)}{\partial \sigma_{l,j}^i}$, we can approximate the ReLU operation using the following Swish function (Ramachandran et al., 2017).

$$\begin{aligned}\sigma_l &\approx \text{sigmoid}(\beta x) \\ \text{ReLU}(x) = x \odot \sigma_l &\approx x \odot \text{sigmoid}(\beta x)\end{aligned} \tag{40}$$

where $\odot$ denotes the element-wise multiplication.

According to Eq. (31), the prior distribution $q(\sigma_l)$ is approximated as follows.

$$q(\sigma_l^i) \approx 1 - \hat{p} + \sigma_l^i(2\hat{p} - 1), \quad \frac{\partial q(\sigma_l^i)}{\partial \sigma_l^i} \approx 2\hat{p} - 1 \tag{41}$$

In implementation, the EBM is a bottom-up ConvNet with six convolutional layers, which takes σ_l as an input and outputs a scalar. During the training phase, we firstly train the EBM using Eq. (37) for every batch of training data. The EBM and the original DNN are trained separately. *I.e.* when training the EBM, parameters in the original DNN are fixed, and vice versa.