Successive Information Bottleneck and Applications in Deep Learning

Yassine Yousfi
Electrical and Computer Engineering
Binghamton University-SUNY
yyousfi1@binghamton.edu

Emrah Akyol
Electrical and Computer Engineering
Binghamton University-SUNY
eakyol@binghamton.edu

***Abstract*—Information Bottleneck (IB) method studies the trade-off between compression and prediction: extracting relevant information from the input variable X while preserving relevant information about another random variable Y, the resulting representation is another random variable Z. Motivated by the deep neural networks implementations, this paper studies a novel variation of the N-layer IB problem where the layers are assumed to be encoded in a successive fashion. We propose a method for performing an N-Layer IB in a greedy fashion and analyze numerical results obtained over a set of synthetic experiments.**

***Index Terms*—Information Bottleneck, Deep Learning**

I. Introduction

Extracting relevant information from high-dimensional data is the core problem of the broad Machine Learning discipline. In supervised learning settings, this task is performed via pre-specified labels given to training data. Deep Learning methods trained via such labeled data outperform most of its competitors. However, a comprehensive understanding of theoretical underpinnings of Deep Learning has remained an elusive goal. Beyond an isolated theoretical interest, this understanding would provide significant practical benefits including determination of optimized design parameters without tuning. One heuristic towards achieving this goal is to utilize lossy compression ideas with the premise that minimal sufficient statistics in information theory can successfully model the relevant information extraction process of Deep Neural Networks (DNNs). This exactly corresponds to the problem of the Information Bottleneck (IB) method, originally proposed in [1]: compressing one random variable (features) to generate a compressed version (features, i.e., low dimensional representation) while preserving as much information on another other variable (labels).

One of the existing open problems of Deep Learning is the determination of the many hyper-parameters involved, in other words, the *architecture design* task. Particularly, it is not clear apriori, how many layers should an optimized DNN use or how many neurons should exist in each layer. Our primary goal in this paper is to investigate the optimal trade-off between the depth (number of layers) and the width (number of neurons) in the presence of a complexity constraint that is measured via the number of synapses, i.e., the number of connections between neurons in fully connected DNNs. Towards this goal, we model the DNN architecture as a successive IB problem and numerically analyze the performance of successive IB schemes that have the same complexity but comprised of different number of layers and neurons. We are particularly interested in obtaining insights into the following question: how different layer-neuron configurations at the same complexity perform through the lens of the Information Bottleneck theory?

The problem of shallow/wide vs deep/thin DNNs has been open for many years in the Deep Learning literature [2]–[4] with first insights pointing at the advantages of deep architectures in terms of expressivity and efficiency to represent common functions. However, recent work [5]–[7] show that in practice, the width vs depth is usually a trade-off that needs to be cautiously tuned in order to avoid over-fitting.

On the other hand, the IB theory has recently received a revived interest due to its modeling ability of the Deep Neural Networks (DNNs). A few notable examples include, [8]–[10], where researchers explore the following question: can the apparent superior performance of DNNs trained via stochastic gradient descent algorithms be attributed to the fact that they form an optimal IB compressed representation of the data? Towards confirming the validity of IB in DNN models, in [11], authors obtained competitive results via a DNN optimizing the IB loss. Beyond modeling the inner machinery of DNN, IB has been used to improve output calibration and detection of out-of-distribution data [12], robustness to adversarial examples [13], and to optimally prune layers of DNNs that are generated via classical (cross-entropy loss, etc.) training methods [14]. In another relevant recent work [15], β-VAE, a variational auto-encoder is learned via an IB inspired objective. In [16], authors argue that IB-based optimization metrics forces the DNN in β-VAE to learn a disentangled latent representation.

This paper is organized as follows. In next section, we present preliminaries, notations, and prior art. In Section III, we describe the problem formulation and the proposed algorithm. Section IV shows numerical results and discussions. We present our conclusions and discuss future research directions in Section V.

This work is supported by NSF CCF grant # 1910715.

II. Preliminaries

A. Notation

We denote the random variables as capital letters e.g., $X, Y, Z,$, and alphabets in calligraphic notations $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$. For simplicity, we take the random variables as discrete valued in finite alphabets. For conciseness of notation, we use $p(x)$ to denote $p(X = x)$.

B. Prior art

1) Point-to-Point IB: In the original IB formulation, the quantized variable Z is found by minimizing the IB functional:

$$\min_{p(z|x)} I(X;Z) - \beta I(Y;Z) \tag{1}$$

Where $\beta > 0$ is a Lagrange multiplier which controls the trade-off between preserving information about Y and compressing X. Varying β produces the IB plane. We note that $Y \to X \to Z$ forms a Markov chain in this order, and it is captured in the problem formulation above (the optimization variable is $p(z|x)$ as opposed to $p(z|x,y)$). Algorithms to solve the IB problem for a given input source $p(x,y)$ are described in [17]. We focus on the iterative IB algorithm which updates using the 3 self constrained equations:

$$\begin{cases} p(z|x) & = \frac{p(z)}{N(x,\beta)} \exp\{\beta \sum_y p(y|x) \log \frac{p(y|z)}{p(y|x)}\}, \\ p(z) & = \sum_x p(z|x)p(x), \\ p(y|z) & = \sum_x p(y|x)p(x|z). \end{cases} \tag{2}$$

where $N(x,\beta)$ serves here as a normalization term.

2) N-Layer Information Bottleneck: Here, we summarize *one* particular generalization of the point-to-point IB method above to N layers to adopt to the DNN architecture. For a N-Layer representation of the data $(Z_1, ..., Z_N)$, a natural generalization of the IB problem follows from N-layer extension of the scalable compression analyzed in the rate-distortion literature [18], [19] with a well-defined distortion function:

$$d_i(X, Z_i) = D_{KL}(p(y|x)||p(y|z_i)). \tag{3}$$

Hence, the N-layer scalable R-D problem formulation minimizes the following Lagrangian functional:

$$\sum_{i=1}^{N} I(X; Z_1, ..., Z_i) - \beta_i I(Y; Z_i) \tag{4}$$

This formulation however is not sufficient to capture the Markov chain condition $Y \to X \to Z_1 \to ... \to Z_N$ which is a fundamental characteristics of a DNN structure, i.e. each "layer" Z_i'should only depend on Z_{i-1} statistically. By simply plugging this Markov chain condition the first term of the optimization problem yields:

$$I(X; Z_1, ..., Z_N) \quad = \quad I(X; Z_1) \tag{5}$$

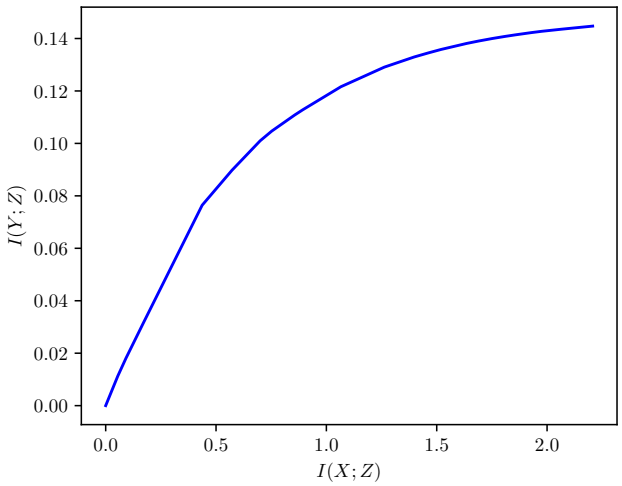

Fig. 1. An illustration of the IB plane for a given input $p(x,y)$ and a constraint on the quantized variable size $|\mathcal{Z}|$

III. Proposed Successive IB Method

Here, as opposed to the prior work in generalizing IB to N layers, we formulate the minimization problem as follows:

$$\min_{p(z_1,...,z_N|x)} \sum_{j=1}^{N} I(Z_{j-1}; Z_j) - \beta_j I(Y; Z_j) \tag{6}$$

Where $Z_0 = X$, under Markov chain constraint $Y \to X \to Z_1 \to ... \to Z_N$. Our primary motivation is the fact that this formulation models, much more accurately than the previous extensions in [20], [21], the mechanics of a feed forward DNN where each layer is a sequential transformation of the previous layer as shown in Figure 2. We also note that this formulation admits the Markov chain condition, which is not possible for the direct adoption of the results in N-Layer scalable source compression as discussed in Section II-B2. While we omit the exact solution of this problem here, to demonstrate the validity of the basic ideas in a practical DNN implementation, we focus on a simple greedy optimization procedure inspired by greedy layer-wise training in Deep Learning [22] where each layer of a fully connected DNN is trained in an unsupervised manner, the training is done independently layer-by-layer which produces a greedy algorithm.

We essentially solve N independent point-to-point IB problems. We note that the successive greedy optimization Algorithm 1 also translates to Deep Neural Networks with intermediate losses [23] where intermediate supervised losses are added to a DNN architecture to help convergence or produce a better internal representation and an improved supervised performance. In our case, each layer is connected to an intermediate "supervised loss" through $-\beta_j I(Y; Z_j)$. Each layer Z_j, $j \leq N$ solves IB algorithm described in Section II-B1:

$$\min_{p(z_j|z_{j-1})} I(Z_{j-1}; Z_j) - \beta_j I(Y; Z_j) \tag{7}$$

Algorithm 1 Greedy successive N-Layer IB

Inputs $p(x,y), \beta j, |\mathcal{Z}_j|, N$
Outputs $p(x, z_1, ..., z_N, y)$
$Z_0 = X$
for j in $1, ..., N$
  $p(x, z_j, y)$=IB$(p(z_{j-1}, y), \beta_j, |\mathcal{Z}_j|)$

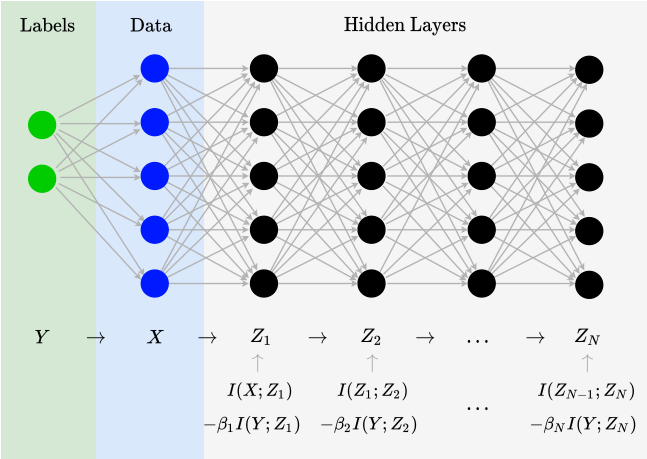

Fig. 2. Structure of a Neural Network, the N layers $Z_1 ... Z_N$ for a successive Markov chain

The algorithm uses the self consistent equations similar to Equations 2 at each layer. β_j controls the trade-off trade-off between preserving information about Y and compressing Z_{j-1} at layer Z_j, Section IV discusses the choices of β_j and their implications on the IB plane.

Algorithm 1 describes the greedy successive N-Layer IB optimization. It is a greedy solution such that each layer Z_j optimizes best for Z_{j-1} and Y independently of $Z_{j+1} ... Z_N$.

IV. Numerical Results

In this section, we present our numerical results. We focus on the case with two layers, denoted here as Z_1 and Z_2, due to space constraints, while noting that the results and the approach can easily be extended to more general, i.e., $N > 2$ layer case. We generate a joint distribution $p(x,y)$, with $|\mathcal{X}| = 64, |\mathcal{Y}| = 2$. We compare the results of the greedy successive 2-Layer IB to the vanilla 1-Layer IB.

Analogous to the IB plane, the 2-Layer IB plane represents $I(Y;Z_2)$ as a function of $I(X;Z_2)$. It corresponds to the IB plane when only considering the last compressed layer Z_2, which is usually viewed as a "feature vector" in DNNs. Figures 3, 4 and 5 show the concave hull of the points $I(Y;Z_2), I(X;Z_2)$ generated by the greedy successive IB algorithm for different β_1, β_2.

Note that at any point of the N-Layer IB plane, we have the following inequalities simply due to the data processing inequality (DPI):

$$\begin{cases} I(Y;Z_2) \leq I(Y;Z_1), \\ I(X;Z_2) \leq I(X;Z_1) \end{cases} \tag{8}$$

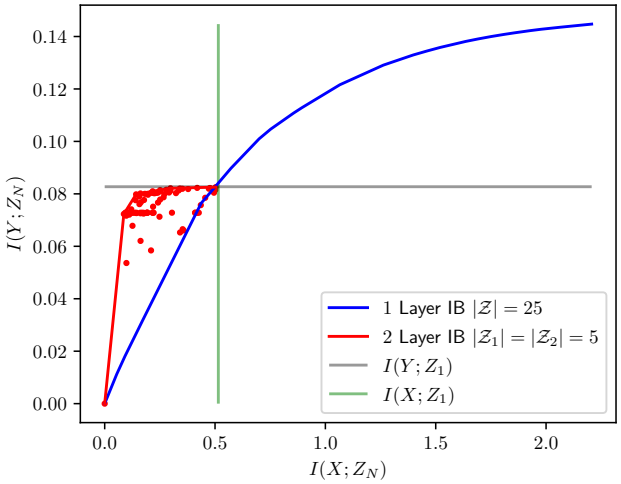

Fig. 3. 2-Layer IB plane for a fixed β_1

Assuming β_1 fixed and varying β_2 to draw the 2-Layer IB plane, Equation 8 implies that the curve will be vertically bounded by $I(Y;Z_1)$ and horizontally bounded by $I(X;Z_1)$ as shown in Figure 3.

A. Comparison of 2-Layer IB with 1-Layer IB

We next assume that $\beta_1 = \beta_2 = \beta$. We compare the performance of the greedy successive 2-Layer IB $Y \to X \to Z_1 \to Z_2$ with the vanilla 1-Layer IB $Y \to X \to Z$. Additionally, we impose a "complexity" constraint on the hidden layers, meaning that $|\mathcal{Z}_1| \cdot |\mathcal{Z}_2| = |\mathcal{Z}|$. The complexity constraint encapsulates a constraint on the number of connections in the DNN's hidden layers (omitting the input layer connections), which is a standard complexity constraint in the deep learning literature as each connection corresponds to a learnable parameter.

Figure 4 shows that the 2-Layer IB is able to achieve higher relevance for stronger compression regimes compared to the 1-Layer IB, which can be explained by the fact that having 2 layers enables more compression of X with less compromise on the relevance with Y.

The trend reverses at low compression regimes, where the relevance is lower than the 1-Layer IB, which is due to Equation 8, and to the fact that $|\mathcal{Z}_1| < |\mathcal{Z}|$, i.e. at $\beta \to \infty$ the 2-Layer IB curve is bounded by a sub-optimal curve to the 1-Layer IB curve.

For the sake of experiment, we also compare 2-Layer IB and 1-Layer IB without the complexity constraint, i.e. $|\mathcal{Z}_1| = |\mathcal{Z}_2| = |\mathcal{Z}|$ representing a 2-Layer IB with each layer having the same size of the one used in the 1-Layer IB. Not surprisingly, Figure 5 shows that the 2-layer IB has a higher curve in the IB plane.

V. Conclusion and Future Work

In this paper, we propose an extension of the original IB problem to a successive N-Layer IB problem. This formulation

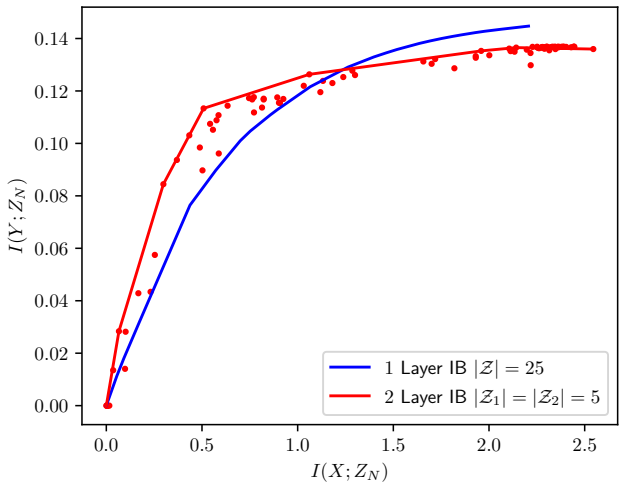

Fig. 4. 2-Layer IB plane for $\beta_1 = \beta_2 = \beta$ compared to 1-Layer IB plane with same complexity $|\mathcal{Z}_1| \cdot |\mathcal{Z}_2| = |\mathcal{Z}|$

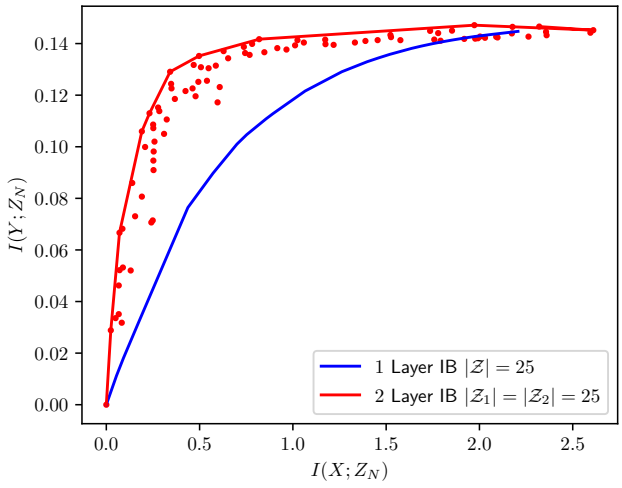

Fig. 5. 2-Layer IB plane for $\beta_1 = \beta_2 = \beta$ compared to 1-Layer IB plane with same layer sizes $|\mathcal{Z}_1| = |\mathcal{Z}_2| = |\mathcal{Z}|$

is motivated by the mechanics of a feed forward DNN, where layers are computed sequentially from the input to the output. Using a simple greedy optimization algorithm to approximate the successive N-Layer IB solution, we study the depth/width trade-off under a complexity constraint of a DNN through the IB theory lens. Numerical experiments for $N = 2$ show the presence of 2 regimes, 2-Layer IB outperforms 1-Layer IB in high compression regimes while the trend reverses for high compression regimes.

Our future research will focus on an exact solution of the successive N-Layer IB problem, as well as interpretations of the 2 observed regimes through an information-theoretic lens and applications in DNNs.

REFERENCES

[1] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[2] Y. Bengio, Y. LeCun *et al.*, "Scaling learning algorithms towards ai," *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[3] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2924–2932.

[4] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 473–480.

[5] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 6105–6114.

[8] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*. IEEE, 2015, pp. 1–5.

[9] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

[10] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124020, 2019.

[11] A. Elad, D. Haviv, Y. Blau, and T. Michaeli, "Direct validation of the information bottleneck principle for deep nets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2019.

[12] A. A. Alemi, I. Fischer, and J. V. Dillon, "Uncertainty in the variational information bottleneck," *arXiv preprint arXiv:1807.00906*, 2018.

[13] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

[14] B. Dai, C. Zhu, B. Guo, and D. Wipf, "Compressing neural networks using the variational information bottleneck," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1135–1144.

[15] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[16] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-vae," *arXiv preprint arXiv:1804.03599*, 2018.

[17] N. Slonim, "The information bottleneck: Theory and applications," Ph.D. dissertation, Citeseer, 2002.

[18] W. H. Equitz and T. M. Cover, "Successive refinement of information," *IEEE Transactions on Information Theory*, vol. 37, no. 2, pp. 269–275, 1991.

[19] E. Tuncel and K. Rose, "Computation and analysis of the n-layer scalable rate-distortion function," *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1218–1230, 2003.

[20] T. T. Nguyen and J. Choi, "Layer-wise learning of stochastic neural networks with information bottleneck," *arXiv preprint arXiv:1712.01272*, 2017.

[21] Q. Yang, P. Piantanida, and D. Gündüz, "The multi-layer information bottleneck problem," in *2017 IEEE Information Theory Workshop (ITW)*. IEEE, 2017, pp. 404–408.

[22] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.