MOLE: MOdular Learning FramEwork via Mutual Information Maximization

Tianchao Li [*1] **Yulong Pei** [*1]

Abstract

This paper is to introduce an asynchronous and local learning framework for neural networks, named Modular Learning Framework (MOLE). This framework modularizes neural networks by layers, defines the training objective via mutual information for each module, and sequentially trains each module by mutual information maximization. MOLE makes the training become local optimization with gradient-isolated across modules, and this scheme is more biologically plausible than BP. We run experiments on vector-, grid- and graph-type data. In particular, this framework is capable of solving both graph- and node-level tasks for graph-type data. Therefore, MOLE has been experimentally proven to be universally applicable to different types of data.

1. Introduction

Deep Neural Networks (DNNs) have attracted attention in many domains over the last decade as a result of their empirical success in a variety of applications, such as image classification (Sornam et al., 2017), natural language processing (Otter et al., 2020), speech recognition (Nassif et al., 2019) and anomaly detection (Pang et al., 2021). Backpropagation (BP, Rumelhart et al., 1986), as the dominant and mature algorithm of training deep neural networks, plays a central role in the success of DNNs. BP is an end-to-end or global optimization algorithm where all the weight parameters are automatically updated by backpropagating the error gradient from the loss function to each layer until the input layer. BP excessively depends on labeled data. Specifically, each input sample for DNNs requires a correct label to compute the loss via the loss function and DNNs cannot feed unlabeled samples. To ensure good generalization of DNNs, BP requires large labeled datasets. In contrast, humans learn from a few shots of labeled samples to recognize the new category: they summarize the special features of this category from the given samples and recognize the unobserved samples using summary knowledge. Therefore, BP makes DNNs, designed by imitating human neural networks in the bio-brain, biologically implausible. Furthermore, the bio-brain is very modular and learns mainly based on local information (Caporale & Dan, 2008).

BP brings other drawbacks to DNNs. During the training, BP needs to compute many gradients to update the corresponding parameters. As all parameters, activations, and gradients of the DNN need to fit into a processing unit's working memory, BP creates a considerable amount of memory when the DNN stacks multi-layers. In addition, BP uses the chain rule to compute the gradients of each layer, namely the parameter's gradient depends on the following parameters in the DNN. So BP leads that implicit correlations across layers would exist, although the parameters only determine the output of their layers. Therefore, the outcome-based internal interpretation and exploration are not reliable enough. Therefore, the gradient-isolated training algorithm or method will bridge this issue.

To relieve these drawbacks from BP, we propose a neural network training framework, named Modular Learning Framework (MOLE). Using this framework, we modularize the neural network by layers with fewer parameters per module. Then we define the training objective for different groups of modules, and each module is sequentially and independently trained by optimizing its objective. Furthermore, we experimentally demonstrate MOLE's universal applicability for various types of data, and we explain the reason for the gap between MOLE and BP. Finally, we compare MOLE with other alternatives to BP and point out potentially feasible improvements.

2. Preliminary

Mutual information (MI) is a measure of the amount of information that one random variable contains about another random variable (Cover et al., 1991). A dual interpretation of MI is the reduction in the uncertainty of one random variable due to the knowledge of the other (Alajaji et al., 2018). MI has the form for two variables X and Z,

$$I(X;Z) = \int_{\mathcal{X}\times\mathcal{Z}} \log \frac{\mathrm{d}\mathbb{P}_{XZ}}{\mathrm{d}\mathbb{P}_X \otimes \mathbb{P}_Z} \mathrm{d}\mathbb{P}_{XZ}, \qquad (1)$$

[*]Equal contribution [1]Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, the Netherland. Correspondence to: Tianchao Li <litianchao1996@outlook.com>, Yulong Pei <y.pei.1@tue.nl>.

In *ICML Workshop on Localized Learning (LLW)*, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

where $\mathbb{P}_{XZ}$ is the joint probability distribution, and $\mathbb{P}_X = \int_{\mathcal{Z}} \mathrm{d}\mathbb{P}_{XZ}$ and $\mathbb{P}_Z = \int_{\mathcal{X}} \mathrm{d}\mathbb{P}_{XZ}$ are the marginal probability distributions. Specifically, MI measures the similarity between two variables' spaces $\mathcal{X}$ and $\mathcal{Z}$ based on the marginal and joint distributions of two variables X and Z. In most settings, these distributions are unknown and would be estimated from the given samples. However, the sample data in deep learning is high-dimensional and even has certain structural properties. Accordingly, the distribution estimation of such data is ill-posed (Vapnik, 1995). Therefore, consistent and precise mutual information estimators on high-dimensional even structural data were proposed by detouring from the distribution estimation.

Some MI estimators were proposed without estimating the underlying data distribution. Giraldo et al. (2014) proposes a non-parametrical matrix-based MI estimator for vector-type data by defining functionals on normalized positive definite matrices. Yu et al. (2021) leverage the above matrix-based estimator and propose a differentiable and statistically more powerful normalized MI estimator. With the success of DNNs, the parametrical and scalable neural estimator is designed for MI estimation with good consistency. MINE (Belghazi et al., 2018) uses the dual representation of KL-divergence (Donsker-Varadhan representation (Donsker & Varadhan, 1983)) to construct the DNN and its expressive power guarantees that the estimation of MINE can be arbitrarily close to the true MI on high-dimensional data. However, MINE does not have a good capability to capture the structural MI. DIM (Hjelm et al., 2018) leverages the local MI using the average MI between the high-level vector and local patches of grid-like data and matches representations to a prior distribution by adversarial learning. DGI (Velickovic et al., 2019) extends the ideas of DIM to the graph domain. GMI (Peng et al., 2020) captures the topological connectivity better than DGI because GMI explicitly captures the correlation between input features and hidden vectors of both nodes and edges by defining the topology-aware MI.

One of the extensions of MI is Information Bottleneck (IB). In line with the information-theoretic concept, the goal of DNNs aligns with that of IB, namely finding the trade-off between compression and prediction (Tishby et al., 2000). Formally, given an input $X \in \mathcal{X}$, the internal representation $T \in \mathcal{T}$ and the label $Y \in \mathcal{Y}$, the optimal representation T should be derived from the trade-off the complexity (rate) of the representation, $I(T; X)$, and the amount of preserved relevant information, $I(T; Y)$ (Tishby & Zaslavsky, 2015). Therefore, IB for DNNs is represented as the maximization of the Lagrangian optimization:

$$\mathcal{L}_{IB} = I(T; Y) - \beta I(T; X), \tag{2}$$

where β is the positive Lagrange multiplier. Previous works (Shamir et al., 2010; Wang et al., 2020; Wu et al., 2020) have proved that IB can improve the robustness of DNNs.

MI derives from the communication system (Shannon, 1948). A communication system consists of five components: information source, transmitter, information channel, receiver, and destination. The information source generates messages or a sequence of messages (X_1). The transmitter (also known as the encoder) encodes the message from the information source into a signal (X_2) that is suitable for transmission over the information channel. The information channel transmits the signal X_2 and outputs the arrival signal X_3 to the following receiver. The receiver (also known as the decoder) decodes the information transmitted via the channel as the original information as much as possible, and the output of the decoder is denoted as X_4. Assuming all is right with the world, the receiver will have the ability to decode the signal and recover the original message. The data flow from X_1 to X_4 in the communication system forms a Markov chain: $X_1 \to X_2 \to X_3 \to X_4$. Cover et al. (1991) defines Data Processing Inequality(DPI) for a Markov chain, so the DPI of communication systems is

$$I(X_2; X_1) \geq I(X_3; X_1) \geq I(X_4; X_1). \tag{3}$$

We propose MOLE in Section 3 inspired by the mechanism of communication systems because the data flow in communication systems is significantly similar to DNNs'.

3. Modular Learning Framework

DNNs process data in a manner quite similar to communication systems. In DNNs, the input layer and the following hidden layers learn a low-dimensional and high-level representation, then the rest of the hidden layers and the output layer work as the predictor using the representation. In communication systems, the encoder compresses the data as the signal for transmission, then the decoder reconstructs the original data using the signal. However, the aim of the decoder is the input of communication systems, while the predictor aims at the label rather than the input of DNNs. Shwartz-Ziv & Tishby (2017) presents that the data flow in DNNs forms a Markov chain, namely,

$$I(X; Y) \geq I(T_1; Y) \geq \cdots \geq I(T_k; Y) \geq I(Y'; Y) \tag{4}$$

where T_k represents the representation of the k-th hidden layer, and $\hat{Y}$ is the prediction. Shwartz-Ziv & Tishby (2017) and Yu et al. (2020) experimentally validated Markov chains on DNNs. Therefore, the label can be regarded as the information source for DNNs, but it has been implicitly processed as the input data. DNNs continue to resolve the label from the input data like communication systems, and each layer performs like either the encoder or decoder. Since the representation of one layer consists of the outputs of all this layer's neurons, the layer is the training atom in MOLE and some layers can be treated as a module for training. Since

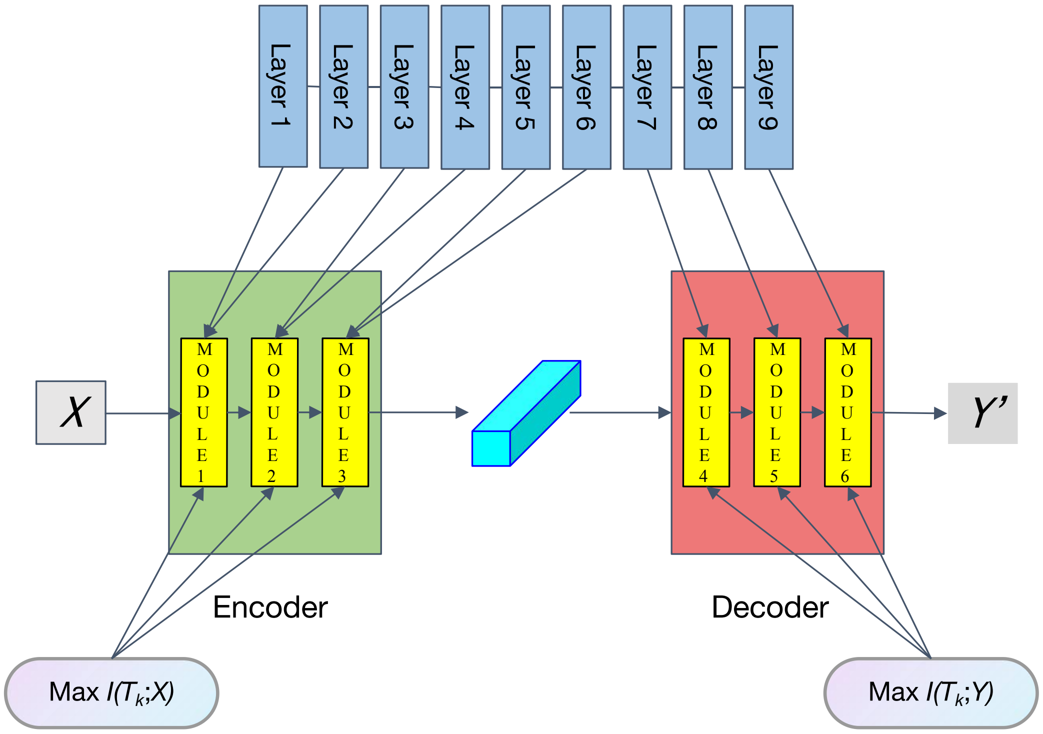


Figure 1. The training pipeline of MOLE is shown. X is the input, Y is the label, Y' is the final output, and T_k represents the output of k-th module. The top part is a DNN stacking 9 layers, the middle is MOLE, and the bottle is the training objective of each module where $I(*;*)$ represents the mutual information.

each layer of DNNs holds InfoMax (Linsker, 1988), each module is trained by maximizing the mutual information between its output and the desired variable. Since the output of one module is the input of its following module, each module is sequentially trained from the input module to the output module.

Intuitively, all modules in DNNs act as decoders that should perform optimal resolution for the labels, namely $I(T_k, Y)$ maximization. However, all the low-level features with equivalent importance are used to resolve, such that the DNN would be struggling to differentiate among the truly similar samples. Similarly to human recognition, we rarely correctly distinguish twins by their entire outlook but by their facial features. According to Tian et al. (2020), training by only $I(T_k; Y)$ maximization leads to substantial noise, causing the representation learning before the predictor vulnerable to adversarial attacks. Therefore, the first several layers should perform as the encoder to extract the important information and embed the low-level features as the high-level representation, and the following layers perform as the encoder to resolve the high-level representation as the label. In addition, the layer or module assigned for encoder-like modules should be trained by the training objective, $I(T_k; X)$ maximization, while the training objective for decoder-like modules is $I(T_k; Y)$ maximization. The pipeline of MOLE is shown in Figure 1. In principle, MI measures the similarity between two variables' space, so the trained model via MOLE would perform better on unobserved samples after $I(Y'; Y)$ is maximized on the output module. However, the optimization by gradient descent techniques approximates the maximum $I(Y'; Y)$ in the batch setting. Because this step is based on the approximation, there would exist groups of outputs with the almost same similarity to the labels if the output module is still trained by empirical $I(Y'; Y)$ maximization and only several groups have relatively good performance. From the geometrical interpretation, the model performs well enough if and only if the output's hyperplane most coverages to the label's hyperplane (Yu & Principe, 2019). To ensure that DNNs always have good performance, the conventional loss function replaces MI on the output module in the current state, and the conventional loss function is equivalent to $I(Y'; Y)$ maximization. As Figure 1 shows, it is clear that using MOLE makes DNNs locally learn. MOLE allows DNNs to learn from unlabeled data for general high-level representation in the encoder-like module, and the few-shot empirical representations with labels would cluster through the decoder-like module. Therefore, MOLE remains biologically plausible. Furthermore, in the case of a DNN with a fixed number of layers, the allocation of a greater number of layers to the encode-like module and a smaller number of layers to the decode-like module enhances the impact of $I(T_k;X)$ while diminishing the influence of $I(T_k;Y)$, and vice versa. MOLE implicitly forms IB by module assignment such that DNNs trained by MOLE would be more robust to some extent.

4. Experiment

This section experimentally shows that MOLE is universally applicable. Since MOLE is proposed by analogizing communication systems, and each module works as a data processor, MOLE generally works as long as it can process all types of data. Considering the most commonly-used data types in DNNs, i.e., vector, grid, and graph, we utilize these datasets: *Adult* (vectors), *MNIST* (grids), *Cora* (a graph) and *Mutagenicity* (graphs). We implement Multi-layer Perceptrons (MLP, Murtagh, 1991) on *Adult*, Convolutional Neural Networks (CNN, LeCun et al., 1998) on *MNIST*, Message-Passing Graph Neural Networks (MPGNN, Gilmer et al., 2017) on *Mutagenicity*, and Graph Convolutional Networks (GCN, Kipf & Welling, 2016) on *Cora*.

In the training, we directly use the high-dimension MI estimators to achieve InfoMax instead of the well-performed contrast learning method or other alternatives. Although such estimators are highly computational and even cannot capture implicit information well, it definitively conforms to the underlying mechanism of communication systems, and straightforwardly shows the quality of the measure mainly affects the final outcomes of MOLE. We implement two main types of estimators, parametric and non-parametric. The non-parametric is the matrix-based estimator (Yu et al., 2021), which can consistently measure between the vector-type data. We further use the multivariate extension (Yu et al., 2020) to extend the matrix-based estimator on the grid- and graph-type data. However, the grid extension estimator does not consider the spatial locality, while the graph extension can partially capture the topological connectivity by involving the adjacent vector. We use parametric estimators including MINE, DIM, and GMI. In detail, MINE performs on the vector-type representation well, as the lower bound is very tight. DIM can capture more information about spatial locality by involving local MI. DGI captures the topological connectivity by learning the topology-aware MI.

The assignment of the training objective and the choice of DNNs are shown in Appendix A.2. The results are tabulated in Table 1. For *Adult*, we run three training: BP, MOLE with Matrix-based estimator, and with MINE. Their accuracies on the unobserved data are close, 83.94%, 84.79%, and 83.69% respectively. MOLE with Matrix-based estimator has the highest test accuracy, which is even higher than its train accuracy. In *MNIST*, the performance of the extended Matrix-based estimator and MINE on MOLE are clearly lower than BP, and the test accuracy differences reach over 4%. The object in *MNIST* is to recognize a grayscale handwritten number on a black background, which means the images have relatively weak spatial locality. If the object and background become more complex with stronger spatial locality, the accuracy difference would enhance. Using DIM to train the encode-like modules narrows the difference. In *Mutagenicity*, the test accuracies on MOLE with Matrix-based estimator, MINE, and GMI+MINE are 66.66%, 69.55%, and 76.01%, while BP's is 77.63%. With the topological connectivity considered by GMI, the model outperforms other models with MOLE, approaching BP. *Cora* is used for the semi-supervised node-level task in GNN, and MOLE with GMI+MINE has close test accuracy to BP. Based on the experiment results, we can conclude the quality of the MI estimator decisively affects the performance of neural networks. Compared with the train and test accuracies on MOLE, we observe that both are very close for most types of neural networks even though the test accuracy is higher. Therefore, we believe that MOLE would improve the generalization of neural networks. Although GCN on *Cora* with MOLE has a large difference between the train and test accuracies, BP even makes this model over-fitting. Therefore, this large difference is reasonable.

After training, we visualize the distribution of samples from raw data to each layer's output in the 2-dimensional plane by t-SNE (Van der Maaten & Hinton, 2008) embedding as shown in Appendix A.3. Ideally, we hope the representation would start to cluster by classes in the encoder-like layer as Figure 5 shows. In most conditions, the output of the encoder-like layer would have a similar or symmetrically similar distribution to the raw data, and the output of the decoder-like layer would cluster by class. For instance, *MNIST* significantly fulfills this condition as shown in Figure 3. So the effect of $I(T_k;Y)$ maximization is to cluster the representations. Therefore, the maximal distance across classes would be an alternative measure to train the decoder-like layer with less computation. After the last layer trained by the conventional loss function, two phenomenons would happen: the samples cluster by class further with less overlapping in the distribution as shown in Figure 4 or distance across cluster enhanced as shown in Figure 3, and the samples and classes become nonlinearly separable in Figure 2. Specifically, the last layer trained by the conventional loss function would boost the cluster in decoder-like layers.

5. Discussion

Recently, Hinton (2022) clarifies the drawbacks of BP and proposes the alternative Forward-Forward (FF) algorithms inspired by contrastive Learning (Gutmann & Hyvärinen, 2010) and Boltzmann machines (Hinton et al., 1986). In FF, the label encoding is added to the input data. The input with correct encoding forms positive samples, while

Table 1. Classification accuracy results on *Adult*, *MNIST*, *CIFAR-10*, *Mutagenicity*, *Cora* with different estimators

Dataset	Estimator	Train Acc.	Diff. of Train Acc. to BP	Test Acc.	Diff of Test Acc. to BP
Adult	BP	91.08%	/	83.94%	/
	Matrix-based	84.74%	-6.34%	84.79%	0.85%
	MINE	83.58%	-7.5%	83.69%	-0.25%
MNIST	BP	99.94%	/	99.28%	/
	Matrix-based	96.65%	-3.29%	94.71%	-4.57%
	MINE	95.05%	-4.89%	94.49%	-4.79%
	DIM+MINE	98.60%	-1.34%	96.85%	-2.43%
Mutagenicity	BP	76.82%	/	77.63%	/
	Matrix-based	67.90%	-8.92%	66.55%	-11.08%
	MINE	70.63%	-6.19%	69.55%	-8.08%
	GMI+MINE	75.56%	-1.26%	76.01%	-1.62%
Cora	BP	99.29%	/	70.60%	/
	GMI+MINE	82.86%	-16.43%	68.40%	-2.20%

the input with incorrect encoding forms negative samples. FF uses the measure of goodness as the training objective. When the positive sample is forwarded, the parameters are updated by maximizing the goodness, and vice versa. The operation of two forward passes equivalently is to resolve the label from the input, namely hidden modules are trained by maximizing $I(T_k; Y)$. Paliotta et al. (2023) extends FF on the graph domain, named GFF, but GFF only adapts the graph-level task. Duan et al. (2021) trains the hidden module by maximizing the inter-class dissimilarity of hidden outputs, and the class of the hidden output is decided by the label of the input. In other words, after training, the hidden output should cluster by the label, which has the same effect of $I(T_k; Y)$ maximization. Similarly, Associated Learning (Wu et al., 2022) adds an autoencoder for the label, and the latent variable has the same dimension as the hidden output. The training mainly depends on the minimization of the difference between the hidden output and the latent variable. However, these approaches would still face the problem of over-reliance on labels, and adding the equivalently $I(T_k, X)$-maximization operation would bring improvement.

Löwe et al. (2019) propose the Greedy InfoMax (GIM) approach for unsupervised representation learning, equivalently encoder-like modules trained by $I(T_k, X)$ maximization. Ma et al. (2020) replaces MI as the optimized measure by HSIC Bottleneck trading-off between the information the hidden representation needs for predicting the output and the information the hidden representation retains about the input. Module assignment in MOLE between encoder- and decoder-like training is such an explicit trade-off. Overall, these backpropagation-free approaches fulfill our proposed MOLE. In addition, MOLE still remains largely unexplored, and it would experience long-time research and exploration to outperform BP. For example, piecewise-linear functions, such as ReLU and LeakyReLU are used to prevent the gradient vanishing and exploding, but these would achieve limited non-linear transformation. The output from the linear transformation is equivalent to the signal in communication systems. This output should be non-linearly transformed as a whole. So using the kernel method (Zhang et al., 2017; Duan et al., 2021) would be a more suitable non-linear transformation for MOLE. In the semi-supervised setting, the imbalanced measure between the output and the label on the decoder-like module would be helpful as well. Furthermore, MOLE would achieve parallel training in the encoder-like modules by designing the parallel architecture in DNNs. Each parallel architecture is trained by maximizing $I(T_k; X)$ and their outputs are concatenated as the input of the following decoder-like module.

6. Conclusion

We introduce Modular Learning Framework, which trains DNNs with gradient-isolated across modules. Although currently, MOLE can not ultimately outperform BP, it deserves further research because of its apparent advantages (biological plausibility, universal applicability, etc). These advantages would help the interpretability of DNNs, for example, it would make the outcome-based interpretation more reliable and reasonable.

During the investigation of MOLE, we also find some questions worth further exploration:

- Can we find the optimal module assignment between the encoder- and decoder-like components?
- In the out-of-experiment setting, the object and background in the grid-type data are much more complex, so the data mainly depends on the spatial locality. Can we design an estimator to better capture the implicit spatial locality?

- The MI estimator for the high-dimensional data is usually time-consumed and not accurate enough, particularly in the encoder-like module. Can we design an alternative technique to achieve InfoMax in the decode-like module? For example, certain self-supervised representation learning methods (SSRL, Ericsson et al., 2022) would work. If it works, this approach could match the accuracy of backpropagation (BP) and significantly expedite the learning process. Furthermore, for more complex data such as audio and video, can we also achieve InfoMax by SSRL?
- After the application of MOLE, each module becomes independent. Is it possible to prove how the parameters of each module span the hyperplane of its output?
- If the span of the hyperplane succeeds, can we design an optimization algorithm that employs geometric projection techniques to obtain an exact optimal solution?

These questions present exciting avenues for further research and could potentially enhance our understanding and utilization of MOLE.

References

Alajaji, F., Chen, P.-N., et al. *An Introduction to Single-User Information Theory*. Springer, 2018.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. Mutual information neural estimation. In *International conference on machine learning*, pp. 531–540. PMLR, 2018.

Caporale, N. and Dan, Y. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

Cover, T. M., Thomas, J. A., et al. Entropy, relative entropy and mutual information. *Elements of information theory*, 2(1):12–13, 1991.

Donsker, M. D. and Varadhan, S. S. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on pure and applied mathematics*, 36 (2):183–212, 1983.

Duan, S., Yu, S., and Príncipe, J. C. Modularizing deep learning via pairwise learning with kernels. *IEEE Transactions on Neural Networks and Learning Systems*, 33 (4):1441–1451, 2021.

Ericsson, L., Gouk, H., Loy, C. C., and Hospedales, T. M. Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62, 2022.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Giraldo, L. G. S., Rao, M., and Principe, J. C. Measures of entropy from data using infinitely divisible kernels. *IEEE Transactions on Information Theory*, 61(1):535–548, 2014.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

Hinton, G. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Hinton, G. E., Sejnowski, T. J., et al. Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1 (282-317):2, 1986.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

Linsker, R. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

Löwe, S., O'Connor, P., and Veeling, B. Putting an end to end-to-end: Gradient-isolated learning of representations. *Advances in neural information processing systems*, 32, 2019.

Ma, W.-D. K., Lewis, J., and Kleijn, W. B. The hsic bottleneck: Deep learning without back-propagation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5085–5092, 2020.

Murtagh, F. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197, 1991.

Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., and Shaalan, K. Speech recognition using deep neural networks: A systematic review. *IEEE access*, 7:19143–19165, 2019.

Otter, D. W., Medina, J. R., and Kalita, J. K. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.

Paliotta, D., Alain, M., Máté, B., and Fleuret, F. Graph neural networks go forward-forward. *arXiv preprint arXiv:2302.05282*, 2023.

Pang, G., Shen, C., Cao, L., and Hengel, A. V. D. Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, 54(2):1–38, 2021.

Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., and Huang, J. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pp. 259–270, 2020.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Shamir, O., Sabato, S., and Tishby, N. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711, 2010.

Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

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

Sornam, M., Muthusubash, K., and Vanitha, V. A survey on image classification and activity recognition using deep convolutional neural network architecture. In *2017 ninth international conference on advanced computing (ICoAC)*, pp. 121–126. IEEE, 2017.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839, 2020.

Tishby, N. and Zaslavsky, N. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pp. 1–5. IEEE, 2015.

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

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Vapnik, V. N. The nature of statistical learning theory. 1995.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

Wang, B., Wang, S., Cheng, Y., Gan, Z., Jia, R., Li, B., and Liu, J. Infobert: Improving robustness of language models from an information theoretic perspective. *arXiv preprint arXiv:2010.02329*, 2020.

Wu, D. Y., Lin, D., Chen, V., and Chen, H.-H. Associated learning: an alternative to end-to-end backpropagation that works on cnn, rnn, and transformer. In *International Conference on Learning Representations*, 2022.

Wu, T., Ren, H., Li, P., and Leskovec, J. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 33:20437–20448, 2020.

Yu, S. and Principe, J. C. Understanding autoencoders with information theoretic concepts. *Neural Networks*, 117: 104–123, 2019.

Yu, S., Wickstrøm, K., Jenssen, R., and Principe, J. C. Understanding convolutional neural networks with information theory: An initial exploration. *IEEE transactions on neural networks and learning systems*, 32(1):435–442, 2020.

Yu, S., Alesiani, F., Yu, X., Jenssen, R., and Principe, J. Measuring dependence with matrix-based entropy functional. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10781–10789, 2021.

Zhang, S., Li, J., Xie, P., Zhang, Y., Shao, M., Zhou, H., and Yan, M. Stacked kernel network. *arXiv preprint arXiv:1711.09219*, 2017.

A. Appendix

A.1. Datasets

In the experiment, we use the public and accessible dataset, and *MNIST*, and *Cora* are well-known and common-used datasets for experiments. Here, we show the details of *Adult* and *Mutagenicity* datasets.

Adult[1] has 48842 samples, of which 3620 samples with *null* values exist. *Adult* is used for salary classification, so salary is used as the label. salary is a binary variable: `>50K` and `<=50K`. Table 2 shows 14 features and their data types in `Adult`. After preprocessing (filter out the samples with *null*, encode the category data by one-hot encoding), there are 104 numerical features.

Mutagenicity[2] is a graph dataset, and each sample consists of a graph and a binary label. The graph describes the structure of molecules, and the binary label indicates whether a molecule can induce genetic mutations. *Mutagenicity* is used for graph classification based on the features of nodes and structures of the graphs, and it contains 4,337 graphs that all consist of 14 basic elements.

Table 2. The attributes of *Adult*

Attribute	**Data Type**
`age`	integer
`workclass`	category
`fnlwgt`	integer
`education`	category
`education-num`	category
`marital-status`	category
`occupation`	category
`relationship`	category
`race`	category
`sex`	category
`capital-gain`	integer
`capital-loss`	integer
`hours-per-week`	integer
`native-country`	category

A.2. Experiment Setup

We experimentally show the viability compared with BP, so we use naive neural networks, stacking 3 to 4 layers.

Multilayer Perceptrons (MLP) on *Adult* Dataset A three-layer MLP backbone is structured, and it is implemented on *Adult* dataset. The first layer has 64 perceptions for 104-dimensional input, and the 64-dimensional output is fed to the second layer with 16 perceptrons. All the perceptrons are followed by ReLU activations for non-linearity. The last layer consists of two perceptrons and uses SoftMax activation to predict the probability of each class. The first layer is trained by $I(T_k; X)$ maximization, the second layer is trained by $I(T_k, Y)$, and the last layer is trained by minimizing the cross entropy.

Convolutional Neural Networks (CNN) on *MNIST* Dataset A four-layer Convolutional Neural Network is initialized: two convolutional layers and two fully-connected layers. It is clear that the convolutional layer is to extract the high-level representation from images, and both are trained by $I(T_k; X)$ maximization. The first fully-connected layer is trained by $I(T_k; Y)$ maximization, and the second is trained by minimizing the cross entropy. When we use DIM+MINE as the MI estimator, we use DIM to estimate the MI between grids and grids on the first convolutional layer, use MINE to estimate the MI between grids and vectors on the second convolutional layer, and use MINE to estimate the MI between vectors and vectors on the first fully-connected layer.

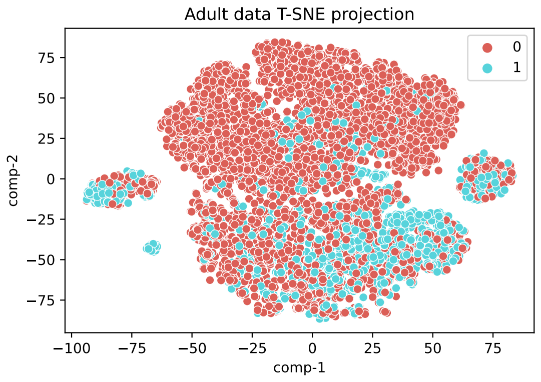
[1]https://archive.ics.uci.edu/ml/datasets/Adult
[2]https://chrsmrrs.github.io/datasets/docs/datasets/

Message Passing Graph Neural Networks (MPGNN) on *Mutagenicity* Dataset A four-layer Message Passing Graph Neural Network is used. The first layer performs the node embedding: a fully-connected layer on the feature matrix, and it is trained by $I(T_k; X)$ maximization. The following layer is a message-passing (MP) layer trained by $I(T_k; X)$ maximization as well. And the next message-passing (MP) layer is trained by $I(T_k; Y)$. And the final layer is a fully-connected with the cross entropy minimized. When we only use MINE as the MI estimator, MINE is to estimate the MI between vectors and vectors. In other words, we only consider the node features. When we use GMI+MINE as the MI estimator, we use MINE to estimate the MI between vectors and vectors on the embedding layer, use GMI to estimate the MI between graphs and graphs on the first message-passing layer, use MINE to estimate the MI between vectors and vectors on the second message-passing layer.

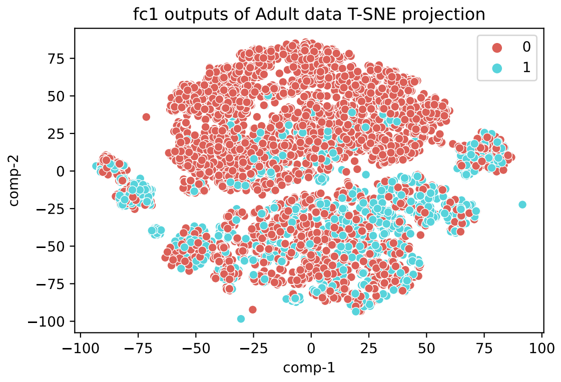
Graph Convolutional Networks (GCN) on *Cora* Dataset A three-layer Graph Convolutional Network has been implemented: two graph convolutional layers and one fully-connected layer. The first layer is trained by $I(T_k; X)$, the second is trained by $I(T_k; Y)$ where the measure only considers MI between the representation with the known label and their label, and the last is trained by minimizing the cross entropy. When we use GMI+MINE as the MI estimator, we use GMI to estimate the MI between two graphs on the first graph convolutional layer and use MINE to estimate the MI between vectors and vectors on the second graph convolutional layer.

A.3. Visualization

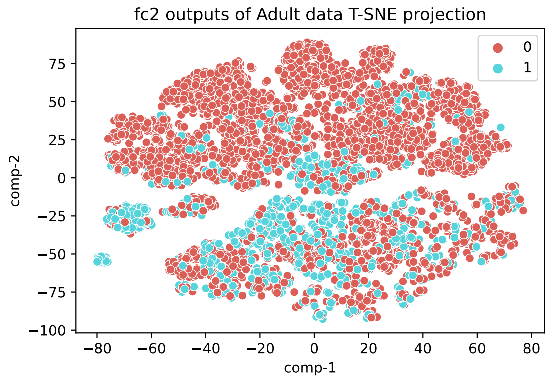
Here, we use t-SNE to embed the high-dimensional data as the 2-dimensional and visualize the embeddings in the 2-dimensional plane. Figure 2 shows the distribution of the raw samples from *Adult* and the corresponding outputs from each layer, and Figure 3 and Figure 5 respectively show *MNIST*'s and *Cora*'s. Figure 4 only the visualization of the last two layers trained by maximizing $I(T_k; Y)$ and minimizing the cross entropy respectively, because the raw samples and the samples' outputs from the first two layers have different dimensions of features.



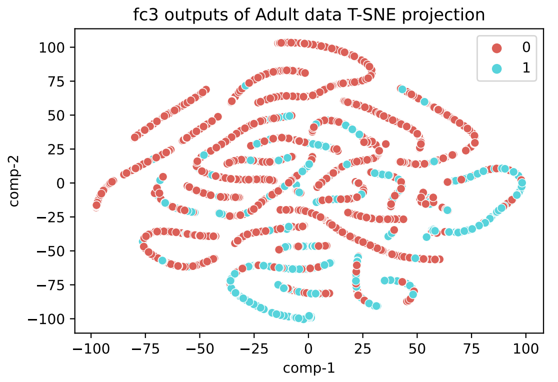
(a) Visualization of raw features' t-SNE embeddings



(b) Visualization of the first-layer outputs' t-SNE embedding

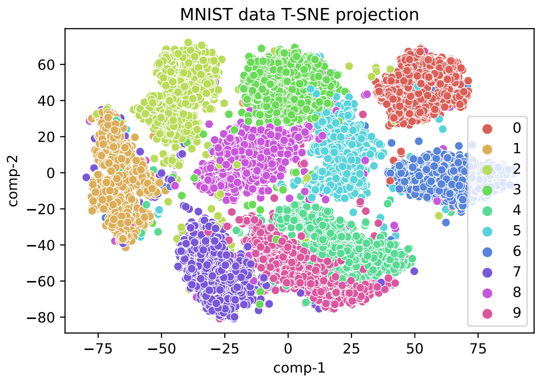


(c) Visualization of the second-layer outputs' t-SNE embedding

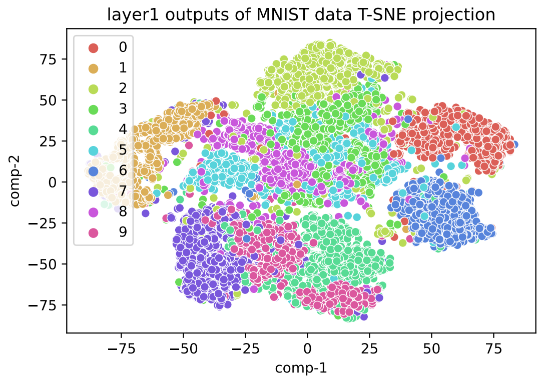


(d) Visualization of the last-layer outputs' t-SNE embedding

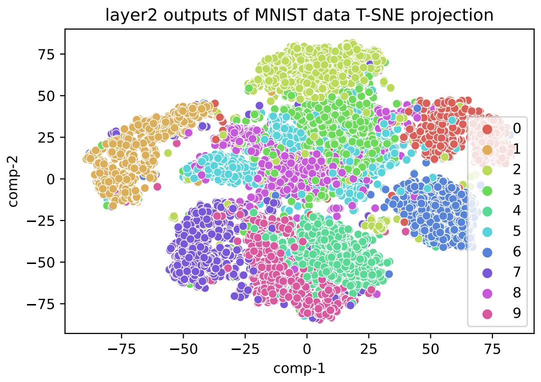
Figure 2. Visualization of t-SNE embeddings from raw features of *Adult* dataset to each output of MLP



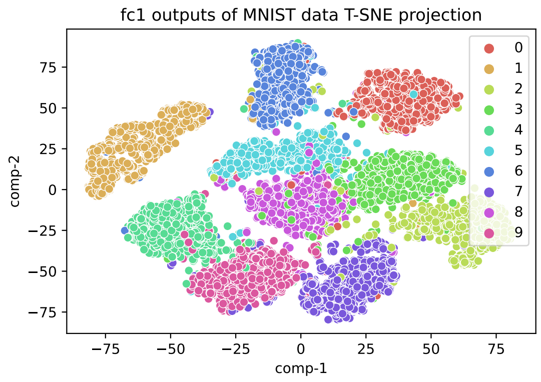
(a) Visualization of raw features' t-SNE embeddings



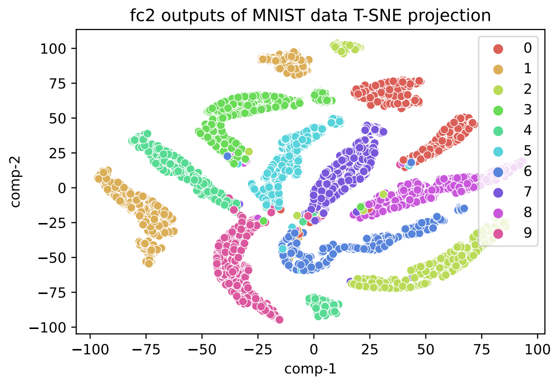
(b) Visualization of the first-layer outputs' t-SNE embedding



(c) Visualization of the second-layer outputs' t-SNE embedding

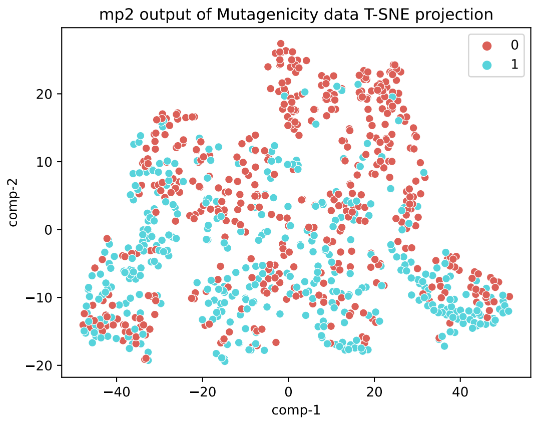


(d) Visualization of the third-layer outputs' t-SNE embedding

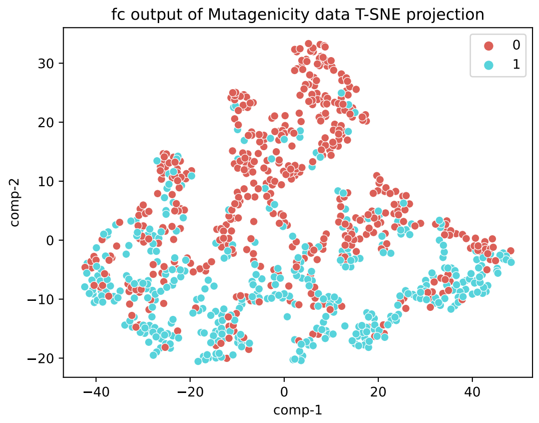


(e) Visualization of the last-layer outputs' t-SNE embedding

Figure 3. Visualization of t-SNE embeddings from raw features of *MNIST* dataset to each output of CNN

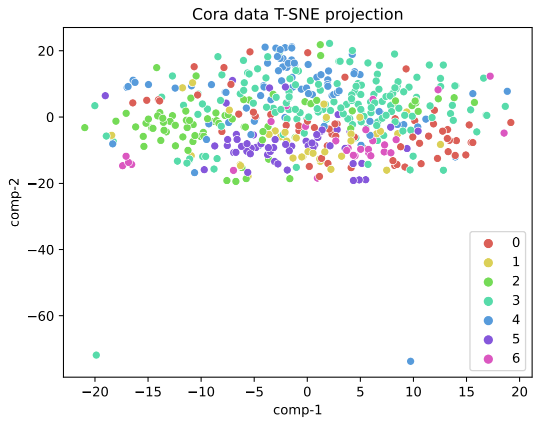


(a) Visualization of the second MP-layer outputs' t-SNE embedding

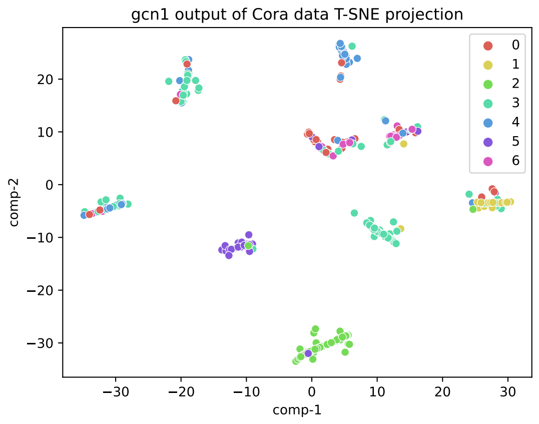


(b) Visualization of the last-layer outputs' t-SNE embedding

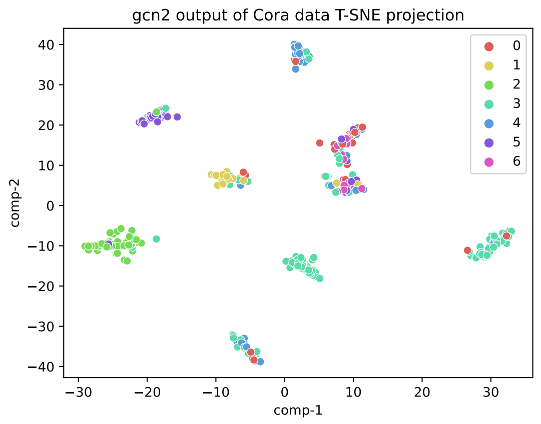
Figure 4. Visualization of t-SNE embeddings from raw features of *Mutagenicity* dataset to each output of MPGNN



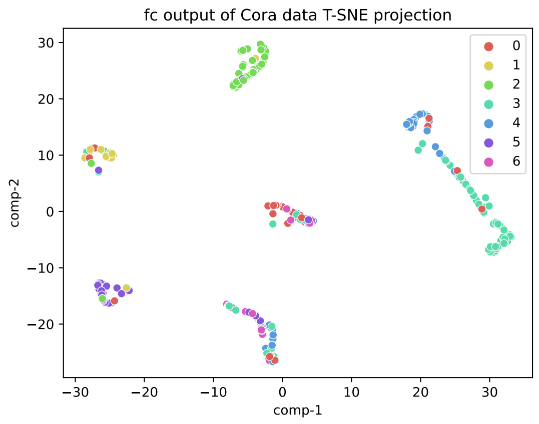
(a) Visualization of raw features' t-SNE embeddings



(b) Visualization of the first-layer outputs' t-SNE embedding



(c) Visualization of the second-layer outputs' t-SNE embedding



(d) Visualization of the third-layer outputs' t-SNE embedding

Figure 5. Visualization of t-SNE embeddings from raw features of *Cora* dataset to each output of GCN