Beyond Directed Acyclic Computation Graph with Cyclic Neural Network

Liangwei Yang[1,2*], **Hengrui Zhang**[2], **Weizhi Zhang**[2], **Zihe Song**[2],
Jing Ma[4], **Jiawei Zhang**[3], **Philip S. Yu**[2]
[1] Salesforce AI Research
[2] Department of Computer Science, University of Illinois Chicago
[3] Department of Computer Science, University of California, Davis
[4] University of Electronic Science and Technology of China
liangwei.yang@salesforce.com, {lyang84, hzhan55, wzhan42, zsong29, psyu}@uic.edu
jingma@uestc.edu.cn, jiawei@ifmlab.org

Abstract

This paper investigates a fundamental yet overlooked design principle of artificial neural networks (ANN): We do not need to build ANNs layer-by-layer sequentially to guarantee the Directed Acyclic Graph (DAG) property. Inspired by biological intelligence, where neurons form a complex, graph-structured network, we introduce the transformative Cyclic Neural Networks (Cyclic NN). It emulates biological neural systems' flexible and dynamic graph nature, allowing neuron connections in any graph-like structure, including cycles. This offers greater flexibility compared to the DAG structure of current ANNs. We further develop the Graph Over Multi-layer Perceptron, the first detailed model based on this new design paradigm. We experimentally validate the advantages of Cyclic NN on widely tested datasets in most generalized cases, demonstrating its superiority over current layer-by-layer DAG neural networks. With the support of Cyclic NN, the Forward-Forward training algorithm also firstly outperforms the current Back-Propagation algorithm. This research illustrates a transformative ANN design paradigm, a significant departure from current ANN designs, potentially leading to more biologically similar ANNs.

1 Introduction

Artificial intelligence (AI) has reshaped our daily lives and is expected to have a much greater impact in the foreseeable future. Lying behind the most profound AI applications [18, 14, 15, 8], artificial neural networks (ANN) such as multi-layer perception (MLP) [16], convolution neural network (CNN) [11] and Transformer [20] are designed specifically for different domains to fit the training data. Regardless of the network structure, neural networks are stacked layer-by-layer to form deep ANNs for greater learning capacity. It has been a *de facto* practice until now that data is first fed into the input layer and then propagated through all the stacked layers to obtain the final representations at the output layer. This paper seeks to answer a fundamental question in ANNs: "Do we really need to stack neural networks layer-by-layer sequentially?".

To answer this question, let's first examine the evidence from biological intelligence (BI). Neuroscientists have studied the biological neurons for decades. The connectome of C. elegans is the most thoroughly studied biological neural system, and biologists depicted the most detailed connection between 302 biological neurons [22, 4] as shown in Figure 1(a). Rather than being stacked layer-by-layer, all the neurons form a complicated connection graph, where each can connect to several

*Work is finished during study at University of Illinois Chicago

1^{st} Workshop on NeuroAI @ 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

other neurons within the system. We cannot even determine which neuron serves as the input/output within the neural system to process information. The same findings have also been observed in the latter more complicated neural systems, such as the biology neural connectome of drosophila larva [23], zebrafish [3], mouse [19] and the human brain [17]. Observed biological intelligence exhibits graph-structured, flexible, and dynamic neural systems, which are apparently different from the current layer-by-layer ANNs we build nowadays, as depicted in Figure 1(b).

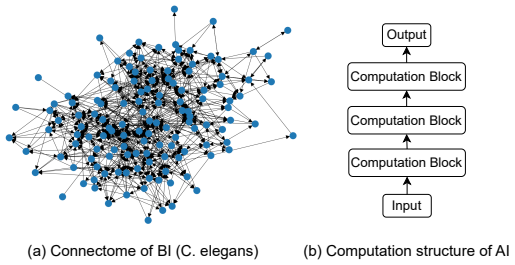

Figure 1: Neuron connection between Biology Neural Network and Artificial Neural Network

The learning rules actually cause the difference in the neural system structure between BI and AI. The Hebb's Rule [6], depicted as "Neurons that fire together wire together", is recognized as the fundamental learning way of biological neurons. The Spike-Timing-Dependent Plasticity (STDP) learning is then proposed to further consider the relative spiking time of pre-synapse and post-synapse neurons. Both learning rules of BI are localized, *i.e.*, the learning occurs on each neuron within its local influence scope. The localized learning rules grant the flexibility of each neuron on its connections to other neurons, which leads to the complicated graph-structured BI system. Conversely, for AI systems, the backward propagation (BP) algorithm [16] has dominated the training of ANNs. Data is fed into the ANNs from the input layer, forward propagates layer by layer to the last layer, calculates a global loss for the whole ANN based on the ground-truth labels, and then reversely backward propagates the error signals layer by layer to the input layer. In this procedure, ANNs are trained by a global loss function, and the ANNs must guarantee the error from global loss can be back-propagated layer by layer. This requirement prevents current ANNs from forming cycles to ease gradient back-propagation. Current ANNs are nearly all DAG structured. To mitigate the biological implausible nature of the BP algorithm, the forward-forward (FF) algorithm [7] is recently proposed to train ANNs. FF algorithm constructs good/bad samples and computes a loss function on each layer to differentiate between these samples. Similar to Hebb's Rule and STDP learning, the FF algorithm is a localized learning method. These advancements have allowed the training of ANNs to no longer rely solely on layer-by-layer back-propagation to design non-DAG-structured Cyclic Neural Networks.

Cyclic NN distinguishes itself from the current layer-by-layer ANNs in several aspects. 1) More flexible neuron connections. Cyclic NN greatly increases the design space of ANNs beyond the DAG structure. In Cyclic NN, the information flow is not as unidirectional as in DAG. Former neurons can also adjust based on the information encoded by the latter neurons, which largely enhances information communication within the network. The flexible connection design also makes Cyclic NN more like the biological neural system. 2) Localized training. Instead of current dominating global loss-guided BP-based training, Cyclic NN is based on localized training, *i.e.*, each neuron is optimized with its own local loss function. There is no gradient propagating between neurons. Localized training has its unique advantages. It frees the need to build DAG dependency between neurons, which is the bedrock of supporting cycles within the network. Also, each neuron is optimized independently without waiting gradients from the latter layers. 3) Computational neuron. Different from current ANNs that a neuron is considered as a d dimension to 1 dimension vector mapping; the neuron within Cyclic NN is considered the computational neuron with greater computation capacity because it is the optimization unit to fit the local task, which requires more parameters. This paper uses a linear layer to parameterize each computational neuron to fit the local classification task. It is also evident by the study of biological neuron [2], which empirically proves the learning capacity of a biological neuron is much larger than a d dimension to 1 dimension vector mapping function as the neuron defined within current ANNs. We take this observation and propose the computational neuron in Cyclic NN with more capable computation to fit the local optimization task. In summary, our contributions can be summarized as follows:

- Conceptually, we compare BI and AI to investigate a fundamental yet overlooked design principle: We do not need to satisfy the DAG constraint when designing ANNs.
- Methodologically, we propose the transformative Cyclic NN, a novel ANN design paradigm that supports a much more flexible connection between neurons beyond directed acyclic graph.

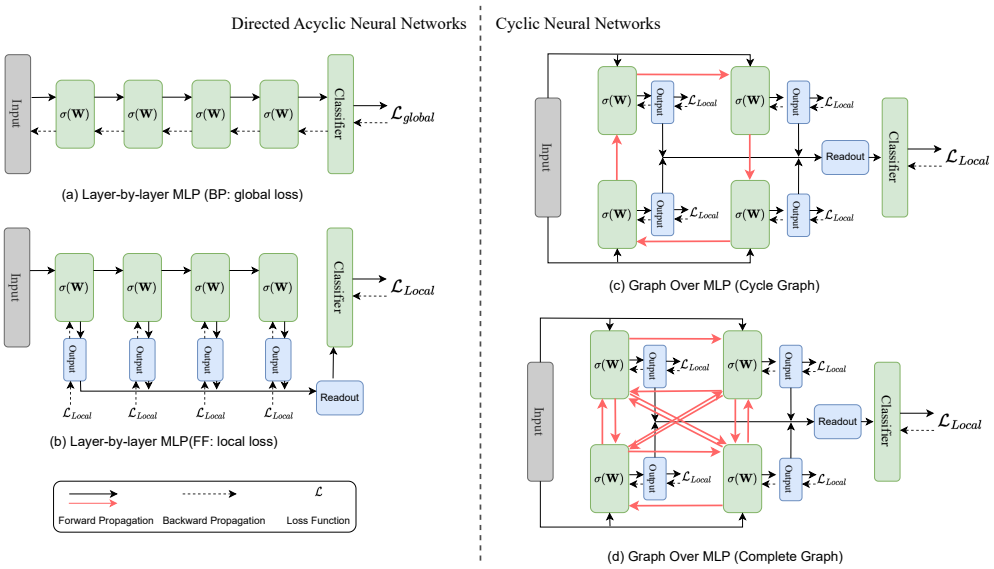

Figure 2: Comparison between different types of MLP structure.

- We test the novel design paradigm on the most generalized case and propose Graph Over Multi-layer Perceptron, the first detailed model based on Cyclic NN.
- Experimentally, we demonstrate the advantage of the proposed Cyclic NN on widely tested datasets. At the same time, we are the first to beat the current dominating BP training using the FF training algorithm by the supported flexible network design proposed in this paper.

2 Cyclic NN: Graph Over Multi-layer Perceptron

We propose the first Cyclic NN under the most generalized case, Graph Over Multi-Layer Perceptron (GOMLP), to show the design principle of Cyclic NN. As shown in Figure 2(c) and (d), GOMLP is designed by building a graph structure over the multi-layer perception to solve the classification task.

2.1 Input Construction

For the classification task, each sample is symbolized as the feature-label pair $(\mathbf{h}_i, y_i)$, where $\mathbf{h}_i$ is the representation of sample i and y_i is the corresponding label. To enable the local optimization illustrated in Section 2.3, a fusion function is used to construct the input as:

$$\begin{aligned} \mathbf{h}_{\text{pos}} &= f_{\text{fusion}}(\mathbf{h}, \mathbf{y}_{\text{true}}) = \mathbf{h}||\mathbf{y}_{\text{true}}, \\ \mathbf{h}_{\text{neg}} &= f_{\text{fusion}}(\mathbf{h}, \mathbf{y}_{\text{false}}) = \mathbf{h}||\mathbf{y}_{\text{false}}, \\ \mathbf{h}_{\text{neu}} &= f_{\text{fusion}}(\mathbf{h}, \mathbf{y}_{\text{neutral}}) = \mathbf{h}||\mathbf{y}_{\text{neu}}, \end{aligned} \tag{1}$$

$\mathbf{h}_{\text{pos}}$, $\mathbf{h}_{\text{neg}}$, and $\mathbf{h}_{\text{neu}}$ are the constructed input for local optimization of different parts. f_{fusion} is a function to fuse information between feature and label, which is defined as a concat function $(||)$. $\mathbf{y}_{\text{true}}$ is the one-hot vector of ground-true label, $\mathbf{y}_{\text{false}}$ is the one-hot vector of a randomly sampled false label. For $\mathbf{y}_{\text{neu}}$, we place an $\frac{1}{\text{Class Number}}$ on all the dimensions of one-hot vector to indicate $\mathbf{h}_{\text{neutral}}$ is neutral to all classes. f_{fusion} can be designed as any proper function to fuse information of the input feature and the label. In our study, we design it as a simple concat function same as [7].

2.2 Computation Graph

The computation graph $\mathcal{G}$ contains the computational neurons $\mathcal{V}$ and the synapses $\mathcal{E}$. Each computational neuron $N \in \mathcal{V}$ is a local module for calculation and optimization, while synapse S defines how the information propagates between computational neurons. $\mathcal{G}$ can be defined as a graph generator:

$$\mathcal{G} = \text{Generator}(|\mathcal{V}|, |\mathcal{E}|). \tag{2}$$

The above-generated graph $\mathcal{G}$ denotes a general graph structure. Meanwhile, to justify the effectiveness of the proposed Cyclic NN, we test multiple graph generators in this paper, including the Chain graph (Figure 2(b)), Cycle graph (Figure 2(c)), Complete graph (Figure 2(d)), Watts-Strogatz (WS) graph [21] and Barabási–Albert (BA) graph [1].

Neuron Update. In GOMLP, each neuron is parameterized by a linear layer. At each propagation, neuron N is updated by $\mathbf{h}_{\text{out}} = \sigma(\mathbf{W}\tilde{\mathbf{h}}_{\text{in}})$ (we omit the notation of N in equation for simplicity) where σ is the Relu activation function [13], $\mathbf{W} \in \mathbb{R}_{d_{\text{out}}^N \times d_{\text{in}}^N}$ is N's parameter. d_{out}^N is N's output dimension, which is a pre-defined dimension size, and d_{in}^N is N's input dimension, which is defined by the output of N's pre-synapse neurons. $\tilde{\mathbf{h}}_{\text{in}}$ is the normalized input as $\tilde{\mathbf{h}}_{\text{in}} = \frac{\mathbf{h}_{\text{in}}}{\|\mathbf{h}_{\text{in}}\|_2}$, where $\mathbf{h}_{\text{in}}$ is computational neuron N's input.

Synapse Propagation. Each synapse $S = (N_i \rightarrow N_j)$ is a directional edge from computational neuron N_i to N_j, which indicates N_i is the pre-synapse neuron of N_j and $\mathbf{h}_{out}^{N_i}$ (the output of N_i) will be propagated to N_j. Assume for neuron N, we obtain a set of pre-synapse neurons $(N_1, N_2, ..., N_n)$ based on the topology of $\mathcal{G}$. Then, in each propagation, N receives the output of all its pre-synapse neurons along the synapses and fuse the information to form its input by a concatenation function as $\mathbf{h}_{\text{in}} = \mathbf{h}||\mathbf{h}_{\text{out}}^{N_1}||\mathbf{h}_{\text{out}}^{N_2}||, ..., ||\mathbf{h}_{\text{out}}^{N_n}$, where $||$ is the concat function, $\mathbf{h}$ is the input representation constructed in Section 2.1. Then we can obtain $\mathbf{h}_{\text{in,pos}}$, $\mathbf{h}_{\text{in,neg}}$, $\mathbf{h}_{\text{in,neu}}$ by providing $\mathbf{h}_{\text{pos}}$, $\mathbf{h}_{\text{neg}}$, $\mathbf{h}_{\text{neu}}$ separately. As we relax the layer-by-layer restriction, the differentiation between the input/hidden/output layers is also relaxed. We directly put the input $\mathbf{h}$ to all computational neurons. Thus, the input dimension size of N, $d_{\text{in}}^N = d_{\mathbf{h}} + d_{\text{out}}^{N_1} + d_{\text{out}}^{N_2} + ... + d_{\text{out}}^{N_n}$.

Readout Layer. Readout layer collects information from all computational neurons and decides on the classification. The input of the readout layer is the concat function of all computational neurons as $\mathbf{h}_{\text{in}}^{\text{readout}} = f_{\text{readout}}(\mathbf{h}_{\text{out}}^{N_*}) = ||_{i=1}^{|\mathcal{V}|}(\mathbf{h}_{\text{out}}^{N_i})$, where $||$ is the concat function. Then, the readout layer casts the representation to output dimension as $\hat{\mathbf{y}} = \text{Softmax}(\mathbf{W}_{\text{readout}}\mathbf{h}_{\text{in}}^{\text{readout}})$. where $\mathbf{W}_{\text{readout}} \in \mathbb{R}^{\text{Class Number} \times d(\mathbf{h}_{\text{in}}^{\text{readout}})}$ is the parameter of the readout layer and $\hat{\mathbf{y}}$ is the prediction vector on classes.

2.3 Local Optimization

Computational Neuron Optimization. Computational neurons are optimized to differentiate the positive examples from negative ones. For computational neuron N, its optimization involves $\mathbf{h}_{\text{in,pos}}$ and $\mathbf{h}_{\text{in,neg}}$. After the computational neuron update, we can get $\mathbf{h}_{\text{out,pos}}$ and $\mathbf{h}_{\text{out,neg}}$, respectively. Then, following [7], a goodness score is calculated as $p(\mathbf{h}) = \sigma(\sum_i h_i^2 - \theta * d(\mathbf{h}))$, where $p(\mathbf{h})$ is the goodness score of $\mathbf{h}$, $d(\mathbf{h})$ is the dimension size of $\mathbf{h}$, σ is the Relu activation function and θ is the threshold. The binary cross-entropy loss is used to optimize each computational neuron as $\mathcal{L}_N = -\frac{1}{|\mathcal{D}|}\sum_{\mathcal{D}}(\log(p(\mathbf{h}_{\text{out,pos}})) - \log(p(\mathbf{h}_{\text{out,neg}})))$, where $\mathcal{D}$ is the dataset. The optimization of computational neurons aims to increase the neuron's output for positive samples while decreasing the neurons' output for negative samples. It enables each computational neuron its own ability to differentiate positive examples from negative ones.

Readout Layer Optimization. To relieve the label leakage issue, the readout layer is only optimized with $\mathbf{h}_{\text{neutral}}$, and we use a multi-class cross-entropy loss to optimize the readout layer as $\mathcal{L}_{\text{Readout}}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{|\mathcal{D}|}\sum_{|\mathcal{D}|}\sum_{c=1}^{C} y_c \log(\hat{y}_c)$, where C is the number of classes, $\mathbf{y}$ is the one-hot vector of ground-truth label and $\hat{\mathbf{y}}$ is the prediction.

During the inference time, we pair each test sample with the neutral label to construct $\mathbf{h}_{\text{neu}}$. It then propagates through the GOMLP to obtain its representation on each computational neuron. Finally, we predict its class with the largest logit from the output of the readout layer.

3 Experiments

Datasets. We conduct experiments on three widely studied datasets from computer vision and natural language processing domains. For each dataset, the training and test split follows the original setting. We further extract 20% samples from the training data as validation sets to tune hyper-parameters. For MNIST [2] [10], we directly use its flattened pixel values as the input of all methods and replace the first 10 pixels with labels as the fusion function, which is the same as [7] and leads to an input

[2] http://yann.lecun.com/exdb/mnist/

dimension of $28 * 28 = 784$. For NLP datasets (NewsGroup [3] [9], IMDB [4] [12]), we use BERT [5] to encode the sentences into a fixed-length tensor (768) as the input. The fusion function is the concat function, which leads to an input dimension of $768 + 20 = 788$ for NewsGroup and $768 + 2 = 770$ for IMDB dataset.

Baselines. In this paper, we aim to reveal the advantages of graph-structured multi-layer perceptron. We compared GOMLP with a variant of different methods, which can be differentiated by two attributes (Training and Graph). Training indicates the training method, where BP indicates Backward Propagation [16] and FF indicates the Forward-forward algorithm [7]. The graph indicates the graph structure of computational neurons. We keep 4 computation neurons for all methods during the experiments. The special cases are further illustrated as:

- BP-Chain*: Layer-by-layer neural networks trained with BP as depicted in Figure 2(a). It is the current default way of building and training ANNs.
- FF-Chain: Layer-by-layer neural networks trained with FF as depicted in Figure 2(b) same as [7].
- BP-Chain: A modified version of BP-Chain*, where we use the structure of Figure 2(b) and trained with BP. It adds direct local supervision on each layer.

We conduct experiments on MNIST [10], NewsGroup [9] and IMDB [12] dataset, and the setup is illustrated in the supplementary material. FF-Cycle, FF-WSGraph, FF-BAGraph, and FF-Complete are different versions of GOMLP, where the training is FF and only the graph generator defined in Eq. 2 differs.

Table 1: Error rate (%) $\downarrow$ on different datasets.

Train	Graph	MNIST	NewsGroup	IMDB
BP	Chain*	$1.77_{\pm 0.16}$	$42.11_{\pm 0.92}$	$\mathbf{17.16}_{\pm 0.19}$
FF	Chain	$1.83_{\pm 0.2}$	$43.88_{\pm 0.28}$	$18.75_{\pm 0.92}$
BP	Chain	$1.74_{\pm 0.11}$	$38.85_{\pm 0.42}$	$17.27_{\pm 0.13}$
FF	Cycle	$1.80_{\pm 0.14}$	$43.54_{\pm 0.41}$	$18.97_{\pm 0.49}$
FF	WSGraph	$1.70_{\pm 0.17}$	$38.28_{\pm 0.13}$	$17.93_{\pm 0.28}$
FF	BAGraph	$1.64_{\pm 0.08}$	$38.41_{\pm 0.14}$	$18.20_{\pm 0.67}$
FF	Complete	$\mathbf{1.54}_{\pm 0.05}$	$\mathbf{38.266}_{\pm 0.06}$	$17.58_{\pm 0.20}$

3.1 Overall Comparison

The overall experiment result is shown in Table 1. We show the error rate of different methods on different datasets (the lower, the better). Best performance is marked bold. From the table, we can have several interesting and exciting findings:

- FF-Complete achieves the best performance on MNIST and NewsGroup datasets and comparable results to the best one on the IMDB dataset. It is the first FF-trained model that outcompetes the BP-trained model. It is an exciting observation of the effectiveness of the FF algorithm compared with the BP algorithm.
- FF-Chain performs worse than BP-Chain* on all datasets. This observation is on par with [7], where the FF lags behind the BP training algorithm when they both follow layer-by-layer organization as a chain graph. However, we can surpass BP-Chain* when organizing the computational neurons as a graph structure. This finding inevitably reveals the advantages of GOMLP by organizing multi-layer perceptron as a flexible graph structure.
- FF-Cycle achieves similar performance with FF-Chain on three datasets. It is reasonable because these two methods have only one edge difference. When we build more complex graphs (WSGraph, BAGraph, Complete Graph), we can observe much better performance immediately. It shows the benefits of enriching the communication between computational neurons by the GOMLP.
- BP-Chain is better than BP-Chain* in most cases. Compared with BP-Chain*, BP-Chain further adds layer-wise optimization directly from the final loss. It indicates the advantageous layer-wise optimization, which provides new guidelines when designing layer-by-layer neural networks.

In summary, the experiment results answer that we do not need to stack neural networks layer-by-layer sequentially, and we can organize the neural networks as a flexible, complex graph structure like the brain. More excitingly, we can outperform the current *de facto* layer-by-layer neural network design paradigm with the Cyclic NN, and provide a totally new way of building ANNs.

[3] http://qwone.com/ jason/20Newsgroups/

[4] https://ai.stanford.edu/ amaas/data/sentiment/

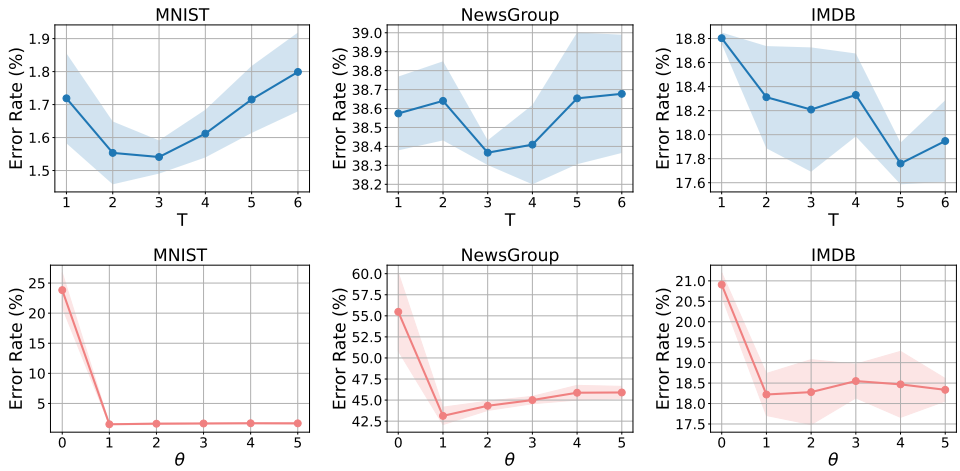

Figure 3: Parameter sensitivity of T and θ

3.2 Hyper-parameter Sensitivity

Experiment results of hyper-parameter (T and θ) sensitivity is shown in Figure 3. T controls the number of propagation between computational neurons. Larger T indicates more times the information is propagated. We can observe an error rate trend that first decreases and then increases on all three datasets. When T is small, computational neurons can not draw sufficient lessons from each other. When T is large, computational neurons are over-propagated, which leads to the over-smoothing problem. θ controls the goodness threshold of each computational neuron. We can observe a sharp error rate decrease when θ increases from 0 to 1, and then it gets stable with larger θ. It indicates the existence of the goodness threshold matters more than the threshold value. When $\theta = 0$, there is little room to optimize the computational neuron towards the negative sample, which can lead to the training collapse as the computational neuron can not differentiate the negative sample.

3.3 Ablation Study.

This section studies the impact of different optimization modules within GOMLP, including the computational neuron optimization $\mathcal{L}_N$ and readout layer optimization $\mathcal{L}_{\text{Readout}}$. We conduct experiments on the FF-Complete structure, and the results are summarized in Table 2. We can have the following observations: 1) The error rate increases when removing any optimization module, indicating the usefulness of each component. 2) GOMLP falls to a very large error rate (nearly random guess) when removing $\mathcal{L}_{\text{Readout}}$. It is reasonable as we depend on the readout layer to complete the final classification task. Without optimization on the readout layer, GOMLP falls into random guess even with optimized computational neuron's input. 3) The error rate increases by removing $\mathcal{L}_N$. It shows the computational neuron's optimization can provide a more informative goodness score for the readout layer to complete the classification task. $\mathcal{L}_N$ and $\mathcal{L}_{\text{Readout}}$ complement each other within GOMLP, and they collectively make the best performance.

Table 2: Error rate (%) ↓ of Ablation study.

Model	MNIST	NewsGroup	IMDB
FF-Complete	**1.54**	**38.26**	**18.20**
-$\mathcal{L}_N$	2.24	47.61	22.94
-$\mathcal{L}_{\text{Readout}}$	95.58	95.55	44.26

4 Conclusion

In summary, this research introduces Cyclic NN, a novel ANN architecture inspired by the complex, graph-like neural networks in biological intelligence. This transformative design diverges from traditional directed acyclic ANN structures. Our findings, demonstrated through the Graph Over Multi-layer Perceptron model and validated on various datasets, showed enhanced performance over conventional DAG networks. This significant development paves the way for more flexible and biologically realistic AI systems, representing a major shift in ANN design.

References

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] David Beniaguev, Idan Segev, and Michael London. Single cortical neurons as deep artificial neural networks. *Neuron*, 109(17):2727–2739, 2021.

[3] Paul Brooks, Andrew Champion, and Marta Costa. Mapping of the zebrafish brain takes shape. *Nature Methods*, 19(11):1345–1346, 2022.

[4] Steven J Cook, Travis A Jarrell, Christopher A Brittin, Yi Wang, Adam E Bloniarz, Maksim A Yakovlev, Ken CQ Nguyen, Leo T-H Tang, Emily A Bayer, Janet S Duerr, et al. Whole-animal connectomes of both caenorhabditis elegans sexes. *Nature*, 571(7763):63–71, 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

[7] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

[8] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[9] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995*, pages 331–339. Elsevier, 1995.

[10] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[11] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[12] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[13] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[14] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL `https://doi.org/10.48550/arXiv.2303.08774`.

[15] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[16] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[17] Alexander Shapson-Coe, Michał Januszewski, Daniel R Berger, Art Pope, Yuelong Wu, Tim Blakely, Richard L Schalek, Peter H Li, Shuohong Wang, Jeremy Maitin-Shepard, et al. A petavoxel fragment of human cerebral cortex reconstructed at nanoscale resolution. *Science*, 384(6696):eadk4858, 2024.

[18] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[19] Olaf Sporns and Edward T Bullmore. From connections to function: the mouse brain connectome atlas. *Cell*, 157(4):773–775, 2014.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[21] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[22] John G White, Eileen Southgate, J Nichol Thomson, Sydney Brenner, et al. The structure of the nervous system of the nematode caenorhabditis elegans. *Philos Trans R Soc Lond B Biol Sci*, 314(1165):1–340, 1986.

[23] Michael Winding, Benjamin D Pedigo, Christopher L Barnes, Heather G Patsolic, Youngser Park, Tom Kazimiers, Akira Fushiki, Ingrid V Andrade, Avinash Khandelwal, Javier Valdes-Aleman, et al. The connectome of an insect brain. *Science*, 379(6636):eadd9330, 2023.