

Appendix of Self-Assembling Graph Perceptrons

Contents

1	Differences between self-assembling networks and other topics	1
1.1	Neural pruning	1
1.2	Neural architecture search	2
1.3	Graph structure learning	2
2	MLPs is a special case of graph perceptrons	2
3	Implementation Details	3
3.1	Dataset description, feature extraction and dataset partitioning	3
3.2	Hyper-parameters	5
3.3	Other details	6
4	More experiments	8
4.1	Assembling dynamic	8
4.2	Model Interpretability	11
4.3	Feature selection in FSDD & ESC-50	11
4.4	Feature selection in Photo & Computers	11
4.5	Ablation study	11
4.6	Hyperparameter analysis	13
5	limitations and Future direction	13
5.1	limitations	13
5.2	Future direction	14

1 Differences between self-assembling networks and other topics

1.1 Neural pruning

In the human brain, the connections between neurons are dynamic, with some unimportant connections being weakened or even removed to improve the network’s efficiency [1]. This characteristic has inspired deep learning researchers to explore how to remove redundant parameters or structures in deep neural networks to enhance computational efficiency and storage performance and reduce over-parameterization [2, 3]. Based on the timing of pruning, model pruning is typically categorized into pruning before training (PBT), pruning during training (PDT), and pruning after training (PAT) [4]. Self-assembling neural networks conceptually resemble PDT, but unlike PDT, which operates on a network whose structure and scale are already predefined [5, 6], self-assembling networks begin with a basic form and develop, with pruning and growth serving as two aspects of achieving self-assembly.

1.2 Neural architecture search

With the widespread application of ANNs, the design and optimization of network architectures have become a key research direction. However, manually designed architectures often fail to leverage the dataset's potential when tackling complex tasks comprehensively. As an automated architecture search method, Neural Architecture Search (NAS) can reduce human intervention and save experimental costs [7]. In NAS, the first step is to define the architecture search space, such as the operations that can be applied to each neural network layer and the selectable hyperparameters. The search process is typically based on methods such as reinforcement learning [8], evolutionary algorithms [9], or gradient optimization [10]. In contrast to NAS, self-assembling networks focus on a finer level of granularity. While NAS explores the optimal combinations of operations, self-assembling networks investigate whether individual operators, exceptionally fully connected layers, can have more streamlined and efficient structures at the neuron and synapse level.

1.3 Graph structure learning

The quality of the graph structure is crucial for effective information propagation on the graph. Standard graph structure learning methods include learning based on node-pair similarity [11], neural network-based learning [12], and direct optimization [13]. In SAGP, we treat neurons connected by synapses as a graph and achieve a dynamic topology by directly optimizing masks on the neurons and the synapses. Additionally, we use the message-passing paradigm [14] commonly employed in graph neural networks for information propagation between neurons, replacing the unidirectional layer-by-layer information flow in traditional MLPs.

2 MLPs is a special case of graph perceptrons

Theorem 1. Let z_{MLP} be the output of the MLP with L hidden layers as follows:

$$\begin{aligned} h_{\text{MLP}}^{l+1} &= \mathbf{W}_l^T \mathbf{x}_{\text{MLP}}^l + \mathbf{b}_l, \quad \mathbf{x}_{\text{MLP}}^{l+1} = \sigma(\mathbf{h}_{\text{MLP}}^{l+1}) \quad \forall l \in [0, L-1], \\ z_{\text{MLP}} &= \mathbf{W}_L^T \mathbf{h}_{\text{MLP}}^L, \quad \mathbf{x}_{\text{MLP}}^0 = \mathbf{x}. \end{aligned} \quad (1)$$

Construct the adjacency matrix and bias vector for this MLP as follows:

$$\mathbf{A}^T = \begin{bmatrix} \mathbf{I} & & & & \\ \mathbf{W}_0^T & & & & \\ & \ddots & & & \\ & & \mathbf{W}_{L-1}^T & & \\ & & & \mathbf{W}_L^T & \mathbf{0} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_{L-1} \\ \mathbf{0} \end{bmatrix}. \quad (2)$$

The output z_{GP} of the Graph Perceptron after $L+1$ message-passing steps is as follows:

$$z_{\text{GP}} = \left[\mathbf{A} \cdot \underbrace{f \circ f \circ \dots \circ f}_{L \text{ times}} \left(\begin{bmatrix} \mathbf{x} \\ \mathbf{0}_{n_1+\dots+n_{L+1}} \end{bmatrix} \right) + \mathbf{b} \right]_{[-n_{L+1}:]}, \quad \text{where } f(\mathbf{a}) \equiv \sigma(\mathbf{A}^T \cdot \mathbf{a} + \mathbf{b}). \quad (3)$$

Then we have:

$$z_{\text{MLP}} \equiv z_{\text{GP}}. \quad (4)$$

Where $\mathbf{W}_l^T \in \mathbb{R}^{n_{l+1} \times n_l}$, $\mathbf{b}^l \in \mathbb{R}^{n_{l+1}}$, $\mathbf{x} \in \mathbb{R}^{n_0}$, and $\mathbf{z} \in \mathbb{R}^{n_{L+1}}$.

61 *Proof.* For any vector $[\mathbf{a}_0 \ \mathbf{a}_1 \ \dots \ \mathbf{a}_L \ \mathbf{a}_{L+1}]^T$, we have:

$$f\left(\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_L \\ \mathbf{a}_{L+1} \end{bmatrix}\right) = \sigma\left(\begin{bmatrix} \mathbf{I} & & & & \\ \mathbf{W}_0^T & & & & \\ & \ddots & & & \\ & & \mathbf{W}_{L-1}^T & & \\ & & & \mathbf{W}_L^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_L \\ \mathbf{a}_{L+1} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_{L-1} \\ \mathbf{0} \end{bmatrix}\right) \quad (5)$$

$$= \begin{bmatrix} \sigma(\mathbf{a}_0) \\ \sigma(\mathbf{W}_0^T \mathbf{a}_0 + \mathbf{b}_0) \\ \vdots \\ \sigma(\mathbf{W}_{L-1}^T \mathbf{a}_{L-1} + \mathbf{b}_{L-1}) \\ \sigma(\mathbf{W}_L^T \mathbf{a}_L) \end{bmatrix} = \begin{bmatrix} \sigma(\mathbf{a}_0) \\ g_0(\mathbf{a}_0) \\ \vdots \\ g_{L-1}(\mathbf{a}_{L-1}) \\ \sigma(\mathbf{W}_L^T \mathbf{a}_L) \end{bmatrix}, \quad (6)$$

62 where $g_i(\mathbf{a}) = \mathbf{W}_i^T \mathbf{a} + \mathbf{b}_i$. Then:

$$\begin{aligned} \underbrace{f \circ f \circ \dots \circ f}_{L \text{ times}}\left(\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_L \\ \mathbf{a}_{L+1} \end{bmatrix}\right) &= \underbrace{f \circ f \circ \dots \circ f}_{L-1 \text{ times}}\left(\begin{bmatrix} \sigma(\mathbf{a}_0) \\ g_0(\mathbf{a}_0) \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}\right) = \underbrace{f \circ f \circ \dots \circ f}_{L-2 \text{ times}}\left(\begin{bmatrix} \sigma^2(\mathbf{a}_0) \\ g_0 \circ \sigma(\mathbf{a}_0) \\ g_1 \circ g_0(\mathbf{a}_0) \\ \vdots \\ \vdots \end{bmatrix}\right) \\ &= \underbrace{f \circ f \circ \dots \circ f}_{L-3 \text{ times}}\left(\begin{bmatrix} \sigma^3(\mathbf{a}_0) \\ g_0 \circ \sigma^2(\mathbf{a}_0) \\ g_1 \circ g_0 \circ \sigma(\mathbf{a}_0) \\ g_2 \circ g_1 \circ g_0(\mathbf{a}_0) \\ \vdots \end{bmatrix}\right) = \dots \end{aligned} \quad (7)$$

63 So we have:

$$\underbrace{f \circ f \circ \dots \circ f}_{L \text{ times}}\left(\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_L \\ \mathbf{a}_{L+1} \end{bmatrix}\right)_{[n_0 + \dots + n_{L-1} : n_0 + \dots + n_L]} = g_{L-1} \circ \dots \circ g_1 \circ g_0(\mathbf{a}_0) \quad (8)$$

64 Let $\mathbf{a}_0 = \mathbf{x}$ and $\mathbf{a}_1 = \mathbf{a}_2 = \dots = \mathbf{a}_{L+1} = \mathbf{0}$, then:

$$\underbrace{f \circ f \circ \dots \circ f}_{L \text{ times}}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{0}_{n_1 + \dots + n_{L+1}} \end{bmatrix}\right)_{[n_0 + \dots + n_{L-1} : n_0 + \dots + n_L]} = g_{L-1} \circ \dots \circ g_1 \circ g_0(\mathbf{x}) = \mathbf{x}_{\text{MLP}}^L. \quad (9)$$

65 So:

$$\mathbf{z}_{\text{GP}} = \mathbf{W}_L^T \mathbf{x}_{\text{MLP}}^L + \mathbf{0} = \mathbf{z}_{\text{MLP}}. \quad (10)$$

66 \square

67 3 Implementation Details

68 3.1 Dataset description, feature extraction and dataset partitioning

69 **ESC-50** ESC-50 [15] is the most commonly used dataset for environmental sound classification,
 70 containing 2,000 segments of environmental sounds across 50 categories, with 40 samples per
 71 category, each lasting 5 seconds (431 time steps) and provided in WAV format. These sounds
 72 encompass a variety of contexts, including natural environments, urban noise, animal sounds, and
 73 human activities. During feature processing, we utilized the librosa library [16] to extract Mel-
 74 frequency cepstral coefficients (MFCC, 40 features), chroma features (Chroma, 12 features), and Mel

Domain Feature Dataset	Text bag-of-words		Audio		Image	
	Photo	Computers	ESC-50	FSDD	Normalized grayscale Fashion-MNIST	CIFAR-10
#Samples	7,650	13,752	2,000	3,000	70,000	60,000
#Features	745	767	40 + 12 + 128	40 + 12 + 128	28 * 28	32 * 32
#Class	8	10	50	10	10	10
#Edges	238,162	491,722	—	—	—	—
#Time steps	—	—	431	7 ~ 99	—	—
#Channels	—	—	—	—	1	3
Partitioning (Train/Val/Test)	800/800/6050	1000/1000/11752	1200/400/400	1800/600/600	60000/-/10000	50000/-/10000

Table 1: Dataset statistics.

spectrogram features (Mel, 128 features) for every time step. After concatenating these features, each audio segment is represented by a feature matrix of 431×180 . For non-recurrent neural networks, we compute the average across all time steps to obtain a vector of length 180 as the input to the network. For recurrent neural networks, we sequentially input the features of each time step and perform classification using the output of the last time step. The ESC-50 dataset is officially provided with a 5-fold split, and we used folds 1-3 as the training set, fold 4 as the validation set, and fold 5 as the test set.

FSDD The Free Spoken Digit Dataset ¹ (FSDD) is a foundational dataset for research in speech recognition. It consists of voice samples of the digits 0 to 9 recorded by six speakers of varying ages, genders, and nationalities. Each speaker recorded each digit 50 times, resulting in 3,000 audio segments, each approximately 1 second long (7 ~ 99 time steps). When using recurrent or non-recurrent neural networks to train on the FSDD dataset, we apply the same feature extraction method for the ESC-50 dataset. Regarding dataset partitioning, we use the first 30 audio segments recorded for each digit by each speaker as the training set, 10 segments as the validation set, and the remaining 10 segments as the test set, resulting in training/validation/test set sizes of 1800/600/600, respectively.

Computers & Photo Based on product descriptions and user records from the Amazon shopping website, many datasets widely used for tasks such as image recognition, text classification, and graph learning have been derived. We used the Computers and Photo datasets from a preprocessed version of Amazon product review data provided by [17]. Each sample in the dataset represents a product in a specific category (Computer or Photo) on the Amazon website, with sample features derived from text descriptions written by product buyers, extracted using a bag-of-words model. These features can be directly applied to text classification tasks. Furthermore, this preprocessed version also provides connectivity between samples, generating an edge between two products when frequently purchased. Therefore, these two datasets are also suitable for node classification tasks in graph deep learning. We selected 100 samples from each category in the dataset for training, 100 for validation, and the remaining samples for testing.

Fashion-MNIST The Fashion MNIST dataset ² is one of the most famous image recognition datasets, consisting of 70,000 grayscale images of size 28x28 pixels with single channel, each depicting a category of clothing, such as clothes, shoes, handbags, etc. The sample features are based on the pixel values of each image (ranging from 0 to 1), followed by normalization pre-processing. This dataset is typically divided into 60,000 training samples and 10,000 test samples, ensuring that the sample proportion for each category is the same, but there is no separate validation set.

CIFAR-10 CIFAR-10 [18] is a widely used image classification dataset commonly employed to evaluate the performance of machine learning and deep learning models. The dataset consists of 60,000 color images of size 32x32 pixels, with each image comprising three color channels (Red, Green, and Blue) divided into 10 classes, with 6,000 images per class. The classes include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is evenly split into 50,000 training images and 10,000 test images. Each image’s pixel values range from 0 to 1 and are typically normalized before use.

¹<https://github.com/Jakobovski/free-spoken-digit-dataset>

²<https://github.com/zalandoresearch/fashion-mnist>

3.2 Hyper-parameters

SAGP The hyperparameters of SAGP are divided into two parts: the first consists of hyperparameters introduced by SAGP itself, for which we use the same settings across all experiments and all datasets (Table 2). The second part consists of shared hyperparameters used in all competing methods. We perform limited hyperparameter tuning for these hyperparameters to ensure that the experimental results accurately reflect the model’s performance and enable a fair comparison. Each hyperparameter’s search space is limited to avoid overfitting the data due to excessively fine-tuned hyperparameters. All methods use the same hyperparameter search space (Table 3). The search results of SAGP on these parameters are shown in Table 4. It is a highly challenging setup, and it is not entirely fair even for SAGP, as careful tuning of hyperparameters can improve the model’s performance. However, we aim to demonstrate the strong adaptability of SAGP in handling different inputs (datasets) through this setup.

Hyperparameter	Description	Value
L	Number of message passing (Eq. 2)	3
α	Temperature in SparseLoss (Eq. 2)	0.2
γ_1	Scaling factor in Auxloss (Eq. 2)	0.05
γ_2	Scaling factor in Auxloss (Eq. 2)	0.001
β	the threshold used in Rule 1	0.5
N	the patience used in Rule 2	10
$ \mathcal{H} _{\max}$	Maximum number of new neurons	128

Table 2: Default settings for SAGP-specific hyperparameters.

Hyperparameter	Search space
Learning rate	[0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001]
Weight decay	[0.01, 0.005, 0.002, 0.001, 0.0005, 0.0002, 0.0001, 0.]
Batchnorm	[True, False]
Dropout	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

Table 3: Search space of common hyperparameters.

	Photo	Computers	ESC-50	FSDD	Fashion-MNIST	CIFAR-10
Learning rate	0.01	0.01	0.002	0.002	0.001	0.001
Weight decay	0.01	0.005	0.001	0.0001	0.0002	0.001
BatchNorm	True	True	True	False	True	True
Dropout	0.5	0.8	0.8	0.4	0.0	0.0

Table 4: Hyperparameters used by SAGP. Take the best value obtained by performing 200 trials in the hyperparameter search space.

MLP For the MLP model, in addition to the hyperparameters listed in Table 3, there are two additional hyperparameters: the number of hidden layers and the size of each layer. The search space for these two hyperparameters is shown in 5. The hyperparameter search results for MLP on six datasets are summarized in Table 6.

Hyperparameter	Search space
#Hidden layer	[0, 1, 2, 3, 4, 5]
#Neurons per hidden layer	[64, 128, 256, 512]

Table 5: Search space of MLP-specific hyperparameters.

	Photo	Computers	ESC-50	FSDD	Fashion-MNIST	CIFAR-10
Learning rate	0.005	0.001	0.001	0.001	0.001	0.001
Weight decay	0.0	0.0	0.005	0.0005	0.0	0.0005
Dropout	0.6	0.6	0.6	0.5	0.0	0.0
BatchNorm	False	False	True	True	True	True
#Hidden layer	1	2	2	4	2	3
#Neurons per hidden layer	512	512	512	512	512	256

Table 6: Hyperparameters used by MLP. Take the best value obtained by performing 200 trials in the hyperparameter search space.

131 **GP** For the 8 different topologies of the non-self-assembling graph perceptron models (described in
132 Section 4.1), we fixed the number of hidden neurons to 128 and performed a hyperparameter search. The results are shown in Table 7.

	Hyperparameter	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
Photo	Learning rate	0.01	0.1	0.05	0.05	N/A	0.02	0.02	0.02
	Weight decay	0.005	0.001	0.0002	0.0002	N/A	0.01	0.005	0.005
	Batchnorm	True	True	True	True	N/A	True	True	True
	Dropout	0.1	0.7	0.8	0.8	N/A	0.4	0.9	0.0
Computers	Learning rate	0.05	0.01	0.1	0.02	N/A	0.05	0.1	0.1
	Weight decay	0.0001	0.01	0.0001	0.005	N/A	0.005	0.0002	0.0002
	Batchnorm	True	False	True	True	N/A	True	True	True
	Dropout	0.9	0.8	0.8	0.9	N/A	0.8	0.8	0.6
ESC-50	Learning rate	0.02	0.02	0.02	0.02	0.01	0.02	0.05	0.02
	Weight decay	0.005	0.002	0.005	0.005	0.005	0.005	0.005	0.005
	Batchnorm	False	True	True	True	True	True	True	False
	Dropout	0.2	0.7	0.6	0.8	0.7	0.4	0.4	0.5
FSDD	Learning rate	0.05	0.05	0.1	0.05	0.05	0.02	0.05	0.05
	Weight decay	0.0002	0.0001	0.0002	0.0005	0.0	0.0001	0.0001	0.0002
	Batchnorm	False	True	False	False	True	True	True	False
	Dropout	0.1	0.6	0.1	0.1	0.5	0.4	0.3	0.2
Fashion-MNIST	Learning rate	0.02	0.01	0.01	0.005	0.005	0.01	0.01	0.02
	Weight decay	0.0	0.0001	0.0	0.0001	0.0	0.0002	0.0	0.0
	Batchnorm	True	True	True	True	True	True	True	True
	Dropout	0.1	0.3	0.2	0.1	0.2	0.2	0.2	0.0
CIFAR-10	Learning rate	0.02	0.01	0.02	0.002	N/A	0.01	0.02	0.01
	Weight decay	0.0	0.0	0.0	0.0002	N/A	0.0	0.0	0.0
	Batchnorm	True	True	True	True	N/A	True	True	True
	Dropout	0.1	0.1	0.1	0.1	N/A	0.0	0.1	0.1

Table 7: Hyperparameters used by fix-topology Graph Perceptrons described in Section 4.1. Take the best value obtained by performing 200 trials in the hyperparameter search space.

133

134 **Other models** SAMLP, LSTM, and LeNet’s basic hyperparameters are consistent with those used in
135 MLP on the same dataset. SAMLP+SAGP, LSTM+SAGP, and LeNet+SAGP’s basic hyperparameters
136 are consistent with those used in SAGP on the same dataset. In the feature selection and parameter
137 pruning experiments, the hyperparameters we used are identical to those in the original backbone
138 models.

139 3.3 Other details

140 **Code** Here, we provide an anonymous implementation of SAGP. Once the paper is accepted, we will
141 promptly release the public version.

142 <https://anonymous.4open.science/r/SAGP-0B7D>

143 **Experiment protocol** All reported data are the average results of repeated experiments conducted
144 with 20 different seeds.

145 **Maximum growth** In the implementation, we constrain the maximum growth size of the network by
 146 setting $|\mathcal{H}|_{\max}$. When the cumulative number of newly grown neurons reaches $|\mathcal{H}|_{\max}$, the grow rule
 147 will no longer be effective. This setting not only simplifies the model implementation but also aligns
 148 with biomimicry principles—i.e., in the human brain, the birth of new neurons primarily occurs in
 149 the early stages of the life cycle rather than throughout the entire lifespan.

150 **Pseudo-code** The neurons are numbered in the order of input, potential hidden, and output neurons.
 151 Specifically, index 1 to $|\mathcal{I}|$ represent the input neurons, index $|\mathcal{I}| + 1$ to $|\mathcal{I}| + |\mathcal{H}|_{\max}$ represent all
 152 possible hidden neurons, and index $|\mathcal{I}| + |\mathcal{H}|_{\max} + 1$ to $|\mathcal{I}| + |\mathcal{H}|_{\max} + |\mathcal{O}|$ represent the output
 153 neurons. Based on this convention, we present the pseudocode for SAGP during the training and
 testing phases.

Algorithm 1 Training process for Self-Assembling Graph Perceptrons (SAGP)

```

1: Input: Training sample  $(x, y)$ , input neuron set  $\mathcal{I} = \{1, \dots, |\mathcal{I}|\}$ , output neuron set  $\mathcal{O} =$ 
    $\{|\mathcal{I}| + |\mathcal{H}|_{\max} + 1, \dots, |\mathcal{I}| + |\mathcal{H}|_{\max} + |\mathcal{O}|\}$ , hyper-parameters  $(\alpha, \beta, \gamma, L, N)$  and trainable
   parameters  $(\{\hat{m}_i\}, \{\hat{m}_{ij}\}, \{A_{ij}\}, \{b_i\})$ 
2:  $\mathcal{H} = \{|\mathcal{I}| + 1\};$  ▷ Hidden neuron set
3:  $n = 1;$  ▷ The number of hidden neurons that have grown
4: repeat
5:   Compute soft neuron mask  $\{m_i^S\}$  and hard neuron mask  $\{m_i^H\}$  with  $\{\hat{m}_i\}$  using Eq. (10)-(12);
6:   Compute soft synapse mask  $\{m_{ij}^S\}$  and hard synapse mask  $\{m_{ij}^H\}$  with  $\{\hat{m}_{ij}\}$  using Eq.
     (10)-(12);
7:    $\mathcal{H} = \mathcal{H} / \{m_i^S < \beta \cdot \text{Mean}_{j \in \mathcal{H}}(m_j^S) | i \in \mathcal{H}\};$  ▷ Apoptosis rule
8:   if  $n < |\mathcal{H}|_{\max}$  and no neurons are removed for  $N$  consecutive epochs then
9:      $n = n + 1;$ 
10:     $\mathcal{H} = \mathcal{H} \cup \{|\mathcal{I}| + n\};$  ▷ Growth rule
11:   end if
12:    $\hat{x}_i^0 = \begin{cases} x_i, & \text{if } i \in \mathcal{I}, \\ 0, & \text{if } i \in \mathcal{H} \cup \mathcal{O}; \end{cases}$ 
13:   for  $l = 0, 1, \dots, L - 1$  do
14:      $h_i^{l+1} = b_i + \sum_{j \in \mathcal{I} \cup \mathcal{H}} m_j^H m_{ij}^H A_{ij} \hat{x}_i^l;$ 
15:      $\hat{x}_i^{l+1} = \sigma(h_i^{l+1});$ 
16:   end for
17:    $z = [h_i^L]_{i \in \mathcal{O}};$ 
18:   Compute task-related loss TaskLoss with  $z$  and  $y$ ;
19:   Compute auxiliary loss AuxLoss with  $\{m_{ij}^S\}$  and  $\{m_{ij}^H\}$ , using Eq. (14);
20:   Loss = TaskLoss +  $\gamma$  AuxLoss;
21:   Back forward Loss to update parameters;
22: until convergent.

```

Algorithm 2 Test process for SAGP

```

1: Input: Test sample  $(x, y)$ , input neuron set  $\mathcal{I}$ , hidden neuron set  $\mathcal{H}$ , output neuron set  $\mathcal{O}$ 
2: Compute hard neuron mask  $\{m_i^H\}$  with  $\{\hat{m}_i\}$ ;
3: Compute hard synapse mask  $\{m_{ij}^H\}$  with  $\{\hat{m}_{ij}\}$ ;
4:  $\hat{x}_i^0 = \begin{cases} x_i, & \text{if } i \in \mathcal{I}, \\ 0, & \text{if } i \in \mathcal{H} \cup \mathcal{O}; \end{cases}$ 
5: for  $l = 0, 1, \dots, L - 1$  do
6:    $h_i^{l+1} = b_i + \sum_{j \in \mathcal{I} \cup \mathcal{H}} m_j^H m_{ij}^H A_{ij} \hat{x}_i^l;$ 
7:    $\hat{x}_i^{l+1} = \sigma(h_i^{l+1});$ 
8: end for
9:  $z = [h_i^L]_{i \in \mathcal{O}};$ 
10: Compute test metrics using  $z$  and  $y$ .

```

155 **Platform** All experiments were run on an Intel(R) Xeon(R) Gold 6240C CPU @ 2.60GHz and a
 156 single Nvidia GeForce RTX 4090. The primary software versions used were PyTorch 2.1.1 and
 157 PyTorch Geometric 2.4.0.

158 **Activation, BatchNorm, and output enhancement** We used Swish [19] as the default activation
 159 function. For SAGP, we implemented a channel-wise activated BatchNorm to replace the traditional
 160 BatchNorm. Additionally, we considered adding self-loops on the output neurons to enhance the
 161 output.

$$A = \begin{bmatrix} I_{|\mathcal{I}| \times |\mathcal{I}|} & A_{\mathcal{I} \rightarrow \mathcal{H}} & A_{\mathcal{I} \rightarrow \mathcal{O}} \\ \mathbf{0}_{|\mathcal{H}| \times |\mathcal{I}|} & A_{\mathcal{H} \rightarrow \mathcal{H}} & A_{\mathcal{H} \rightarrow \mathcal{O}} \\ \mathbf{0}_{|\mathcal{O}| \times |\mathcal{I}|} & \mathbf{0}_{|\mathcal{O}| \times |\mathcal{H}|} & I_{|\mathcal{O}| \times |\mathcal{O}|} \end{bmatrix}. \quad (11)$$

162 4 More experiments

163 4.1 Assembling dynamic

164 We provide the self-assembly dynamics for all six datasets we used, including changes in the number
 165 of hidden neurons and synapses, variations in training and test losses, and visualizations of the
 166 topology of hidden neurons and their synapses. We broadly divide the self-assembly process into
 167 three stages. Stage 1: The topology expands rapidly in size. Stage 2: Intense competition occurs
 168 among neurons, typically accompanied by the apoptosis of many neurons (with some exceptions, such
 169 as ESC-50, where the number of neurons continues to increase, indicating that the dataset’s features
 170 are more complex and require stronger model capabilities). Stage 3: Neuron numbers stabilize, but
 synapses continue to prune slowly.

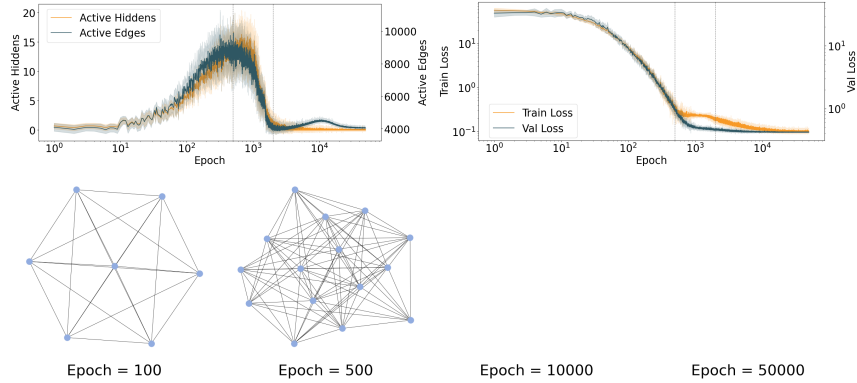


Figure 1: Assembling dynamic for the Photo dataset. We only depicted the hidden neurons and their connections. When Epochs are 10000 and 50000, there are no hidden neurons in the SAGP topology, with only synaptic connections from input neurons to output neurons remaining.

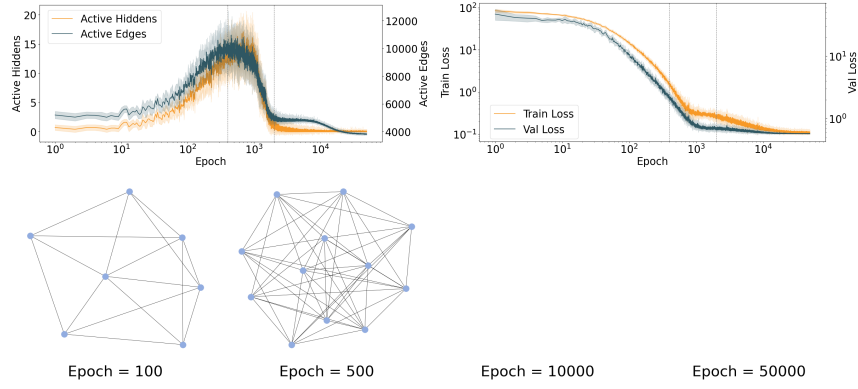


Figure 2: Assembling dynamic for the Computers dataset. We only depicted the hidden neurons and their connections. When Epochs are 10000 and 50000, there are no hidden neurons in the SAGP topology, with only synaptic connections from input neurons to output neurons remaining.

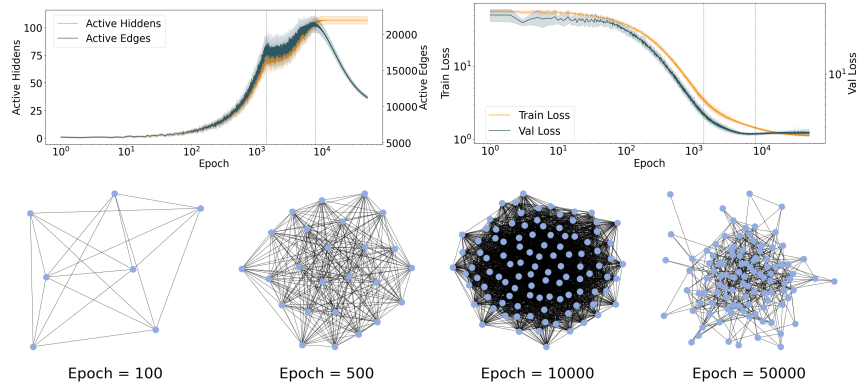


Figure 3: Assembling dynamic for the ESC-50 dataset.

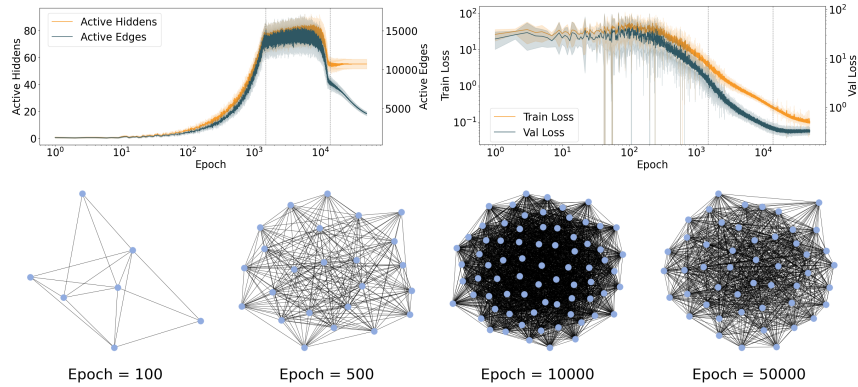


Figure 4: Assembling dynamic for the FSDD dataset.

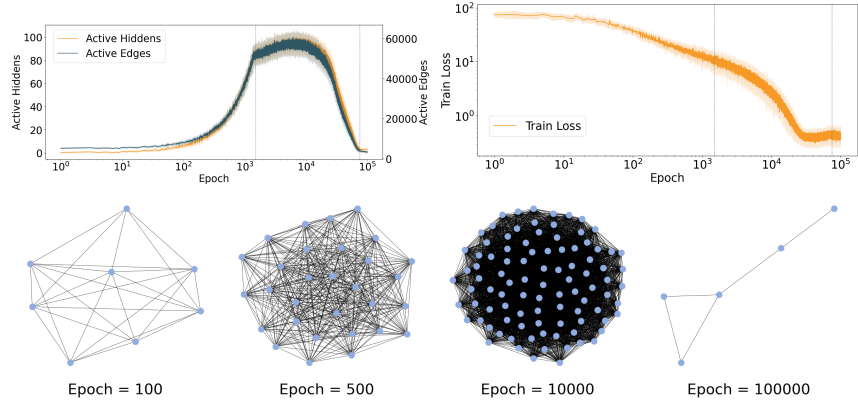


Figure 5: Assembling dynamic for the Fashion-MNIST dataset.

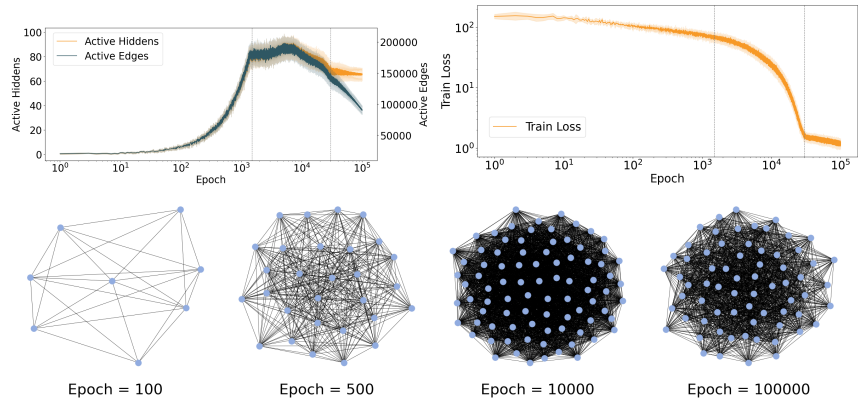


Figure 6: Assembling dynamic for the CIFAR-10 dataset.

4.2 Model Interpretability

Building on the model interpretability experiments in Section 4.3, we added results for the Fashion-MNIST dataset. Specifically, when SAGP runs on image datasets, we visualize the out-degree of each pixel after the assembly process reaches stability. For CIFAR-10, high out-degree neurons were concentrated in the image’s center, where the subject often appears. For Fashion-MNIST, the high out-degree neurons exhibited a ring-like distribution, indicating that the network primarily relies on the edges separating the subject from the background for classification.

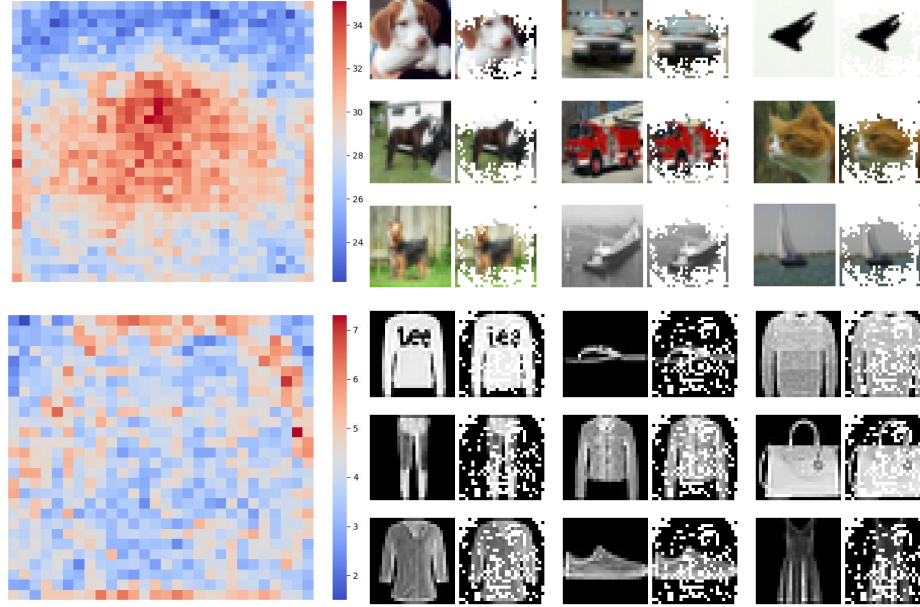


Figure 7: In the CIFAR-10 dataset (upper) and Fashion-MNIST dataset (lower), each pixel is treated as an input neuron. We visualize the out-degree of all pixels and display an image in which only the top 50%/70% of the pixels with the largest out-degrees are retained.

4.3 Feature selection in FSDD & ESC-50

Building on the feature selection experiments in Section 4.3, we added results for the ESC-50 dataset. Specifically, for the audio classification datasets obtained using MFCC, Chroma, and Mel feature extraction methods, we used the out-degree of input neurons after the assembly process stabilized to indicate feature channel importance. We found that for both the FSDD and ESC-50 datasets, the out-degree of the Chroma feature was smaller, and experimental results confirmed that this feature contributed less to the classification. This demonstrates the potential of self-assembling networks for feature selection.

4.4 Feature selection in Photo & Computers

Once SAGP training converges, we take the outdegree of each input neuron as an indicator of the corresponding feature’s importance. We then select the top 5%, 10%, and 20% of features by outdegree, train them on different backbone models, and report their performance. The results show that although SAGP is not specifically designed for feature selection, it still achieves performance comparable to current state-of-the-art methods (Table 8).

4.5 Ablation study

We conducted ablation experiments on the three key components of SAGP. For SAGP w/o Growth, we set the initial number of hidden neurons to $|\mathcal{H}|_{\max}$ and turned off the growth rule. For SAGP w/o

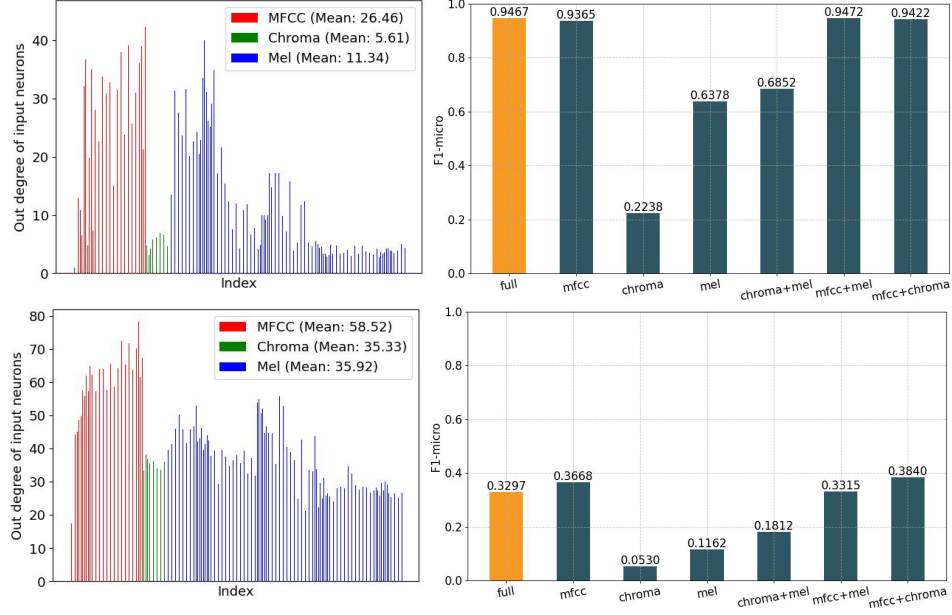


Figure 8: When training SAGP on the FSDD dataset (upper) and ESC-50 dataset (lower), we present the out-degree statistics for three types of input features and the results of training an MLP with different combinations of input features.

Backbone	Ratio	XGBoost	LassoNet	GradEnFS	SAGP
Photo					
GCN	5%	.8569 \pm .0073	.9162\pm.0029	.9043 \pm .0036	.9147 \pm .0029
	10%	.9037 \pm .0042	.9197\pm.0031	.9085 \pm .0030	.9185 \pm .0038
	20%	.9079 \pm .0037	.9201 \pm .0029	.9161 \pm .0025	.9213\pm.0034
GAT	5%	.8840 \pm .0110	.9141\pm.0029	.9104 \pm .0042	.9122 \pm .0033
	10%	.9112 \pm .0033	.9184\pm.0038	.9149 \pm .0058	.9165 \pm .0049
	20%	.9135 \pm .0070	.9190 \pm .0046	.9158 \pm .0061	.9194\pm.0049
Computers					
GCN	5%	.7794 \pm .0064	.8301 \pm .0062	.8041 \pm .0056	.8345\pm.0050
	10%	.7879 \pm .0126	.8435 \pm .0050	.8236 \pm .0061	.8450\pm.0057
	20%	.8270 \pm .0043	.8543\pm.0054	.8451 \pm .0085	.8529 \pm .0059
GAT	5%	.8137 \pm .0094	.8473 \pm .0078	.8233 \pm .0070	.8490\pm.0042
	10%	.8305 \pm .0118	.8480 \pm .0068	.8359 \pm .0054	.8517\pm.0047
	20%	.8407 \pm .0037	.8619\pm.0052	.8527 \pm .0106	.8599 \pm .0036

Table 8: We selected the top 5%, 10%, and 20% of features based on the out-degree as subsets, and compared them with state-of-the-art feature selection methods.

Competition, we did not include the auxiliary loss used to drive Competition during backpropagation. For SAGP w/o Apoptosis, we turned off the apoptosis rule during the self-assembly process (Table 9). Based on the experimental results, we draw the following conclusions: First, the growth rule allows the network to develop from a minimal scale, significantly reducing memory overhead compared to starting from a maximum scale. Moreover, the absence of the growth rule may prevent the model from recovering from suboptimal topologies, thereby degrading performance. Second, the lack of Competition prevents the network from simplifying its topology, increasing computational costs. Specifically, for the computer dataset, the network size of SAGP w/o Competition is much larger than that of SAGP, yet its performance does not show any advantage. Finally, the absence of the apoptosis rule prevents the network from simplifying to an optimal structure. Additionally, since apoptotic neurons no longer participate in computations, the lack of the apoptosis rule can potentially increase computational complexity.

			SAGP	SAGP w/o Growth	SAGP w/o Competition	SAGP w/o Apoptosis
FSDD	Best epoch	$ \mathcal{H} $	55.10	23.30	86.70	62.20
		#Synapse	6246	3345	18018	7278
		F1-micro	.9142 \pm .0112	.8033 \pm .0149	.9227 \pm .0098	.9137 \pm .0162
	Last epoch	$ \mathcal{H} $	55.00	1.00	86.70	61.00
		#Synapse	4312	968	17605	4735
		F1-micro	.9120 \pm .0142	.7843 \pm .0134	.9227 \pm .0040	.9080 \pm .0091
Peak memory usage		0.75 Gb	0.96 Gb	0.94 Gb	0.81 Gb	
Computers	Best epoch	$ \mathcal{H} $	0.00	0.00	55.60	0.00
		#Synapse	4376	4127	32980	4996
		F1-micro	.7687 \pm .0032	.7667 \pm .0032	.7683 \pm .0023	.7664 \pm .0068
	Last epoch	$ \mathcal{H} $	0.00	0.00	52.00	0.00
		#Synapse	3810	3827	31084	3884
		F1-micro	.7677 \pm .0040	.7650 \pm .0034	.7662 \pm .0024	.7663 \pm .0035
Peak memory usage		0.39 Gb	1.34 Gb	1.29 Gb	0.39 Gb	

Table 9: Ablation experiments were conducted on the three key mechanisms of SAGP: growth, competition, and apoptosis.

4.6 Hyperparameter analysis

In this experiment, we focus on the impact of the hyperparameter α on the performance of SAGP. As the temperature parameter in the competition loss, α controls the intensity of competition between neurons or synapses. As shown in Figure 4(b) in main text, when α is smaller, more masks tend to approach 0, indicating that the competition becomes more intense. As shown in Table 10, increasing α leads to more hidden neurons and synapses. While a larger topology can expand the capacity of the perceptron model and potentially bring some performance improvement (such as on the FSDD dataset), it may also result in a less efficient topology. For example, in the Computers dataset, when $\alpha = 0.5$, the number of synapses is 254 times larger than when $\alpha = 0.2$, yet the performance remains almost the same. Therefore, fixing α at 0.2 balances performance and efficiency while reducing the burden of hyperparameter tuning.

			0.05	0.1	0.2	0.5	0.8
FSDD	Best epoch	$ \mathcal{H} $	38.50	42.60	55.10	96.70	112.6
		#Synapse	4583	5411	6246	28024	34897
		F1-micro	.8840 \pm .0164	.9003 \pm .0131	.9142 \pm .0112	.9263 \pm .0084	.9267 \pm .0073
	Last epoch	$ \mathcal{H} $	24.30	38.70	55.00	96.70	113.0
		#Synapse	1768	2863	4312	28247	35507
		F1-micro	.8693 \pm .0181	.8963 \pm .0202	.9120 \pm .0142	.9220 \pm .0041	.9280 \pm .0066
Computers	Best epoch	$ \mathcal{H} $	0.00	0.00	0.00	116.4	127.6
		#Synapse	3701	4216	4376	111252	118374
		F1-micro	.7690 \pm .0044	.7680 \pm .0027	.7687 \pm .0032	.7671 \pm .0078	.7678 \pm .0092
	Last epoch	$ \mathcal{H} $	0.00	0.00	0.00	116.4	127.6
		#Synapse	3661	3718	3810	110891	118566
		F1-micro	.7660 \pm .0034	.7658 \pm .0040	.7677 \pm .0040	.7666 \pm .0020	.7674 \pm .0044

Table 10: The performance of SAGP when α is set to 0.05, 0.1, 0.2, 0.5, and 0.8.

5 limitations and Future direction

5.1 limitations

Due to the generality of the graph perceptron architecture, we employ a message-passing mechanism [14] for communication between neurons, with the scatter function serving as its core implementation. Assuming the number of synapses and feature dimensions is s and d , respectively, implementing message passing requires memory and time on the order of $\mathcal{O}(sd)$. When the graph topology is relatively dense, this cost is empirically 10 to 50 times higher than that of an MLP that performs only matrix multiplication. However, when using a SAGP, self-assembly capability allows us to achieve a highly sparse perceptron topology. Consequently, during the actual inference phase, the spatial and temporal costs of SAGP are only slightly higher than those of an MLP. Nonetheless, the convergence

229 of the self-assembly process requires more iterations, which we set to 50,000 by default—quite much
 230 more than standard settings.

	Training Time	GPU memory peak usage
Multi-layer connected GP	5.108 s	0.824 Gb
MLP	1.587 s	0.023 Gb

Table 11: A comparison of the computational cost between MLP and multi-layer connected GP on an RTX 4090 GPU. The number of training epochs is fixed at 1000.

231 5.2 Future direction

232 **Efficiency** As mentioned in the limitations, SAGP still has disadvantages in terms of computational
 233 cost and convergence speed compared to MLP. Therefore, researching solutions to improve SAGP’s
 234 efficiency is a key future direction.

235 **Assemble other models** SAGP has demonstrated how to integrate the biomimetic concept of self-
 236 assembly in the most basic deep model — MLP. However, for many current state-of-the-art deep
 237 learning models, constructing self-assembly mechanisms in other complex models will be an interest-
 238 ing research direction.

239 **Assembly rules** The self-assembly mechanism designed in SAGP is relatively simple, utilizing
 240 specific rules to implement neuron growth and apoptosis. Developing more reasonable assembly
 241 rules remains an area for further study.

References

- [1] Paolicelli, R. C., G. Bolasco, F. Pagani, et al. Synaptic pruning by microglia is necessary for normal brain development. *science*, 333(6048):1456–1458, 2011.
- [2] Han, S., H. Mao, W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- [3] Chang, X., Y. Li, S. Oymak, et al. Provable benefits of overparameterization in model compression: From double descent to pruning neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pages 6974–6983. 2021.
- [4] Cheng, H., M. Zhang, J. Q. Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [5] Huang, Z., N. Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320. 2018.
- [6] Zhao, C., B. Ni, J. Zhang, et al. Variational convolutional neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2780–2789. 2019.
- [7] Elsken, T., J. H. Metzen, F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.
- [8] Zoph, B. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.
- [9] Liu, Y., Y. Sun, B. Xue, et al. A survey on evolutionary neural architecture search. *IEEE transactions on neural networks and learning systems*, 34(2):550–570, 2021.
- [10] Liu, H., K. Simonyan, Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
- [11] Li, R., S. Wang, F. Zhu, et al. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32. 2018.
- [12] Jiang, B., Z. Zhang, D. Lin, et al. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11313–11320. 2019.
- [13] Gao, X., W. Hu, Z. Guo. Exploring structure-adaptive graph learning for robust semi-supervised classification. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.
- [14] Gilmer, J., S. S. Schoenholz, P. F. Riley, et al. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.
- [15] Piczak, K. J. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018. 2015.
- [16] McFee, B., C. Raffel, D. Liang, et al. librosa: Audio and music signal analysis in python. In *SciPy*, pages 18–24. 2015.
- [17] Shchur, O., M. Mumme, A. Bojchevski, et al. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.
- [18] Krizhevsky, A., G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [19] Ramachandran, P., B. Zoph, Q. V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.