Article

Semi-Supervised Classification via Hypergraph Convolutional Extreme Learning Machine

Zhewei Liu ¹, Zijia Zhang ¹ , Yaoming Cai ¹ , Yilin Miao ¹ and Zhikun Chen ^{2,*}

¹ School of Computer Science, China University of Geosciences, Wuhan 430078, China; zhewei.liu@cug.edu.cn (Z.L.); zhangzijia@cug.edu.cn (Z.Z.); caiyaom@cug.edu.cn (Y.C.); yilin.miao@cug.edu.cn (Y.M.)

² Guangxi Key Laboratory of Marine Disaster in the Beibu Gulf, Beibu Gulf University, Qinzhou 535011, China

* Correspondence: chzhikun@163.com; Tel.: +86-139-7779-8494

Abstract: Extreme Learning Machine (ELM) is characterized by simplicity, generalization ability, and computational efficiency. However, previous ELMs fail to consider the inherent high-order relationship among data points, resulting in being powerless on structured data and poor robustness on noise data. This paper presents a novel semi-supervised ELM, termed Hypergraph Convolutional ELM (HGCELM), based on using hypergraph convolution to extend ELM into the non-Euclidean domain. The method inherits all the advantages from ELM, and consists of a random hypergraph convolutional layer followed by a hypergraph convolutional regression layer, enabling it to model complex intraclass variations. We show that the traditional ELM is a special case of the HGCELM model in the regular Euclidean domain. Extensive experimental results show that HGCELM remarkably outperforms eight competitive methods on 26 classification benchmarks.

Keywords: graph convolutional network; extreme learning machine; semi-supervised learning; hypergraph learning

Citation: Liu, Z.; Zhang, Z.; Cai, Y.; Miao, Y.; Chen, Z. Semi-Supervised Classification via Hypergraph Convolutional Extreme Learning Machine. *Appl. Sci.* **2021**, *11*, 3867. <https://doi.org/10.3390/app11093867>

Academic Editor: Antonio Fernández

Received: 24 March 2021

Accepted: 22 April 2021

Published: 25 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Extreme Learning Machine (ELM) [1,2] was developed as a simple but effective learning model for classification and regression problems. As it is a special form of random vector functional-link network (RVFL) [3], ELM suggests that the hidden layer parameters of a neural network play an important role but does not need update during training [4,5]. Inspired by this, a large number of ELM variants have been proposed and widely applied to biomedical data analysis [6], computer vision [7], system modeling and prediction [8,9], and so on.

The key to the classic ELM is random mapping [10,11]. Despite helpful, the random mapping often suffers from poor robustness due to its randomness. To remedy the drawback, a number of works have been devoted to seeking the optimal hidden parameters. Wu et al. [12] presented a multi-objective evolutionary ELM to jointly optimize the structural risk and empirical risk of ELM. Similarly, many popular heuristic search methods, including differential evolution [13], are adopted for this purpose. However, the heuristic search is often time-consuming. Alternatively, kernel ELM (KELM) [4,14] implicitly implements the ELM hidden mapping in the reproducing kernel Hilbert space, which usually results in more stable performance. Considering that the random mapping also provides good diversity, it allows us to design an ensemble of ELMs [5,15]. It has been proven to be useful to improve the robustness of single ELM. Another tendency to enhance ELM is to make ELM deeper [16,17]. Compared with popular deep learning models, e.g., Convolutional Neural Networks (CNNs) [18], deep ELMs lack the potential to capture deep semantics from large-scale complex data.

The above-mentioned ELMs typically belong to supervised ELM, which desires adequate training samples for model training. For most general situations, where training

samples are significantly limited, the auxiliary information of unlabeled data is surprisingly helpful to improve generalization ability [19,20]. To this end, Semi-supervised ELM (SS-ELM) [21] has drawn increasing attention. SS-ELM adopts a frequently used scheme, i.e., graph Laplacian regularization, to incorporate manifold structure. To combine multiple graphs into SS-ELM, Yi et al. [22] introduced an adaptive multiple graph regularized SS-ELM approach. These efforts have demonstrated that structured information is beneficial for ELM. Nonetheless, the simple graph can only reveal the pairwise relationship of data points, thus ignoring the high-order relationship. Moreover, these methods basically work in the Euclidean domain, leading to indirect and insufficient use of structured information.

Recently, Graph Neural Networks (GNNs) [23] is emerging as a powerful technique for generalizing neural networks on graph-structured data. Technically, GNNs capture the dependence of graphs via message passing between the nodes of graphs [24]. Graph Convolutional Network (GCN) [25] is one of the most popular GNN models for node-level tasks [26] and graph-level tasks [27]. It has been proven that GCN is equivalent to a fixed low-pass filter [28,29] followed by a linear classifier. Thus, it often suffers from the over-smoothing problem when model depth increased. More recently, some works have attempted to extending GCN into hypergraph scenarios [30,31] so that the high-order relationship of data can be embedded. Despite considerable success, current GCNs highly rely on arduous model update via gradient descent.

Depending on the observation that GCN is equivalent to a multilayer perceptron multiplied by an adjacent matrix, [32], Zhang et al. proposed a Graph Convolutional ELM (GCELM) for semi-supervised learning. As a pioneer work, GCELM follows the same pipeline as ELM but extends it into the non-Euclidean domain, enabling ELM to deal with graph-structured data directly. However, GCELM mainly focuses on the graph embedding of pairwise connections, resulting in failure to capture intra-class variations between data points. This motivates us to further develop an enhanced version of HGCELM. We refer to the method as Hypergraph Convolutional Extreme Learning Machine (HGCELM). Specifically, HGCELM contains a random hypergraph convolutional layer that produces hypergraph embeddings, and a hypergraph regression layer with a closed-form solution.

To sum up, the main contributions of this paper are two-fold:

1. We propose a simple but effective hypergraph convolutional ELM, i.e., HGCELM, for semi-supervised classification. The HGCELM method not only inherits all the advantages from ELM but enables ELM to model the high-order relationship of data. The successful attempt signifies that structured information, especially high-order relationships, among data is important for ELM, which offers an alternative orientation for ELM representation learning.
2. We have shown that the traditional ELMs are the special cases of HGCELM on the Euclidean data. We conduct extensive experiments on 26 popular datasets for semi-supervised classification task. Comparisons with state-of-the-art methods demonstrate that the proposed GCELM can achieve superior performance.

The rest of the paper is organized as follows. In Section 2, we briefly review hypergraph learning, ELMs, and graph neural networks. In Section 3, we systematically introduce the framework, formulation, and implementation of the proposed method. Experimental evaluations and comparisons are presented in Section 4, followed by the conclusions and future work given in Section 5.

2. Preliminary and Related Work

2.1. Notations

Through this paper, symbols for vectors are boldface lowercase italics (e.g., x), symbols for matrices are boldface uppercase roman letters (e.g., $\mathbf{X}$), and symbols for scalars are italics (e.g., x_{ij}). Let $\mathcal{X} = \{x_i \in \mathbb{R}^m; y_i \in \mathbb{R}^C\}_{i=1}^N$ be the sample set consisting of N m -dimensional data points and C distinct classes, in which x_i and y_i denote i -th data point and its one-hot target vector, respectively. For clarity, we denote the labeled data and unlabeled data

with a subscript $\mathcal{L}$ and $\mathcal{U}$, e.g., $\mathbf{X}_{\mathcal{L}}$ and $\mathbf{X}_{\mathcal{U}}$ are the feature matrices for training and test, respectively. $\mathbf{I}_N$ signifies an identity matrix with the size of $N \times N$. The matrix Frobenius norm is defined as $\|\mathbf{X}\|_F = \left(\sum_{ij} |x_{ij}|^2\right)^{1/2}$. The main notations involved in this paper and corresponding definitions are given in Table 1.

Table 1. Important notations used in this paper and their definitions.

Notation	Definition
N	The number of data points.
m	The number of features.
C	The number of classes.
$\mathbf{X}$	The feature matrix of dataset, $\mathbb{R}^{N \times m}$.
$\mathbf{Y}$	The label matrix with one-hot encoding, $\mathbb{R}^{N \times C}$.
$N_{\mathcal{T}}$	The number of labeled samples.
$\mathcal{T}$	The labeled set.
$\mathcal{U}$	The unlabeled set.
Θ	The hidden layer parameter matrix, $\mathbb{R}^{m \times L}$.
β	The output layer parameter matrix, $\mathbb{R}^{L \times C}$.
$\mathbf{Z}$	The matrix of latent representation, $\mathbb{R}^{N \times L}$.
$\mathbf{Z}^{\dagger}$	The Moore–Penrose generalized inverse of matrix $\mathbf{Z}$.
L	The number of hidden neurons.
$\mathcal{V}$	The set of vertices in the hypergraph.
$\mathcal{E}$	The set of hyperedges in the hypergraph.
$\mathbf{W}$	The diagonal matrix of the hyperedge weights, $\mathbb{R}^{ \mathcal{E} \times \mathcal{E} }$.
$\mathcal{G}$	A hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$.
$\mathbf{H}$	The incidence matrix of the hypergraph, $\mathbb{R}^{N \times \mathcal{E} }$.
$d(v)$	The degree of the vertex v .
$\delta(e)$	The degree of the hyperedge e .
$\mathbf{D}_v$	The diagonal matrix of the vertex degrees, $\mathbb{R}^{N \times N}$.
$\mathbf{D}_e$	The diagonal matrix of the hyperedge degrees, $\mathbb{R}^{ \mathcal{E} \times \mathcal{E} }$.
$\mathbf{L}$	The hypergraph Laplacian matrix, $\mathbb{R}^{N \times N}$.

2.2. Hypergraph Preliminary

Hypergraph is the generalization of the simple graph, in which an edge can join any number of vertices. We refer to edges in a hypergraph as hyperedges. By contrast, we denote the regular graph in which each edge only connects two vertexes as a simple graph. A visual comparison between the simple graph and the hypergraph is illustrated in Figure 1. It can be seen that hypergraph can reveal a more complex data relationship, which is superior to the simple graph. For a learning task, hypergraph is usually used to represent the high-order intraclass variations among data points.

Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ be a hypergraph composed of a vertex set $v \in \mathcal{V}$ with the size of N , a hyperedge set $e \in \mathcal{E}$ with the size $|\mathcal{E}|$, and a weight set of hyperedge $\mathcal{W}$, where the weight of hyperedge e is indicated as $w(e)$. A hypergraph is often described as an incidence matrix $\mathbf{H} \in \mathbb{R}^{N \times |\mathcal{E}|}$ whose elements indicate whether a vertex joins a corresponding hyperedge (e.g., Figure 1b). Mathematically, the incidence matrix is defined by

$$h(v, e) = \begin{cases} 1 & v \in e \\ 0 & v \notin e. \end{cases} \quad (1)$$

The degree of a vertex v and the degree of a hyperedge e are given as follows, respectively

$$d(v) = \sum_{e \in \mathcal{E}} w(e) h(v, e), \quad (2)$$

$$\delta(e) = \sum_{v \in \mathcal{V}} h(v, e), \quad (3)$$

Figure 1. Visual comparison between a simple graph and a hypergraph. (a) An example of the simple graph (left) and its corresponding adjacent matrix **A** (right). (b) An example of hypergraph (left) and its corresponding incidence matrix **H** (right), where edges with the same color denote a hyperedge. Best view in color.

By analogy with the simple graph, spectral analysis can be used as an efficient tool for the analysis of hypergraph. The normalized hypergraph Laplacian matrix [33,34] is calculated by

$$\mathbf{L} = \mathbf{I} - \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^T \mathbf{D}_v^{-1/2}. \quad (4)$$

For a semi-supervised learning task, hypergraph is usually used by incorporating with an empirical error [35], as follows

$$\arg \min_{\mathbf{F}} \mathcal{R}_{emp}(\mathbf{F}) + \lambda \text{tr}(\mathbf{F}^T \mathbf{L} \mathbf{F}), \quad (5)$$

where $\mathcal{R}_{emp}(\mathbf{F})$ denotes the empirical error term over a problem-dependent prediction $\mathbf{F}$.

2.3. ELMs

The basic ELM can be interpreted as two components, i.e., random hidden mapping and ridge regression classifier. Formally, ELM's hidden layer can be expressed as

$$\mathbf{Z} = \sigma(\mathbf{X}\mathbf{\Theta} + \mathbf{b}), \quad (6)$$

where $\mathbf{Z}$ is the hidden layer output matrix parameterized by hidden weight matrix $\mathbf{\Theta}$ and bias vector $\mathbf{b}$, σ denotes a nonlinear activation function such as Sigmoid. In the second stage, ELM computes prediction by

$$\mathbf{Y} = \mathbf{Z}\mathbf{\beta}. \quad (7)$$

Here, $\mathbf{\beta}$ is the output weight matrix. Since $\mathbf{Z}$ is known to the output layer, Equation (7) essentially is a least-squares optimization problem and can be solved as

$$\mathbf{\beta} = \mathbf{Z}^\dagger \mathbf{Y}_{\mathcal{T}}. \quad (8)$$

ELM avoids iterative parameters tuning, thus significantly faster than gradient descent-based neural networks. SS-ELM is the semi-supervised version of ELM by introducing a graph Laplacian regularization term [21]. Its formulation is given by

$$\arg \min_{\mathbf{\beta}} \|\mathbf{Z}_{\mathcal{T}} \mathbf{\beta} - \mathbf{Y}_{\mathcal{T}}\|_F^2 + \lambda \text{tr}(\mathbf{\beta}^T \mathbf{Z}^T \mathbf{L} \mathbf{Z} \mathbf{\beta}). \quad (9)$$

SS-ELM also has a closed-form solution. It should be noted that the graph structure information is considered in SS-ELM but it essentially works in the regular Euclidean domain.

2.4. GCNs

There is an increasing interest in generalizing convolutions to the graph domain [23]. The recent development of GNNs allows us to efficiently approximate convolution on graph-structured data. GNNs can typically divide into two categories [23,28]: spectral convolutions [36], which perform convolution by transforming node representations into the spectral domain using the graph Fourier transform or its extensions, and spatial convolutions [37], which perform convolution by sampling from neighborhood signals.

In [25], Kift et al. developed a Graph Convolutional Network (GCN) by simplifying the spectral convolution with the 1th-order Chebyshev polynomials and setting the largest eigenvalue of the normalized graph Laplacian to be 2. Formally, GCN defines spectral convolution over a graph as follows

$$\mathbf{Z} = \sigma(\tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{X} \Theta). \quad (10)$$

Here, $\tilde{\mathbf{A}} = \mathbf{I}_N + \mathbf{A}$ is the so-called augmented normalized adjacency, $\tilde{\mathbf{D}}$ is given by $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. However, the current GNNs highly rely on gradient descend optimizers, which is often time-consuming and prone to suffering from a locally optimal solution. To overcome the shortcomings, Zhang et al. [32] proposed a randomization-based GCN (i.e., GCELM) by combining the advantages of ELM with GCN. Instead of updating all trainable parameters, GCELM employs a random graph convolutional layer and keeps it fixed. Thus, this allows GCELM to calculate the closed-form solution for the training phase, and further resulting in a faster learning speed.

3. HGCELM

In Figure 2, we provide an illustration of the proposed HGCELM framework. The framework first constructs a hypergraph from the given dataset, and then feeds it into the HGCELM model consisting of a random hypergraph convolutional layer and a hypergraph convolutional regression layer. The details for HGCELM are introduced as follows.

Figure 2. The proposed HGCELM framework.

3.1. Hypergraph Construction

We represent the high-order relationship of a dataset by constructing a hypergraph $\mathcal{G}$. For this purpose, each data point x_i is treated as a vertex v , and furthermore, we consider that x_i is the center vertex and its k nearest neighbors are associated with the hyperedge e . As a result, each hyperedge connects $k + 1$ vertices. The incidence matrix of the hypergraph is defined by

$$h(v_i, e_j) = \begin{cases} 1 & x_j \in \mathcal{N}_k(x_i) \\ 0 & \text{otherwise} \end{cases}. \quad (11)$$

The degree of the vertex set $\mathcal{V}$ and the degree of the hyperedge set $\mathcal{E}$ can be expressed in diagonal matrix forms, i.e., $\mathbf{D}_v$ and $\mathbf{D}_e$. There are various ways to assign the weights for hyperedges, such as the sum of similarities of vertices within a hyperedge [34]. In this paper, we view all hyperedges as equal weights and thus can be indicated as an identity matrix

$\mathbf{I} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$. The samples belonging to the same class often have a higher probability of being assigned to $\mathcal{N}_k$. Therefore, it is reasonable to use hypergraph to describe the intraclass variations of data.

3.2. Random Hypergraph Convolution

Inspired by the previous GCELM [32], we propose a novel ELM mapping, called random hypergraph convolution (RGC), to incorporate the high-order relationship among data into the feature mapping. We define the random hypergraph convolution following that proposed in [30,31], except that ours does not need to be updated iteratively. It is expressed as follows:

$$\mathbf{Z} = \sigma(\mathbf{S}\mathbf{X}\mathbf{\Theta}), \quad (12)$$

where $\mathbf{S} = \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{D}_e^{-1} \mathbf{H}^{-1} \mathbf{D}_v^{-1/2}$ referring to as augmented normalized incidence matrix by imposing a symmetric normalization. It should be noted that $\mathbf{S}$ incorporates structured information of data and can be precomputed during implementation. Thus, the random hypergraph convolution enables ELM to embed the high-order information. Following the ELM's theory, the filter parameters of the random hypergraph convolution, $\mathbf{\Theta}$, is randomly generated under a specific probability distribution, e.g., Gaussian distribution $\Theta_{ij} \sim \mathcal{N}(0, 1)$.

3.3. Hypergraph Convolutional Regression

Based on the hidden hypergraph embedding $\mathbf{Z}$, we use a hypergraph convolutional regression layer to predict labels. Formally, the layer can be written as:

$$\mathbf{Y} = \mathbf{S}\mathbf{Z}\boldsymbol{\beta}. \quad (13)$$

To solve $\boldsymbol{\beta}$, the above equation can be rewritten as the following ridge regression problem:

$$\arg \min_{\boldsymbol{\beta}^*} \|\boldsymbol{\beta}\|_F, s.t., \tilde{\mathbf{Y}} = \mathbf{S}\mathbf{Z}\boldsymbol{\beta}. \quad (14)$$

Here, $\tilde{\mathbf{Y}} = [\mathbf{Y}_{\mathcal{T}}; \mathbf{Y}_{\mathcal{U}}]$ is an augmented training target matrix. Since $\mathbf{Y}_{\mathcal{U}}$ is invisible during training stage, thus it is set to be a 0 matrix. Let $\mathbf{M}$ be a diagonal mask matrix with its first $N_{\mathcal{T}}$ diagonal elements $M_{ii} = 1$ and the rest equal to 0. We further rewrite Equation (14) as

$$\arg \min_{\boldsymbol{\beta}^*} \frac{1}{2} \|\mathbf{M}(\tilde{\mathbf{Y}} - \mathbf{S}\mathbf{Z}\boldsymbol{\beta})\|_F^2 + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_F^2. \quad (15)$$

It is easy to prove that Equation (15) holds a optimal solution and its closed form is expressed

$$\boldsymbol{\beta}^* = \left(\mathbf{Z}^T \mathbf{S}^T \mathbf{M} \mathbf{S} \mathbf{Z} + \lambda \mathbf{I} \right)^{-1} \mathbf{Z}^T \mathbf{S}^T \mathbf{M} \tilde{\mathbf{Y}}. \quad (16)$$

The labels of the unlabeled data points can be determined depended on the obtained $\boldsymbol{\beta}^*$, which is given by

$$\tilde{\mathbf{Y}}_{\mathcal{U}} = [\mathbf{S}\mathbf{Z}]_{\mathcal{U}} \boldsymbol{\beta}^*. \quad (17)$$

We show the overall learning steps of the proposed HGCELM in Algorithm 1. Since there is no iteration in HGCELM, it would be computationally efficient for training such model.

Algorithm 1: HGCELM

Input: Dataset $\mathbf{X}$, training labels $\mathbf{Y}_T$, hyper-parameters λ and L .

- 1 Construct hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ by Equation (11);
- 2 Generate Θ using the standard normal distribution;
- 3 Calculate random hypergraph embedding: $\mathbf{Z} = \sigma(\mathbf{S}\mathbf{X}\Theta)$;
- 4 Solve hypergraph conv regression: $\beta^* = (\mathbf{Z}^T \mathbf{S}^T \mathbf{M} \mathbf{S} \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^T \mathbf{S}^T \mathbf{M} \tilde{\mathbf{Y}}$;
- 5 Predict test levels by Equation (17);

Output: $\tilde{\mathbf{Y}}_U$.

3.4. Computation Complexity and Connection to Existing Methods

We give a theoretical analysis for the computation complexity of our method. Ignoring the hypergraph generation procedure and assuming $L \ll N$, the time complexity of HGCELM is approximately $O(N^3)$. Compared with ELM, HGCELM has a higher time complexity, which is determined by the hypergraph convolution. Fortunately, the augmented normalized incidence matrix can be precomputed and is sparse. Thus, the hypergraph embedding can be efficiently implemented. HGCELM is more efficient than GCN since it does not use iteration.

Our HGCELM is closely related to the classic ELMs. First, HGCELM maintains the advantages of ELMs, i.e., fast learning speed achieved by a closed-form solution. Second, our HGCELM will degenerate to be the classic ELM by casting away $\mathbf{S}$. Consequently, the classic ELM is a special case of the HGCELM model in the regular Euclidean domain. Furthermore, our HGCELM remedies the drawback that traditional ELMs cannot deal with structured data.

4. Results and Discussion

In this section, we conduct experiments to verify the effectiveness of the proposed HGCELM.

*4.1. Experimental Configurations**4.1.1. Datasets*

We evaluate our method on 26 widely used classification datasets taken from University of California at Irvine (UCI) repository. For each dataset, we scale its feature values into $[0, 1]$ using a min-max standardization method and randomly choose 5 samples per class as the training set and the rest as the test set. The details of these datasets are reported in Table 2.

Table 2. Details of the benchmark datasets.

Dataset	#Classes	#Instance	#Feature	#Train	#Test	Dataset	#Classes	#Instance	#Feature	#Train	#Test
austra	2	680	14	10	680	weather	2	22	4	10	12
australian	2	690	14	70	620	Wine	3	178	13	19	159
breast	2	277	9	29	248	X8D5K	5	1000	8	100	900
cleve	2	296	13	30	266	zoo	7	101	16	13	88
diabetes	2	768	8	78	690	cloud	2	1024	10	103	921
dnatest	3	1186	180	120	1066	bupa	2	345	6	10	335
german	2	1000	24	100	900	air	3	359	64	37	322
heart	2	270	13	27	243	segmentation	7	210	18	21	189
ionosphere	2	351	32	10	341	pima In. D.	2	768	8	10	758
iris	3	150	4	15	135	Xinp	3	178	13	19	159
sonar	2	208	60	10	198	wdbc	2	569	30	58	511
vote	2	435	16	10	425	ecoli label	2	335	2	10	325
WBC	2	683	9	10	673	appendicitis	2	106	7	12	94

4.1.2. Baseline Methods

We compare our method with eight baselines including the basic ELM [2], KELM [38], SS-ELM [21], Transductive Support Vector Machine (TSVM) [19], Self-training Semi-supervised ELM (ST-ELM) [39], Laplacian Support Vector Machine (LapSVM) [20], GCELM [32], and GCN [25]. For a fair comparison with previous works, we set 50 hidden neurons for these methods that contain hidden layer, i.e., GCELM, ELM, SS-ELM, ST-ELM, and GCN. The hyperparameter settings of these methods are provided in Table 3. We implement all the methods with Python 3.5 running on an Intel i5-6500 3.20 GHz CPU with 8.00 GB RAM.

Table 3. Hyperparameter settings of different methods.

Method	Hyper-Parameters
HGCELM (Ours)	$L = 50, \lambda = 10^{-3}$
GCELM [32]	$L = 50, \lambda = 10^{-3}$
ELM [2]	$L = 50, \lambda = 10^{-5}$
KELM [38]	$\lambda = 10^{-5}$
SS-ELM [21]	$L = 50, \lambda_1 = 10^{-5}, \lambda_2 = 10^{-2}$
TSVM [19]	$C = 10^3, C^* = 1.5, \text{kernel} = \text{RBF}$
ST-ELM [39]	$L = 50, \lambda_1 = \lambda_2 = 10^{-5}$
LapSVM [20]	$\lambda_1 = 0.001, \lambda_2 = 0.05$
GCN [25]	$L = 50, lr = 0.02, \text{epoch} = 100$

4.2. Qualitative Study

4.2.1. Comparisons with Baselines

Table 4 reports the test accuracy of different methods over the 26 datasets. Each method is evaluated for 30 independent runs. At the bottom of the table, two summative metrics are calculated, that is, an arithmetic mean of test accuracy over the 26 datasets and a statistical value for how many datasets our HGCELM win/tie/loss (W/T/L) than a competitor. On the basis of these results, we conclude that the proposed HGCELM method consistently outperforms the other baselines. Compared with GCELM, HGCELM achieves 2.29% improvement in terms of the averaged accuracy with a lower standard deviation. It demonstrates that hypergraph is more helpful for semi-supervised learning. This is because the hypergraph convolution aims to embed the intraclass variations, while the graph convolution focuses only on the pairwise relationship. There is also a significant improvement comparing HGCELM against the classic semi-supervised methods (i.e., SS-ELM, TSVM, ST-ELM, and LapSVM). Specifically, our method wins on 25, 26, 26, and 24 datasets, respectively. Notably, ELM and KELM are purely supervised, thus their classification accuracy is lower than most semi-supervised methods caused by the limited training samples. Despite being optimized, GCN cannot achieve better test accuracy than our HGCELM. It further signifies the effectiveness and reliability of our proposal.

4.2.2. Performance with Varying Training Size

To investigate the performance of HGCELM, we visualize the test accuracy under varying training sizes. In this experiment, we increase the training size from 5% to 50% and present the corresponding results of 30 evaluations with a box plot. As shown in Figure 3, the test accuracy of HGCELM tends to increase when the training sample increased. Meanwhile, HGCELM becomes more robust and more stable. When the training size is larger than 25%, HGCELM's accuracy closes to its highest accuracy and tends to be convergent. For the Iris dataset, HGCELM can achieve a remarkable accuracy (higher than 90%) using only 5 samples per class (10%). It means that HGCELM has the ability to explore and make better use of the useful information from unlabeled data.

Table 4. Experimental results of different algorithms on 26 different datasets (MEAN $\pm$ STD, preferably in **BOLD**).

Data Sets	HGCELM	GCELM	SS-ELM	ELM	KELM	TSVM	ST-ELM	LapSVM	GCN
austra	82.57 $\pm$ 3.72	80.78 $\pm$ 2.99	76.24 $\pm$ 8.14	73.32 $\pm$ 10.48	79.47 $\pm$ 4.12	77.43 $\pm$ 10.59	75.17 $\pm$ 9.99	55.70 $\pm$ 0.21	71.38 $\pm$ 8.36
australian	81.90 $\pm$ 4.73	78.46 $\pm$ 4.82	78.10 $\pm$ 6.24	71.35 $\pm$ 11.25	79.32 $\pm$ 6.11	75.38 $\pm$ 10.81	78.57 $\pm$ 9.56	75.79 $\pm$ 6.61	75.36 $\pm$ 7.56
breast	66.52 $\pm$ 10.39	61.61 $\pm$ 8.61	55.30 $\pm$ 7.67	52.00 $\pm$ 7.38	55.56 $\pm$ 8.97	55.77 $\pm$ 6.78	53.18 $\pm$ 10.68	71.54 $\pm$ 0.00	60.17 $\pm$ 6.18
cleve	74.51 $\pm$ 2.09	72.90 $\pm$ 6.42	72.88 $\pm$ 5.30	68.95 $\pm$ 7.31	72.38 $\pm$ 3.79	72.80 $\pm$ 5.70	68.85 $\pm$ 8.30	54.20 $\pm$ 0.00	70.17 $\pm$ 6.14
diabetes	69.22 $\pm$ 3.75	70.10 $\pm$ 3.09	66.27 $\pm$ 8.92	57.31 $\pm$ 8.51	65.71 $\pm$ 5.64	64.97 $\pm$ 5.26	58.43 $\pm$ 8.56	65.30 $\pm$ 0.06	62.93 $\pm$ 6.62
dnatest	78.55 $\pm$ 2.34	67.87 $\pm$ 5.03	51.60 $\pm$ 4.63	50.77 $\pm$ 4.79	50.81 $\pm$ 4.26	60.99 $\pm$ 2.94	42.86 $\pm$ 8.51	48.27 $\pm$ 4.10	59.33 $\pm$ 3.95
german	67.38 $\pm$ 2.93	67.10 $\pm$ 10.01	54.48 $\pm$ 5.35	57.95 $\pm$ 9.42	51.07 $\pm$ 7.48	59.52 $\pm$ 6.89	59.64 $\pm$ 7.63	70.21 $\pm$ 0.03	59.54 $\pm$ 5.32
heart	78.50 $\pm$ 2.83	76.53 $\pm$ 3.63	72.54 $\pm$ 6.06	69.15 $\pm$ 9.07	73.25 $\pm$ 8.99	70.23 $\pm$ 7.77	67.21 $\pm$ 8.62	55.77 $\pm$ 0.00	70.00 $\pm$ 8.58
ionosphere	90.32 $\pm$ 0.00	81.07 $\pm$ 7.51	75.98 $\pm$ 4.31	74.65 $\pm$ 8.96	75.57 $\pm$ 6.18	77.05 $\pm$ 5.43	77.20 $\pm$ 5.68	64.52 $\pm$ 0.00	71.32 $\pm$ 8.21
iris	95.56 $\pm$ 1.72	94.32 $\pm$ 3.20	80.04 $\pm$ 4.79	80.70 $\pm$ 10.76	93.04 $\pm$ 3.88	93.00 $\pm$ 3.70	81.04 $\pm$ 8.36	92.70 $\pm$ 2.58	91.19 $\pm$ 3.61
sonar	67.93 $\pm$ 4.60	67.44 $\pm$ 6.64	63.28 $\pm$ 5.10	64.66 $\pm$ 6.37	62.56 $\pm$ 5.37	66.41 $\pm$ 5.27	62.21 $\pm$ 5.10	46.46 $\pm$ 0.00	67.22 $\pm$ 7.09
vote	90.12 $\pm$ 2.60	84.19 $\pm$ 5.72	83.75 $\pm$ 3.49	84.39 $\pm$ 6.17	87.98 $\pm$ 1.97	87.97 $\pm$ 3.59	87.65 $\pm$ 6.21	38.35 $\pm$ 0.00	86.62 $\pm$ 3.87
WBC	97.21 $\pm$ 0.27	96.53 $\pm$ 0.74	92.30 $\pm$ 1.83	89.35 $\pm$ 5.63	95.30 $\pm$ 2.78	95.40 $\pm$ 1.90	92.14 $\pm$ 3.50	65.26 $\pm$ 0.07	95.93 $\pm$ 2.28
weather	84.17 $\pm$ 6.92	76.94 $\pm$ 14.38	69.72 $\pm$ 15.14	73.61 $\pm$ 12.74	69.17 $\pm$ 11.21	76.39 $\pm$ 14.29	70.00 $\pm$ 15.00	41.67 $\pm$ 0.00	50.83 $\pm$ 11.66
Wine	96.38 $\pm$ 1.38	91.25 $\pm$ 2.28	89.37 $\pm$ 3.33	79.75 $\pm$ 6.72	90.31 $\pm$ 3.92	90.51 $\pm$ 3.34	79.59 $\pm$ 7.68	88.45 $\pm$ 1.60	90.84 $\pm$ 2.79
X8D5K	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00	96.93 $\pm$ 3.07	99.96 $\pm$ 0.06	99.98 $\pm$ 0.05	99.93 $\pm$ 0.21	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00
zoo	100.00 $\pm$ 0.00	99.22 $\pm$ 1.32	98.02 $\pm$ 1.45	98.18 $\pm$ 1.51	98.65 $\pm$ 1.60	99.48 $\pm$ 1.09	99.69 $\pm$ 0.94	99.95 $\pm$ 0.28	100.00 $\pm$ 0.00
cloud	90.38 $\pm$ 2.92	90.18 $\pm$ 3.68	89.92 $\pm$ 3.22	73.38 $\pm$ 10.32	88.48 $\pm$ 4.36	89.71 $\pm$ 4.65	75.95 $\pm$ 9.54	81.32 $\pm$ 4.86	88.10 $\pm$ 3.90
bupa	57.37 $\pm$ 2.72	56.33 $\pm$ 6.93	52.55 $\pm$ 4.01	54.07 $\pm$ 6.63	56.15 $\pm$ 4.98	55.95 $\pm$ 5.62	55.41 $\pm$ 5.85	41.79 $\pm$ 0.00	51.72 $\pm$ 3.71
air	81.72 $\pm$ 6.33	71.30 $\pm$ 6.25	66.52 $\pm$ 4.50	67.44 $\pm$ 5.83	70.89 $\pm$ 4.18	70.68 $\pm$ 6.38	69.53 $\pm$ 7.55	72.49 $\pm$ 7.63	79.65 $\pm$ 5.78
segmentation	84.51 $\pm$ 2.00	83.22 $\pm$ 3.14	78.91 $\pm$ 3.42	65.71 $\pm$ 7.74	83.01 $\pm$ 3.22	81.85 $\pm$ 4.53	67.05 $\pm$ 8.36	80.84 $\pm$ 3.51	80.74 $\pm$ 3.61
pima In. D.	68.97 $\pm$ 3.19	68.45 $\pm$ 5.55	67.28 $\pm$ 7.67	58.43 $\pm$ 7.06	63.52 $\pm$ 6.78	64.84 $\pm$ 8.34	62.35 $\pm$ 5.65	35.29 $\pm$ 1.03	65.47 $\pm$ 4.70
Xinp	97.33 $\pm$ 0.85	93.48 $\pm$ 1.95	93.04 $\pm$ 2.31	80.31 $\pm$ 6.64	92.76 $\pm$ 3.00	93.90 $\pm$ 1.76	81.56 $\pm$ 8.59	92.45 $\pm$ 1.32	94.97 $\pm$ 1.82
wdbc	94.44 $\pm$ 1.42	92.13 $\pm$ 0.16	88.94 $\pm$ 4.59	81.20 $\pm$ 8.63	91.72 $\pm$ 4.20	91.15 $\pm$ 4.02	89.51 $\pm$ 5.39	62.97 $\pm$ 0.00	90.86 $\pm$ 2.44
ecoli_label	67.45 $\pm$ 1.78	64.46 $\pm$ 7.16	67.09 $\pm$ 2.29	57.92 $\pm$ 11.95	61.17 $\pm$ 12.58	58.92 $\pm$ 13.97	63.28 $\pm$ 9.71	61.80 $\pm$ 8.44	63.14 $\pm$ 8.21
appendicitis	86.98 $\pm$ 2.25	85.83 $\pm$ 2.80	73.33 $\pm$ 14.10	58.12 $\pm$ 11.00	69.06 $\pm$ 12.88	63.70 $\pm$ 15.29	57.86 $\pm$ 15.23	16.67 $\pm$ 0.00	76.15 $\pm$ 10.68
Average	82.69 $\pm$ 2.84	80.4 $\pm$ 4.61	75.64 $\pm$ 5.14	71.26 $\pm$ 7.78	76.9 $\pm$ 5.29	77.53 $\pm$ 5.94	72.62 $\pm$ 7.68	65.9 $\pm$ 1.57	76.63 $\pm$ 5.12
W/T/L	–	24/1/1	25/1/0	26/0/0	26/0/0	26/0/0	26/0/0	24/1/1	24/2/0

4.2.3. Analysis on Decision Boundaries

To provide an intuitive understanding of the superiority of our HGCELM, we visualize the decision boundaries of ELM, SS-ELM, GCELM, and HGCELM. In Figure 4, we synthesize three types of representative data distributions, i.e., linear separable data (the first row), linear inseparable data with half circles (the second row), and linear inseparable data with concentric circles (the third row), each of which contains 100 samples with two classes. The complexity of these datasets gradually increases from top to bottom. In this experiment, we select 10 samples for training and initialize each classifier with 10 hidden neurons. Owing to the simplicity, all the four classifiers can correctly find a reliable decision boundary on the linear separable dataset. However, for the first linear inseparable dataset, semi-supervised methods (i.e., HGCELM, GCELM, and SS-ELM) can obtain more generalized decision boundaries than the supervised method (i.e., ELM). However, spectral convolution-based methods (HGCELM and GCELM) are superior to the graph Laplacian-based method (SS-ELM) in terms of decision boundary, as well as test accuracy. From the second linear inseparable dataset, HGCELM can accurately classify the data points with a better decision boundary, while the other three methods show relatively poorer ones. In particular, ELM fails to work on this dataset under the same experiment settings. This is because the ELM cannot use the unlabeled samples, making it more likely to overfitting the limited training set. Although SS-ELM shows same accuracy as HGCELM, its decision boundary cannot separate two circles. We can naturally conclude that HGCELM has a better generalization ability than the other methods. The ability is benefited from the fact that the high-order relationship among the whole data points can improve the quality of decision making. It should be noticed that due to the geometric property of data distribution, the Euclidean-distance-based graph construction strategy would result in an inappropriate graph structure on the linear inseparable datasets. Therefore, a small neighbor size is desired when constructing a graph or hypergraph for these datasets.

Figure 3. Test accuracy of HGCELM with varying labeled sample sizes on (a) Iris, (b) Wine, (c) WDBC, and (d) Segmentation datasets.

Figure 4. Visualization of the decision boundaries for ELM, SS-ELM, GCELM, and HGCELM over the three synthetic data distribution. This figure is best viewed in color.

4.3. Parameter Sensitivity Study

Two parameter sensitivity studies are carried out to further explore the robustness of the proposed method.

4.3.1. Impact of Hidden Neurons

Figure 5 shows the impact of different hidden neurons. We compare our method with five baselines that require setting hidden neurons. As seen from Figure 5a–d, all the compared methods can achieve better performance when using more hidden neurons on all the datasets. Nevertheless, our proposed HGCELM enjoys a more competitive

edge than the other methods. In particular, when our method uses more than 20 hidden neurons. Notice that GCN can guarantee relatively better accuracy than other competitors when using small hidden neurons. This is because the filter parameters of GCN are fully optimized. By contrast, ELMs require larger hidden neurons, which is repeatedly demonstrated by many previous works. From Figure 5c,d, the classic ELM suffers from overfitting (accuracy declined with hidden neurons) caused by the inadequate training data. The problem is overcome in other semi-supervised ELMs including our HGCELM.

4.3.2. Impact of λ and k

In this experiment, we further explore the impact of the other two important hyper-parameters, i.e., λ and k . Figure 6a–d shows the results by grid searching. Here, we set $\lambda = \{10^{-7}, 10^{-6}, \dots, 10^2\}$ and $k = \{3, 6, \dots, 30\}$, respectively. We can observe two tendencies from the results. First, to obtain a better performance, the regularization coefficient λ should be set to be less than 1. Theoretically, using a larger λ will punish HGCELM to be a compact model but also resulting in an increased risk of under-fitting the training data. Second, a larger neighbor size k will guarantee a better classification accuracy in general. However, there will exist noise samples in hyperedges if k becomes too large. We provide the suitable ranges of the two parameters as $\lambda \in [10^{-3}, 10^{-1}]$ and $k \in [5, 10]$.

Figure 5. Classification performance comparison under different number of hidden neurons on (a) Iris, (b) Wine, (c) WDBC, and (d) Segmentation datasets.

Figure 6. Parameter sensitivity analysis for λ and k on (a) Iris, (b) Wine, (c) WDBC, and (d) Segmentation datasets. The x axis is denoted as $\log_{10}(\lambda)$.

5. Conclusions

We have proposed a novel ELM called HGCELM for semi-supervised classification. The idea behind HGCELM is to combine hypergraph convolution with ELM so that embedding the high-order relationship of data. The resulting model extends ELM into the non-Euclidian domain and endows ELM with the capability of modeling structure data. The proposed HGCELM is characterized by a very light computational burden and good generalization ability, making it easy to implement and apply in practice. Extensive experiments on 26 datasets demonstrate that HGCELM is superior to many existing methods. The successful attempt clues a promising avenue for designing randomized neural networks and graph neural networks.

Author Contributions: Conceptualization, Z.L. and Y.C.; Data curation, Z.L.; Formal analysis, Z.Z.; Investigation, Y.M.; Methodology, Z.L. and Y.C.; Project administration, Y.C. and Z.C.; Software, Z.Z.; Supervision, Z.C.; Validation, Y.M.; Writing—original draft, Z.L.; Writing—review & editing, Z.Z. and Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Fundamental Research Funds for National University, China University of Geosciences (Wuhan) under grant 1910491T06, in part by Guangxi Natural Science Fund General Project under grant 2021GXNSFAA075029, and Guangxi Science and Technology Base and Talent Project under grant Guike AD20159036.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The UCI datasets used in this work are available from <https://archive.ics.uci.edu/ml/datasets.php> accessed on 14 March 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Huang, G.; Huang, G.B.; Song, S.; You, K. Trends in extreme learning machines: A review. *Neural Netw.* **2015**, *61*, 32–48. [[CrossRef](#)] [[PubMed](#)]
- Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [[CrossRef](#)]
- Zhang, Y.; Wu, J.; Cai, Z.; Du, B.; Yu, P.S. An unsupervised parameter learning model for RVFL neural network. *Neural Netw.* **2019**, *112*, 85–97. [[CrossRef](#)]
- Huang, G.B.; Zhou, H.; Ding, X.; Zhang, R. Extreme Learning Machine for Regression and Multiclass Classification. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2012**, *42*, 513–529. [[CrossRef](#)] [[PubMed](#)]
- Cai, Y.; Liu, X.; Zhang, Y.; Cai, Z. Hierarchical ensemble of extreme learning machine. *Pattern Recognit. Lett.* **2018**, *116*, 101–106. [[CrossRef](#)]
- Song, Y.; Crowcroft, J.; Zhang, J. Automatic epileptic seizure detection in EEGs based on optimized sample entropy and extreme learning machine. *J. Neurosci. Methods* **2012**, *210*, 132–146. [[CrossRef](#)]
- Zeng, Y.; Xu, X.; Shen, D.; Fang, Y.; Xiao, Z. Traffic Sign Recognition Using Kernel Extreme Learning Machines with Deep Perceptual Features. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1647–1653.
- Xu, Y.; Dong, Z.Y.; Zhao, J.H.; Zhang, P.; Wong, K.P. A Reliable Intelligent System for Real-Time Dynamic Security Assessment of Power Systems. *IEEE Trans. Power Syst.* **2012**, *27*, 1253–1263. [[CrossRef](#)]
- Chen, X.; Dong, Z.Y.; Meng, K.; Xu, Y.; Wong, K.P.; Ngan, H.W. Electricity Price Forecasting With Extreme Learning Machine and Bootstrapping. *IEEE Trans. Power Syst.* **2012**, *27*, 2055–2062. [[CrossRef](#)]
- Zhang, L.; Suganthan, P. A survey of randomized algorithms for training neural networks. *Inf. Sci.* **2016**, *364–365*, 146–155. [[CrossRef](#)]
- Cao, W.; Wang, X.; Ming, Z.; Gao, J. A review on neural networks with random weights. *Neurocomputing* **2018**, *275*, 278–287. [[CrossRef](#)]
- Wu, Y.; Zhang, Y.; Liu, X.; Cai, Z.; Cai, Y. A multiobjective optimization-based sparse extreme learning machine algorithm. *Neurocomputing* **2018**, *317*, 88–100. [[CrossRef](#)]
- Zhang, Y.; Wu, J.; Cai, Z.; Zhang, P.; Chen, L. Memetic Extreme Learning Machine. *Pattern Recognit.* **2016**, *58*, 135–148. [[CrossRef](#)]
- Xiao, L.; Shao, W.; Jin, F.; Wu, Z. A self-adaptive kernel extreme learning machine for short-term wind speed forecasting. *Appl. Soft Comput.* **2021**, *99*, 106917. [[CrossRef](#)]
- Wang, X.B.; Zhang, X.; Li, Z.; Wu, J. Ensemble extreme learning machines for compound-fault diagnosis of rotating machinery. *Knowl. Based Syst.* **2020**, *188*, 105012. [[CrossRef](#)]
- Huang, G.B.; Bai, Z.; Kasun, L.L.C.; Vong, C.M. Local Receptive Fields Based Extreme Learning Machine. *IEEE Comput. Intell. Mag.* **2015**, *10*, 18–29. [[CrossRef](#)]
- Cai, Y.; Zhang, Z.; Yan, Q.; Zhang, D.; Banu, M.J. Densely connected convolutional extreme learning machine for hyperspectral image classification. *Neurocomputing* **2021**, *434*, 21–32. [[CrossRef](#)]
- Cai, Y.; Liu, X.; Cai, Z. BS-Nets: An End-to-End Framework for Band Selection of Hyperspectral Image. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 1969–1984. [[CrossRef](#)]
- Joachims, T. Transductive inference for text classification using support vector machines. In Proceedings of the ICML, Bled, Slovenia, 27–30 June 1999; Volume 99, pp. 200–209.
- Belkin, M.; Niyogi, P.; Sindhvani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.* **2006**, *7*, 2399–2434.
- Huang, G.; Song, S.; Gupta, J.N.D.; Wu, C. Semi-Supervised and Unsupervised Extreme Learning Machines. *IEEE Trans. Cybern.* **2014**, *44*, 2405–2417. [[CrossRef](#)] [[PubMed](#)]
- Yi, Y.; Qiao, S.; Zhou, W.; Zheng, C.; Liu, Q.; Wang, J. Adaptive multiple graph regularized semi-supervised extreme learning machine. *Soft Comput.* **2018**, *22*, 3545–3562. [[CrossRef](#)]
- Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24. [[CrossRef](#)]
- Garg, V.; Jegelka, S.; Jaakkola, T. Generalization and Representational Limits of Graph Neural Networks. In Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, 12–18 July 2020; Daumé, H., III, Singh, A., Eds.; Volume 119, pp. 3419–3430.
- Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
- Cai, Y.; Zhang, Z.; Cai, Z.; Liu, X.; Jiang, X.; Yan, Q. Graph Convolutional Subspace Clustering: A Robust Subspace Clustering Framework for Hyperspectral Image. *IEEE Trans. Geosci. Remote Sens.* **2020**, *59*, 4191–4202. [[CrossRef](#)]
- Lee, J.; Lee, I.; Kang, J. Self-Attention Graph Pooling. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Chaudhuri, K., Salakhutdinov, R., Eds.; Volume 97, pp. 3734–3743.
- Li, Q.; Han, Z.; Wu, X.M. Deeper insights into graph convolutional networks for semi-supervised learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
- Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; Weinberger, K. Simplifying Graph Convolutional Networks. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Chaudhuri, K., Salakhutdinov, R., Eds.; Volume 97, pp. 6861–6871.

30. Feng, Y.; You, H.; Zhang, Z.; Ji, R.; Gao, Y. Hypergraph Neural Networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 3558–3565.
31. Bai, S.; Zhang, F.; Torr, P.H. Hypergraph convolution and hypergraph attention. *Pattern Recognit.* **2021**, *110*, 107637. [[CrossRef](#)]
32. Zhang, Z.; Cai, Y.; Gong, W.; Liu, X.; Cai, Z. Graph Convolutional Extreme Learning Machine. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [[CrossRef](#)]
33. Schölkopf, B.; Platt, J.; Hofmann, T. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*; MITP: Cambridge, MA, USA, 2007; pp. 1601–1608.
34. Jin, T.; Cao, L.; Zhang, B.; Sun, X.; Deng, C.; Ji, R. Hypergraph Induced Convolutional Manifold Networks. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), Macao, China, 10–16 August 2019; pp. 2670–2676.
35. Cai, Y.; Zhang, Z.; Cai, Z.; Liu, X.; Jiang, X. Hypergraph-Structured Autoencoder for Unsupervised and Semisupervised Classification of Hyperspectral Image. *IEEE Geosci. Remote Sens. Lett.* **2021**, 1–5. [[CrossRef](#)]
36. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16), Barcelona, Spain, 4–9 December 2016; pp. 3844–3852.
37. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), Long Beach, CA, USA, 4–9 December 2017; pp. 1025–1035.
38. Chen, C.; Li, W.; Su, H.; Liu, K. Spectral-Spatial Classification of Hyperspectral Image Based on Kernel Extreme Learning Machine. *Remote Sens.* **2014**, *6*, 5795–5814. [[CrossRef](#)]
39. Li, Y.; Guan, C.; Li, H.; Chin, Z. A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system. *Pattern Recognit. Lett.* **2008**, *29*, 1285–1294. [[CrossRef](#)]