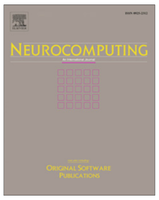
Neurocomputing 499 (2022) 93–105

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



# Investigating the geometric structure of neural activation spaces with convex hull approximations

Yuting Jia [a], Shao Zhang [a], Haiwen Wang [a], Ying Wen [a,*], Luoyi Fu [a], Huan Long [a], Xinbing Wang [a], Chenghu Zhou [b]

[a] Shanghai Jiao Tong University, China
[b] Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, China

ARTICLE INFO

*Article history:*
Received 10 January 2022
Revised 8 March 2022
Accepted 2 May 2022
Available online 12 May 2022

*Keywords:*
Neural networks
Activation space understanding
Convex hull

ABSTRACT

Neural networks have achieved great success in many tasks, including data classification and pattern recognition. However, how neural networks work and what representations they learn are still not fully understood. For any data sample fed into a neural network, we wondered how its corresponding vectors expanded by activated neurons change throughout the layers and why the final output vector could be classified or clustered. To formally answer these questions, we define the data sample outputs of each layer as *activation vectors* and the space expanded by them as the *activation space*. Then, we investigate the geometric structure of the high-dimensional activation spaces of neural networks by studying the geometric characters of the massive activation vectors through approximated convex hulls. We find that the different layers of neural networks have different roles, where the former and latter layers can disperse and gather data points, respectively. Moreover, we also propose a novel classification method based on the geometric structures of activation spaces, called nearest convex hull (NCH) classification, for the activation vectors in each layer of a neural network. The empirical results show that the geometric structure can indeed be utilized for classification and often outperforms original neural networks. Finally, we demonstrate that the relationship among the convex hulls of different classes could be a good metric to help us optimize neural networks in terms of over-fitting detection and network structure simplification.

## 1. Introduction

In recent years, neural networks have been widely used in areas including pattern recognition and data classification [1–4]. While they have delivered state-of-the-art performance on various tasks, little information about their intrinsic mechanism is known by people [5,6]. If we only use ‘accuracy’, the most widely used metric today, to evaluate neural networks, some fundamental information hidden behind the complicated architectures may be neglected. There is a major problem in that the prediction mechanisms of neural networks are black boxed, and it is difficult for us to understand and explain the reasons for their prediction results [7]. Moreover, a variety of activation functions have been proposed in recent years [8,9], and how they affect the training and application of neural networks is also a critical problem [10]. To solve the problems, in this paper, we try to uncover the intrinsic properties of the activation spaces and corresponding activation vectors of neural networks. For any data sample fed into a neural network, through several linear or nonlinear transformations, the output data vector could be classified, clustered, recognized, etc. Thus, we consider how the output vectors of each layer, called activation vectors, change during the whole transformation process of a neural network.

To date, it is still difficult for us to understand how activation vectors are distributed and change in high-dimensional activation spaces. Many research approaches have been proposed in recent years with the aim of solving the problem. One straightforward attempt to understand the structure of activation spaces would be dimension reduction and visualization. Many related works have been proposed, including stochastic neighbor embedding (SNE) [11], t-distributed SNE (t-SNE) [12] and LargeVis [13]. However, dimension reduction methods may lead to data point complications (e.g., overlapping), which do not happen in the original high-dimensional space. The other solution is to directly study the original high-dimensional activation spaces [14–17]. Although

* Corresponding author.
*E-mail addresses:* hnxxjyt@sjtu.edu.cn (Y. Jia), shaozhang@sjtu.edu.cn (S. Zhang), wanghaiwen@sjtu.edu.cn (H. Wang), ying.wen@sjtu.edu.cn (Y. Wen), yiluofu@sjtu.edu.cn (L. Fu), longhuan@sjtu.edu.cn (H. Long), xwang8@sjtu.edu.cn (X. Wang), zhouch@lreis.ac.cn (C. Zhou).

https://doi.org/10.1016/j.neucom.2022.05.019
0925-2312/© 2022 Elsevier B.V. All rights reserved.

some metrics are proposed in these works to evaluate and compare different activation spaces, they still fail to give a clear and intuitive description and analysis of the geometric structure of the activation spaces. Such a lack of understanding can affect the performance and efficiency of neural network training and restrict the application of neural networks in some areas like medical treatment. In this paper, we consider using convex hull to help investigate the geometric feature of activation spaces. Convex hull illustrates the minimized geometric space of any group of data points [18] and has been widely used in various application scenarios [19–21]. Specifically, for a group of data points, we want to represent their features through a convex hull. Then, the comparison among activation vectors from different groups can be transformed to the investigation of their convex hulls' geometric relationship. As shown in Fig. 1, with data samples of different classes fed into a neural network classifier, for any layer, we capture the high-level structure of the activation vectors in the activation space by characterizing the convex hull of each class and exploring the relationship (e.g., the distance and intersection ratio) among them. The captured high-level structures could help us to understand the function of each neural layer and then guide us to train better neural networks more effectively [22,23].

However, in actual applications and experimental processes, considering the high dimensionality of common activation spaces, it is computationally infeasible to obtain the exact convex hull of high-dimensional activation vectors. Many approximation algorithms have been proposed to solve this problem, but most of them only work under some rather strong theoretical assumptions [24–26]. Only a few algorithms could be applied on arbitrary data points, but they still face problems such as computational inefficiency [27] and low accuracy [28]. Thus, this paper proposes a new efficient approximate convex hull algorithm, called the Revised Greedy Expansion (RGE) algorithm, which outperforms existing state-of-the-art algorithms.

We investigate the activation spaces by studying the intrinsic geometric property of the convex hull of each class and their relationships with RGE. We find that different layers of a neural network play different roles: the former layers disperse data points as far as possible, while the last few layers gather data strongly to some clusters. Furthermore, even though a mixture of original data samples of different classes exists, through a few layers, the convex hulls of different classes can be completely separated.

Inspired by these findings, we combine RGE with the Nearest Convex Hull (NCH) classifier [29] to evaluate the activation space of each layer. The empirical results show that the geometric structure of convex hulls of different classes can be utilized for classification and often outperforms the prediction result of the original neural network. Moreover, the NCH classifier quantitatively illustrates the occurrence of overfitting in the neural network training process at the layer level, which gives a new reasonable metric for overfitting detection and neural network architecture design.

In summary, the contributions of this paper are threefold:

- We propose a novel efficient approximate convex hull algorithm in high-dimensional spaces that outperforms existing state-of-the-art algorithms.
- We present an improved understanding of the geometric structure of activation spaces through a convex hull investigation, which helps us determine the working process of neural networks and the role of each layer.
- We show the outstanding performance of the NCH classifier and give a new reasonable metric for detecting overfitting as well as redundant layers.

## 2. Related work

### 2.1. Neural network investigation

In recent years, many researchers have investigated deep neural networks and proposed various approaches with the aim of understanding neural networks.

- **Neuron Investigation**. These approaches try to determine the role of each individual neuron in a trained network. For example, Watanabe et al. [17] clustered neurons of one network layer into several communities and interpreted the role of each community. Bau et al. [30] attempted to identify the semantics of individual hidden neuron units of neural networks.
- **Layer Investigation**. These approaches consider each layer of a neural network as a whole and investigate their corresponding activation spaces. The intrinsic properties of neural networks and the comparison among networks are studied with different methods, including vector correlation analysis [31], space match model [32] and convex analytics [16].
- **Data Investigation**. These approaches focus on the relationship among data samples and the changes of their corresponding activation vectors through layers. Considering that classification is the most common task of neural networks, most studies investigating data samples are conducted with neural network classifiers. For example, Yousefzadeh [15] and Serpa [33] found

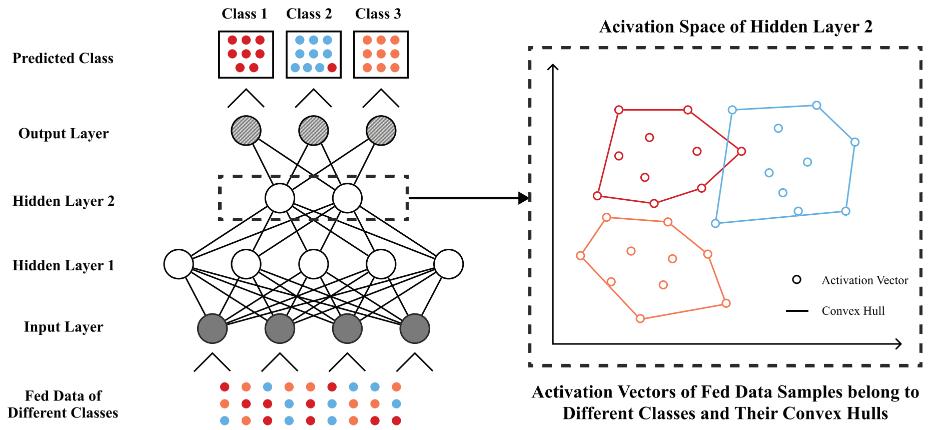

Class 1
Class 2
Class 3
Predicted Class
Output Layer
Hidden Layer 2
Hidden Layer 1
Input Layer
Fed Data of Different Classes
Acivation Space of Hidden Layer 2
Activation Vector
Convex Hull
Activation Vectors of Fed Data Samples belong to Different Classes and Their Convex Hulls

**Fig. 1.** Geometric structure investigation of activation spaces in neural networks. With fed data samples of different classes, for a specific layer, we capture the geometric structure of the activation vectors in the activation space by characterizing the convex hull of each class.

the convex hulls of different classes of data in the dataset and tried to explain the robustness and confidence bounds of neural networks, respectively. Other works focused on manifolds composed of activation vectors in a neural network to reveal the microstructures of activation spaces [34] and explore the classification capacity of neural networks [14].

The methods described above try to determine the working mechanisms of neural networks from different aspects. However, all of them fail to answer the basic and important question of how mixed data samples change through layers and why the output activation vectors could be classified or clustered. The methods focusing on manifolds composed of activation vectors are the best at answering this question, but the manifold method still cannot provide a simple and clear illustration of the geometric structure of activation vectors in high-dimensional activation spaces.

### *2.2. Convex hull approximation*

The convex hull approximation task is a significant problem, and there have been many theoretical studies on it. In recent years, various efficient algorithms have been proposed in low dimensions [35–38]. However, in high dimensions, most previous research either only gave some theoretical calculation bounds [39,40] or worked under some strong theoretical assumptions [25,24]. In this paper, we do not make any theoretical assumptions or restrictions, and thus, the approximated convex hull is more reasonable. To approximate convex hull in high dimensions, a straightforward approach is modeled as an enumeration problem: *for each node to find out if it lies in the convex hull of the other nodes*. Generally, there are two common ways to solve the problem: *linear programming* [41,27] and *triangle algorithm* [42–44]. The complexities in calculation of both ways are independent of the number of dimensions, which means the methods can scale well to high-dimensional spaces. However, in the enumeration process, they would have worst case complexities that depend on the square or cube of the number of remaining nodes. Huang et al. [28] proposed a semi-nonnegative matrix factorization method with kernel trick to accelerate the identification of the extreme nodes in high dimensions. However, the accuracy of the approximate convex hull constructed by the kernelized extreme nodes is lower than that of other methods. In this paper, we try to combine the advantages of different approaches in a unified framework to approximate high-quality convex hulls efficiently in arbitrary high-dimensional spaces.

Recently, deep learning methods have also been introduced into convex hull approximation problems, in which the most typical model is the pointer network (Ptr-Net) [45]. However, a critical problem for applying such models in high-dimensional convex hull approximation tasks is that it is difficult to prepare training datasets for them. Generally, there are two aspects of this problem: 1) In very high-dimensional spaces (e.g., spaces with thousands of dimensions), according to [46], the expected vertices of the convex hull could be exponential to the number dimensions, which means it is impossible for us to prepare proper and sufficient training datasets; 2) For each dimension, we have to prepare a new training dataset and retrain the model. In recent years, transfer learning has been adopted to help neural networks to merge knowledge from different domains and work with different source domains without retraining [47–49]. However, it requires the semantic definition of features so that it is mostly utilized in fields like computer vision. In the scenario of convex hull detection, the value of each dimension of nodes contains no semantic meaning. Therefore, it will be difficult to apply one trained convex hull detection model in different dimensions through transfer learning.

## 3. Notations

In this paper, a high-dimensional point is an activation vector formed by the output of all neurons at any layer. We start by focusing on fully connected layers of neural networks for classification. For a well-trained neural network $\mathscr{N}$ with $\ell$ fully connected layers, where the $i$th layer contains $d^i$ neurons, when any one data point $\mathbf{x}$ from the whole dataset $\mathscr{X}$ is fed into $\mathscr{N}$, the output at the $i$th layer can be regarded as a vector of dimension $d^i$. We use the symbol $\mathbf{x}^i \in \mathbb{R}^{d^i}$ to represent the vector and call it the activation vector of input $\mathbf{x}$ at layer $i$. Moreover, for each layer, we call the $d^i$-dimensional space containing all activation vectors the activation space of that layer. Furthermore, for a certain class $C$, we denote $\mathbf{x}^{i,C}$ as activation vector $\mathbf{x}^i$ (at the $i$th layer) of the input $\mathbf{x}$ of class $C$. When we are discussing a set of data, we use $\mathscr{Y}^{i,C} = \left\{\mathbf{y}_1^{i,C}, \mathbf{y}_2^{i,C}, \ldots, \mathbf{y}_Y^{i,C}\right\}$ to represent a collection of $Y = |\mathscr{Y}^{i,C}|$ activation vectors of the same class $C$ in the $i$th layer.

## 4. Convex hull approximation in activation spaces

In this paper, we study the convex hulls of vectors in activation spaces and utilize them for data classification. We believe this is an important step towards revealing the intrinsic properties and geometric structures of neural networks. Furthermore, we try to characterize the geometric structures of activation spaces to explain the phenomena found during the investigation.

For a set of points $\mathscr{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_S\}$ in $d$ dimensions, its convex hull $CH(\mathscr{S})$ is defined as the smallest convex set that contains $\mathscr{S}$, and it can be written as follows:

$$CH(\mathscr{S}) := \left\{ \sum_{j=1}^{S} t_j \mathbf{x}_j | \mathbf{x}_j \in S, \sum_{j=1}^{S} t_j = 1, t_j \in [0,1] \right\}. \tag{1}$$

When the dimension $d$ is low, the classical algorithms for convex hull detection [50–52] work in $O(S\log S)$ time. However, in regard to high-dimensional data, the exact convex hull becomes computationally infeasible with a lower bound relying exponentially on the dimension $d$ [46]. As dimensions in neural networks are usually very high, it is natural to pursue an efficient approximation algorithm instead.

### *4.1. Revised greedy expansion algorithm*

In this paper, we propose our algorithm, which is called Revised Greedy Expansion (RGE), to find the convex hull of high-dimensional points by using the greedy expansion (GE) algorithm [27] as our reference. This algorithm provides an approximate convex hull with a balance between complexity and accuracy.

The most critical problem in finding a convex hull is how to judge whether a point $\mathbf{v}$ is inside or outside a set of points $\mathscr{S} = \{\mathbf{x}_1, \cdots, \mathbf{x}_S\}$. We solve this problem by calculating the distance from $\mathbf{v}$ to $\mathscr{S}$. The distance $d(\mathbf{v}, \mathscr{S})$ is defined as the following dimension-independent quadratic programming problem:

$$\begin{aligned} d(\mathbf{v}, \mathscr{S}) = \quad & \min_{\alpha_i} \left\| \mathbf{v} - \sum_{i=1}^{S} \alpha_i \mathbf{x}_i \right\|_2 \\ & s.t. \quad \alpha_i \geqslant 0, \ \sum_{i=1}^{S} \alpha_i = 1. \end{aligned} \tag{2}$$

Obviously, $\mathbf{v} \in \mathrm{CH}(\mathscr{S})$ if and only if $d(\mathbf{v}, \mathscr{S}) = 0$. Actually, we relax this criterion to reduce the computational complexity. If the distance $d(\mathbf{v}, \mathscr{E})$ is less than a small positive number $\epsilon$ instead of exactly zero, we regard it as an interior point.

Finally, for the given set $\mathscr{S}$, we call $\mathscr{E} \subseteq \mathscr{S}$ an $\epsilon$-approximation of CH($\mathscr{S}$) if for any $\mathbf{u} \in \mathscr{S}$ it holds that

$$d(\mathbf{u}, \mathscr{E}) \leqslant \epsilon, \tag{3}$$

and it should be of minimal size. It requires that every point in $\mathscr{S}$ is within distance $\epsilon$ from its $\epsilon$-approximate convex hull CH($\mathscr{S}$). Generally, with smaller $\epsilon$, the approximate convex hull will be more precise, while its calculation process will consume more time. There is a trade-off between approximation precision and calculation time by tuning $\epsilon$, and it should be determined depending on specific application scenarios and approximation precision requirements.

Then, we adopt a greedy expansion method to select points from $\mathscr{S}$ to compose the $\epsilon$-approximation convex hull. For each iteration, supposing $\mathscr{E}$ is the set of selected points with size $E = |\mathscr{E}|$, we will find

$$\mathbf{x} = \arg \min_{\mathbf{x} \in \mathscr{S} \setminus \mathscr{E}} \max_{\mathbf{v} \in \mathscr{S} \setminus \mathscr{E}} d(\mathbf{v}, \mathscr{E} \cup \mathbf{x}) \tag{4}$$

and add it to $\mathscr{E}$.

## 4.2. Convex hull initialization

It is noted that the time complexity for finding $\mathbf{x}$ is $O\left((S-E)^2\right)$, which is extremely high when $E$ is small. We have to check every point one by one according to our algorithm, so we would like to have an initial set that is as large as possible. We follow [28] to utilize the Kernel Convex Hull Approximation (KCHA) approach to obtain the startup set, in which the kernel trick is utilized to speed up finding extreme points. KCHA is based on the semi-nonnegative matrix factorization (Semi-NMF) for the following optimization problem:

$$\begin{aligned} \min_{\mathbf{C} \geqslant 0} \quad & ||\phi(\mathbf{X}) - \phi(\mathbf{X})\mathbf{C}||^2, \\ s.t. \quad & \mathbf{1}^T\mathbf{C} = \mathbf{1}^T, \end{aligned} \tag{5}$$

where $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_S]$ is a matrix containing $S$ column vectors; $\mathbf{C} = [\mathbf{c}_1 \cdots \mathbf{c}_S]$ with shape $(S \times S)$ encodes valuable identification information of the extreme points; $\mathbf{1}$ is an $S$-dimensional column vector with all elements of 1; and $\phi(\cdot)$ denotes the kernel projection function. In this paper, we adopt the Gaussian kernel $K(\mathbf{X}, \mathbf{X}) = \phi(\mathbf{X})^T\phi(\mathbf{X})$, whose elements are defined as follows:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma^2}\right). \tag{6}$$

Algorithm 1: Kernelized Convex Hull Approximation

```
Input: matrix $\mathbf{X}$ consisting of points $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_S\}$, the expected
       number of extreme points $n$.
Output: $n$ approximated extreme points.
1 Initialize $\mathbf{C}$ with $\mathbf{C}_{ij} = \frac{1}{N}$;
2 while $\mathbf{C}$ does not converge do
3     set $\mathbf{C} = \mathbf{C} \circ \sqrt{\frac{K(\mathbf{X},\mathbf{X})}{K(\mathbf{X},\mathbf{X})\mathbf{C}}}$;
4 end
5 return the set of $n$ points whose values in $diag(\mathbf{C})$ (i.e., the diagonal
    elements in $\mathbf{C}$) are among the top $n$;
```

Algorithm 2: Revised Greedy Expansion Algorithm

```
Input: points $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_S\}$, approximation rate $\epsilon$.
Output: $\epsilon$-approximation convex hull $\mathcal{E}$.
1 Find the kernelized extreme points and initialize $\mathcal{E}$ as the set of these points;
2 Assign the set of outside points $\mathcal{R} = \{\mathbf{x} \in \mathcal{S} | d(\mathbf{x}, \mathcal{E}) > 0\}$;
3 while $\max_{\mathbf{x} \in \mathcal{S}} d(\mathbf{x}, \mathcal{E}) > \epsilon$ do
4     Select point $\mathbf{x} = \arg\min_{\mathbf{x} \in \mathcal{R}} \max_{\mathbf{v} \in \mathcal{R}} d(\mathbf{v}, \mathcal{E} \cup \mathbf{x})$ and add it to $\mathcal{E}$;
5     Remove point $\mathbf{x}$ from $\mathcal{E}$ if $d(\mathbf{x}, \mathcal{E} \setminus \mathbf{x}) \leq \epsilon$;
6     Remove point $\mathbf{x}$ from $\mathcal{R}$ if $d(\mathbf{x}, \mathcal{E}) \leq \epsilon$;
7 end
8 return $\epsilon$-approximation convex hull $\mathcal{E}$;
```

Considering $K(\mathbf{x},\mathbf{x}) = 1$ and $K(\mathbf{x},\mathbf{y}) > 0$, one possible way to solve Eq. (5) is to regard it as a Semi-NMF problem [28,53] and utilize a solver with the multiplicative updating rule:

$$\begin{aligned} \mathbf{C}^{k+1} &= \mathbf{C}^k \circ \sqrt{\frac{\left[\phi(\mathbf{X})^T\phi(\mathbf{X})\right]_+ + \left[\phi(\mathbf{X})^T\phi(\mathbf{X})\right]_- \mathbf{C}^k}{\left[\phi(\mathbf{X})^T\phi(\mathbf{X})\right]_- + \left[\phi(\mathbf{X})^T\phi(\mathbf{X})\right]_+ \mathbf{C}^k}} \\ &= \mathbf{C}^k \circ \sqrt{\frac{K(\mathbf{X},\mathbf{X})}{K(\mathbf{X},\mathbf{X})\mathbf{C}^k}}, \end{aligned} \tag{7}$$

where $\mathbf{C}^k$ is the calculated $\mathbf{C}$ after the $k$th iteration of updating; $\mathbf{A} \circ \mathbf{B}$ and $\frac{\mathbf{A}}{\mathbf{B}}$ denote elementwise multiplication and division between matrices $\mathbf{A}$ and $\mathbf{B}$, respectively; $[\cdot]_+$ and $[\cdot]_-$ are defined as $[\mathbf{A}]_+ = \frac{\mathbf{A}+abs(\mathbf{A})}{2}$ and $[\mathbf{A}]_- = \frac{\mathbf{A}-abs(\mathbf{A})}{2}$, respectively, where $abs(\mathbf{A})$ is a matrix consisting of the absolute values of the elements of matrix $\mathbf{A}$. With such an updating rule, the whole KCHA process is summarized in Algorithm 1.

### 4.3. Performance and efficiency of RGE

With the greedy expansion process and the method for initializing the extreme point set, the whole RGE approach is presented as Algorithm 2.

We compare RGE with three representative baselines from three categories to evaluate its performance. KCHA aims to achieve very high efficiency basing on semi-nonnegative matrix factorization and kernel tricks to achieve very high efficiency. GE and AVTA are representative algorithms to solve the convex hull expansion enumeration problem through linear programming and triangle algorithm, respectively.

In this section, we generate four 2-dimensional toy datasets and apply the four algorithms to them. The detected convex hull will be evaluated from two aspects: 1) The most important is whether the detected convex hull cover all the nodes or not. We calculate the largest distance from outside nodes to the detected convex hull and denote it as Approximation Error Distance. 2) Besides, each

**Table 1**
Approximation error distance and inner point ratio of the algorithms on four toy datasets.

| | Approximation Error Distance | | | | Inner Point Ratio | | | |
|---|---|---|---|---|---|---|---|---|
| | KCHA | GE | AVTA | RGE | KCHA | GE | AVTA | RGE |
| Center | 0.192 | 0.005 | 0.200 | 0.000 | 50.0% | 0.0% | 0.0% | 0.0% |
| Circles | 0.671 | 0.158 | 0.062 | 0.053 | 55.0% | 8.7% | 0.0% | 4.3% |
| Moons | 2.037 | 0.043 | 0.117 | 0.039 | 55.0% | 30.4% | 0.0% | 16.7% |
| Centers | 0.793 | 0.038 | 0.326 | 0.038 | 55.0% | 10.0% | 0.0% | 0.0% |

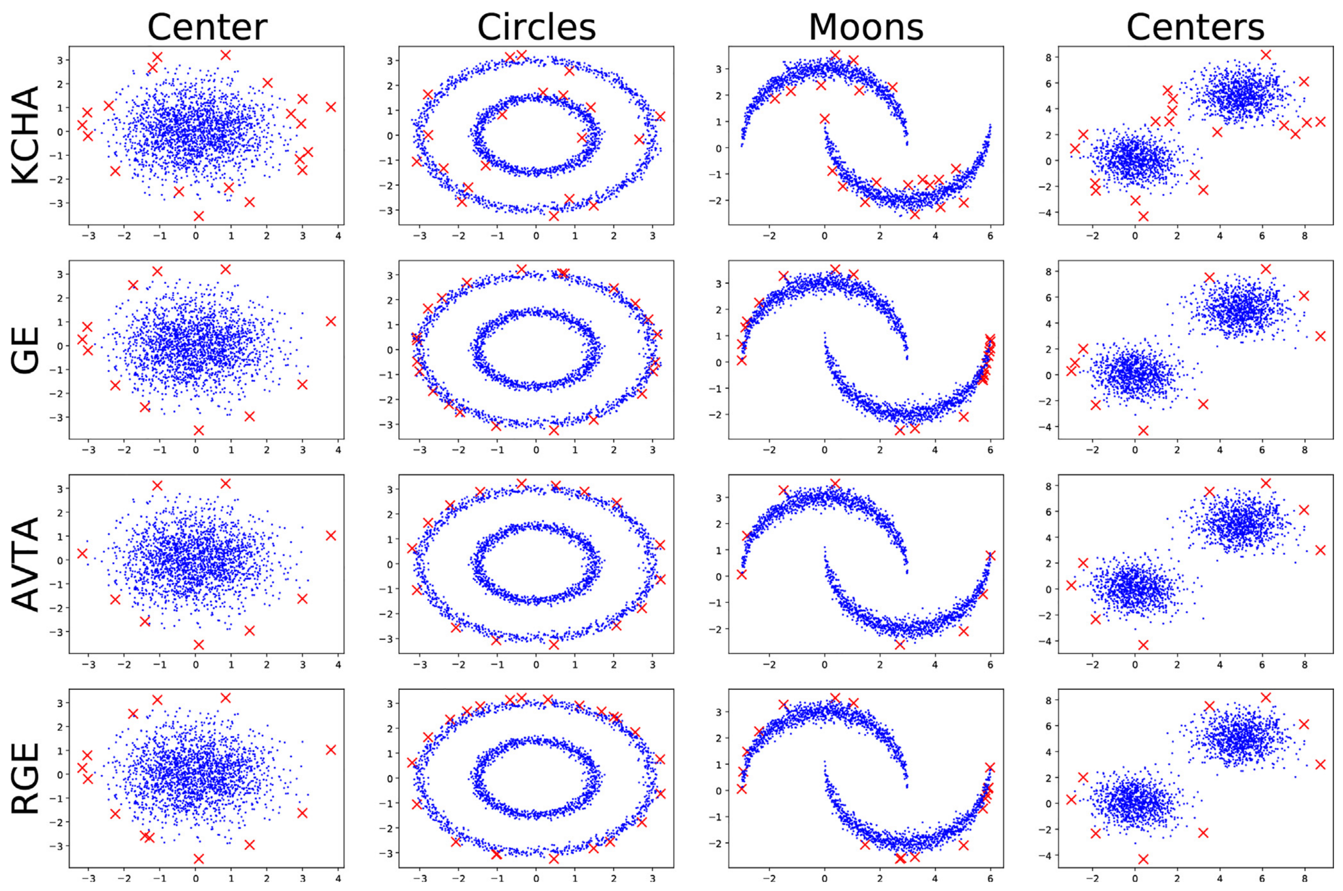


**Fig. 2.** Approximate convex hulls found by the three algorithms on four toy datasets. Both the blue nodes and red crosses denote input points, and the red crosses represent points selected as extreme points by the respective algorithm.

**Table 2**
Running times (sec.) of the algorithms on four toy datasets.

| Algorithm | Center | Circles | Moons | Centers |
|---|---|---|---|---|
| KCHA | 7.78 | 16.15 | 18.19 | 18.06 |
| GE | 2074.81 | 2855.46 | 6655.60 | 7568.68 |
| AVTA | 98.61 | 120.38 | 59.63 | 61.26 |
| RGE | 9.13 | 32.63 | 38.88 | 45.73 |

**Table 3**
Neural network statistics. $\ell$: number of hidden layers, $d$: number of neurons per layer, TR: training accuracy, and TE: test accuracy.

| Dataset | $\ell$ | $d$ | ReLU | | Sigmoid | |
|---|---|---|---|---|---|---|
| | | | TR | TE | TR | TE |
| MNIST | 4 | 1024 | 1.00 | 0.94 | 0.98 | 0.96 |
| CIFAR-10 | 4 | 1024 | 0.94 | 0.46 | 0.93 | 0.44 |
| FOS-CS-5 | 4 | 1024 | 0.79 | 0.59 | 0.93 | 0.62 |
| GOOGLE-8 | 4 | 1024 | 0.99 | 0.95 | 0.99 | 0.95 |

node in the detected convex hull should be a crucial node that does not lie in the convex hull of the other nodes. We count the ratio of nodes lying in the convex hull of the other nodes and denote it as Inner Point Ratio.

The results of the four algorithms on the four toy datasets are summarized in Table 1. As we can see, the error distance of RGE is always lower than or equal to other methods, which shows the high accuracy of RGE. As for the aspect of the inner point ratio, both AVTA and RGE outperform the other two methods. Although there are some inner points for RGE on dataset Circles and Moons, considering its improvement on the error distance, the result of RGE is still overall excellent. To more intuitively illustrate the difference among them, we plot the detected convex hulls of the four algorithms in Fig. 2. Further more, we evaluate the efficiency of the three algorithms by directly comparing their running times, as shown in Table 2. Benefiting from obtaining the startup convex hull by KCHA, compared to GE and AVTA, RGE achieves a huge improvement in efficiency.

In summary, although some baselines may outperform RGE in some specific points, RGE takes a trade-off among accuracy and efficiency, and achieves better usability and reliability.

## 5. Geometric structures of activation spaces

In this part, we build geometric descriptions for data points of different classes in the activation spaces. A geometric description is more intuitive than pure quantitative metrics, such as accuracy, and it provides a better understanding of some intrinsic properties of a neural network. We first introduce our experimental platform and then address the three most significant results that give a clearer picture of the macrostructure of activation spaces.

### *5.1. Experimental Platform*

In the experiments, we adopt four datasets from two categories, one with continuous vectors and one with discrete embeddings.

Specifically, the first category contains two well-studied image classification datasets, MNIST[54] and CIFAR-10 [55], which contain 70 k and 60 k images in 10 classes, respectively. For each dataset, we sample 10 k images as a test set, and the others are used for the training set.

The second category contains two datasets for the network embedding task extracted from a large-scale knowledge graph in the academic domain called AceKG [56]: **1)** FOS-CS-5: a network containing all the papers, authors and venues in AceKG in the five subfields of computer science[1], which contains 6.5 M nodes in 5 classes. **2)** GOOGLE-8: a network containing eight venue categories from Google Scholar[2] and AceKG; the heterogeneous network contains 1.2 M nodes from 8 classes. For more details about the two datasets, refer to [56].

In this paper, to feed these two network datasets into the neural networks, we adopt the well-known graph representation learning algorithm, DeepWalk [57], to embed both networks into 64-dimensional vectors and randomly select 60 k (50 k for training and 10 k for testing) vectors for the following study. The two embedded vector datasets are available online[3].

There is a distinct difference among these datasets. Although all of the data can be seen as vectors, the images in MNIST and CIFAR-10 are originally continuous dense vectors, while the data in other two datasets are embeddings of graph structures. As our work concentrates on the study of vectors, we believe it would be more persuasive to consider both types of data.

In this paper, we focus on neural networks with a fully connected layer and softmax layer, i.e., the Multilayer Perceptron Model (MLP). The reason for this choice is that a fully connected structure is the most representative and fundamental factor in neural networks for studying activation spaces. Although this paper does not discuss state-of-the-art neural networks, all the methods and experiments proposed and deployed in this paper can also be applied to those networks. Thus, to begin, we train two neural networks on each dataset with different activation functions, i.e., ReLU or sigmoid. ReLU and Sigmoid are the two most representative and commonly used activation functions in neural networks. Moreover, the output ranges of them are $[0, +\infty)$ and $(0, 1)$, respectively, which can show the characteristics of the activation spaces without constraint and with constraint. Except for the activation functions, all the network structures are the same: they contain four fully connected layers with 1024 neurons per layer. The detailed statistics of the well-trained neural networks are shown in Table 3. The accuracies of the networks on the training set are quite high, where the lowest accuracy (approximately 80% on FOS-CS-5 with ReLU) is still acceptable considering the difficulty of 5-class classification. Even though the networks on CIFAR-10 and FOS-CS-5 seem to overfit, they still have the ability to classify the data points in the training set correctly. Therefore,

[1] computer security, operating systems, algorithms, world wide web and machine learning

[2] https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng

[3] https://bit.ly/embedding_datasets

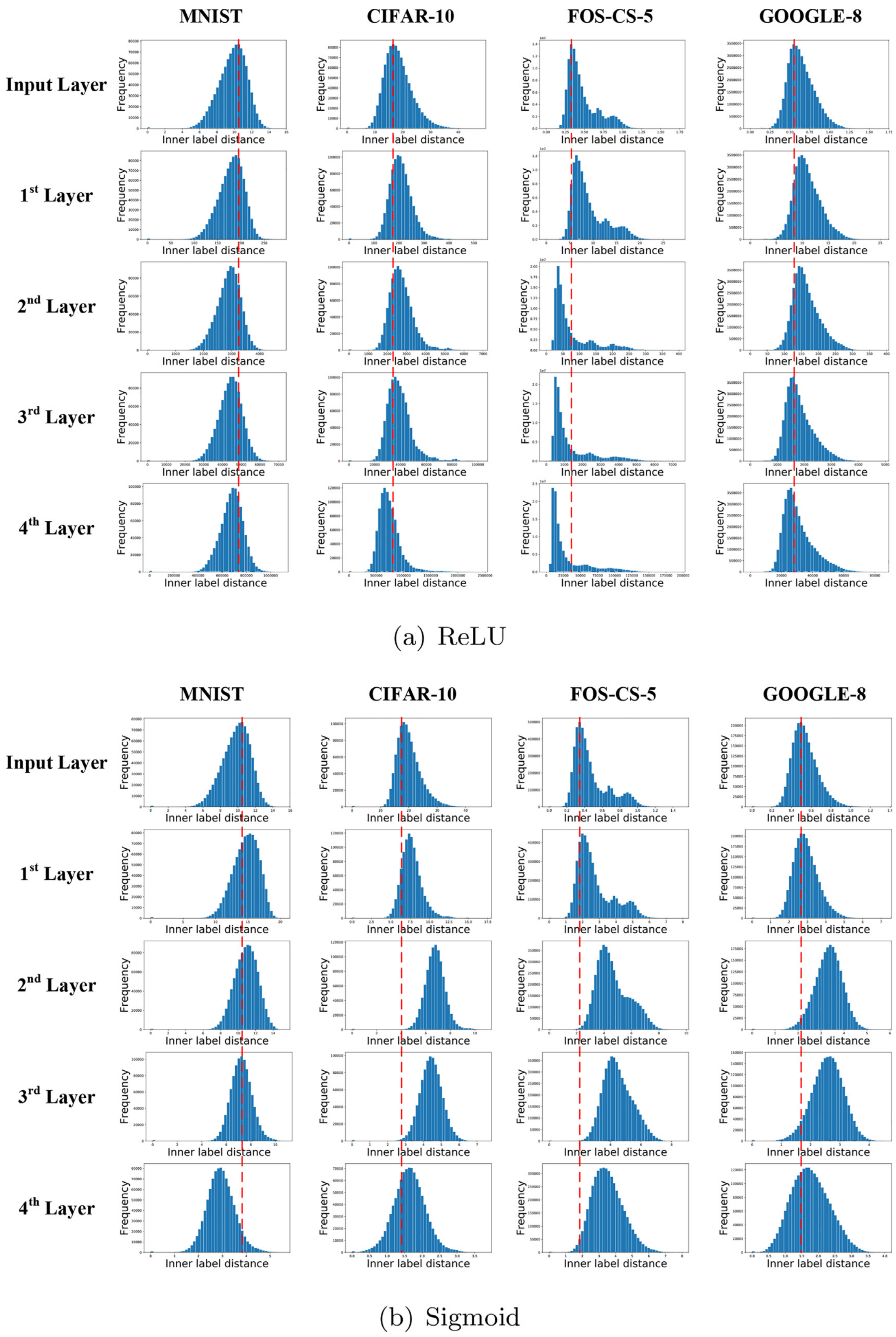


**Fig. 3.** Frequency distribution histogram of the inner class distances for five layers on four datasets. The red dashed line marks the peak of the input layer and helps to show the shift in the peak through the layers.

**Table 4**
The number of data points included in the convex hulls of other nodes within the same class in the 4th layer.

| | ReLU | | Sigmoid | |
|---|---|---|---|---|
| | Train | Test | Train | Test |
| MNIST | 0.0% | 0.0% | 9.5% | 10.4% |
| CIFAR-10 | 0.0% | 0.0% | 90.0% | 79.2% |
| FOS-CS-5 | 28.7% | 28.7% | 5.7% | 5.2% |
| GOOGLE-8 | 0.0% | 0.0% | 82.2% | 79.2% |

it is also meaningful to study the properties of their activation spaces.

### 5.2. Distribution shift of the inner class distance

First, the most critical problem is the spatial distribution of activation vectors. Here, we try to figure this out by studying the Euclidean distances between any two activation vectors of the same class, called the inner class distances. In other words, for layer $i$ and class $C$, we calculate

$$d\left(\mathbf{y}_{k_1}^{i,C}, \mathbf{y}_{k_2}^{i,C}\right) = ||\mathbf{y}_{k_1}^{i,C} - \mathbf{y}_{k_2}^{i,C}||_2, \ \forall k_1 \neq k_2. \tag{8}$$

We visualize the results by drawing the distribution histogram of the distance distribution for each layer and each class. Due to space limitations, we only present the result of one class for each dataset and each network in Fig. 3. It should be mentioned that the results of the other classes are very similar.

In general, the distribution of the inner class distances is unimodal. This means that activation vectors of the same class are distributed almost uniformly, and thus, there are not many small clusters that are distant from the others. Such a conclusion inspires us to draw the convex hull of the activation vectors of each class, and then the characteristics of each class could be represented by the convex hull.

Moreover, for deeper layers, i.e., those that are closer to the output layer, there are two apparent phenomena: **1)** The absolute value of the inner class distance of the network with ReLU increases, which is closely related to the existence of adversarial examples [58]. On the other hand, with the maximum output limitation of the sigmoid function, the inner class distance of the network with sigmoid increases first and then decreases. **2)** Through the four hidden layers, overall, the peaks of the distribution gradually first shift to the right and then back to the left. This means that data samples fed into the network are first scattered and then gathered again. Especially for the 4th layer of the network with sigmoid, there exists an obvious shift on all four datasets.

### 5.3. More crucial points in the lower layers

In the previous section, we found that we can analyse the characteristics of the activation vectors of each class through their convex hulls. Thus, in this section, we investigate the convex hull of each class and summarize some intrinsic features of the convex hulls. Moreover, in the following section, we investigate the relationship among the convex hulls of the different classes.

For each class, there should be some activation vectors that form the convex hull of the class and other activation vectors that lie in the inner part of the convex hull. Specifically, for any activation vector $\mathbf{y}_k^{i,C}$, it is obvious that if $\mathbf{y}_k^{i,C} \in \mathrm{CH}\left(\mathscr{Y}^{i,C} \setminus \mathbf{y}_k^{i,C}\right)$, then $\mathbf{y}_k^{i,C}$ is not an extreme point, which means it is redundant for constructing the convex hull.

For all the discussed datasets, we implement the RGE algorithm to identify whether any data are redundant. However, experimental results show that, from the 1st to 3rd layers, for any class $C$ and layer $i$, $\mathbf{y}_k^{i,C} \notin \mathrm{CH}\left(\mathscr{Y}^{i,C} \setminus \mathbf{y}_k^{i,C}\right)$ is set up for over $95\%$ $k$. The number of inner points in the 4th layer (i.e., the final layer) is shown in Table 4, which shows that, in general, activation vectors of networks with sigmoid are denser. Especially on CIFAR-10 and GOOGLE-8, over $79\%$ of the data points are included in the convex hulls of other points.

Intuitively speaking, if we consider a convex hull as an approximation of the knowledge learned by a neural network, this result shows the following: **1)** In the lower layers, all vectors are essential to the corresponding conceptions; **2)** In the final layer, the domain of the knowledge contracts, and some data points can be represented by the features of other samples. Take dataset FOS-CS-5 as an example, the ratio of crucial points through all layers is shown in Fig. 4(a). As we can see, the crucial points first increase dramatically and then fall down. In other words, in a 4-layer MLP, the former layers extract the features and distinguish data points as much as possible, while the latter layers, especially the final layer, try to cluster the dispersed data points of the same class.

### 5.4. Few misinclusions in the lower layers

Now, we continue to study the relationship between an activation vector and convex hulls of different classes. For each layer, we calculate the distance according to Eq. (2) for all possible combinations of nodes and convex hulls. To reduce space, same as the previous section, we focus on FOS-CS-5, and the other datasets are similar. Fig. 4(b) shows the ratio of data points included in the convex hull of other classes through all layers of the two networks on FOS-CS-5. We can find that the ratio of misinclusions is very low in the 1st to 3rd layers, while the ratio increases substantially in the final layer, which supports the conclusion presented in the last section. Although there are some misincluded data points in the original datasets, after one or two layer(s) of data mapping, the misinclusions can be completely eliminated. Theoretically, there should exist some hyperplanes that can split data points from different classes. However, in the following layers, misinclusions occur again, representing a decrease in the ability to distinguish data points. In other words, neural networks that are too deep experience the problem of data mixing, and convex hull analysis of the activation spaces could help us to determine the appropriate depth of a neural network. Considering the activation vectors in the 4th layer are directly related to the performance of neural networks, we also summarize the ratio of misincluded points in the 4th layer for all networks in Table 5. As we can see, in the training stage, the misinclusion ratio of the network with ReLU on FOS-CS-5 is much more than other networks, which indicates that the neural network with ReLU on FOS-CS-5 is less accurate. Moreover, when we focus on the comparison between ‘Train’ and ‘Test’, we can see that there are three networks with the missinclusion ratio on

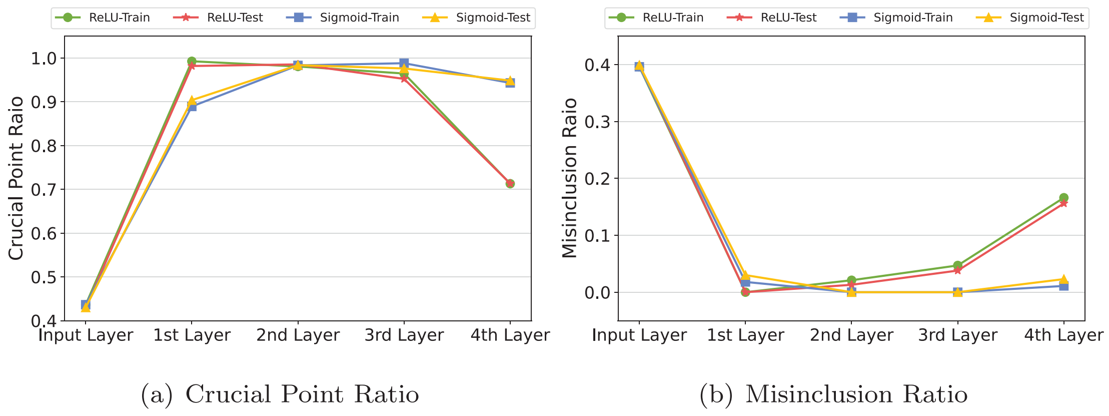


**Fig. 4.** The ratio of crucial points and misinclusions through all layers on FOS-CS-5.

**Table 5**
The number of data points included in the convex hulls of other classes in the 4th layer.

| | ReLU | | Sigmoid | |
|---|---|---|---|---|
| | Train | Test | Train | Test |
| MNIST | 0.0% | 0.0% | 0.0% | 0.0% |
| CIFAR-10 | 0.2% | 2.0% | 4.8% | 49.6% |
| FOS-CS-5 | 16.6% | 15.6% | 1.1% | 2.3% |
| GOOGLE-8 | 0.5% | 0.7% | 0.2% | 0.3% |

Test over 2-times more than that on Train: ReLU on CIFAR-10, Sigmoid on CIFAR-10, and Sigmoid on FOS-CS-5. The results also indicate that the corresponding neural networks are overfitting where the training accuracy exceeds the test accuracy 0.31 and more.

## 6. Nearest convex hull classification

Considering that there could be zero misinclusions among different classes, for any new data sample, we want to determine whether there are some novel classification methods based on the geometric relationship between its activation vector and the convex hulls of classes. More specifically, in this section, we adopt Nearest Convex Hull (NCH) classification to analyse the functionality of neural networks in each activation space (i.e., each layer). Several experiments are carried out to establish the observations in this section. Although the convex hull may seem to be an over-simple estimation of the distribution of activation vectors, further experiments show that this is not the case.

### *6.1. Methodology*

The NCH classification method was proposed in [29] and assigns a data point to the class whose convex hull is the closest. NCH has a good geometric interpretation and has been widely extended [25,26].

Similar to the traditional strategy in neural networks, we divide the datasets into training and test sets. For each layer in the trained network, denoted as the $i$th layer, we run the NCH classification algorithm as follows: **1)** For any activation vector $\mathbf{x}^i$ in the training set, take $n$, the number of classes, and the convex hulls formed by the activation vectors of all training data except $\mathbf{x}^i$. Different convex hulls correspond to different classes, and $\mathbf{x}^i$ takes the class of the closest one. **2)** For the test set data, we label all the activation vectors of the $i$th layer as above. Now, convex hulls can be formed by the whole training set.

### *6.2. Comparative methods*

We compared the performance of the NCH algorithm to the performance of neural networks and classical classifiers, including logistic regression (LR), the support vector machine (SVM), and the $k$-nearest neighbour (KNN) algorithm. One large difference from common neural networks is that now we can calculate the training/test accuracy for *all* layers, rather than for only the last layer.

### *6.3. Experiment setup*

In this paper, we will evaluate the performance of NCH classifier and comparative baselines in activation spaces of neural networks. For each classifier and each activation space, activation vectors in training and test sets are all fed into the classifier for classification. Then, we will define training accuracy and test accuracy as the percent of vectors whose predicted class is the same as its ground truth in training and test sets, respectively.

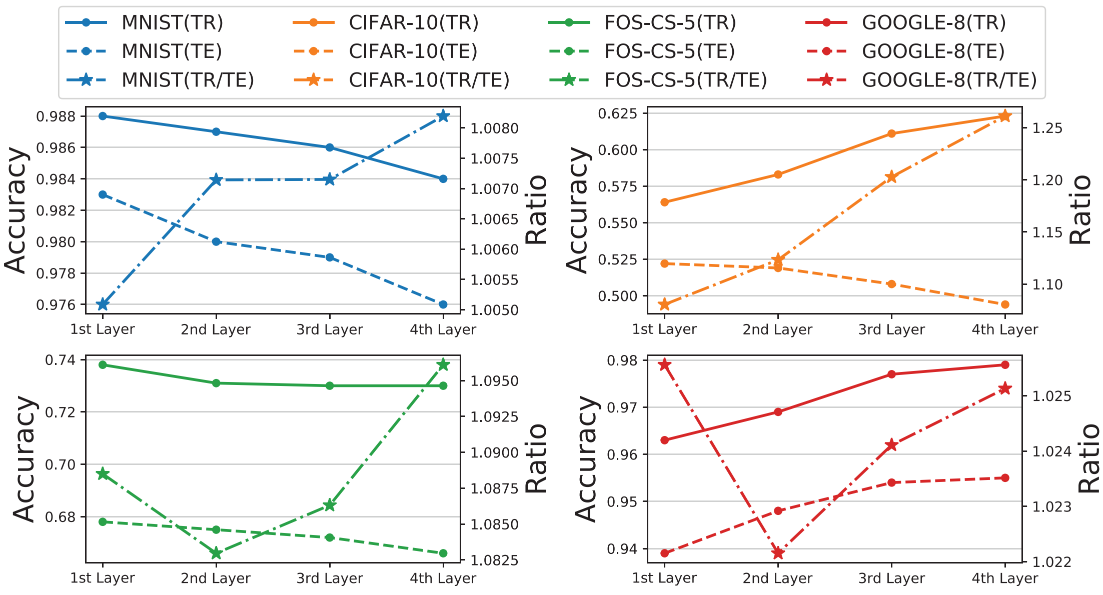
### *6.4. Comparison with neural networks*

We run the NCH classification algorithm on all four datasets above, and the experimental results are summarized in Table 6. Compared to the results in Table 3, the NCH classification method performs better on nearly every test set in all four layers. This is a interesting phenomenon, of which three key points should be addressed.

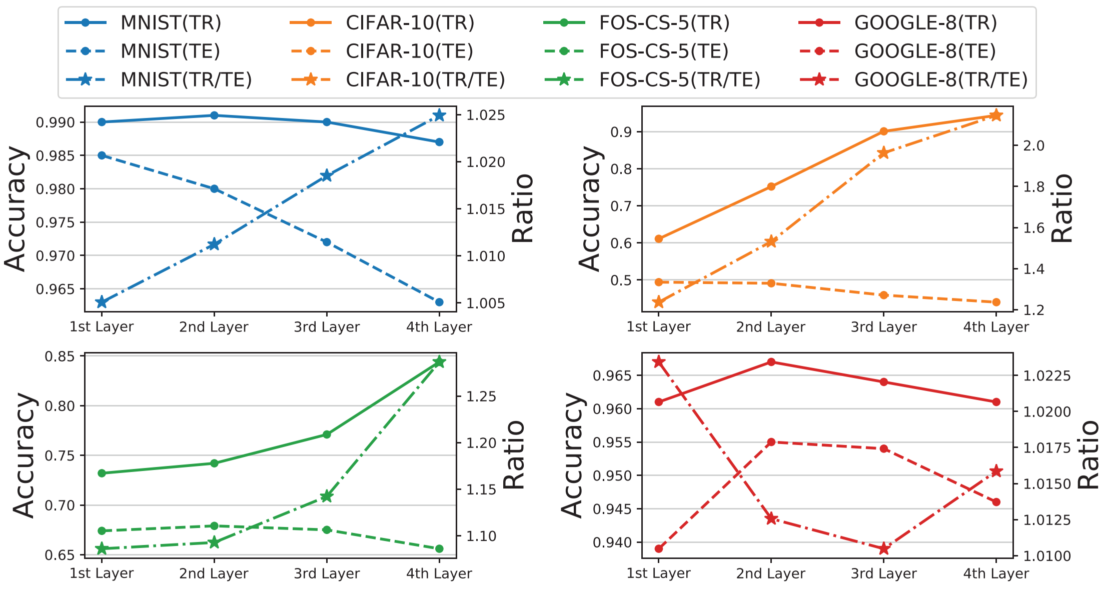
First, intuitively, NCH classification is less involved than neural networks because it has a rather clear geometric meaning. The better accuracy could be seen as a proof of the soundness of the conjecture: a relatively simple geometric structure that can be utilized for classification exists. Moreover, in the last section, we have also built some geometric descriptions in activation spaces that support such conjectures.

**Table 6**
Accuracy of NCH classification for the activation space of each layer of the two neural networks.

| | | ReLU | | | | Sigmoid | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1st Layer | 2nd Layer | 3rd Layer | 4th Layer | 1st Layer | 2nd Layer | 3rd Layer | 4th Layer |
| MNIST | train | 0.988 | 0.987 | 0.986 | 0.984 | 0.990 | 0.991 | 0.990 | 0.987 |
| | test | 0.983 | 0.980 | 0.979 | 0.976 | 0.985 | 0.980 | 0.972 | 0.963 |
| CIFAR-10 | train | 0.564 | 0.583 | 0.611 | 0.623 | 0.611 | 0.752 | 0.901 | 0.944 |
| | test | 0.522 | 0.519 | 0.508 | 0.494 | 0.494 | 0.491 | 0.459 | 0.440 |
| FOS-CS-5 | train | 0.738 | 0.731 | 0.730 | 0.730 | 0.732 | 0.742 | 0.771 | 0.844 |
| | test | 0.678 | 0.675 | 0.672 | 0.666 | 0.674 | 0.679 | 0.675 | 0.656 |
| GOOGLE-8 | train | 0.963 | 0.969 | 0.977 | 0.979 | 0.961 | 0.967 | 0.964 | 0.961 |
| | test | 0.939 | 0.948 | 0.954 | 0.955 | 0.939 | 0.955 | 0.954 | 0.946 |



(a) ReLU



(b) Sigmoid

**Fig. 5.** Accuracy of NCH classification for each layer of the two neural networks on the four datasets. 'TR' and 'TE' represent the training and test accuracy, respectively, and they refer to the left scale. 'TR/TE' means the ratio of TR to TE and refers to the right scale.

Second, compared to ReLU, networks with sigmoid tend to show more clustering features in the training process. As shown in Table 6, the accuracy of NCH classification on the training set for the activation spaces of networks with sigmoid is always very high (at least 0.844), regardless of whether the network is overfit. In contrast, for networks with ReLU, if one network is overfit, then its activation space for the training set will have worse geometric clustering features; i.e., the accuracy of NCH classification on the training set will decrease much. Such result corresponds to the characteristics of different activation functions. The output range of sigmoid is limited to $(0, 1)$ so that the activation vectors are also strongly clustered, which is consistent with the classification metric of NCH. No matter the neural network is overfitting or not, the training accuracy of NCH is always pretty high. On the contrary, the output range of ReLU is $[0, +\infty)$, which means the activation vectors are more likely to be discrete and have poorer geometric clustering. Although there could also be some specific hyperplane to separate different classes in the wide activation space (i.e., the accuracy of the original neural network is pretty high), the robustness of such hyperplane has been insufficient. In such situation, the overfitting occurs, and the training accuracy of NCH classifier decreases. This opens a new perspective on the study of overfitting. We believe that, at least for networks with ReLU, the NCH classification provides a more reliable accuracy metric that can be used to evaluate the quality of the models in the training process. Considering ReLU has been the most popular activation function in neural networks [59], even if the overfitting detection only works for ReLU, it also has great application value.

Finally, as shown in Fig. 5, for networks with both activation functions, there is manifested monotonicity considering the statistics of the NCH classification's accuracy. Especially on the MNIST and CIFAR-10 data, the ratio of the training accuracy to the test accuracy monotonically increases, while the test accuracy monotonically decreases. This implies that if we take the NCH classification algorithm, then it will be better to keep the output in the first activation space, and the rest of the neural network can be simply removed. In other words, NCH classification, or the geometric metric, can also help us to choose neural networks with appropriate depths.

### 6.5. Comparison with traditional classifiers

One natural question about the NCH classification algorithm for the activation spaces is can we replace it with simpler classification algorithms, such as the KNN algorithm. For a better understanding of the four datasets and the trained neural networks, we also apply some benchmark algorithms on the four datasets. As shown in Table 7, LR, SVM and KNN with $k = 5$ are adopted, and their training and test accuracies on the four datasets are listed.

It is shown that the NCH classification algorithm outperforms the traditional algorithms on the MNIST, CIFAR-10 and FOS-CS-5 datasets and achieves almost the same score on GOOGLE-8. This means that compared to the other strategy, the convex hull reveals more information about the intrinsic properties of the activation vectors.

**Table 7**
Results of the benchmarks on the four datasets. LR: logistic regression, SVM: support vector machine, KNN: $k$-nearest neighbor.

| | | LR | SVM | KNN |
|---|---|---|---|---|
| MNIST | train | 0.928 | 0.942 | 0.981 |
| | test | 0.920 | 0.944 | 0.968 |
| CIFAR-10 | train | 0.495 | 0.445 | 0.505 |
| | test | 0.388 | 0.440 | 0.340 |
| FOS-CS-5 | train | 0.589 | 0.547 | 0.747 |
| | test | 0.587 | 0.546 | 0.638 |
| GOOGLE-8 | train | 0.944 | 0.941 | 0.966 |
| | test | 0.940 | 0.938 | 0.955 |

## 7. Conclusion and future works

Our work tries to bridge the gap between the great success of modern neural networks and the very poor understanding of why they work so well. Compared to the previous efforts on this topic, we propose a simple and neat idea, i.e., to study the convex hulls of the corresponding conceptions formed by every layer of a neural network. To fulfil this goal, we build a new convex hull approximation algorithm that works efficiently in high-dimensional spaces. With such an algorithm, we investigate the geometric properties of activation spaces and propose a novel NCH classifier to evaluate the roles and performance of the neural network layers. Several interesting results on the functionality of neural networks are given in this paper. 1) The neural networks are intelligent that there are very few misinclusions in the sense of convex hulls, especially in the lower layers. 2) Different layers of a neural network play different roles, and the geometric properties of the activation spaces could help us choose an appropriate depth for a neural network. 3) The NCH classifier provides a new perspective on the study of overfitting detection and helps us to evaluate neural networks in the training process through their intrinsic characteristics. As the convex hull has explicit geometric meanings, we believe our work sheds light on the intrinsic geometric properties of neural networks.

There are two important future areas we want to investigate in the future: 1) One is to consider more state-of-the-art neural networks. This paper mainly focuses on MLP, one of the most fundamental architectures of neural networks. In recent years, more and more different neural networks (e.g., Convolutional Neural Network, Residual Network, and Transformer) have been proposed successively. No matter how the architectures of neural networks change, the fed data samples and their corresponding representations in the hidden layers are still high-dimensional vectors. Therefore, we can also utilize RGE and NCH classifier to analyze how the activation vectors and spaces change in these networks as well as their geometric properties. 2) The other one is to combine the convex hull investigation into the optimization of neural networks. This paper aims to investigate well-trained neural networks with convex hulls. In the future, we wish to explore how to train better neural networks by tuning network architectures and hyper-parameters according to the results of real-time convex hull investigation.

## CRediT authorship contribution statement

**Yuting Jia:** Conceptualization, Methodology, Software, Writing - original draft. **Shao Zhang:** Data curation, Writing - original draft. **Haiwen Wang:** Validation, Visualization. **Ying Wen:** Writing - review & editing, Supervision. **Luoyi Fu:** Writing - review & editing. **Huan Long:** Formal analysis, Writing - review & editing. **Xinbing Wang:** Supervision. **Chenghu Zhou:** Project administration.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This work is supported by the National Natural Science Foundation of China (42050105, 62020106005, 62061146002, 61960206002, 61822206, 61832013, 61829201).

## References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.

[2] J. Li, K. Jin, D. Zhou, N. Kubota, Z. Ju, Attention mechanism-based cnn for facial expression recognition, Neurocomputing 411 (2020) 340–350.

[3] F. Zhang, K. Sun, X. Wu, A novel variable selection algorithm for multi-layer perceptron with elastic net, Neurocomputing 361 (2019) 110–118.

[4] G. Hernández, E. Zamora, H. Sossa, G. Téllez, F. Furlán, Hybrid neural networks for big data classification, Neurocomputing 390 (2020) 327–340.

[5] F. Bartolucci, E.D. Vito, L. Rosasco, S. Vigogna, Understanding neural networks with reproducing kernel banach spaces, 2021.

[6] L. Wang, L. Hu, J. Gu, Y. Wu, Z. Hu, K. He, J.E. Hopcroft, Towards understanding learning representations: To what extent do different neural networks learn the same representation, 2018, CoRR abs/1810.11750.

[7] T. Gebhart, P. Schrater, Adversarial examples target topological holes in deep networks, CoRR abs/1901.09496 (2019).

[8] M. Zhu, W. Min, Q. Wang, S. Zou, X. Chen, Pflu and fpflu: Two novel non-monotonic activation functions in convolutional neural networks, Neurocomputing 429 (2021) 110–117.

[9] D. Kim, J. Kim, J. Kim, Elastic exponential linear units for convolutional neural networks, Neurocomputing 406 (2020) 253–266.

[10] X. Zou, Z. Wang, Q. Li, W. Sheng, Integration of residual network and convolutional neural network along with various activation functions and global pooling for time series classification, Neurocomputing 367 (2019) 39–45.

[11] G.E. Hinton, S.T. Roweis, Stochastic neighbor embedding, in: S. Becker, S. Thrun, K. Obermayer (Eds.), Advances in Neural Information Processing Systems 15, MIT Press, 2003, pp. 857–864.

[12] L. van der Maaten, Accelerating t-sne using tree-based algorithms, J. Mach. Learn. Res. 15 (2014) 3221–3245.

[13] J. Tang, J. Liu, M. Zhang, Q. Mei, Visualizing large-scale and high-dimensional data, in: WWW'16, International World Wide Web Conferences Steering Committee, pp. 287–297.

[14] U. Cohen, S. Chung, D.D. Lee, H. Sompolinsky, Separability and geometry of object manifolds in deep neural networks, Nat. Commun. 11 (2020) 1–13.

[15] R. Yousefzadeh, Deep learning generalization and the convex hull of training sets, CoRR abs/2101.09849 (2021).

[16] T. Ergen, M. Pilanci, Convex geometry of two-layer relu networks: Implicit autoencoding and interpretable models, in: S. Chiappa, R. Calandra (Eds.), Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 4024–4033.

[17] C. Watanabe, K. Hiramatsu, K. Kashino, Understanding community structure in layered neural networks, Neurocomputing 367 (2019) 84–102.

[18] R.T. Rockafellar, Convex analysis, Princeton Mathematical Series, Princeton University Press, Princeton, N.J., 1970.

[19] R. Zdunek, T. Sadowski, Image completion with approximate convex hull tensor decomposition, Signal Processing: Image Commun. 95 (2021) 116276.

[20] H. Zhang, Y. Cai, W. Li, et al., Containment control of general linear multi-agent systems by event-triggered control mechanisms, Neurocomputing 433 (2021) 263–274.

[21] X. Cao, M. Ren, J. Zhao, H. Lu, L. Jiao, Non-overlapping classification of hyperspectral imagery based on set-to-sets distance, Neurocomputing 378 (2020) 422–434.

[22] A.G. Anderson, C.P. Berg, The high-dimensional geometry of binary neural networks, CoRR abs/1705.07199 (2017).

[23] K. Schürholt, D. Borth, An investigation of the weight space for version control of neural networks, CoRR abs/2006.10424 (2020).

[24] A. Blum, S. Har-Peled, B. Raichel, Sparse approximation via generating point sets, in: Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms, Soc. Ind. Appl. Math., pp. 548–557.

[25] H. Cevikalp, B. Triggs, R. Polikar, Nearest hyperdisk methods for high-dimensional classification, in: ICML, ACM, pp. 120–127.

[26] C. Thurau, Nearest archetype hull methods for large-scale data classification, in: 2010 20th International Conference on Pattern Recognition, IEEE, pp. 4040–4043.

[27] H. Sartipizadeh, T.L. Vincent, Computing the approximate convex hull in high dimensions, arXiv preprint arXiv:1603.04422 (2016).

[28] C. Huang, Y. Wu, G. Min, Y. Ying, Kernelized convex hull approximation and its applications in data description tasks, in: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 1–8.

[29] G. Nalbantov, P. Groenen, C. Bioch, Nearest convex hull classification, Technical Report (2006).

[30] D. Bau, J.-Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, A. Torralba, Understanding the role of individual units in a deep neural network, Proc. Nat. Acad. Sci. 117 (2020) 30071–30078.

[31] M. Raghu, J. Gilmer, J. Yosinski, J. Sohl-Dickstein, Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability, in: Advances in Neural Information Processing Systems, pp. 6076–6085.

[32] L. Wang, L. Hu, J. Gu, Z. Hu, Y. Wu, K. He, J. Hopcroft, Towards understanding learning representations: To what extent do different neural networks learn the same representation, in: Advances in Neural Information Processing Systems, pp. 9607–9616.

[33] L.R. Pineda, A.L. Serpa, Determination of confidence bounds and artificial neural networks in non-linear optimization problems, Neurocomputing 463 (2021) 495–504.

[34] T. Yu, H. Long, J.E. Hopcroft, Curvature-based comparison of two neural networks, in: 2018 24th International Conference on Pattern Recognition (ICPR), IEEE, pp. 441–447.

[35] L. Li, S. Luo, X.-C. Tai, J. Yang, A variational convex hull algorithm, in: J. Lellmann, M. Burger, J. Modersitzki (Eds.), Scale Space and Variational Methods in Computer Vision, Springer International Publishing, Cham, 2019, pp. 224–235.

[36] G. Klimenko, B. Raichel, Fast and exact convex hull simplification, arXiv preprint arXiv:2110.00671 (2021).

[37] J.F. Castro, M. Romero, G. Gutiérrez, M. Caniupán, C. Quijada-Fuentes, Efficient computation of the convex hull on sets of points stored in a k-tree compact data structure, Knowl. Inf. Syst. 62 (2020) 4091–4111.

[38] R. Graham, A.M. Oberman, Approximate convex hulls: sketching the convex hull using curvature, arXiv preprint arXiv:1703.01350 (2017).

[39] G. Klimenko, B. Raichel, G. Van Buskirk, Sparse convex hull coverage, Computational Geometry 98 (2021) 101787.

[40] G. Van Buskirk, Finding Features via Hull Approximation, The University of Texas at Dallas, 2021.

[41] A. Ruano, H.R. Khosravani, P.M. Ferreira, A randomized approximation convex hull algorithm for high dimensions, IFAC-PapersOnLine 48 (2015) 123–128. 2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics CESCIT 2015.

[42] B. Kalantari, A characterization theorem and an algorithm for a convex hull problem, Ann. Oper. Res. 226 (2015) 301–349.

[43] P. Awasthi, B. Kalantari, Y. Zhang, Robust vertex enumeration for convex hulls in high dimensions, in: A. Storkey, F. Perez-Cruz (Eds.), Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, volume 84 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 1387–1396.

[44] B. Kalantari, A triangle algorithm for semidefinite version of convex hull membership problem, arXiv preprint arXiv:1904.09854 (2019).

[45] O. Vinyals, M. Fortunato, N. Jaitly, Pointer networks, in: Advances in Neural Information Processing Systems, pp. 2692–2700.

[46] R.A. Dwyer, Average-case Analysis of Algorithm for Convex Hulls and Voronoi Diagrams., Ph.D. thesis, CMU, USA, 1988.

[47] Z. Wan, R. Yang, M. Huang, N. Zeng, X. Liu, A review on transfer learning in eeg signal analysis, Neurocomputing 421 (2021) 1–14.

[48] Q. Wu, H. Wu, X. Zhou, M. Tan, Y. Xu, Y. Yan, T. Hao, Online transfer learning with multiple homogeneous or heterogeneous sources, IEEE Trans. Knowl. Data Eng. 29 (2017) 1494–1507.

[49] Z. Yang, R. Yang, M. Huang, Rolling bearing incipient fault diagnosis method based on improved transfer learning with hybrid feature extraction, Sensors 21 (2021) 7894.

[50] D.R. Chand, S.S. Kapur, An algorithm for convex polytopes, J. ACM 17 (1) (1970) 78–86.

[51] R. Graham, An efficient algorithm for determining the con-vex hull of a finite planar set, Inform. Process. Lett. 1 (4) (1972) 132–133.

[52] R. Jarvis, On the identification of the convex hull of a finite set of points in the plane, Inform. Process. Lett. 2 (1) (1973) 18–21.

[53] C.H. Ding, T. Li, M.I. Jordan, Convex and semi-nonnegative matrix factorizations, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 45–55.

[54] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.

[55] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical Report, Citeseer, 2009.

[56] R. Wang, Y. Yan, J. Wang, Y. Jia, Y. Zhang, W. Zhang, X. Wang, Acekg: A large-scale knowledge graph for academic data mining, CIKM '18, ACM, New York, NY, USA, 2018, pp. 1487–1490.

[57] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: SIGKDD, ACM, pp. 701–710.

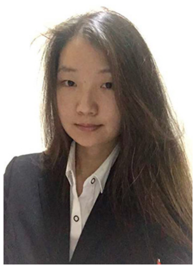
[58] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572 (2014).
[59] T. Szandała, Review and comparison of commonly used activation functions for deep neural networks, in: Bio-inspired neurocomputing, Springer, 2021, pp. 203–224.

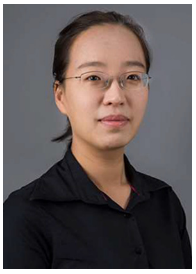
**Yuting Jia** received his B.E. degree in Computer Science from Shanghai Jiao Tong University, China, in 2017. He is currently pursuing his Ph.D. degree at the same university. His research of interests are in the area of data mining, community detection and network embedding.

**Shao Zhang** is currently a Ph.D. student at Shanghai Jiao Tong University, supervised by Prof. Xinbing Wang and Prof. Ying Wen. She obtained her MSc in Multimedia and Entertainment Technology (Game Development stream) from School of Design, The Hong Kong Polytechnic University in 2020. Before that, Shao earned her B.Eng in Industrial Design from School of Design, Hunan University in 2019. Her research of interests are in the area of human-AI collaboration, game ai and HCI4AI.

**Haiwen Wang** received the BE degree in Electronic Engineering from Sichuan University, China, in 2018. He is currently working toward the Ph.D. degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research of interests are in the area of data mining, machine learning, deep learning and social networks.

**Ying Wen** is a tenure-track Assistant Professor in John Hopcroft Center for Computer Science at Shanghai Jiao Tong University. He recieved his Ph.D. from the Department of Computer Science at the University College London in 2020. His research interests include machine learning, multi-agent systems and human-centered interactive systems etc. He has published over 20 research papers about machine learning on top-tier international conferences(ICML, NeurIPS, ICLR, IJCAI, and AAMAS). He has been serving as a PC member at ICML, NeurIPS, ICLR, AAAI, IJCAI, ICAPS and a reviewer at TIFS,Operational Research etc. He was granted Best Paper Award in AAMAS 2021 Blue Sky Track and the Best System Paper Award in CoRL 2020.


**Luoyi Fu** received her B. E. degree in Electronic Engineering from Shanghai Jiao Tong University, China, in 2009 and Ph.D. degree in Computer Science and Engineering in the same university in 2015. She is currently an Assistant Professor in Department of Computer Science and Engineering in Shanghai Jiao Tong University. Her research of interests are in the area of social networking and big data, scaling laws analysis in wireless networks, connectivity analysis and random graphs. She has been a member of the Technical Program Committees of several conferences including ACM MobiHoc 2018-2019, IEEE INFOCOM 2018-2019.


**Huan Long** is currently an associate professor in the department of computer science and engineering at Shanghai Jiao Tong University. She got her PhD degree from Shanghai Jiao Tong University in 2009. Her main research interest is in theoretical computer science, especially concurrency theory and data science.

**Xinbing Wang** received the B.S. degree (with hons.) from the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, in 1998, and the M.S. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2001. He received the Ph.D. degree, major in the Department of Electrical and Computer Engineering, minor in the Department of Mathematics, North Carolina State University, Raleigh, in 2006. Currently, he is a professor in the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. Dr. Wang has been an associate editor for IEEE/ACM Transactions on Networking and IEEE Transactions on Mobile Computing, and the member of the Technical Program Committees of several conferences including ACM MobiCom 2012, 2018-2019, ACM MobiHoc 2012-2014, IEEE INFOCOM 2009-2017.

**Chenghu Zhou** received the B.S. degree in geography from Nanjing University, Nanjing, China, in 1984, and the M.S. and Ph.D. degrees in geographic information system from the Chinese Academy of Sciences (CAS), Beijing, China, in 1987 and 1992, respectively. He is currently an Academician with CAS, where he is also a Research Professor with the Institute of Geographical Sciences and Natural Resources Research, and a Professor with the School of Geography and Ocean Science, Nanjing University. His research interests include spatial and temporal data mining, geographic modeling, hydrology and water resources, and geographic information systems and remote sensing applications.