This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

Local-Aggregation Graph Networks

Jianlong Chang, Lingfeng Wang, *Member, IEEE*, Gaofeng Meng*, *Senior, IEEE*, Qi Zhang, Shiming Xiang, *Member, IEEE* and Chunhong Pan, *Member, IEEE*

Abstract—Convolutional neural networks (CNNs) provide a dramatically powerful class of models, but are subject to traditional convolution that can merely aggregate permutation-ordered and dimension-equal local inputs. It causes that CNNs are allowed to only manage signals on Euclidean or grid-like domains (*e.g.*, images), not ones on non-Euclidean or graph domains (*e.g.*, traffic networks). To eliminate this limitation, we develop a local-aggregation function, a sharable nonlinear operation, to aggregate permutation-unordered and dimension-unequal local inputs on non-Euclidean domains. In the context of the function approximation theory, the local-aggregation function is parameterized with a group of orthonormal polynomials in an effective and efficient manner. By replacing the traditional convolution in CNNs with the parameterized local-aggregation function, Local-Aggregation Graph Networks (LAGNs) are readily established, which enable to fit nonlinear functions without activation functions and can be expediently trained with the standard back-propagation. Extensive experiments on various datasets strongly demonstrate the effectiveness and efficiency of LAGNs, leading to superior performance on numerous pattern recognition and machine learning tasks, including text categorization, molecular activity detection, taxi flow prediction, and image classification.

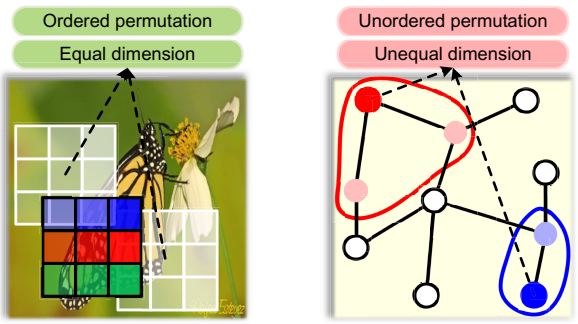
Index Terms—Local-aggregation function, local-aggregation graph neural network, non-Euclidean structured signal.

1 Introduction

Recently, convolutional neural networks (CNNs) have achieved significant success on a wide variety of pattern recognition and machine learning fields, ranging from natural language processing [1], [2], [3], [4] and computer version [5], [6], [7], [8], [9], [10], [11], [12] to playing games [13], [14], [15], [16]. The success mainly relies on an important assumption on the stationarity through local statistics, which are present in natural speeches, images and videos. Owing to this property, CNNs share parameters in the whole spatial domain, which can extremely reduce the number of parameters, without sacrificing the expressive capability.

In many contexts, however, one may face with non-Euclidean structured signals embedded on an underlying non-Euclidean or graph structure [17], [18], [19], [20], [21]. In practice, there are a large number of notable instances, such as traffic flow data in traffic networks, relational data in social networks, gene data in biological regulatory networks, text documents in word embedding, and active data in molecule structure networks. To represent these signals, graphs provide a generic framework, which consists of two main components, *i.e.*, vertices and edges. Generally, vertices store values of signals by using finite dimensional vectors, just as speeches and images do. Edges in graphs are utilized to reserve the structural information of signals.

Although traditional CNNs are potent architectures for managing extensive pattern recognition and machine learning tasks [11], [22], they are inherently incapable of dealing with non-Euclidean structured signals. The limitation originates from two key properties of signals on non-Euclidean domains, *i.e.*, unordered permutation and unequal dimension. One can see many practical instances, such as entry/exit routes of junctions in traffic networks, friends of users in social networks, and so on. These properties are the bottlenecks of extending CNNs to graphs in a straightforward way, since the traditional convolution is hard to handle permutation-unordered and dimension-unequal local inputs, as illustrated in Figure 1.



(a) Euclidean domain (b) non-Euclidean domain

Figure 1. Two intrinsic differences between (a) Euclidean and (b) non-Euclidean structured signals. On Euclidean domains, local inputs at each location (vertex) are permutation-ordered and dimension-equal, which allow a single sharable filter to aggregate these local inputs. On non-Euclidean domains, however, local inputs at each vertex are permutation-unordered and dimension-unequal, which are beyond the scope of the traditional convolution. For clarity, we exemplify these two properties in the figure (b) where the neighbors of the red and blue vertices are depicted in the red and blue circles, respectively. Specifically, the neighbors of the red and blue vertices are unordered, and their numbers are unequal.

In order to eliminate such limitation, we develop a local-aggregation function to establish local-aggregation graph networks (LAGNs) on non-Euclidean domains. Contrary to the convolution in the traditional CNNs, the

- *Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Qi Zhang, Shiming Xiang and Chunhong Pan are with the Department of National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. E-mail: {jianlong.chang, lfwang, gfmeng, qi.zhang2015, smxiang, chpan}@nlpr.ia.ac.cn.*
- *Jianlong Chang, Qi Zhang and Shiming Xiang are also with the School of artificial intelligence, University of Chinese Academy of Sciences, Beijing 100049, China.*
- ** Corresponding author.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

local-aggregation function is not only shareable across vertices but also capable of aggregating permutation-unordered and dimension-unequal local inputs. In the context of the function approximation theory, the local-aggregation function can be instantiated with a group of orthonormal polynomials in an effective and efficient manner. By replacing the convolution in traditional CNNs with the local-aggregation function, LAGNs are readily built, which can capture high-level representations of non-Euclidean structured signals and fit nonlinear functions without activation functions in deep models. Because of the differentiability of local-aggregation function, LAGNs can be expediently trained end-to-end by the standard back-propagation.

To sum up, the main contributions of this work are:

- In the context of the function approximation theory, we explore a *local-aggregation function*, a sharable operation, which provides a successful attempt to aggregate permutation-unordered and dimension-unequal local inputs on graphs.
- By stacking the local-aggregation function layer-by-layer, local-aggregation graph networks are readily established to manage both Euclidean and non-Euclidean structured signals, which dramatically broadens the reach of deep learning.
- By taking advantage of a set of linearly independent univariate functions, the local-aggregation function is instantiated in an effective and efficient manner, which permits that LAGNs can be end-to-end trained with the standard back-propagation.
- Contrary to the linear traditional convolution, the local-aggregation function can be considered as a nonlinear operation, which exploits a new precept for the nonlinear function estimation without activation functions in deep neural networks.

The remainder of this paper is organized as follows: in Section 2, we make a brief review on the related work of deep learning on graphs. Section 3 describes the details of the developed local-aggregation function. Section 4 introduces the established local-aggregation graph networks. Section 5 evaluates LAGNs with extensive experiments. Finally, Section 6 concludes this paper.

2 Related Work

Over the last decade, lots of efforts have been made to extend deep learning on graphs, *i.e.*, forming learnable functions on non-Euclidean domains [23], [24], [25], [26], [27]. Existing methods can be roughly summarized into two types, namely spatial and spectral methods.

For the spatial methods, two preliminary versions were developed to generalize neural networks to graph by combining recurrent neural networks and random walk models, proposed in [28], [29]. To exploit a shift-invariance operator on non-Euclidean domains, in recent, the geodesic convolution [30], the anisotropic diffusion [31], and the diffusion convolution [32] were developed to learn filters to aggregate the pre-defined patch operators. Further, Monti *et al.* proposed mixture model network [33] to learn filters and the patch operators jointly, which generalizes the previous models, *e.g.*, the geodesic convolution and the anisotropic diffusion. From the viewpoint of message passing, Gilmer *et al.* formulated message passing neural network [34], which is a general framework for managing signals on non-Euclidean domains. Compared with these spatial methods, the spectral methods produce convolution on graphs by feat of spectral graph theory [35]. The graph convolution on non-Euclidean domains was first defined in [36], [37]. Despite the conceptual breakthroughs, horrible time consumption limits the applicability of these spectral methods [36], [37]. To eliminate such limitation, Defferrard *et al.* introduced an efficient filtering scheme that does not require explicit computation of the Laplacian eigenvectors [38], [39]. Specifically, the details of these existing graph networks are reviewed in the following.

2.1 Spatial Methods

In analogy with the traditional convolution, spatial convolution methods devote to aggregating local inputs on spatial domains directly. A major problem in applying a convolution operator to non-Euclidan domains is shift-invariance, namely the operator should be position-dependent. To deal with such problem, several methods were explored by representing local patches in some intrinsic system of coordinates, *i.e.*,

$$D_j(x)f = \sum_{\bar{x}\in\mathcal{X}(x)} f(\bar{x})v_j(x,\bar{x})d\bar{x}, j = 1, \cdots, J,$$

where $\bar{x} \in \mathcal{X}(x)$ implies that $\bar{x}$ is a neighbor of x, $D_j(x)$ is the j-th operator to integrate the message at the neighbors of x, and $v_j(x,\cdot)$ is the j-th predefined weighting function located near x. According to this formulation, an intrinsic equivalent convolution providing for spatial domains can be formulated as follows:

$$(f * g)(x) = \sum_{j}^{J} g_j D_j(x) f,$$

where g indicates a group of learnable parameters for each operator $D_j(x)$, and acts as a filter on spatial domains. Furthermore, these filters are always localized on graphs and the learning complexities is $\mathcal{O}(1) = J$, which implies that a generally efficient convolutional framework in spatial domains is developed.

By employing with different weighting functions, various spatial models have been presented. Based on a local charting procedure in geodesic polar coordinates, Masci *et al.* [30] proposed geodesic convolution by introducing weighting functions as follows:

$$v_{ij}(x,\bar{x}) = \exp\left(-\frac{\left(\rho(\bar{x})-\rho_i\right)^2}{2\delta_\rho^2} - \frac{\left(\rho\left(\bar{x}\right)-\theta_j\right)^2}{2\delta_\theta^2}\right),$$

where i and j denote the indices of the radial and angular bins, respectively. Contrary to employing polar coordinates, the non-Euclidean heat equations can be also utilized as weighting functions around each vertex.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

Formally, Boscaini *et al.* [31] cast the anisotropic diffusion equation on graph domains, *i.e.*,

$$f_t(x,t) = -\text{div}\left(\mathbf{A}(x)\nabla f(x,t)\right),$$

where $f(x,t)$ signifies the temperature at vertex x and the diffusion time t, and $\mathbf{A}(x)$ is a heat conductivity tensor formulated as follows:

$$\mathbf{A}_{\alpha\theta}(x) = \mathbf{R}(x)\begin{pmatrix}\alpha & \\ & 1\end{pmatrix}\mathbf{R}_\theta^{\mathrm{T}}(x),$$

where $\mathbf{R}(x) \in \mathbb{R}^{2\times 2}$ executes rotation of θ, and α determines the degree of anisotropy. Furthermore, Monti et al. [33] generalized such models into a more generic spatial domain framework. Technically, the weighting function with learnable parameters $\mathbf{\Sigma}_j$ and μ_j are introduced rather than using the predefined ones, *i.e.*,

$$v_j\left(\mathbf{u}\right) = \exp\left(-\frac{1}{2}\left(\mathbf{u}-\mu_j\right)^{\mathrm{T}}\mathbf{\Sigma}_j^{-1}\left(\mathbf{u}-\mu_j\right)\right),$$

where $\mathbf{u}(x,\bar{x})$ indicates a local system of d-dimensional pseudo coordinates around x.

Different from the aforementioned methods that require a group of weighting functions, Gilmer *et al.* [34] proposed a message passing neural networks framework, which consists of two phase, a message passing phase and a readout phase. During the message passing phase, given an input state x_i^t of each vertex, hidden state x_i^{t+1} of each vertex can be obtained based on message functions M_t and vertex update functions U_t, *i.e.*,

$$\begin{aligned} x_i^{t+1} &= \sum_{j\in\mathcal{X}(j)} M_t\left(h_i^t, h_j^t, w_{ij}\right), \\ h_i^{t+1} &= U_t\left(h_i^t, x_i^{t+1}\right), \end{aligned}$$

where h_i^t is the i-th element in a graph, w_{ij} implies the connected weight between i-th vertex and j-th vertex, $\mathcal{X}(i)$ means the neighbors of the i-th vertex. For the readout phase, a readout function R is defined to calculated an output feature vector as follows:

$$\mathbf{y} = R\left(\mathbf{h}_i | i \in G\right).$$

where $\mathbf{y}$ is the output feature vector. Concurrent with our work, in addition, the graph network [40] generalizes various graph neural networks into a framework, including the message passing neural networks [34], [41], [42], [43], [44], [45], and the graph attention networks [46].

2.2 Spectral Methods

Contrary to the spatial methods, spectral methods explore an analogical convolution operator over non-Euclidean domains by feat of the spectral graph theory. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with n vertices, where $\mathcal{V}$ is a set of vertices with n elements, $\mathcal{E}$ stores a set of edges on the graph $\mathcal{G}$ and $\mathbf{W} \in \mathbb{R}^{n\times n}$ is a weighted adjacency matrix indicating connected weights between vertices. By generalizing the convolution on Euclidean domains, the spectral convolution is defined based on the graph Laplacian $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ represents the degree matrix. The Laplacian has a decomposition $\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}$, where $\mathbf{\Lambda} = diag(\lambda_1, \cdots, \lambda_n)$ signifies the diagonal matrix of eigenvalues and $\mathbf{\Phi} = (\Phi_1, \cdots, \Phi_n)$ is the matrix that consists of the eigenvectors corresponding to the eigenvalues. For two signals $\mathbf{f}$ and $\mathbf{g}$ on the graph $\mathcal{G}$, similar to the convolution in Euclidean domains, their spectral convolution can be calculated as follows:

$$\mathbf{f} * \mathbf{g} = \mathbf{\Phi}\left(\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{f}\right) \circ \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{g}\right)\right),$$

where $\circ$ is the element-wise Hadamard product. To introduce learnable filters, Bruna *et al.* [36] built spectral convolutional layer as follows:

$$\mathbf{f}_j^{out} = s\left(\sum_{i=1}^{p}\mathbf{\Phi}_k\mathbf{K}_{ij}\mathbf{\Phi}_k^{\mathrm{T}}\mathbf{f}_i^{in}\right),$$

where $\mathbf{f}_i^{in}$ and $\mathbf{f}_j^{out}$ signify the i-th input and j-th output respectively, $\mathbf{\Phi}_k = (\Phi_1, \cdots, \Phi_k)$ implies the first k eigenvectors, $\mathbf{K}_{ij} \in \mathbb{R}^{k\times k}$ is a learnable filter from $\mathbf{f}_i^{in}$ to $\mathbf{f}_j^{out}$, and $s(\cdot)$ is a nonlinearity activation function. There are three inescapable limitations in such spectral convolution. First, the convolution may be not localized in the spatial domain, which results in that a spectral convolutional filter may require $O(n)$ parameters to learn. Second, horrible time consumption limits the applicability of the spectral methods, since the eigenvectors of the Laplacian need to be calculated. Third, the convolution relies on graph domains, *i.e.*, a spectral CNN model learned on one graph cannot be transferred to other graphs.

In order to acquire spatially-localized filters and reduce the time consumption, Henaff *et al.* [37] formulated a parametric filter of the from

$$\bar{g}_i = \sum_{j=1}^{r}\alpha_j\beta_j(\lambda_i),$$

where $\beta_j(\cdot)$ and α_j are the j-th fixed cubic spline and the corresponding coefficient, respectively. Then, the spectral convolutional filter is denoted as $diag(\bar{\mathbf{G}}) = \mathbf{B}\alpha$, where $\mathbf{B} = (\beta_j(\lambda_i)) \in \mathbb{R}^{k\times r}$. Note that the learnable parameters in filters are independent of the input dimension.

To further alleviate the time consumption in the spectral convolution, ChebNets in [38] employe the Chebyshev polynomials to learn the spectral filters, *i.e.*,

$$g_\alpha\left(\bar{\mathbf{\Delta}}\right) = \sum_{j=0}^{r-1}\alpha_j\mathbf{\Phi}T_j\left(\bar{\mathbf{\Lambda}}\right)\mathbf{\Phi}^{\mathrm{T}} = \sum_{j=0}^{r-1}\alpha_j T_j\left(\bar{\mathbf{\Lambda}}\right),$$

where $\bar{\mathbf{\Delta}} = 2\lambda_n^{-1}\mathbf{\Delta} - \mathbf{I}$ and $\bar{\mathbf{\Lambda}} = 2\lambda_n^{-1}\mathbf{\Lambda} - \mathbf{I}$ are satisfied, λ_n signifies the maximum eigenvalues of the Laplacian $\mathbf{\Delta}$. Since the eigenvectors of the Laplacian are not required in this convolution, a fast convolution in graph domains is developed. Further, GCNs in [39] simplify the spectral convolution by parameterizing filters with a single parameter only, *i.e.*,

$$g_\alpha\left(\mathbf{f}\right) = \alpha\left(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{1/2}\mathbf{f}\right),$$

where $\mathbf{f}$ and $g_\alpha\left(\mathbf{f}\right)$ indicate the input and output signals on graph domains, respectively. In addition, since the ChebNets [38] and GCNs [39] are independent of the graph domains, which implies that spectral CNN models learned on one graph is sharable across different domains. Furthermore, these knowledgeable insights also form the bases of a number of methods, including adaptive graph convolutional neural networks [47], and CayleyNets [48].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

3 From Euclidean to non-Euclidean

As a core module in CNNs, intrinsically, convolution is a weighted summation operation defined on local inputs, which can be shared at each local location. The shareability mainly originates from the regular local inputs, which are provided with two key properties, *i.e.*, ordered permutation and equal dimension. The properties offer a fundamental basis to encode local inputs with dimension-equal vectors, resulting in the shareability of the defined sharable operation on local inputs, *i.e.*, Hadamard product between two vectors. In practice, Euclidean structured signals (*e.g.*, speeches, images and videos), inherently present these properties, and can be managed with CNNs compatibly. Compared with Euclidean structured signals, however, local inputs in non-Euclidean structured signals are permutation-unordered and dimension-unequal. It results in that the traditional convolution is hardly utilized to manage non-Euclidean structured signals.

Analogy to the structures of CNNs, we consider to develop a sharable local operation to replace the traditional convolution in CNNs, yielding a unified deep model of dealing with both Euclidean and non-Euclidean structured signals. Taking into account these properties of local inputs in non-Euclidean structured signals, two essential problems have to be tackled. First, a general coding scheme beyond the traditional vector-based technique is required to represent local inputs on Euclidean and non-Euclidean domains. Second, a sharable operation needs to be developed, which should be invariant to the permutation of elements in local inputs and capable of adapting dimension-unequal local inputs. In the following, Section 3.1 and 3.2 focus on tackling these two problems, respectively.

3.1 Local Inputs Coding

To adapt the variation of permutation and dimension, we code local inputs with sets. Technically, both Euclidean and non-Euclidean structured signals can be represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertices in $\mathcal{V}$ store the values of signals, the edges in $\mathcal{E}$ indicate whether two vertices are connected. For a signal $\mathbf{x}$ embedded on the graph $\mathcal{G}$, the local inputs at the i-th vertex can be represented as follows:

$$\mathcal{X}_i = \{x_j \mid e_{ji} \in \mathcal{E}\},\ i \in \mathcal{V}, \tag{1}$$

where $e_{ji} \in \mathcal{E}$ signifies that the j-th vertex is a neighbor of the i-th vertex. Such coding scheme possesses two considerable advantages. First, this scheme is independent of the permutations and dimensions of inputs, which suffices to represent arbitrary local inputs on both Euclidean and non-Euclidean domains. Second, the coding scheme is irrespective of specific weighted adjacency matrices, which can elevate the independence of our model and broaden the reach of applications consequently.

3.2 Local-Aggregation Function

Similar to the traditional convolution in CNNs, we introduce a local-aggregation function that is capable of aggregating any local input on graphs. In mathematical, any local-aggregation function possesses the same form [49]

Theorem 1. *For an arbitrary function $f(\cdot)$ defined on a set $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$, it is invariant to the permutation of elements in $\mathcal{X}$ if and only if it has the representation*

$$f(\mathcal{X}) = \rho\left(\sum_{x \in \mathcal{X}} \eta(x)\right) \tag{2}$$

for some continuous outer and inner functions $\rho(\cdot) : \mathbb{R}^{n+1} \to \mathbb{R}$ and $\eta(\cdot) : \mathbb{R} \to \mathbb{R}^{n+1}$, respectively.

From Theorem 1, we observe several constructive perspectives. First, any permutation invariant multivariate continuous function can be formed by Eq. (2). It guarantees that the formula is in a position to learn arbitrary mappings of sets. Second, the same decomposition is obtained for any permutation-unordered and dimension-unequal inputs. These perspectives indicate that we can share this operation at each vertex on both Euclidean and non-Euclidean domains, just as the traditional convolution does. Significantly, the perspectives are reassuring and supporting by the requisite theoretical guarantee.

Under the guidance of Theorem 1, we instantiate the local-aggregation function as a superposition of univariate continuous functions to aggregate permutation-unordered and dimension-unequal local inputs. For a signal $\mathbf{x}$ on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the output at the i-th vertex can be formulated as follows:

$$y_i = f(\mathcal{X}_i) = \rho\left(\sum_{e_{ji} \in \mathcal{E}} \eta(x_j)\right), \tag{3}$$

where $\mathcal{X}_i = \{x_j \mid e_{ji} \in \mathcal{E}\}$ is the local input at the i-th vertex, $e_{ji} \in \mathcal{E}$ signifies that the j-th vertex is a neighbor of the i-th vertex, $f(\cdot)$ is a sharable local-aggregation function defined on any local input, and y_i is the output at the i-th vertex. From Theorem 1, any permutation invariant multivariate continuous function can be decomposed as two parts, *i.e.*, the inner function $\eta(\cdot) : \mathbb{R} \to \mathbb{R}^T$ and the outer function $\rho(\cdot) : \mathbb{R}^T \to \mathbb{R}$, where T indicates the truncated dimensionality. In the first part, the local input $\mathcal{X}_i$ is mapped as a T-dimensional vector via the inner function $\eta(\cdot)$. In the second part, the outer function $\rho(\cdot)$ maps the T-dimensional vector to the desired target value. In the following, we instantiate the inner and outer functions with univariate continuous functions, respectively.

3.2.1 Inner Function Instantiation

Equivalently, the inner function $\eta(\cdot) : \mathbb{R} \to \mathbb{R}^T$ can be decomposed as T univariate functions $\{\psi_k(\cdot) : \mathbb{R} \to \mathbb{R}\}_{k=1}^T$. Denote the T-dimensional output of the inner function in Eq. (3) as $\mathbf{z} = [z_1, \cdots, z_k, \cdots, z_T] \in \mathbb{R}^T$, the k-th element z_k in $\mathbf{z}$ can be formulated as

$$z_k = \sum_{e_{ji} \in \mathcal{E}} \psi_k(x_j),\ k = 1, 2 \cdots, T, \tag{4}$$

where $\{\psi_k(\cdot) : \mathbb{R} \to \mathbb{R}\}_{k=1}^T$ are a group of sharable inner functions defined on scalars. Consequently, the local-aggregation function in Eq. (3) can be rewritten as

$$y_i = f(\mathcal{X}_i) = \rho(\mathbf{z}),\ \mathbf{z} \in \mathbb{R}^T, \tag{5}$$

where $\rho(\cdot) : \mathbb{R}^T \to \mathbb{R}$ is the outer function defined on the T-dimensional output of the inner functions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

Table 1
Various polynomial basis functions. For each basis function, the orthogonal interval $[a, b]$ and weight function $\sigma(x)$ are shown in the second and third columns, respectively. Each polynomial basis function can be generated based on the recurrence formula in the forth column.

Method	$[a, b]$	$\sigma(x)$	Recurrence formula: $\{h_1(x), h_2(x), \cdots, h_j(x), \cdots\}$ $(j \geq 3)$
Chebyshev	$[-1, 1]$	$\sqrt{1-x^2}$	$h_1(x) = 1, h_2(x) = 2x, h_j(x) = 2xh_{j-1}(x) - h_{j-2}(x)$.
Legendre	$[-1, 1]$	1	$h_1(x) = 1, h_2(x) = x, h_j(x) = \frac{2j-1}{j}xh_{j-1}(x) - \frac{j-1}{j}h_{j-2}(x)$.
Laguerre	$[0, +\infty]$	e^{-x}	$h_1(x) = 1, h_2(x) = 1 - x, h_j(x) = (2j - 1 - x)h_{j-1}(x) - (j-1)^2 h_{j-2}(x)$.
Hermite	$[-\infty, +\infty]$	e^{-x^2}	$h_1(x) = 1, h_2(x) = 2x, h_j(x) = 2xh_{j-1}(x) - 2(j-1)h_{j-2}(x)$.

3.2.2 Outer Function Instantiation

For the outer function $\rho(\cdot) : \mathbb{R}^T \rightarrow \mathbb{R}$, a group of univariate functions $\{\varphi_k(\cdot) : \mathbb{R} \rightarrow \mathbb{R}\}_{k=1}^T$ are employed for instantiating, beyond a linear transformation in the traditional convolution, *i.e.*,

$$y_i = f(\mathcal{X}_i) = \sum_{k=1}^{T} \varphi_k(z_k) = \sum_{k=1}^{T} \varphi_k \left(\sum_{e_{ji} \in \mathcal{E}} \psi_k(x_j) \right), \quad (6)$$

where z_k is the k-th element in the T-dimensional output of the inner functions, and $\{\varphi_k(\cdot) : \mathbb{R} \rightarrow \mathbb{R}\}_{k=1}^T$ are a group of sharable outer functions. Although such instantiation is simple, similar to the traditional CNNs, the expressive capability is profound when multiple local-aggregation functions are stacked.

4 Local-Aggregation Graph Networks

By replacing the traditional convolution in CNNs with the defined local-aggregation function, local-aggregation graph networks (LAGNs) are established readily. However, there is an essential problem that requires to be tackled before training LAGNs. That is, all the inner and outer functions in Eq. (5) are univariate functions, *i.e.*, infinite dimensional vectors, which need infinite parameters to determine. This implies that LAGNs cannot be learned in a common way, an effective and efficient strategy is requisite to learn the local-aggregation function with finite parameters. In the following, we devote to handling this problem and model LAGNs conclusively.

4.1 Local-Aggregation Function Parametrization

We parameterize univariate functions in the context of the approximation theory of functions [50]. Theoretically, for any continuous univariate function $h(\cdot)$, it can be composed of a group basis functions $\{h_1(x), h_2(x), \cdots\}$ and a set of coefficients $\{\theta_1, \theta_2, \cdots\}$, *i.e.*,

$$h(x) \simeq \lim_{m \rightarrow +\infty} \sum_{j=1}^{m} \theta_j \cdot h_j(x), \quad (7)$$

where $h_j(x)$ indicates the j-th basis function, θ_j is the corresponding coefficient, and the $h(x) = \sum_{j=1}^{m} \theta_j \cdot h_j(x)$ is satisfied when the basis functions are complete and k tends to infinity. According to Eq. (7), $h(\cdot)$ can be recoded by the coefficients $\{\theta_1, \theta_2, \cdots\}$, which implies that we can learn the coefficients to approximate the original function $h(\cdot)$ based on the basis functions.

Because of the high efficiency and linear independence [27], [38], [51], both the inner and outer functions are parameterized with orthonormal basis functions. Typically, four famous orthonormal polynomial bases in Table 1, *i.e.*, Chebyshev, Legendre, Laguerre and Hermite polynomials, are utilized. For each polynomial basis, the polynomial $h_j(x)$ of order $j-1$ $(j \geq 3)$ can be generated by a stable recurrence relation, with specific $h_1(x)$ and $h_2(x)$. These polynomials form an orthogonal basis for $L^2([a, b], \sigma(x)dx)$, the Hilbert space of square integrable functions with respect to the measure $\sigma(x)dx$, which guarantees the requisite fitting capability. Specifically, the truncated polynomials are employed to approximate both inner and outer functions, *i.e.*,

$$\psi_k(x) = \sum_{j=1}^{m} u_{kj} \cdot h_j(x), \ k = 1, 2, \cdots, T, \quad (8)$$

and

$$\varphi_k(x) = \sum_{j=1}^{m} v_{kj} \cdot h_j(x), \ k = 1, 2, \cdots, T, \quad (9)$$

where u_{kj} and v_{kj} are the j-th parameters corresponding to the k-th inner and outer functions, respectively.

The parameterization offers indispensable constant learning complexity and efficient computational strategy in practice. Technically, the local-aggregation function in Eq. (6) can be rewritten as

$$y_i = f(\mathcal{X}_i; \mathbf{u}, \mathbf{v}), \quad (10)$$

where $\mathbf{u} \in \mathbb{R}^{T \times m}$ and $\mathbf{v} \in \mathbb{R}^{T \times m}$ indicate learnable parameters in the inner and outer functions, respectively. It indicates that the learning complexities of this parameterization is $\mathcal{O}(1) = 2mT$, which is akin to the traditional convolution. Furthermore, both inner and outer functions can be cumulatively computed based on the stable recurrence relations, leading to an efficient computing strategy.

4.2 Local-Aggregation Graph Networks

We establish LAGNs by exploiting the developed local-aggregation function, building profound deep models for both Euclidean and non-Euclidean structured signals. During training we need to compute the gradients with respect to the parameters in the local-aggregation function, *i.e.*, $\mathbf{u}$ and $\mathbf{v}$. Given a loss ℓ, we have

$$\frac{\partial \ell}{\partial u_{kj}} = \frac{\partial \ell}{\partial y_i} \cdot \frac{\partial y_i}{\partial \varphi_k} \cdot h_j(x), \quad (11)$$

and

$$\frac{\partial \ell}{\partial v_{kj}} = \frac{\partial \ell}{\partial y_i} \cdot \frac{\partial y_i}{\partial \psi_k} \cdot h_j(x). \quad (12)$$

Therefore, the local-aggregation function is differentiable, enabling the end-to-end training of the established LAGNs with the standard back-propagation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

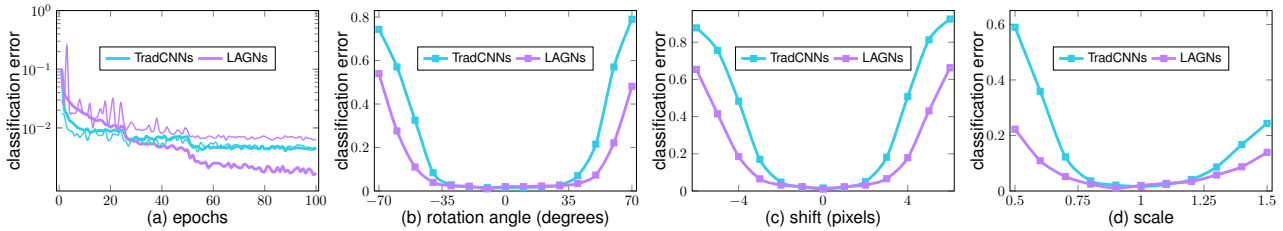


Figure 2. Invariance. (a) Classification errors during training (thick line) and validation (fine line), respectively. (b) Rotation. (c) Shift. (d) Scale.

5 Experiments

In this section, we carry out a series of experiments to demonstrate the effectiveness and efficiency of LAGNs on various pattern recognition and machine learning tasks, including classification and regression. Furthermore, extensive ablation experiments are conducted to systematically and comprehensively analyze the proposed LAGNs models. Specifically, the core code[1] of LAGNs is released at https://github.com/vector-1127/LAGNs.

5.1 Compared Methods

With respect to the mode of convolution, two types of methods are compared, *i.e.*, spatial and spectral graph convolutional methods. For the spatial methods, the fully connected neural networks (FCNs), the traditional CNNs (TradCNNs), the local connected networks (LCNs) [36], graph convolutional networks (GCNs) [54] and the mixture-model networks (MoNets) [33] (which is a generalization of the geodesic CNNs [30], the anisotropic CNNs [31], and the diffusion CNNs [32]) are adopted to compare with LAGNs. As for the spectral methods, we employ the spectral networks (SpeNets) [37] and the Chebyshev spectral networks (ChebNets) [38] to verify the performance of our method. In each specific task, traditional approaches that are proposed to handle the task, are also employed for comprehensive comparisons.

5.2 Experimental Settings

The hyper-parameters in our experiments are set as follows. In LAGNs, the max pooling and the Graclus method [55] are employed as the pooling operations to coarsen the feature maps in LAGNs when managing Euclidean and non-Euclidean structured signals respectively, the ReLU function [56] is utilized as the activation function, batch normalization [57] is employed to normalize the inputs of all layers, the whole parameters are randomly initialized with a uniform distribution $U(-0.05, 0.05)$, the biases are initialized to 0, the order of polynomials T is set to the maximum number of neighbors among the whole spatial domains (*e.g.*, $T = 9$ if we attempt to aggregate values of 3×3 pixels in images). During the training stage, the RMSprop optimizer [58] with the initial learning rate 0.001 is utilized to train LAGNs, the mini-batch size is set to 32, the categorical cross entropy loss is used in the classification tasks, and the mean squared error loss is used in the regression tasks. For a reasonable evaluation, we perform 5 random restarts and the average results are used for comparisons.

1. Relies on Keras [52] with the Tensorflow [53] backend.

Table 2
Classification accuracies on MNIST.

Model	Architecture	ACC
Logistic Regression	-	0.9173
KNN	-	0.9593
Random Forest	-	0.9710
Linear SVM	-	0.9784
FCNs1	*FC1024-FC512*	0.9905
FCNs2	*FC1024-FC1024-FC512*	0.9916
TradCNNs	*C32-P2-C64-P2-FC512*	**0.9942**
LCNs [36]	*C32-P2-C64-P2-FC512*	0.9924
MoNets [33]	*C32-P2-C64-P2-FC512*	0.9919
GCNs [54]	*C32-P2-C64-P2-FC512*	0.9847
SpeNets [37]	*C32-P2-C64-P2-FC512*	0.9848
ChebNets [38]	*C32-P2-C64-P2-FC512*	0.9887
LAGNs	*LA32-P2-LA64-P2-FC512*	0.9936

5.3 Revisiting Euclidean Structured Signals

To validate the capability of LAGNs on the Euclidean domains, we establish a LAGN to handle the classification problem on MNIST. This is an important extensible check for our model, which must be in a position to learn high-level representations on any graph, including the 2-dimensional regular grid. In the MNIST dataset, there are 70000 digital images (60000 for training and 10000 for testing, respectively) encoded on a 2-dimensional grid of size 28×28. In our experiment, we construct a graph to encode these digital images. Specifically, every pixel in images acts as a vertex in graphs, and there is an edge between vertices if two pixels are adjacent (*i.e.*, 8-pixel neighborhood). Consequently, the images are recast as graphs of $|\mathcal{V}| = 784$ vertices ($28 \times 28 = 784$) and $|\mathcal{E}| = 5940$ edges ($4 \times 3 + 104 \times 5 + 676 \times 8 = 5940$). During the comparison stage, a basic architecture similar to LeNet5 [59] is utilized to compare diversity approaches.

In Table 2, we report the quantitative results of the modeled networks with diverse convolution operations on the MNIST dataset. From the table, we observe that the performance of LAGN approximates to the TradCNNs and surpasses to the other graph convolution networks stably. These results demonstrate that LAGNs have enough capability to manage Euclidean structured signals with competitive performance.

In Figure 2, we empirically verify the invariance of the TradCNNs and LAGNs on the MNIST dataset. In this experiment, we disturb the testing data in MNIST with three typical transformations, including rotation, shift, and scale. Then, these disturbed data is utilized to validate the trained networks with the evaluated convolution operations. From Figure 2, the results assuredly prove that LAGNs are in possession of excellent robustness than the TradCNNs to such transformations. Further analysis,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

Table 3
Classification accuracies on 20NEWS.

Model	Architecture	ACC
Linear SVM	-	0.6590
Multinomial Naive Bayes	-	0.6851
Softmax	-	0.6628
FCNs1	*FC2500*	0.6464
FCNs2	*FC2500-FC500*	0.6576
LCNs [36]	*C20-FC512*	0.6625
MoNets [33]	*C20-FC512*	0.6929
GCNs [54]	*C20-FC512*	0.6675
SpeNets [37]	*C20-FC512*	0.6663
ChebNets [38]	*C20-FC512*	0.6826
LAGNs	*LA20-FC512*	**0.7254**

Table 4
Squared correlations on DPP4.

Model	Architecture	R^2
OLS Regression	-	0.135
Random Forest	-	0.232
Merck Winner DNN	-	0.224
FCNs1	*FC300*	0.173
FCNs2	*FC300-FC100*	0.195
LCNs [36]	*C20-FC512*	0.225
MoNets [33]	*C20-FC512*	0.243
GCNs [54]	*C20-FC512*	0.258
SpeNets [37]	*C10-C20-FC32*	0.248
ChebNets [38]	*C10-C20-FC32*	0.256
LAGNs	*LA10-LA20-FC32*	**0.273**

LAGNs represent local inputs as sets, where can definitively endow LAGNs desirable robustness.

5.4 Text Categorization on 20NEWS

To verify the performance of LAGNs to manage the non-Euclidean structured signals, we model a LAGN to classify the texts on the 20NEWS dataset. The 20NEWS dataset is a popular dataset in text applications, such as text classification and text clustering. It is a collection of 18846 text documents (11314 for training and 7532 for testing, respectively) associated with 20 categories. In this experiment, every document sample is represented as a graph instance. Following the previous work [38], specifically, 10000 frequently-used words are generated from the 93953 diverse words in the dataset. The bag-of-words model is employed to encode each document with the generated words, and the cosine similarity is used to estimate whether an edge is existed between pairwise vertices. In addition, three traditional models, linear SVM, multinomial naive bayes, and softmax classifiers, are employed to compare with LAGNs.

Table 3 lists the classification accuracies of the compared methods on 20NEWS. Note that the devised LAGN dramatically outperforms the other methods, which indicates that LAGNs are effective to deal with the classification problem. Furthermore, in contrast to the other graph convolution methods, more obvious enhances are achieved. It verifies that more reasonable local information can be captured by our local-aggregation function.

5.5 Merck Molecular Activity Challenge

We carry out an experiment on the DPP4 dataset[2] to handle the molecular activity prediction task with the developed LAGNs. Since molecules in DPP4 possess different structures and are represented as graphs, the task can be considered a regression problem on non-Euclidean domains. In the DPP4 dataset, there are 8193 molecules with 2796 features severally. Several baseline models, *i.e.*, ordinary least squares regression (OLS regression) and random forest, are employed for comparison. In addition, the result of the winner in this challenge is also collected to compare with our LAGN model. Following the previous work [37], [54], the squared correlation (R^2) is employed to measure the errors between the actual activity levels and the predicted activity levels.

The results of the compared methods are illustrated in Table 4. From the table, LAGNs are superior than the other methods consistently. Significant improvements originate from two main reasons. First, compared with these traditional methods (*e.g.*, random forest), the structure information is considered in LAGNs, which can assist our model to learn more beneficial feature representations for regression. Second, in contrast to other graph models, LAGNs can loose the requirement of the precise connected weights and learn a mapping from signals purely. Benefiting from such modeling, LAGNs enable to capture beneficial representations by avoiding learning connected weights or utilizing fixed connected weights.

5.6 Taxis Flow Prediction

In this experiment, we built LAGNs to manage the taxis flow prediction task. The target of this task is to calculate taxis flows at each traffic junction, which can be treated as a regression problem on non-Euclidean domains. For this task, we employ a taxis flow dataset based on the GPS trajectory data of 29950 taxis in Beijing from November 2015 to May 2016 [60]. In this dataset, 12549 samples are collected (10000 for training and 2549 for testing, respectively). For each sample, vertices are instantiated by 198 important traffic junctions in Beijing, and two vertices are connected if two corresponding traffic junctions are directly connected. In every vertex, six history observations (120 minutes) and the taxis flow of next time interval (20 minutes) are acted as inputs and outputs, respectively. In experiment, we employ ordinary least squares regression (OLS regression) as a baseline method for comparison. In addition, the root mean square error (RMSE) is used to evaluate performance of models.

The prediction results of various models are illustrated in Table 5. Notice that our method improves the baselines with a large margin. Intuitively, the linear method (*e.g.*, OLS regression) is insufficient for dealing with this problem, and the nonlinear approaches are required. For FCNs, although they possess the capability of nonlinear fitting, deeper models produce inferior results. A possible reason is that the over-fitting risk is occurred since more parameters need to be learned. Contrary to FCNs, the graph network methods (*e.g.*, SpeNets [37], ChebNets [38], *etc.*) can improve the performance sub-

2. https://www.kaggle.com/c/MerckActivity

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

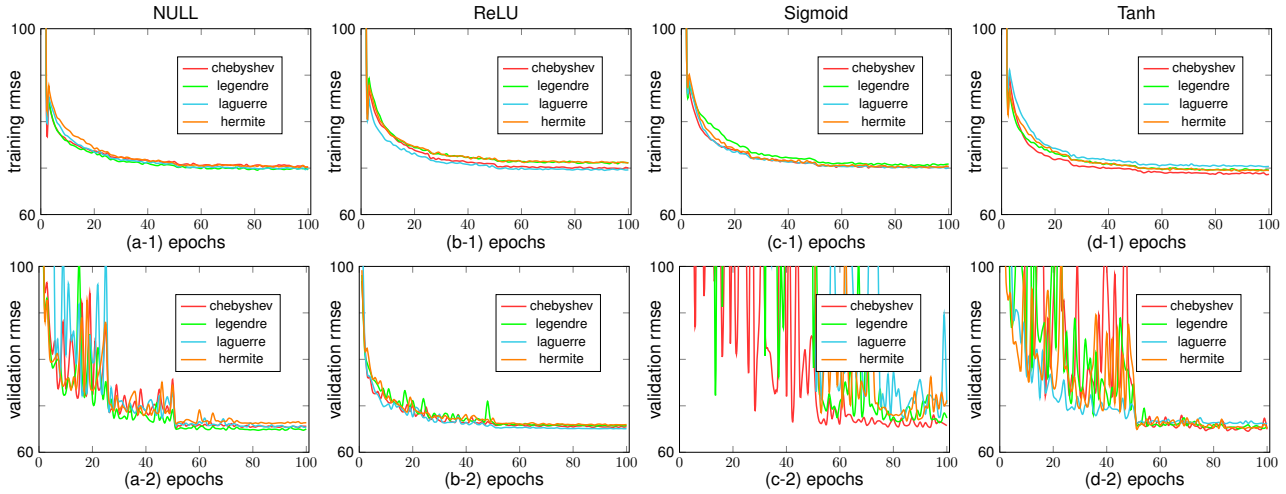


Figure 3. The root mean square error (rmse) during training and validation stages with diverse basis functions and activation functions.

Table 5
RMSE among various methods on the taxi flow data.

Model	Architecture	RMSE
OLS regression	-	84.20
FCNs1	*FC512-FC198*	75.30
FCNs2	*FC512-FC512-FC198*	80.60
LCNs [36]	*C32-C32-C1*	68.83
MoNets [33]	*C32-C32-C1*	69.35
GCNs [54]	*C32-C32-C1*	71.54
SpeNets [37]	*C32-C32-C1*	75.83
ChebNets [38]	*C32-C32-C1*	70.52
LAGNs	*LA32-LA32-LA1*	**62.71**

Table 6
Time consumption to process one epoch on MNIST during training.

Model	CPU (s)	GPU (s)	Speedup†
FCNs1	**40**	**16**	2.5×
FCNs2	65	22	3.0×
TradCNNs	112	**16**	7.0×
LCNs [36]	175	42	4.2×
MoNets [33]	252	59	4.3×
GCNs [54]	470	58	8.1×
SpeNets [37]	2768	262	10.6×
ChebNets [38]	1952	148	13.2×
LAGNs	863	46	**18.8**×

† Speedup indicates the acceleration from CPU to GPU.

stantially. This means that utilizing the structure information is effective to deal with non-Euclidean structured signals. However, the achievement of these graph convolutional methods is slight contrary to the random forest model. The highest promotion is achieved by the proposed LAGNs, which verifies that our model can learn more informative feature representations for predicting.

5.7 Ablation Study

In this subsection, extensive ablation studies are performed to investigate the performance of the developed LAGNs. Specifically, the taxi flow prediction task is considered as a basic task in the following experiments, since the graph structure in the traffic dataset is unambiguous. Furthermore, the LAGN designed in Section 5.6 is employed as a base model in the following experiments.

5.7.1 Computational Efficiency Comparison

We empirically investigate the computational complexity of the proposed LAGNs. Table 6 reports the time consumption of networks designed in Section 5.3 when one epoch is executed on the MNIST dataset during training. From Table 6, we observe that the spatial models (*e.g.*, LAGNs) are faster than the spectral models dramatically. It is to be expected, since a great number of operations related with Laplacian matrix need to be executed in the spectral models. Compared with the spatial models, the proposed method requires to spend more time in CPUs. Fortunately, the cumulative computing formulation offers the expedite opportunity on GPUs, which is exemplified by the results and guarantees the practicability of LAGNs.

5.7.2 Influence of Basis Functions

To investigate the influence of basis functions and diverse activation functions, four bases, *i.e.*, Chebyshev, Legendre, Laguerre and Hermite polynomials, are orderly employed to evaluate LAGNs. Intuitively, Figure 3 illustrates the results of LAGNs with different basis functions. In the figure, one tendency can be observed, namely the basis functions can slightly impact performance of LAGNs. This means that the back-propagation algorithm has enough capability to guide LAGNs to yield acceptable solutions with tolerable differences. Such differences may originate from the orthogonal intervals of these polynomial, *i.e.*, the orthogonality may influence the performance of the local-aggregation function.

5.7.3 Capability of Nonlinear Function Fitting

To explore the capability of nonlinear function fitting of the developed local-aggregation function, we compare the performance of LAGNs when the activation functions are equipped or unequipped (denoted as "NULL" in Figure 3). From the figure, we observe that the performance of LAGNs will not be degenerated after the activation function is omitted. It significantly demonstrates that our local-aggregation function inherently suffices to estimate nonlinear function, without activation functions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

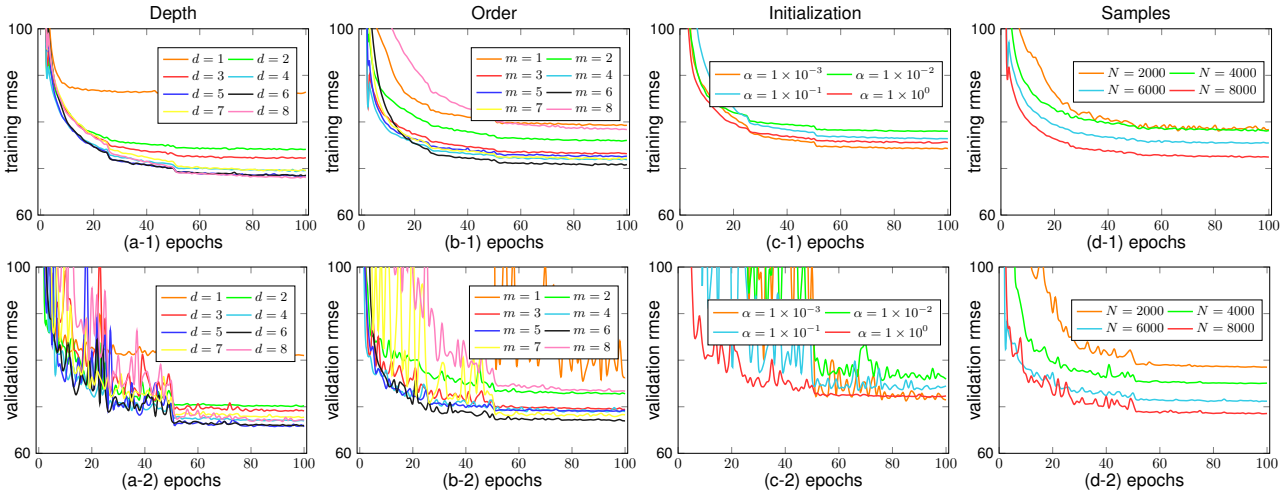


Figure 4. Training and validation errors with diverse (a) depth of LAGNs, (b) polynomial orders, (c) initializations, and (d) number of samples.

5.7.4 Impact of Activation Function

In order to reveal the impact of the activation function, several LAGNs are modeled with various activation functions, *i.e.*, sigmoid, tanh, ReLU [56] and NULL (no activation functions). In Figure 3, we observe that the influence of activation functions is greater than the influence of basis functions. One reason is that the gradient vanishing problem may be introduced by the traditional activation functions (*e.g.*, sigmoid and tanh). For ReLU [56], it can weaken the gradient vanishing problem effectively and improve the performance of LAGNs consequently. This observation is similar to the impact of activation functions in the TradCNNs, namely the activation function can definitively determine the performance of networks.

5.7.5 Contribution of Depth

We establish several LAGNs with different depths to evaluate the contribution of the depth. Specifically, each layer in these networks consist of the 32 local-aggregation functions. By stacking this layer repeatedly, the LAGNs with d layers are obtained. Intuitively, Figure 4 (a) illustrates the results when the number of layers various between 1 and 8 with an interval 1. From Figure 4 (a), several observation can be received. First, the performance gradually improves if we increase the depth d, then the performance tends to be saturated when filters can be well approximated, *i.e.*, $d = 5$ or $d = 6$. It corresponds to our scenario that deeper models possess stronger capability of feature learning and representing than shallow models. Second, deeper LAGNs also have the over-fitting risk, which is similar to the traditional deep neural networks.

5.7.6 Effect of Polynomial Order

In this experiment, we evaluate the effect of the various order of polynomials basis functions in LAGNs. The results are illustrated in Figure 4 (b), in which m indicates the order of the utilized polynomial. Notice that appropriately improving the order is beneficial for learning. This is because that the higher-order polynomial can improve the power of nonlinear function fitting to capture high-level feature representations. However, excessive order may degrade capability of LAGNs. There are two main reasons for the degeneration. First, utilizing more complex basis functions to approximate graph-based filters increases the complexity of learning. Second, the over-fitting risk may be proliferated, as the order increases.

5.7.7 Sensitivity of Initialization

We conduct an experiment to study the sensitivity of initialization in LAGNs. Specifically, parameters of graph-based filters are initialized with a uniform distribution $U(-1 \times 10^{\alpha}, 1 \times 10^{\alpha})$, where α various from -3 to 0 with an interval 1. From Figure 4 (c), a conspicuous tendency can be observed, *i.e.*, the initial state of LAGNs has an obvious effect on the performance of LAGNs. In the figure, a proper α contributes to train LAGNs better (*e.g.*, $\alpha = -3$). If a smaller α is selected, the training will be unstable (*e.g.*, $\alpha = -2$) or convergent slowly.

5.7.8 Performance on Various Number of Samples

To observe the effect of the number of samples N to our method, we vary N in $\{2000,\ 4000,\ 6000,\ 8000\}$ on the taxi flow dataset. Intuitively, Figure 4 (d) shows that the performance of LAGNs improves with more samples, which is similar to the traditional deep models. The main reason is that more training samples will describe the original distribution of data more accurately and make the estimated discrimination functions more robust. Furthermore, it corresponds to the actual scenario that the influence of the number of samples will be declined when the performance of LAGNs tends to saturation.

5.7.9 Feature Maps Visualization

In this experiment, we carry out an additional experiment on STL-10 to qualitatively analyze the feature maps learned by LAGNs. The STL-10 dataset consists of 13000 color images (10 classes) with size of $96 \times 96 \times 3$. For each class, 500 images are utilized to train models and 800 images are used for evaluation. Generally, we design a TradCNN and a LAGN (listed in Table 7) to visualize the learned feature maps. Similar to the experiment on MNIST, we construct a graph on the 2-dimensional grid to yield three-channels graph of $|\mathcal{V}| = 96 \times 96 \times 3 = 27648$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

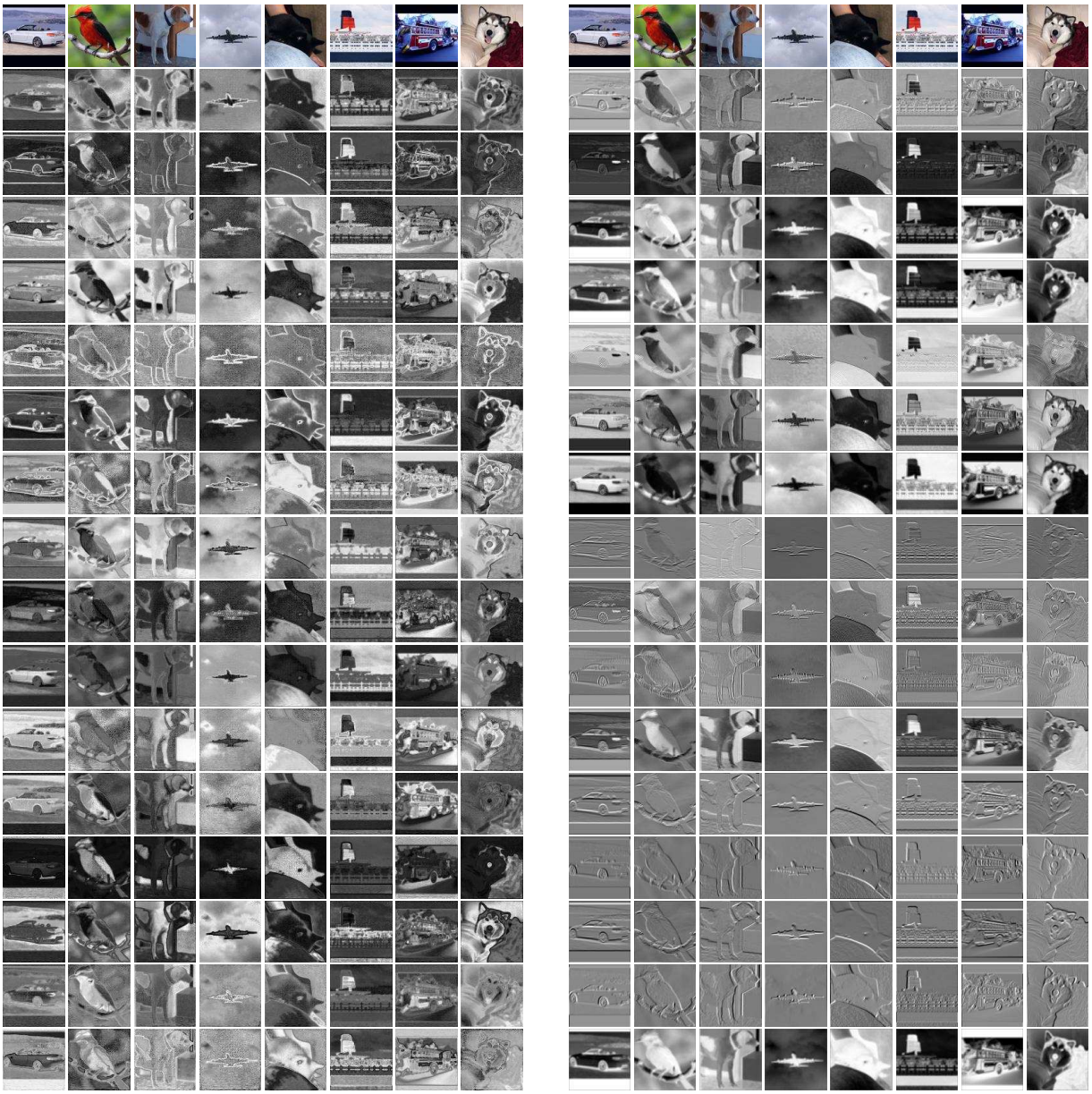


Figure 5. The feature maps learned by the experimental (left) LAGN and (right) TradCNN on the STL-10 dataset. Specifically, the input images are illustrated on the upward side, and the sixteen feature maps in the first layers of the LAGN and TradCNN are shown orderly.

vertices and $|\mathcal{E}| = (4\times3+376\times5+8836\times8)\times3 = 217740$ edges, which corresponds to three channels in images.

The quantitative and qualitative results are illustrated in Table 7 and Figure 5, respectively. In Table 7, the classification results of two networks indicate that TradCNNs are more suitable for image classification tasks than the proposed LAGNs. A possible reason is that LAGNs can estimate classification function without precise local permutation. With the tolerable accuracies, however, LAGNs are more robust than TradCNNs, as demonstrated in Figure 2. Visually, Figure 5 illustrates the whole learned feature maps in the experimental TradCNN and LAGN. From Figure 5 (left), notice that TradCNNs pay more attention to edge information, which can be treated as the low-level features of original images. Compared with TradCNNs, Figure 5 (right) shows that LAGNs can focus on not only edge information, but also objects or part of objects in images, which are profitable for various tasks.

Table 7
Classification accuracies of TradCNNs and LAGNs on STL-10.

Model	Architecture	ACC
TradCNNs	*C16-P4-FC512*	63.43
LAGNs	*LA16-P4-FC512*	62.66

6 Conclusion

We introduce a local-aggregation function, which suffices to aggregate permutation-unordered and dimension-unequal local inputs. In the context of the function approximation theory, the explicit form of the local-aggregation function is acquired. Based on such theoretical guidance, we parameterize the local-aggregation function with a group of orthonormal polynomials, which can be cumulatively computed in an efficient computing manner. By replacing the traditional convolution in CNNs with the parameterized local-aggregation function,

LAGNs are expediently established to manage both Euclidean and non-Euclidean structured signals. In comparison with the existing graph convolutional approaches, the proposed method has achieved superior performance on various pattern recognition and machine learning tasks, including text categorization, molecular activity detection, taxi flow prediction, and image classification.

Future work includes generating more accurate relationships between different vertices and exploring preferable approaches to learn the local-aggregation function. For the first problem, we have used the cosine distance to estimate relationships between vertices as some pioneering work (*e.g.*, [37], [38], [54]). However, unavoidable errors may be introduced because of the noise during training. It will be interesting in the future to automatically generate relationships to improve the performance. As for the second problem, we have parameterized graph-based filters relying on a set of predefined basis functions in this work. Nevertheless, the flexibility of the local-aggregation function may be limited. In the future, more appropriate methods could be developed for significant promotions.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grants 91646207, 61773377, and 61573352, and the Beijing Natural Science Foundation under Grant L172053. We would like to thank Lele Yu, Jie Gu, Cheng Da, Xue Ye, and Tingzhao Yu for their discussions in shaping the early stage of this work. We furthermore thank the anonymous reviewers for their constructive comments earnestly.

REFERENCES

[1] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit." in *ACL*, 2014, pp. 55–60.

[2] R. Sarikaya, G. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *T-ASLP*, vol. 22, no. 4, pp. 778–784, 2014.

[3] O. Irsoy and C. Cardie, "Deep recursive neural networks for compositionality in language," in *NeurIPS*, 2014, pp. 2096–2104.

[4] J. Hirschberg and C. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.

[5] A. Coates and A. Ng, "Selecting receptive fields in deep networks," in *NeurIPS*, 2011, pp. 2528–2536.

[6] K. Gregor and Y. LeCun, "Emergence of complex-like cells in a temporal product network with local receptive fields," *CoRR*, vol. abs/1006.0448, 2010.

[7] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Good practice in large-scale learning for image classification," *T-PAMI*, vol. 36, no. 3, pp. 507–520, 2014.

[8] M. Cheng, N. Mitra, X. Huang, P. Torr, and S. Hu, "Global contrast based salient region detection," *T-PAMI*, vol. 37, no. 3, pp. 569–582, 2015.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *T-PAMI*, vol. 38, no. 1, pp. 142–158, 2016.

[10] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *ICCV*, 2017, pp. 5880–5888.

[11] J. Pont-Tuset, P. Arbelaez, J. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *T-PAMI*, vol. 39, no. 1, pp. 128–140, 2017.

[12] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep unsupervised learning with consistent inference of latent representations," *Pattern Recognition*, vol. 77, pp. 438–453, 2018.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[15] V. Mnih, A. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016, pp. 1928–1937.

[16] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[17] M. Schlichtkrull, T. Kipf, P. Bloem, R. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*, 2018, pp. 593–607.

[18] N. Verma, E. Boyer, and J. Verbeek, "Dynamic filters in graph convolutional networks," *CoRR*, vol. abs/1706.05206, 2017.

[19] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *NAACL-HLT*, 2018, pp. 464–468.

[20] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.

[21] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *CVPR*, 2017, pp. 29–38.

[22] J. Chang, G. Meng, L. Wang, S. Xiang, and C. Pan, "Deep self-evolution clustering," *T-PAMI*, 2019.

[23] M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, 2017.

[24] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive model," *CoRR*, vol. abs/1802.08773, 2018.

[25] R. Ying, R. He, K. Chen, P. Eksombatchai, W. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," *arXiv preprint arXiv:1806.01973*, 2018.

[26] J. Chen and J. Zhu, "Stochastic training of graph convolutional networks," *arXiv preprint arXiv:1710.10568*, 2017.

[27] J. Chang, J. Gu, L. Wang, G. Meng, S. Xiang, and C. Pan, "Structure-aware convolutional neural networks," in *NeurIPS*, 2018, pp. 11–20.

[28] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *IJCNN*, vol. 2. IEEE, 2005, pp. 729–734.

[29] F. Scarselli, M. Gori, A. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *T-NN*, vol. 20, no. 1, pp. 61–80, 2009.

[30] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *ICCV Workshop*, 2015, pp. 832–840.

[31] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *NeurIPS*, 2016, pp. 3189–3197.

[32] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *NeurIPS*, 2016, pp. 1993–2001.

[33] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *CVPR*, 2017, pp. 5425–5434.

[34] J. Gilmer, S. Schoenholz, P. Riley, O. Vinyals, and G. Dahl, "Neural message passing for quantum chemistry," in *ICML*, 2017, pp. 1263–1272.

[35] F. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[36] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2915591, IEEE Transactions on Pattern Analysis and Machine Intelligence

[37] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *CoRR*, vol. abs/1506.05163, 2015.

[38] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NeurIPS*, 2016, pp. 3837–3845.

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

[40] P. Battaglia, J. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *CoRR*, vol. abs/1806.01261, 2018.

[41] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *NeurIPS*, 2015, pp. 2224–2232.

[42] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *CoRR*, vol. abs/1511.05493, 2015.

[43] P. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in *NeurIPS*, 2016, pp. 4502–4510.

[44] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, pp. 595–608, 2016.

[45] K. Schütt, F. Arbabzadah, S. Chmiela, K. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nature communications*, vol. 8, p. 13890, 2017.

[46] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *CoRR*, vol. abs/1710.10903, 2017.

[47] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *AAAI*, 2018.

[48] R. Levie, F. Monti, X. Bresson, and M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *CoRR*, vol. abs/1705.07664, 2017.

[49] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. Smola, "Deep sets," in *NeurIPS*, 2017, pp. 3394–3404.

[50] E. Kreyszig, *Introductory functional analysis with applications*. wiley New York, 1989, vol. 1.

[51] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[52] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[53] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," http://tensorflow.org/, 2015.

[54] Y. Hechtlinger, P. Chakravarti, and J. Qin, "A generalization of convolutional neural networks to graph-structured data," *arXiv preprint arXiv:1704.08165*, 2017.

[55] I. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *T-PAMI*, vol. 29, no. 11, 2007.

[56] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011, pp. 315–323.

[57] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456.

[58] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[59] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

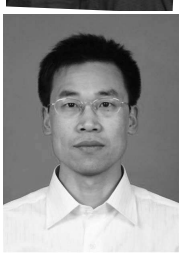
[60] Q. Zhang, Q. Jin, J. Chang, S. Xiang, and C. Pan, "Kernel-weighted graph convolutional network: A deep learning approach for traffic forcasting," in *ICPR*, 2018, pp. 1018–1023.

Jianlong Chang received the B.S. degree in School of Mathematical Sciences from University of Electronic Science and Technology of China, Chengdu, China, in 2015. He is currently pursuing the Ph.D. degree at the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include relation-based deep learning, deep graph networks, Auto ML.

Lingfeng Wang received his B.S. degree in computer science from Wuhan University, Wuhan, China, in 2007. He received his Ph.D. degree from Institute of Automation, Chinese Academy of Sciences in 2013. He is currently an associate professor with the National Laboratory of Pattern Recognition of Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision and image processing.

Gaofeng MENG received the B.S. degree in Applied Mathematics from Northwestern Polytechnical University in 2002, and the M.S. degree in Applied Mathematics from Tianjing University in 2005, and the Ph.D degree in Control Science and Engineering from Xi'an Jiaotong University in 2009. He is currently an associate professor of the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. He also serves as an Associate Editor of Neurocomputing. His current research interests include document image processing, computer vision and pattern recognition.

Qi Zhang received the B.S. degree in automatic control from Huazhong University of Science & Technology (HUST), Wuhan, China, in 2015. Currently, he is a second-year master student in National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include pattern recognition, and data mining.



Shiming Xiang received the B.S. degree in mathematics from Chongqing Normal University, Chongqing, China, in 1993, the M.S. degree from Chongqing University, Chongqing, China, in 1996, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2004. From 1996 to 2001, he was a Lecturer with the Huazhong University of Science and Technology, Wuhan, China. He was a Postdoctorate Candidate with the Department of Automation, Tsinghua University, Beijing, China, until 2006. He is currently a Professor with the National Lab of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include pattern recognition, and machine learning.

Chunhong Pan received his B.S. Degree in automatic control from Tsinghua University, Beijing, China, in 1987, his M.S. Degree from Shanghai Institute of Optics and Fine Mechanics, Chinese Academy of Sciences, China, in 1990, and his Ph.D. degree in pattern recognition and intelligent system from Institute of Automation, Chinese Academy of Sciences, Beijing, in 2000. He is currently a professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision, image processing, computer graphics, and remote sensing.