例会

◎ 论文1

Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." IEEE Transactions on Neural Networks and Learning Systems (2020)

A Comprehensive Survey on Graph Neural Networks

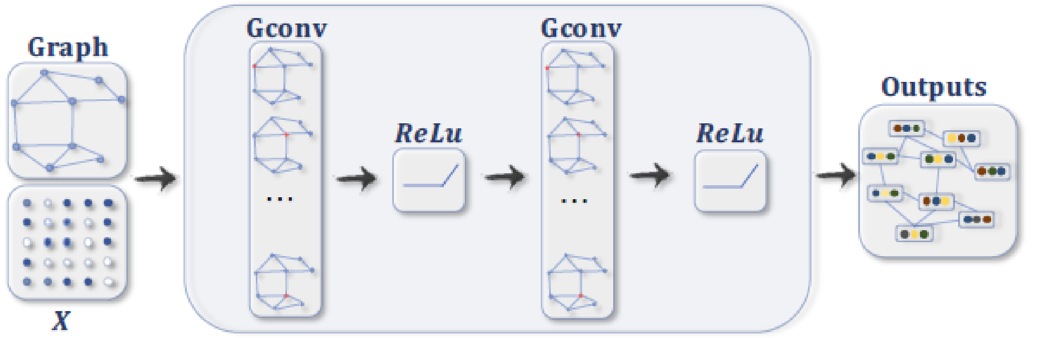
Zonghan Wu, Shirui Pan, *Member, IEEE*, Fengwen Chen, Guodong Long,
Chengqi Zhang, *Senior Member, IEEE*, Philip S. Yu, *Fellow, IEEE*

Abstract—Deep learning has revolutionized many machine learning tasks in recent years, ranging from image classification and video processing to speech recognition and natural language understanding. The data in these tasks are typically represented in the Euclidean space. However, there is an increasing number of applications where data are generated from non-Euclidean domains and are represented as graphs with complex relationships and interdependency between objects. The complexity of graph data has imposed significant challenges on existing machine learning algorithms. Recently, many studies on extending deep learning approaches for graph data have emerged. In this survey, we provide a comprehensive overview of graph neural networks (GNNs) in data mining and machine learning fields. We propose a new taxonomy to divide the state-of-the-art graph neural

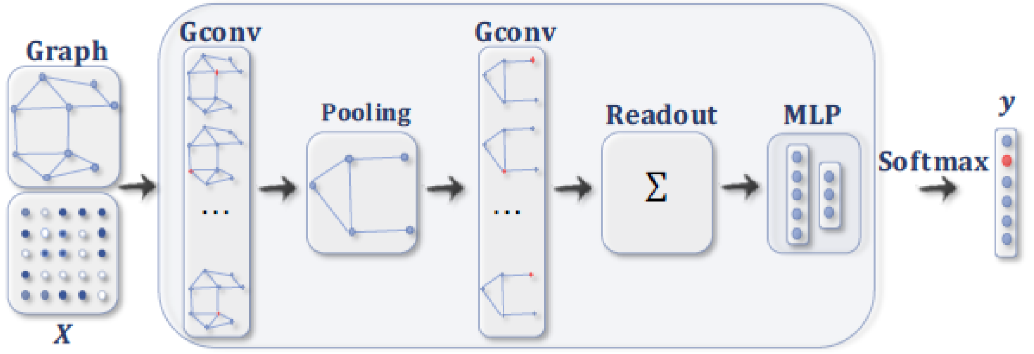
example, we can represent an image as a regular grid in the Euclidean space. A convolutional neural network (CNN) is able to exploit the shift-invariance, local connectivity, and compositionality of image data [9]. As a result, CNNs can extract local meaningful features that are shared with the entire data sets for various image analysis.

While deep learning effectively captures hidden patterns of Euclidean data, there is an increasing number of applications where data are represented in the form of graphs. For examples, in e-commence, a graph-based learning system can exploit the interactions between users and products to make highly accurate recommendations. In chemistry, molecules

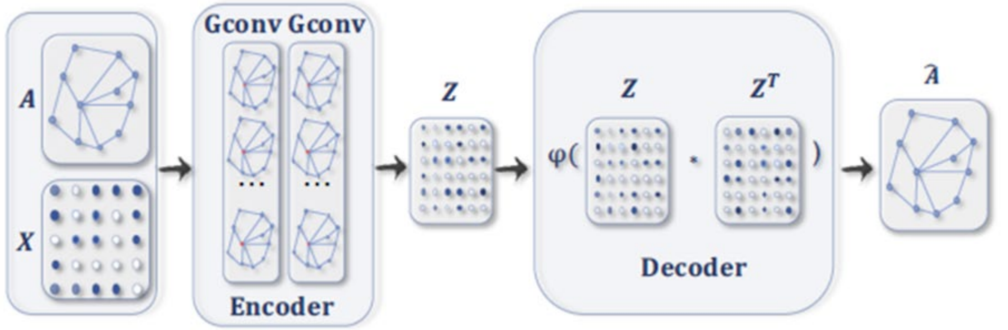
◎ 论文1



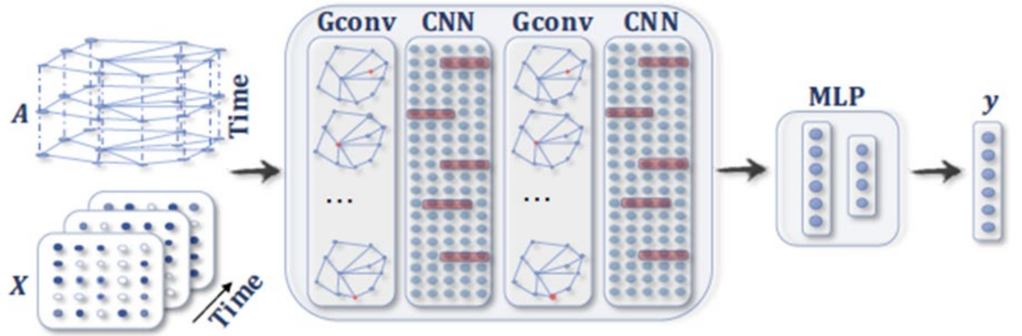
(a) A ConvGNN with multiple graph convolutional layers. A graph convolutional layer encapsulates each node's hidden representation by aggregating feature information from its neighbors. After feature aggregation, a non-linear transformation is applied to the resulted outputs. By stacking multiple layers, the final hidden representation of each node receives messages from a further neighborhood.



(b) A ConvGNN with pooling and readout layers for graph classification [21]. A graph convolutional layer is followed by a pooling layer to coarsen a graph into sub-graphs so that node representations on coarsened graphs represent higher graph-level representations. A readout layer summarizes the final graph representation by taking the sum/mean of hidden representations of sub-graphs.



(c) A GAE for network embedding [61]. The encoder uses graph convolutional layers to get a network embedding for each node. The decoder computes the pair-wise distance given network embeddings. After applying a non-linear activation function, the decoder reconstructs the graph adjacency matrix. The network is trained by minimizing the discrepancy between the real adjacency matrix and the reconstructed adjacency matrix.



(d) A STGNN for spatial-temporal graph forecasting [74]. A graph convolutional layer is followed by a 1D-CNN layer. The graph convolutional layer operates on A and $X^{(t)}$ to capture the spatial dependency, while the 1D-CNN layer slides over X along the time axis to capture the temporal dependency. The output layer is a linear transformation, generating a prediction for each node, such as its future value at the next time step.

◎ 论文1

TABLE II: Taxonomy and representative publications of Graph Neural Networks (GNNs)

Category		Publications
Recurrent Graph Neural Networks (RecGNNs)		[15], [16], [17], [18]
Convolutional Graph Neural Networks (ConvGNNs)	Spectral methods	[19], [20], [21], [22], [23], [40], [41]
	Spatial methods	[24], [25], [26], [27], [42], [43], [44] [45], [46], [47], [48], [49], [50], [51] [52], [53], [54], [55], [56], [57], [58]
Graph Autoencoders (GAEs)	Network Embedding	[59], [60], [61], [62], [63], [64]
	Graph Generation	[65], [66], [67], [68], [69], [70]
Spatial-temporal Graph Neural Networks (STGNNs)		[71], [72], [73], [74], [75], [76], [77]

ABLE III: Summary of RecGNNs and ConvGNNs. Missing values ("-") in pooling and readout layers indicate that the method nly experiments on node-level/edge-level tasks.

Approach	Category	Inputs	Pooling	Readout	Time Complexity
GNN* (2009) [15]	RecGNN	A, X, X^e	-	a dummy super node	$O(m)$
GraphESN (2010) [16]	RecGNN	A, X	-	mean	$O(m)$
GGNN (2015) [17]	RecGNN	A, X	-	attention sum	$O(m)$
SSE (2018) [18]	RecGNN	A, X	-	-	-
Spectral CNN (2014) [19]	Spectral-based ConvGNN	A, X	spectral clustering+max pooling	max	$O(n^3)$
Henaff et al. (2015) [20]	Spectral-based ConvGNN	A, X	spectral clustering+max pooling		$O(n^3)$
ChebNet (2016) [21]	Spectral-based ConvGNN	A, X	efficient pooling	sum	$O(m)$
GCN (2017) [22]	Spectral-based ConvGNN	A, X	-	-	$O(m)$
CayleyNet (2017) [23]	Spectral-based ConvGNN	A, X	mean/graclus pooling	-	$O(m)$
AGCN (2018) [40]	Spectral-based ConvGNN	A, X	max pooling	sum	$O(n^2)$
DualGCN (2018) [41]	Spectral-based ConvGNN	A, X	-	-	$O(m)$
NN4G (2009) [24]	Spatial-based ConvGNN	A, X	-	sum/mean	$O(m)$
DCNN (2016) [25]	Spatial-based ConvGNN	A, X	-	mean	$O(n^2)$
PATCHY-SAN (2016) [26]	Spatial-based ConvGNN	A, X, X^e	-	sum	-
MPNN (2017) [27]	Spatial-based ConvGNN	A, X, X^e	-	attention sum/set2set	$O(m)$
GraphSage (2017) [42]	Spatial-based ConvGNN	A, X	-	-	-
GAT (2017) [43]	Spatial-based ConvGNN	A, X	-	-	$O(m)$
MoNet (2017) [44]	Spatial-based ConvGNN	A, X	-	-	$O(m)$
LGCN (2018) [45]	Spatial-based ConvGNN	A, X	-	-	-
PGC-DGCNN (2018) [46]	Spatial-based ConvGNN	A, X	sort pooling	attention sum	$O(n^3)$

◎ 论文1

Node-level输出用于点回归和分类任务

Edge-level输出与边分类和链路预测任务相关

Graph-level的输出用来做图分类任务

Semi-supervised learning：节点分类

Supervised learning：图分类

Unsupervised learning：图嵌入

◎ 论文1

TABLE VI: Summary of selected benchmark data sets.

Category	Data set	Source	# Graphs	# Nodes(Avg.)	# Edges (Avg.)	#Features	# Classes	Citation
Citation Networks	Cora	[117]	1	2708	5429	1433	7	[22], [2 [49], [5
	Citeseer	[117]	1	3327	4732	3703	6	[22], [4 [56], [6
	Pubmed	[117]	1	19717	44338	500	3	[18], [2 [49], [5 [70], [9
	DBLP (v11)	[118]	1	4107340	36624464	-	-	[64], [7
Bio-chemical Graphs	PPI	[119]	24	56944	818716	50	121	[18], [4 [56], [5
	NCI-1	[120]	4110	29.87	32.30	37	2	[25], [2
	MUTAG	[121]	188	17.93	19.79	7	2	[25], [2
	D&D	[122]	1178	284.31	715.65	82	2	[26], [4
	PROTEIN	[123]	1113	39.06	72.81	4	2	[26], [4
	PTC	[124]	344	25.5	-	19	2	[25], [2
	QM9	[125]	133885	-	-	-	-	[27], [6
	Alchemy	[126]	119487	-	-	-	-	-
Social Networks	Reddit	[42]	1	232965	11606919	602	41	[42], [4
	BlogCatalog	[127]	1	10312	333983	-	39	[18], [5
Others	MNIST	[128]	70000	784	-	1	10	[19], [2
	METR-LA	[129]	1	207	1515	2	-	[48], [7
	Nell	[130]	1	65755	266144	61278	210	[22], [4

TABLE VIII: A Summary of Open-source Implementations

Model	Framework	Github Link
GGNN (2015)	torch	https://github.com/yujiali/ggnn
SSE (2018)	c	https://github.com/Hanjun-Dai/steady_state_embeddi
ChebNet (2016)	tensorflow	https://github.com/mdeff/cnn_graph
GCN (2017)	tensorflow	https://github.com/tkipf/gcn
CayleyNet (2017)	tensorflow	https://github.com/amoliu/CayleyNet.
DualGCN (2018)	theano	https://github.com/ZhuangCY/DGCN
GraphSage (2017)	tensorflow	https://github.com/williamleif/GraphSAGE
GAT (2017)	tensorflow	https://github.com/PetarV-/GAT
LGCN (2018)	tensorflow	https://github.com/divelab/lgcn/
PGC-DGCNN (2018)	pytorch	https://github.com/dinhinfotech/PGC-DGCNN
FastGCN (2018)	tensorflow	https://github.com/matenure/FastGCN
StoGCN (2018)	tensorflow	https://github.com/thu-ml/stochastic_gcn
DGCNN (2018)	torch	https://github.com/muhanzhang/DGCNN
DiffPool (2018)	pytorch	https://github.com/RexYing/diffpool
DGI (2019)	pytorch	https://github.com/PetarV-/DGI
GIN (2019)	pytorch	https://github.com/weihua916/powerful-gnns
Cluster-GCN (2019)	pytorch	https://github.com/benedekrozemberczki/ClusterGCN

论文1

模型的深度

可扩展性的权衡

异质性

动态性

◎ 论文2

S. Liang, C. Liu, Y. Wang, H. Li and X. Li, "DeepBurning-GL: an Automated Framework for Generating Graph Neural Network Accelerators," 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA, 2020, pp. 1-9.

CCF-B

DeepBurning-GL: an Automated Framework for Generating Graph Neural Network Accelerators

Shengwen Liang[1,2], Cheng Liu[1], Ying Wang[1,2], Huawei Li[1,2], Xiaowei Li[1,2]
State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences[1],
University of Chinese Academy of Sciences[2]

ABSTRACT

Building FPGA-based graph learning accelerators is very time-consuming due to the low-level RTL programming and the complicated design flow of FPGA development. It also requires the architecture and hardware expertise from the Graph Neural Network (GNN) application developers to tailor efficient accelerator designs on FPGAs. This work proposes an automation framework, DeepBurning-GL, which is compatible with state-of-the-art graph

1 INTRODUCTION

Recently, graph neural networks (GNNs) that operate on unstruc tured data are becoming a rapidly progressing field with divers applications such as social networks [16], knowledge graph [14 and point cloud [22]. The success of GNNs propelled the deploy ment of GNNs to the production system on the cloud and edg platform, such as Pinterest [28], Alibaba [27], and Baidu [13].

Similar to DNNs, a typical GNN-layer is depicted in Fig. 1 and i

◎ 论文2

1. 分析GNN常见的瓶颈，构建了三类模板去根据用户约束产生RTL代码
2. 自动生成内存和缓存策略。
3. 自动资源分配和设计参数探索
4. 与架构比如PyG 和DGL兼容的API（应用程序接口）

◎ 论文2

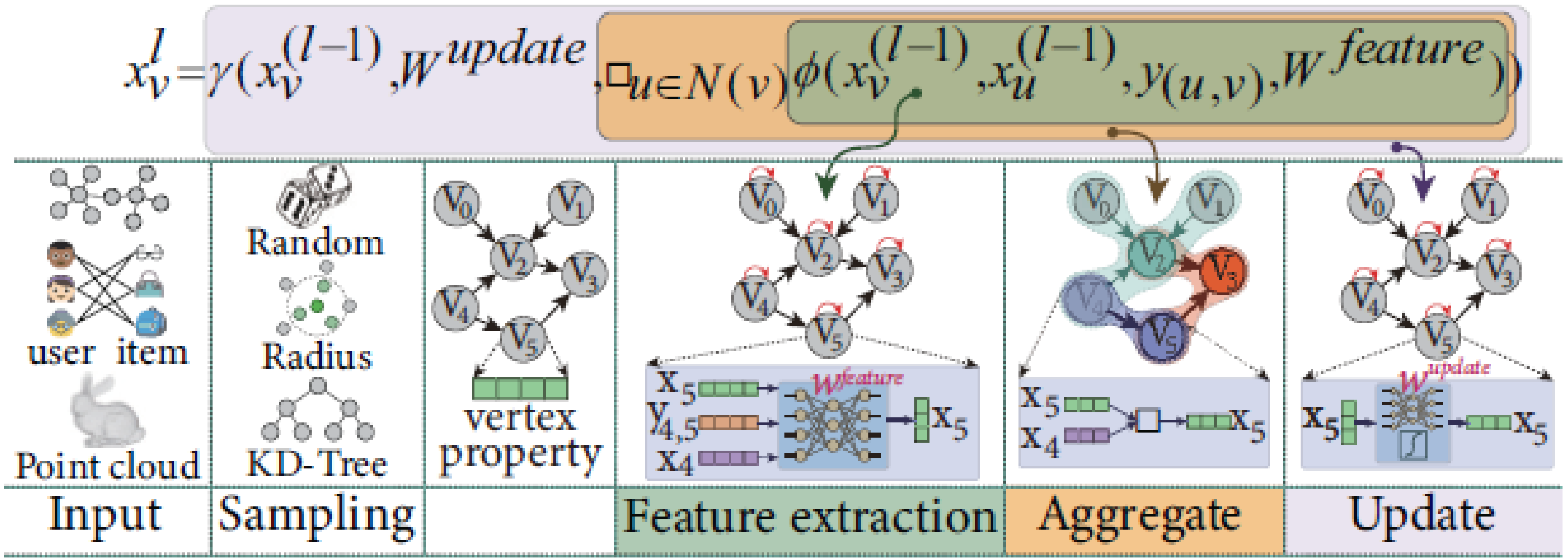


Fig. 1. An example of GNN model. The feature extraction, aggregate, and update stage are performed iteratively.

◎ 论文2

影响GNN性能的因素

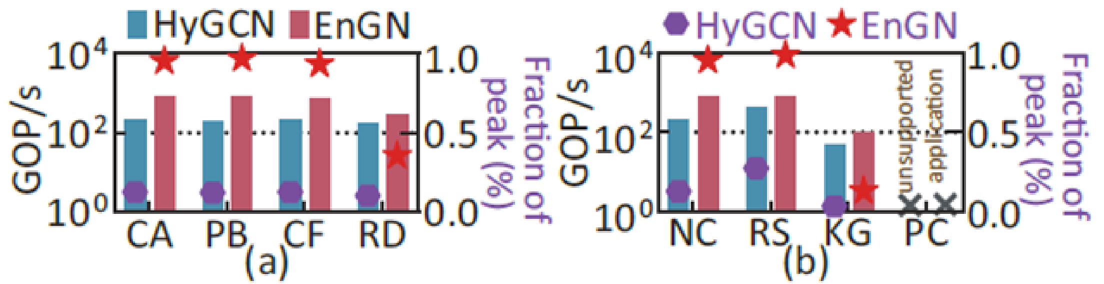


Fig. 4. Inefficiency of general purpose GCN processors. (a) The performance of GCN models on different scale datasets. (b) The performance of GNN models for different applications, where they fail to support point cloud processing.

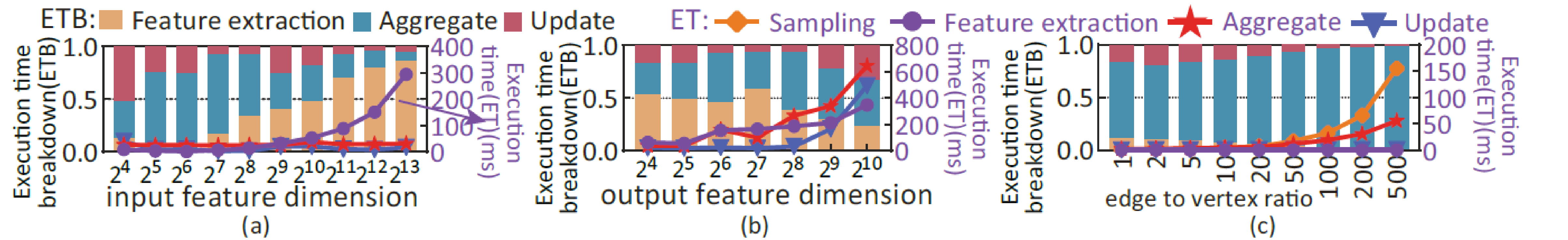


Fig. 3. The factors that affect the performance of GNNs. (a) Varied input feature dimension. (b) Varied output feature dimension. (c) Varied edge to vertex ratio.

◎ 论文2

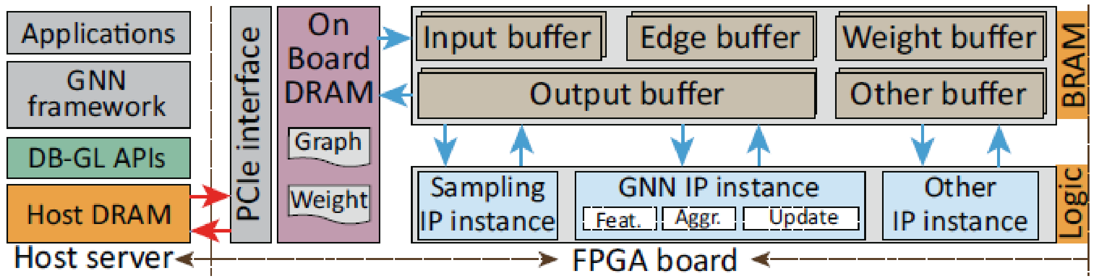


Fig. 6. Accelerator architecture.

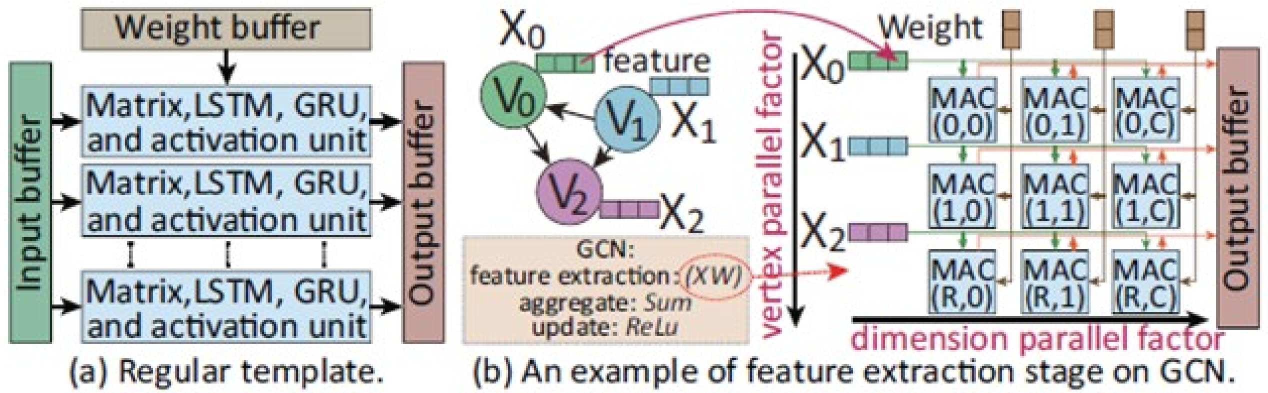


Fig. 7. Regular computing template and data mapping.

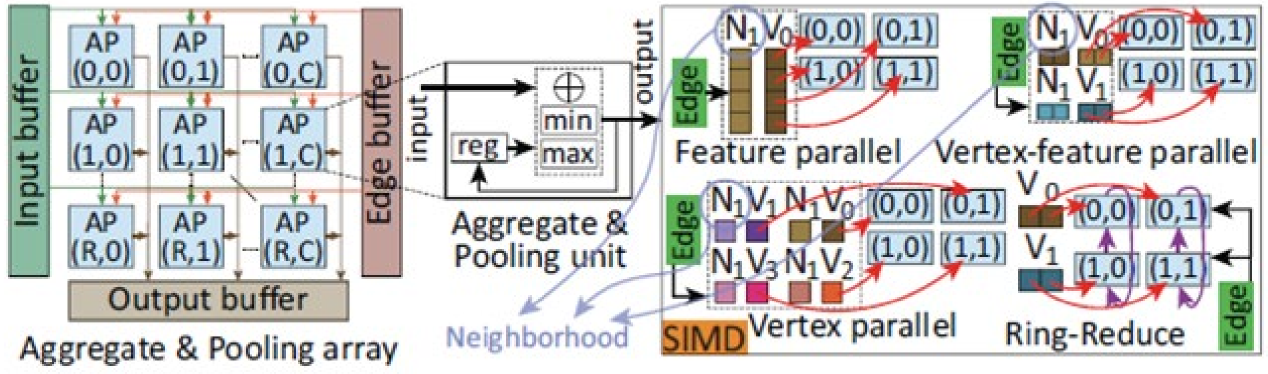


Fig. 8. Irregular computing template and data mapping.

◎ 论文2

选取的模型和数据集

Table 2: Baseline architecture.

	CPU	GPU	HyGCN	ZC706	KCU1500	Alveo U50
Compute unit	3.0GHz @ 65 cores	1.25GHz @ 5120 cores	4608 DSPs	900 DSPs	5520 DSPs	5952 DSPs
'On-chip memory	42.75MB	34MB	24.125MB	19.2Mb	75.9Mb	227.3Mb
Off-chip memory	255.9GB/s	~900GB/s	316GB/s	12.8GB/s	76.8GB/s	316GB/s

Table 3: GNN models and datasets.

Applications	Model	Graph	#Vertices	#Edges	#Feature/ #Relation	Label
Node Classification	GCN	Cora (CA)	2708	10556	1433	7
		Pubmed (PB)	19717	88651	500	3
Recommender system	GS-Pool	Cora-Full (CF)	19793	126842	8710	67
		Reddit (RD)	232965	114.6M	602	41
Knowledge graph	R-GCN	AIFB (AF)	8285	29043	91	4
		MUTAG (MG)	23644	192098	47	2
		BGS (BG)	333845	2166243	207	2
		AM (AM)	1666764	13643406	267	11
Point cloud	EdgeConv	ModelNet10 (MN)	1024	20480	6	10
		ModelNet40 (M4)	2048	51200	6	40

◎ 论文2

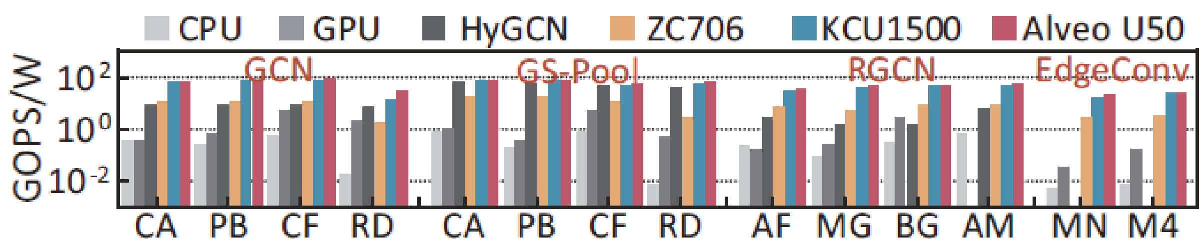


Fig. 10. Energy efficiency.

generated accelerators on ZC706, KCU1500, and Alveo U50 are 8.6 GOPS/W, 47.1 GOPS/W, and 53.5 GOPS/W respectively. They are 28.7X, 157.8X, and 179.4X higher than the implementations on CPUs, and are 6.4X, 35.2X, and 40.1X higher than the GPU implementations. According to the experiments, HyGCN implemented

Table 5: Resource utilization of the three FPGA platforms.

		ZC706				KCU1500				Alveo U50				
		LUT	FF	BRAM	DSP	LUT	FF	BRAM	DSP	LUT	FF	BRAM	URAM	DSP
Total		218K	437K	545	900	663K	1326K	2160	5520	872K	1743K	1344	640	5952
GCN	CA	52%	20%	23%	90%	38%	35%	35%	85%	77%	29%	29%	29%	99%
	PB	47%	19%	77%	86%	42%	42%	56%	84%	93%	36%	47%	46%	98%
	CF	54%	23%	94%	96%	45%	44%	88%	88%	90%	34%	71%	70%	98%
	RD	16%	7%	95%	31%	16%	16%	95%	35%	87%	34%	85%	84%	99%
GS-Pool	CA	53%	22%	82%	99%	36%	36%	76%	79%	88%	31%	43%	42%	91%
	PB	50%	19%	86%	93%	37%	37%	82%	83%	76%	30%	49%	48%	83%
	CF	53%	20%	94%	96%	38%	39%	94%	85%	90%	35%	83%	83%	100%
	RD	22%	9%	99%	38%	41%	21%	97%	43%	90%	34%	87%	87%	95%
RGCN	AF	52%	20%	37%	95%	34%	33%	9%	76%	81%	33%	5%	5%	89%
	MG	49%	19%	81%	86%	38%	37%	29%	84%	82%	30%	12%	12%	88%
	BG	52%	21%	91%	94%	42%	43%	71%	93%	91%	34%	19%	19%	95%
	AM	53%	22%	99%	99%	40%	41%	92%	89%	79%	30%	85%	85%	88%
EdgeConv	MN	90%	17%	39%	97%	51%	20%	9%	63%	67%	34%	5%	5%	88%
	M4	95%	19%	70%	98%	59%	27%	21%	81%	77%	37%	12%	11%	99%

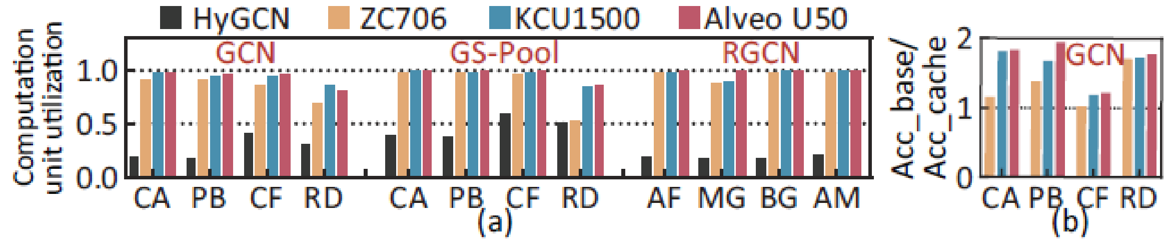


Fig. 11. (a) Computation unit utilization. (b) Normalized performance speedup of accelerators with degree-aware cache to that with the baseline cache on GCN model.

◎ 论文3

C. Hao et al., "FPGA/DNN Co-Design: An Efficient Design Methodology for IoT Intelligence on the Edge," 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2019, pp. 1-6.

CCF-A

FPGA/DNN Co-Design: An Efficient Design Methodology for IoT Intelligence on the Edge

Cong Hao[1*], Xiaofan Zhang[1*], Yuhong Li[1], Sitao Huang[1], Jinjun Xiong[2], Kyle Rupnow[3], Wen-mei Hwu[1], Deming Chen[1,3]
[1]University of Illinois at Urbana-Champaign, [2]IBM T. J. Watson Research Center, [3]Inspirit IoT, Inc.
{congh, xiaofan3, leeyh, shuang91, w-hwu, dchen}@illinois.edu, jinjun@us.ibm.com, kyle.rupnow@inspirit-iot.com

ABSTRACT

While embedded FPGAs are attractive platforms for DNN acceleration on edge-devices due to their low latency and high energy efficiency, the scarcity of resources of edge-scale FPGA devices also makes it challenging for DNN deployment. In this paper, we propose a simultaneous FPGA/DNN co-design methodology with both bottom-up and top-down approaches: a bottom-up hardware-oriented DNN model search for high accuracy, and a top-down FPGA accelerator design considering DNN-specific characteristics. We also build an automatic co-design flow, including an *Auto-DNN* engine to perform hardware-oriented DNN model search, as well as

combined DNN and FPGA accelerator co-design space, and constrains the solutions to have both high QoR and efficient FPGA implementations. Consequently, the co-design task will be extremely time-consuming, as we must perform training of each candidate DNN to evaluate its quality. Even using Neural Architecture Search (NAS) [8, 9] for DNN development and the High Level Synthesis (HLS) for fast FPGA development [10, 11], both tasks still need a large amount of engineering hours.

Facing the opportunities and challenges, in this work, we propose a simultaneous FPGA/DNN co-design approach, which effectively searches the design space to both generate high quality DNNs suit-