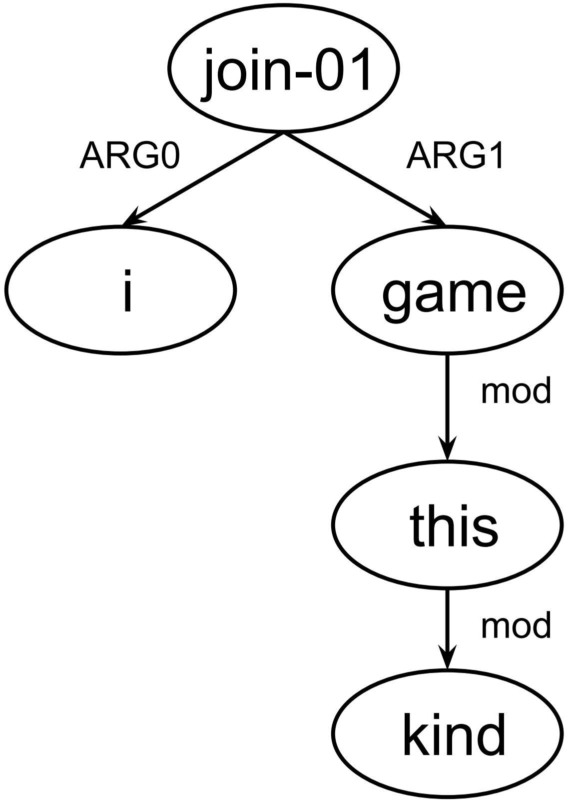
Lightweight, Dynamic Graph Convolutional Networks for AMR-to-Text Generation

Yan Zhang*, Zhijiang Guo*, Zhiyang Teng, Wei Lu
Shay B. Cohen, Zuozhu Liu, Lidong Bing

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN
西湖大學 WESTLAKE UNIVERSITY
THE UNIVERSITY of EDINBURGH
浙江大學 ZHEJIANG UNIVERSITY
达摩院 Alibaba DAMO Academy

AMR-to-Text Generation: Task Definition

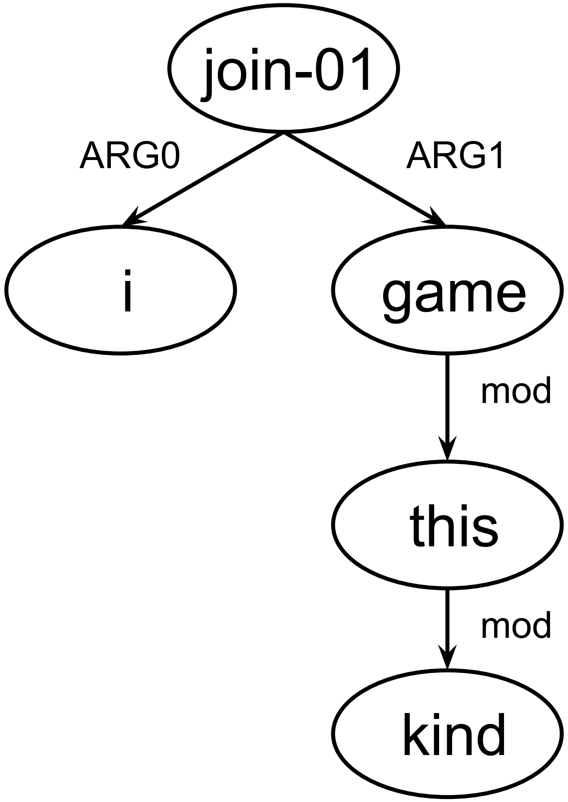


Source: AMR Graph

I will join this kind of game.

Target: Text Sequence

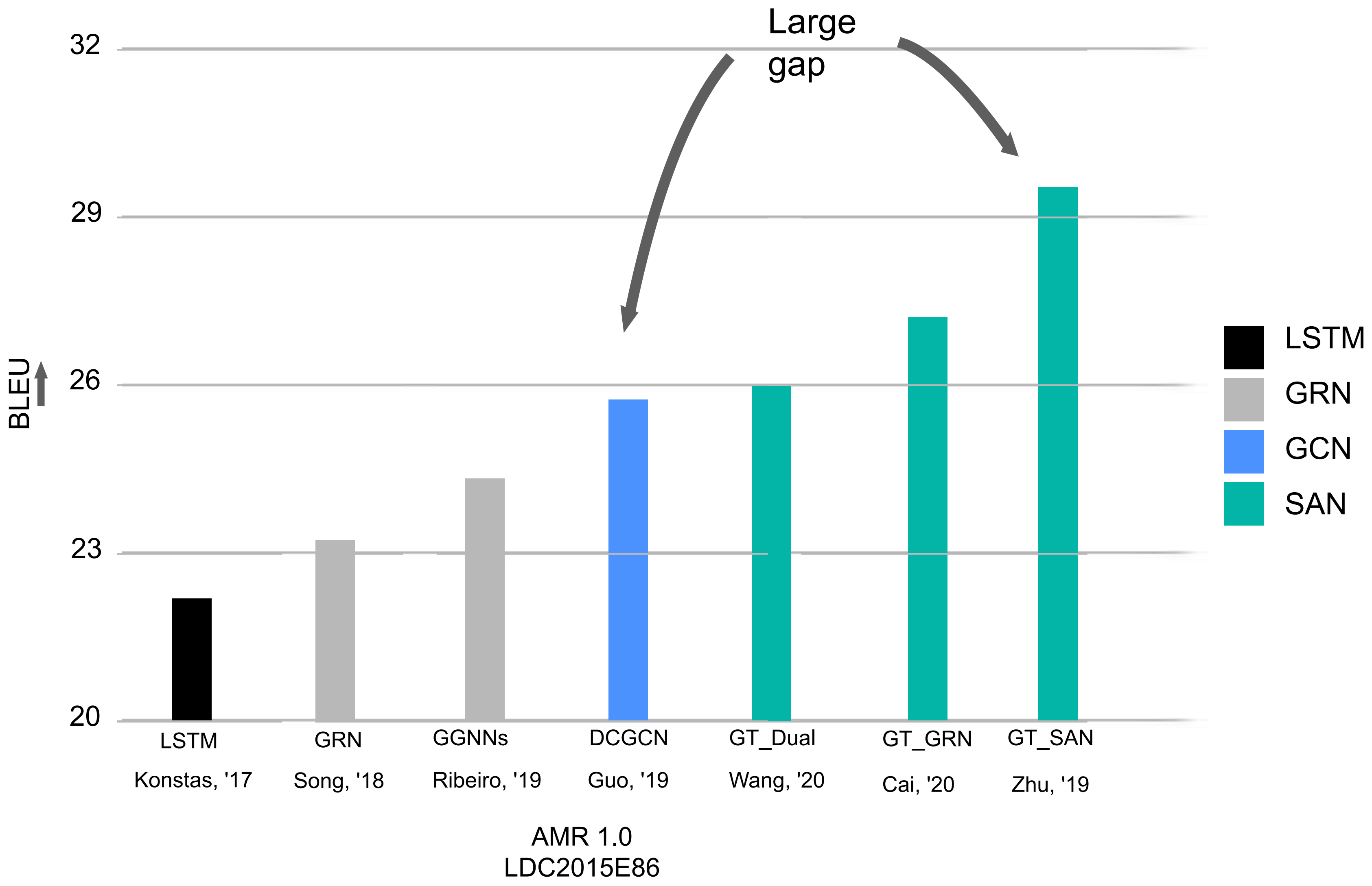
AMR-to-Text Generation: Main Challenge



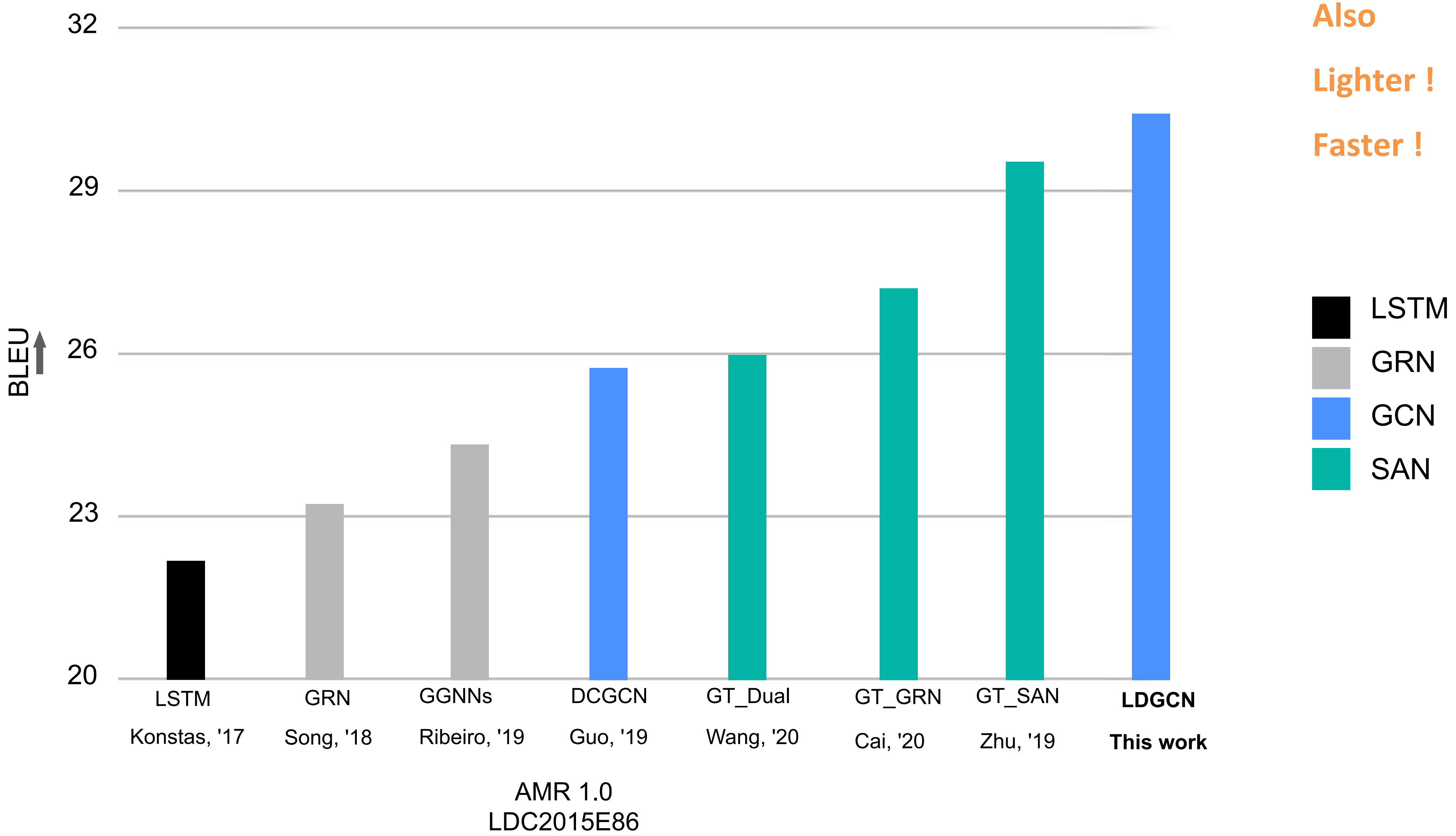
Source: AMR Graph

How to learn a *good representation* of the AMR graph?

Progress in AMR-to-Text Generation



Progress in AMR-to-Text Generation



Also

Lighter !

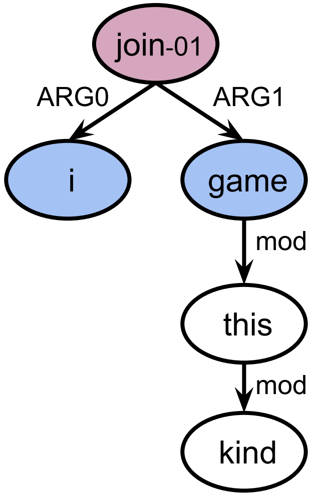
Faster !

Graph Convolution v.s. Self-Attention

GCN Guo et al., 2019 Damonte et al., 2019

$$h_i = \sigma(\sum_{j=1}^{n} A_{i,j} W h_j + b)$$

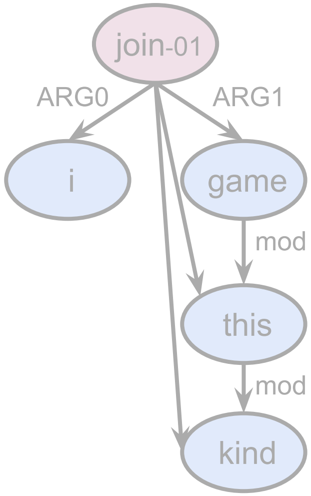
$A_{i,j} = 1$ if there is a edge between i and j



SAN vaswani et al., 2017

$$h_i = \sigma(\sum_{j=1}^{n} A_{i,j} W h_j + b)$$

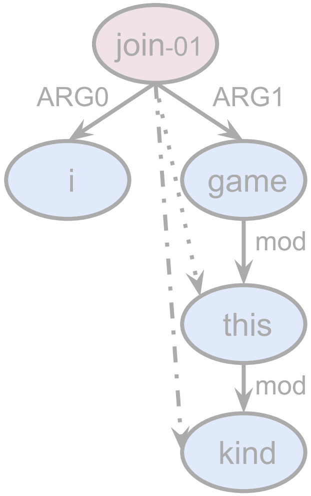
$$A_{i,j} = f(h_i, h_j)$$



Structured SAN Zhu et al., 2019 Cai et al., 2020 Wang et al., 2020

$$h_i = \sigma(\sum_{j=1}^{n} A_{i,j} W h_j + b)$$

$$A_{i,j} = f(h_i, h_j, r_{ij})$$

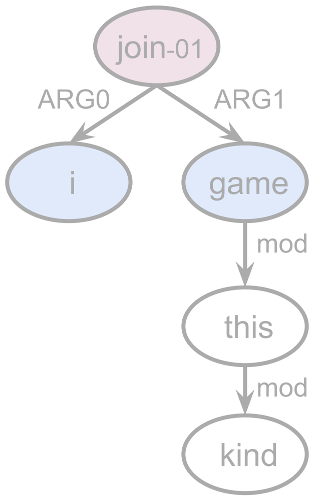


Graph Convolution v.s. Self-Attention

GCN Guo et al., 2019 Damonte et al., 2019

$$h_i = \sigma(\sum_{j=1}^{n} A_{i,j} W h_j + b)$$

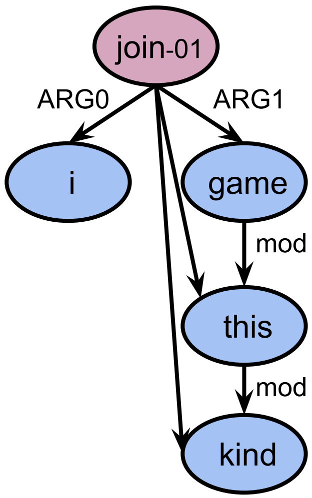
$A_{i,j} = 1$ if there is a edge between i and j



SAN vaswani et al., 2017

$$h_i = \sigma(\sum_{j=1}^{n} A_{i,j} W h_j + b)$$

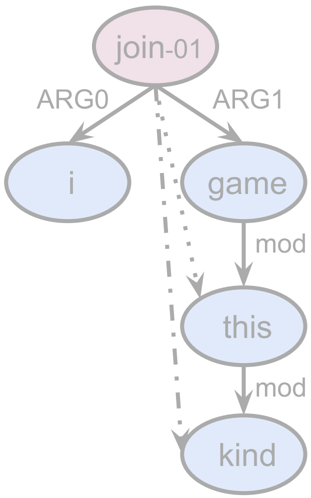
$$A_{i,j} = f(h_i, h_j)$$



Structured SAN Zhu et al., 2019 Cai et al., 2020 Wang et al., 2020

$$h_i = \sigma(\sum_{j=1}^{n} A_{i,j} W h_j + b)$$

$$A_{i,j} = f(h_i, h_j, r_{ij})$$

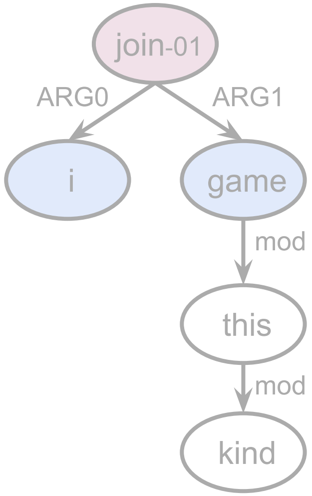


Graph Convolution v.s. Self-Attention

GCN Guo et al., 2019 Damonte et al., 2019

$$h_i = \sigma(\sum_{j=1}^{n} A_{i,j} W h_j + b)$$

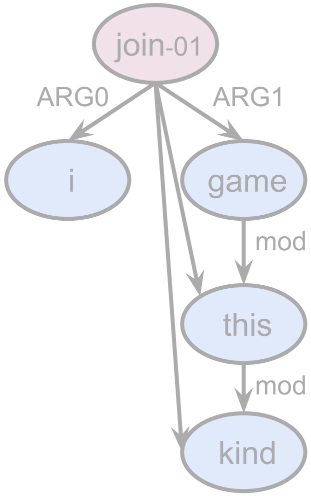
$A_{i,j} = 1$ if there is a edge between i and j



SAN vaswani et al., 2017

$$h_i = \sigma(\sum_{j=1}^{n} A_{i,j} W h_j + b)$$

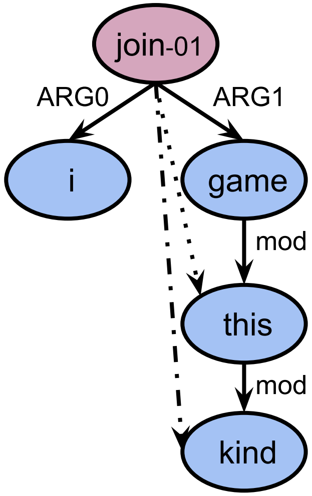
$$A_{i,j} = f(h_i, h_j)$$



Structured SAN Zhu et al., 2019 Cai et al., 2020 Wang et al., 2020

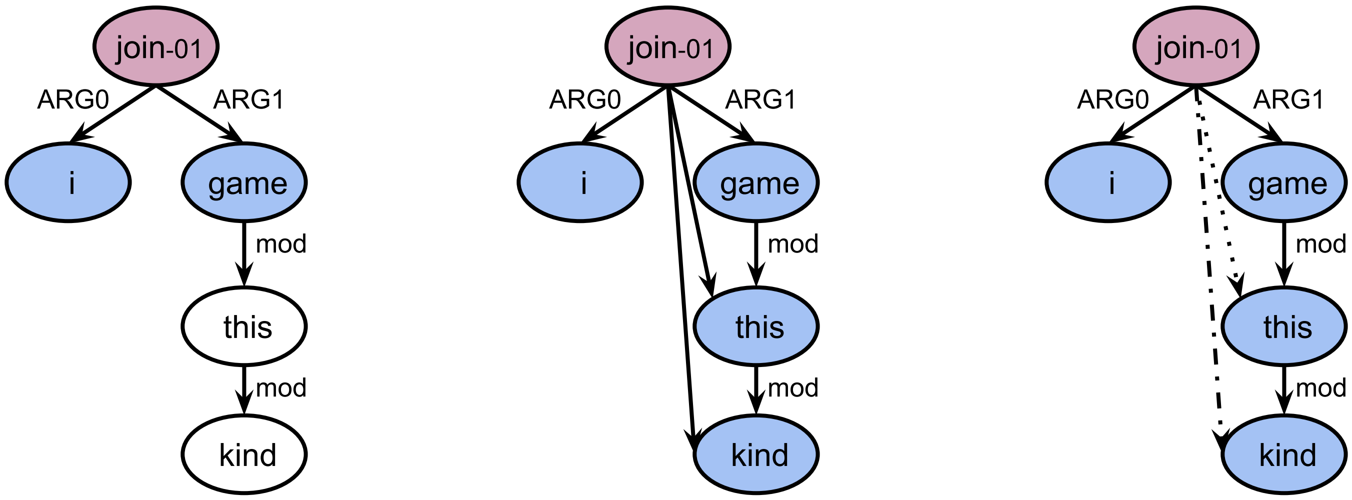
$$h_i = \sigma(\sum_{j=1}^{n} A_{i,j} W h_j + b)$$

$$A_{i,j} = f(h_i, h_j, r_{ij})$$



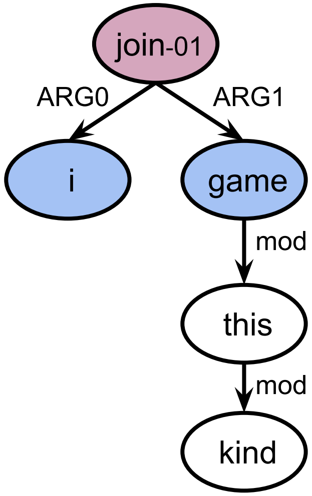
Graph Convolution v.s. Self-Attention

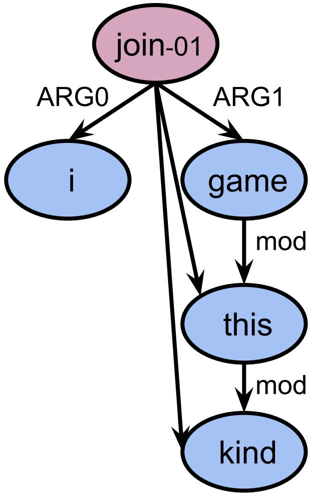
	GCN Guo et al., 2019 Damonte et al., 2019	SAN vaswani et al., 2017	Structured SAN Zhu et al., 2019 Cai et al., 2020 Wang et al., 2020
Time Complexity	$O(n)$	$O(n^2)$	$O(n^2+n)$

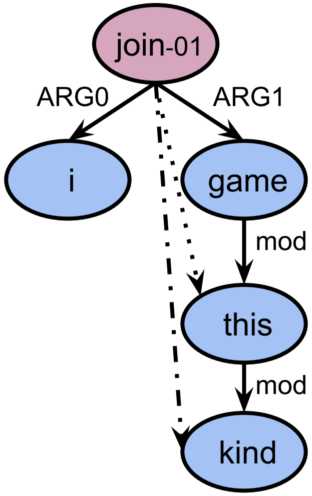


Graph Convolution v.s. Self-Attention

	GCN Guo et al., 2019 Damonte et al., 2019	SAN vaswani et al., 2017	Structured SAN Zhu et al., 2019 Cai et al., 2020 Wang et al., 2020
Time Complexity	$O(n)$	$O(n^2)$	$O(n^2+n)$
Neighbour Info	***1*-order**	***n*-order**	***n*-order**

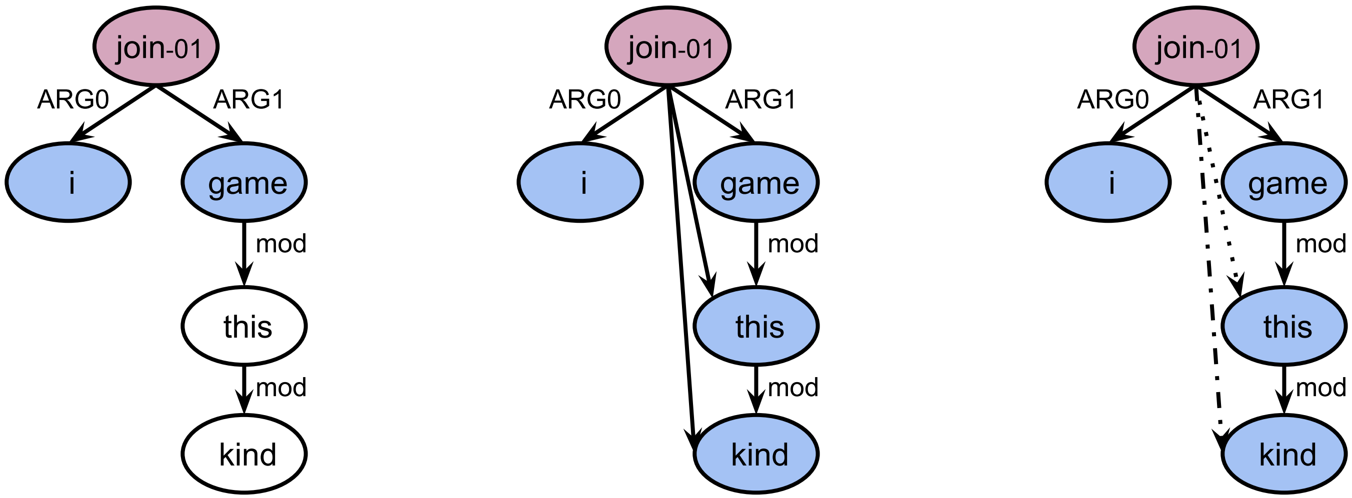






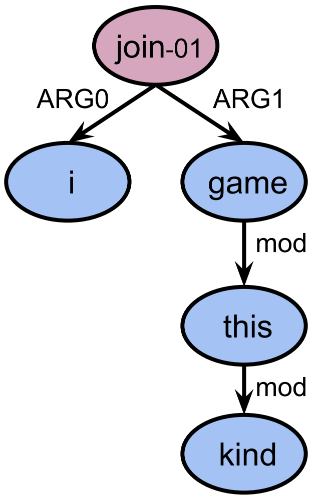
Graph Convolution v.s. Self-Attention

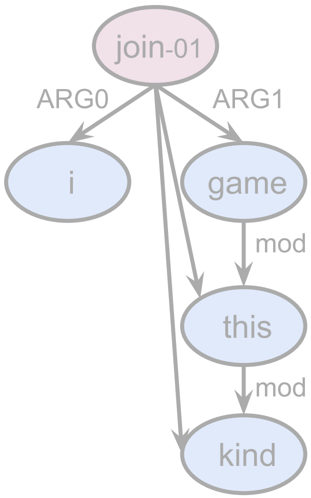
	GCN Guo et al., 2019 Damonte et al., 2019	SAN vaswani et al., 2017	Structured SAN Zhu et al., 2019 Cai et al., 2020 Wang et al., 2020
Time Complexity	$O(n)$	$O(n^2)$	$O(n^2+n)$
Neighbour Info	1-order	n-order	n-order
Layers	**36**	**8**	**8**

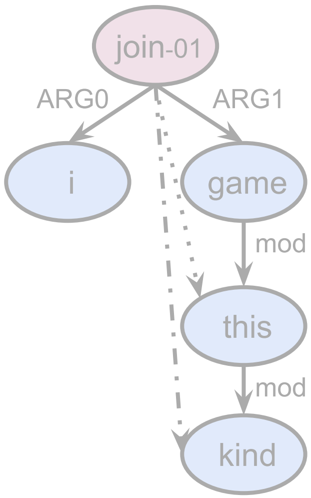


Graph Convolution v.s. Self-Attention

	LDGCN (This work)	SAN vaswani et al., 2017	Structured SAN Zhu et al., 2019 Cai et al., 2020 Wang et al., 2020
Time Complexity	$O(n)$	$O(n^2)$	$O(n^2+n)$
Neighbour Info	k-order (k<<n)	n-order	n-order
Layers	36 **(fewer #params)**	8	8







Research Questions

- **Q1:** Can we build a more effective graph encoder solely based on graph convolution?
- **Q2:** Existing models require large model size to maintain model capacity. Can we build a model with fewer parameters while have similar model capacity?

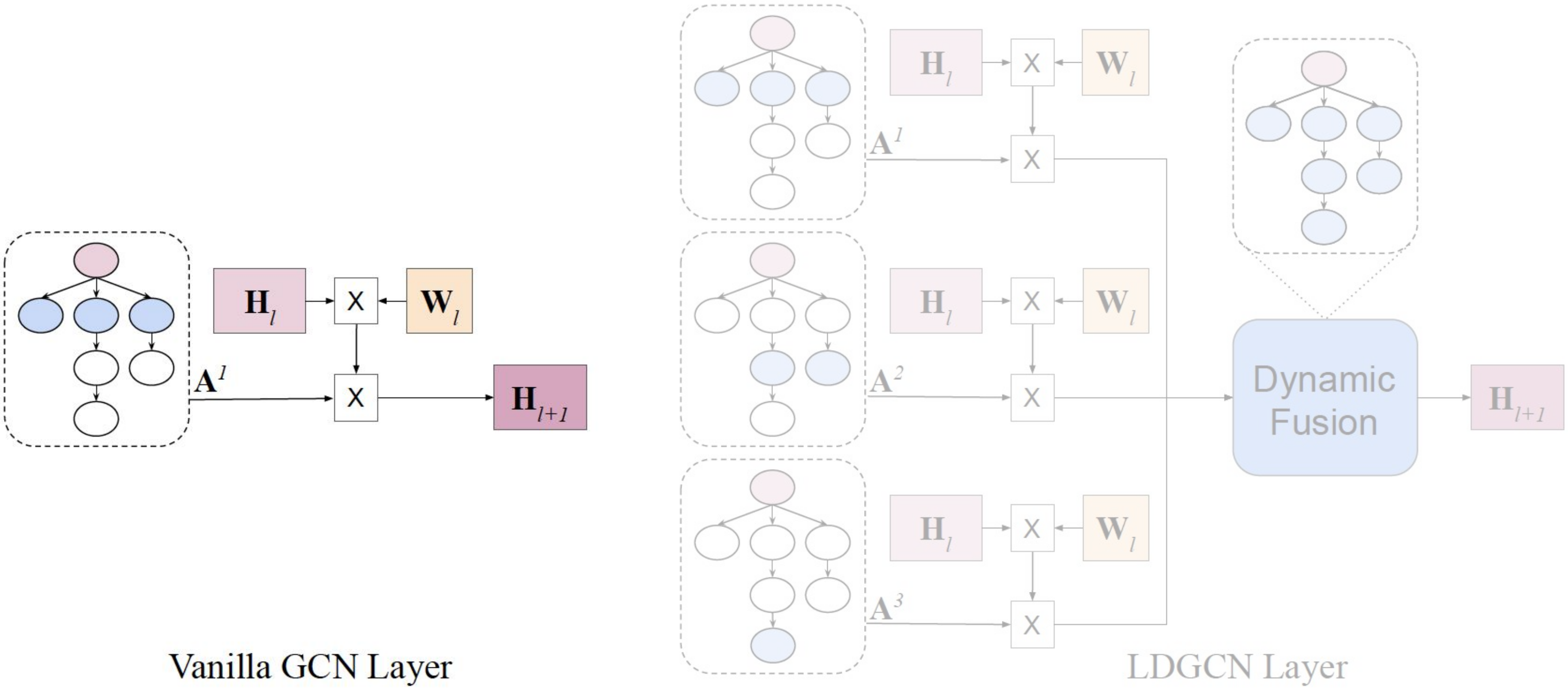
Research Questions

- **Q1:** Can we build a more effective graph encoder solely based on graph convolution?

- **A: Dynamic fusion mechanism** is introduced to graph convolutions to integrate information from ***higher order*** neighbors

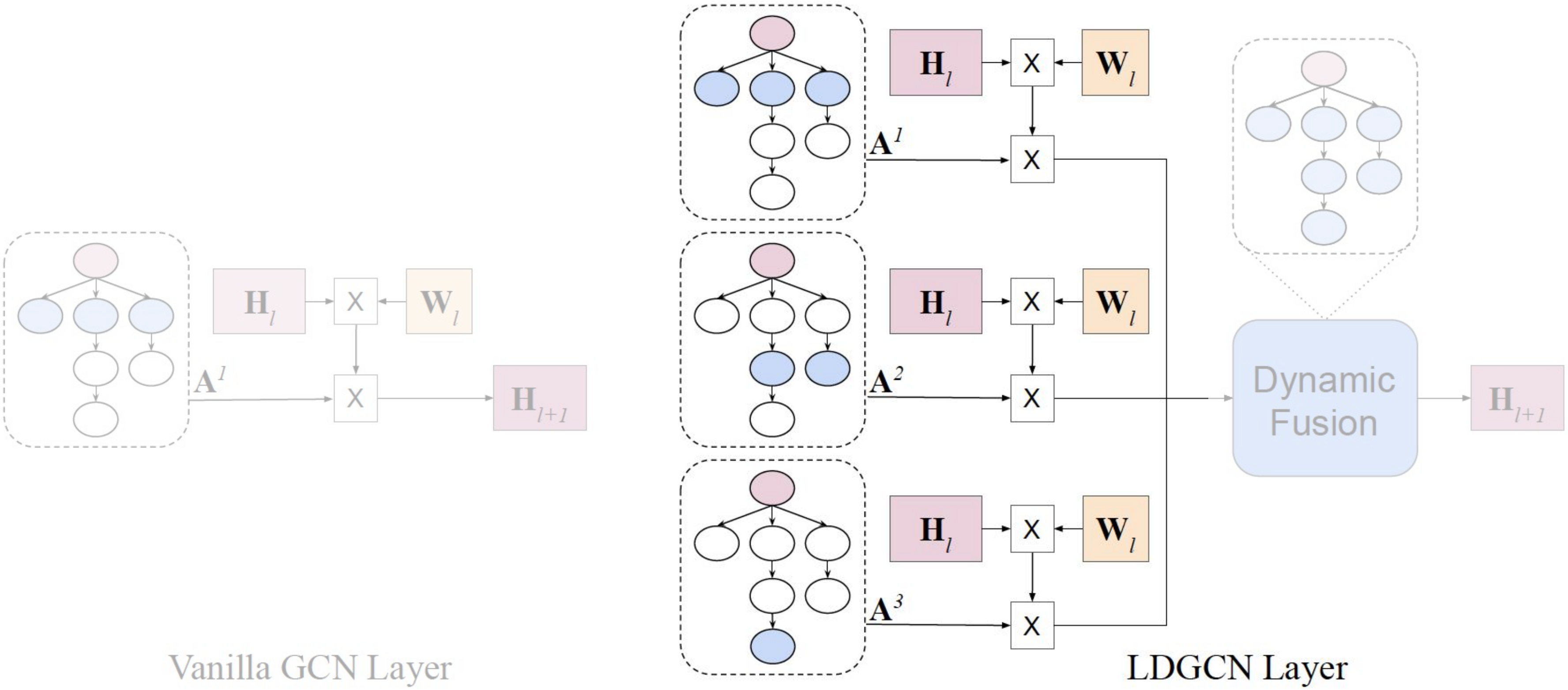
Dynamic Fusion Mechanism

- ***Vanilla graph convolutional layer*** only takes ***1***-order adjacency matrix as the input.



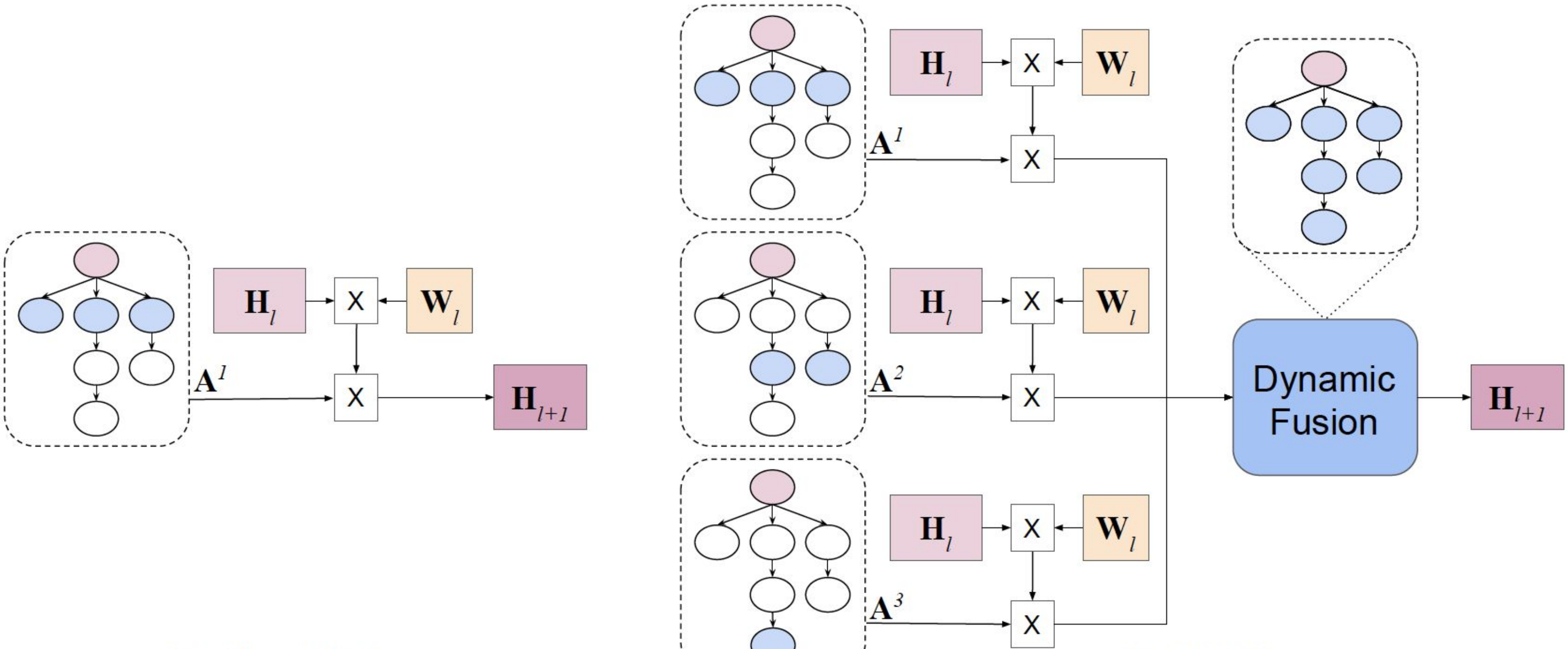
Dynamic Fusion Mechanism

- Each graph convolutional layer takes ***k*** number of ***k***-order adjacency matrices as inputs (here ***k=3***).



Dynamic Fusion Mechanism

- Each graph convolutional layer takes ***k*** number of ***k***-order adjacency matrices as inputs (here ***k=3***).
- The dynamic fusion mechanism is able to integrate information from ***1***- to ***k-hop*** neighbors.

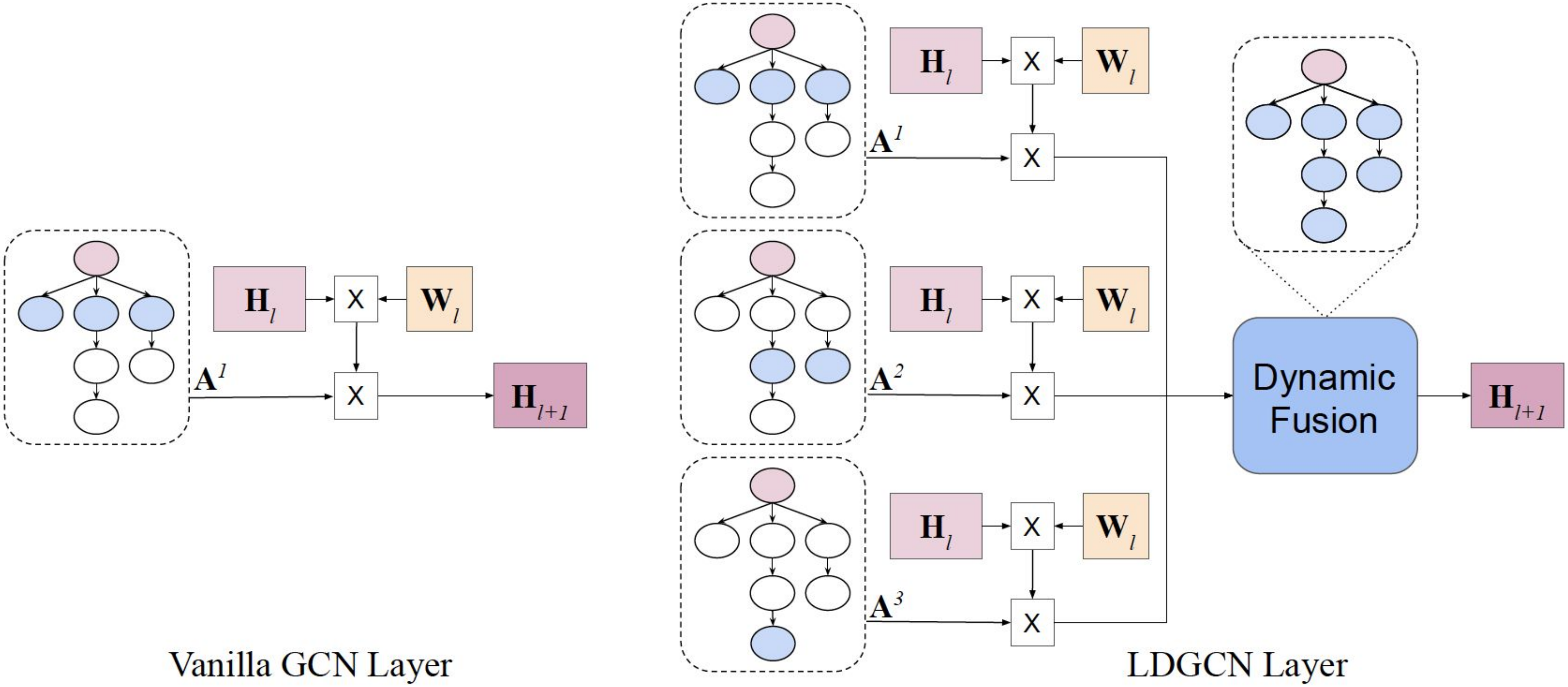


Vanilla GCN Layer

LDGCN Layer

Dynamic Fusion Mechanism

- **Computational Overhead**: In practice, there is no need to calculate or store $\mathbf{A}^k$. $\mathbf{A}^k\mathbf{H}_l$ is computed with right-to-left multiplication.
- If ***k=3***, we can calculate $\mathbf{A}^3\mathbf{H}_l$ as $(\mathbf{A}(\mathbf{A}(\mathbf{A}\mathbf{H}_l)))$. $\mathbf{A}$ is stored as vanilla GCNs.



Research Questions

- **Q2:** Can we build a lighter model that still achieves competitive performance?

- **A: Weight sharing strategies** are proposed to ***reduce memory usage*** and ***speed up inference***.

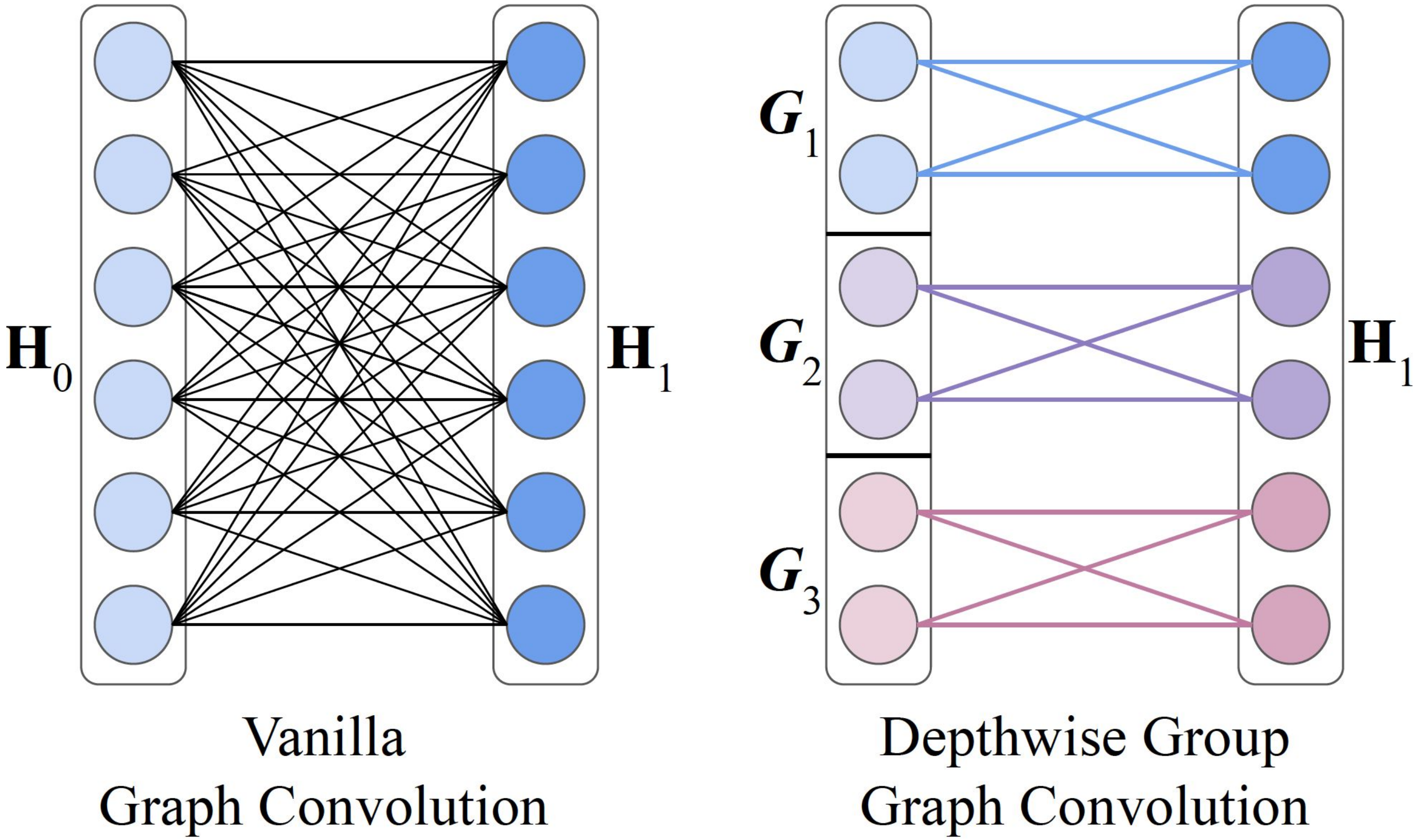
Research Questions

- **Q2:** Existing models require large model size to maintain model capacity. Can we build a model with fewer parameters while have similar model capacity?

- **A: Weight sharing strategies** are proposed to ***reduce memory usage*** and ***speed up inference***.
  - *Group Graph Convolution*
  - *Weight Tied Convolution*

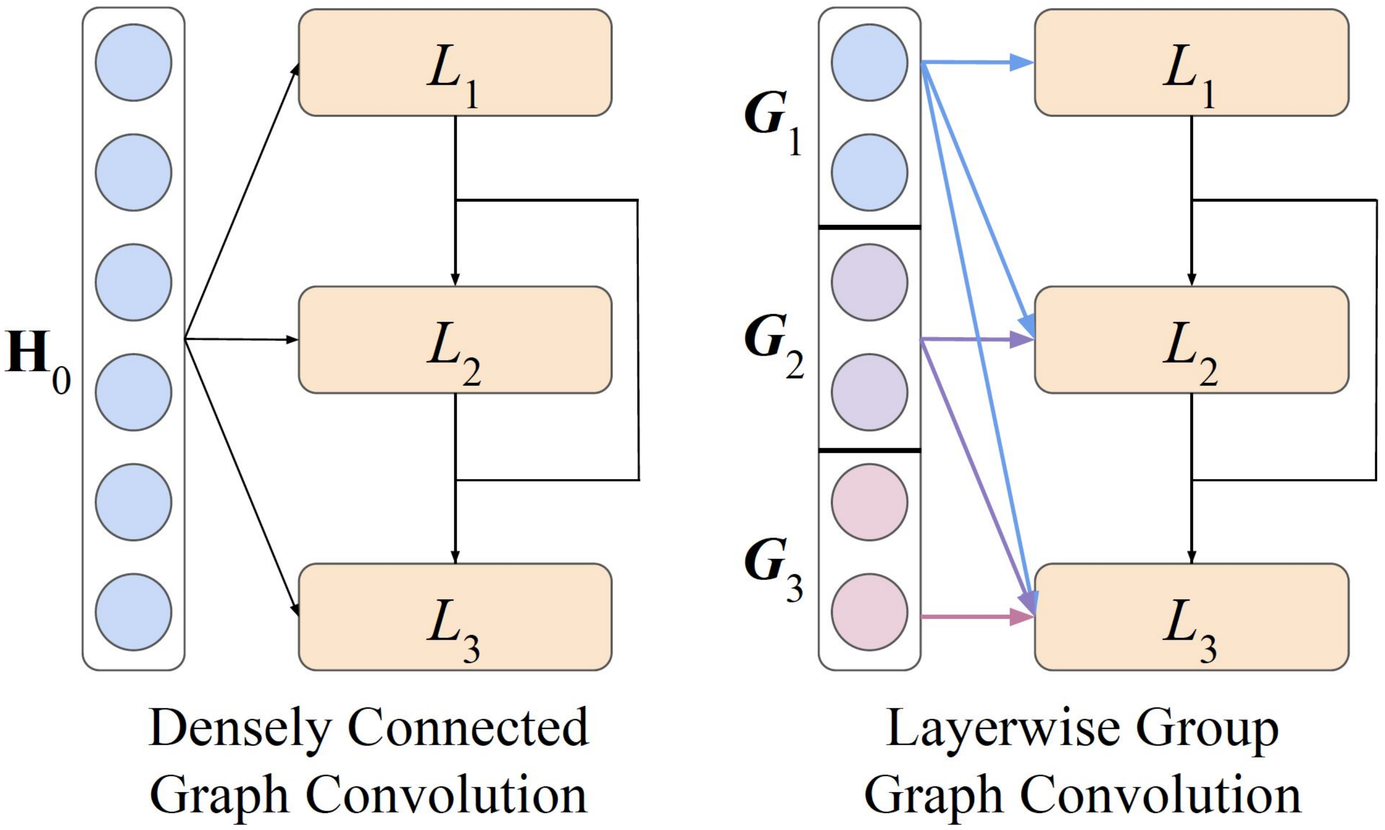
Group Graph Convolution: Depthwise

- ***Depthwise graph convolution***: the input and output representation of the ***l***-th layer $\mathbf{H}_l$ and $\mathbf{H}_{l+1}$ are partitioned into N disjoint groups (here N=3).



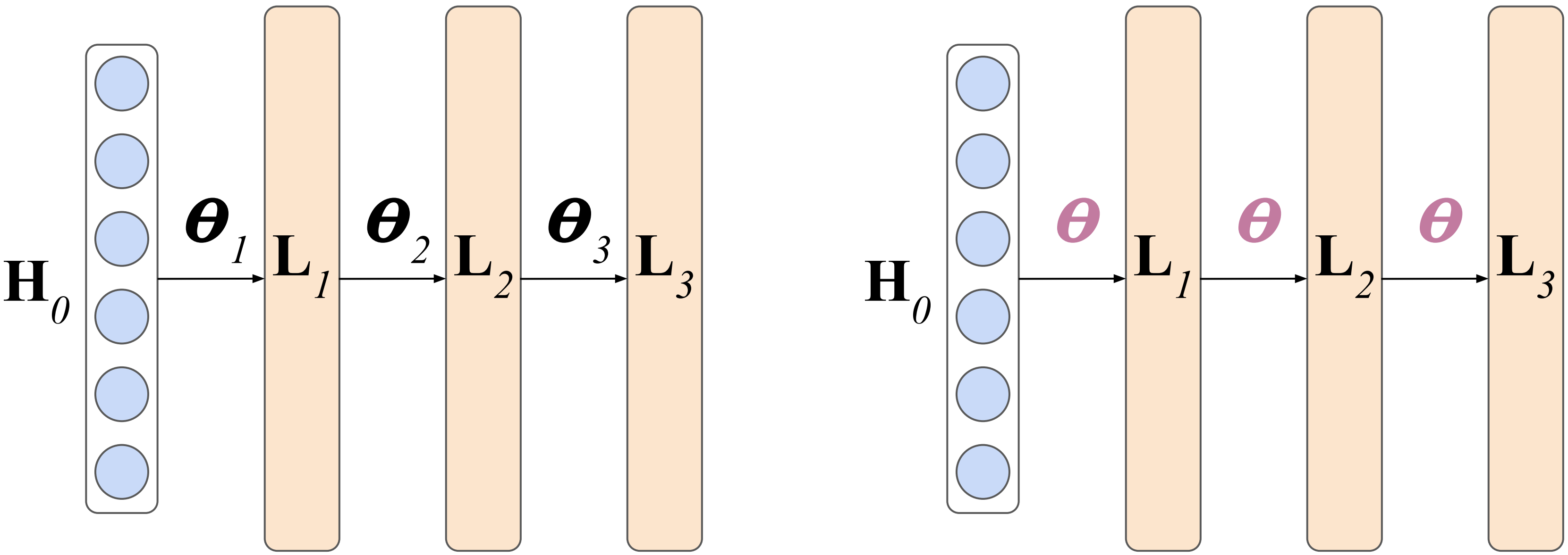
Group Graph Convolution: Layerwise

- ***Densely Connected graph convolution:*** each layer takes the concatenation of outputs from all preceding layers as its input
- ***Layerwise graph convolution***: The input representation $\boldsymbol{H_0}$ is partitioned into $\boldsymbol{M=3}$ disjoint groups.



Weight Tied Graph Convolution

- We further adopt a more aggressive strategy where ***parameters are shared across all layers.***
- Theoretically, weight tied networks ***can be unrolled to any depth***, typically with improved feature abstractions as depth increases (Bai et al., 2019).



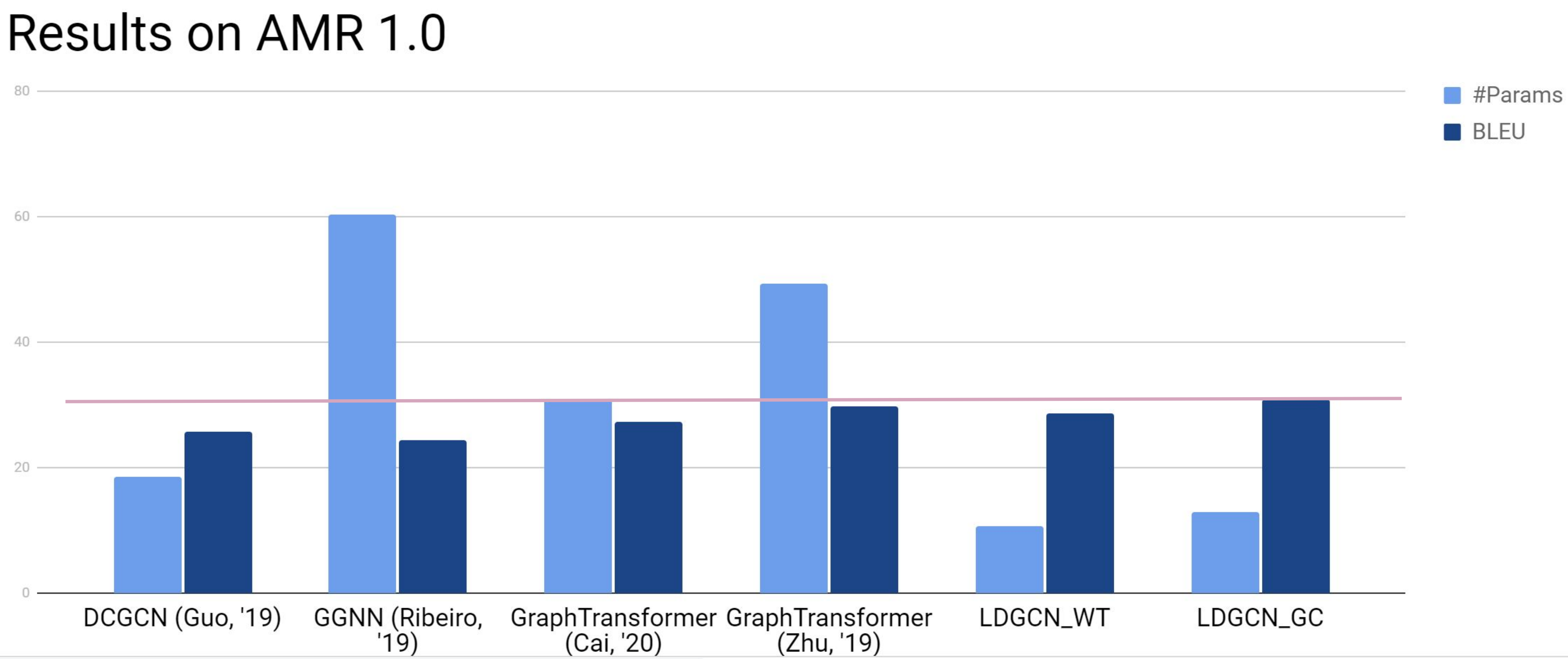
Vanilla
Graph Convolutions

Weight Tied
Graph Convolutions

Data Statistics

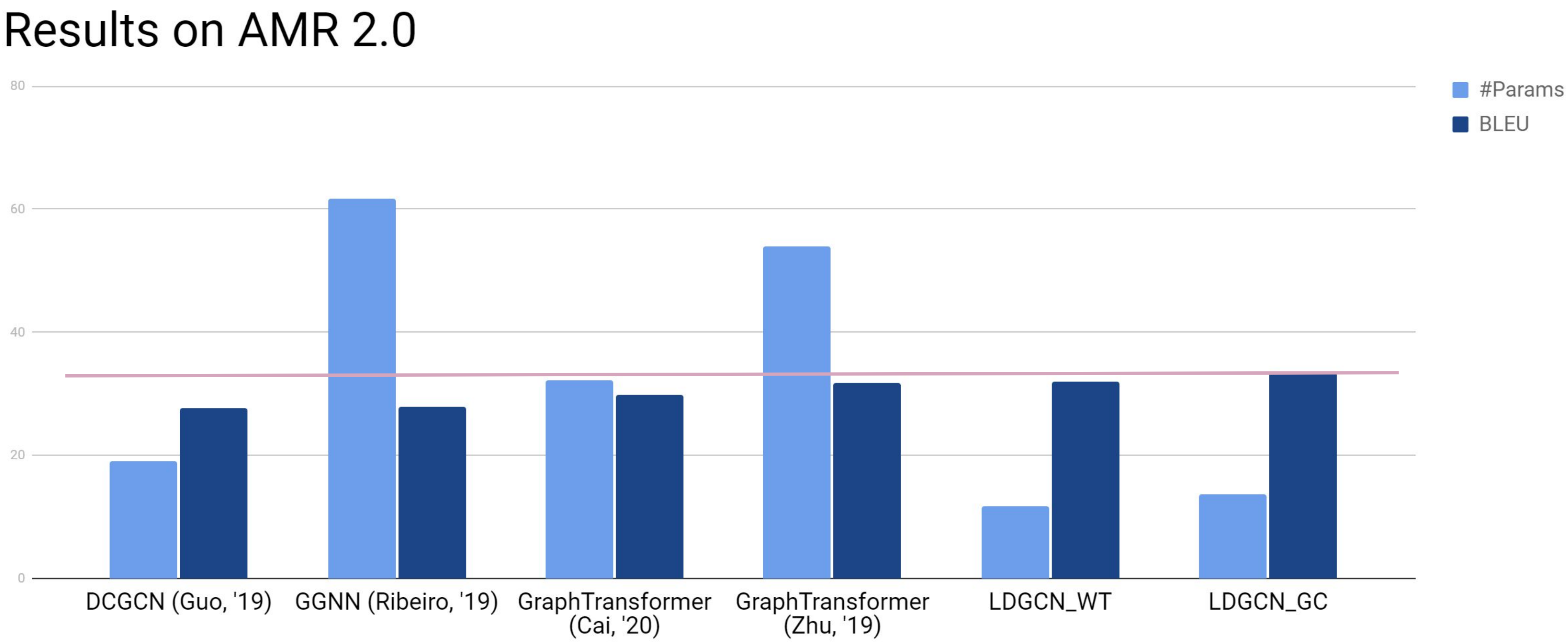
Dataset	Train	Dev	Test
AMR 1.0 (LDC2015E86)	16,833	1,368	1,371
AMR 2.0 (LDC2017T10)	36,521	1,368	1,371
AMR 3.0 (LDC2020T02)	55,635	1,722	1,898

Main Results



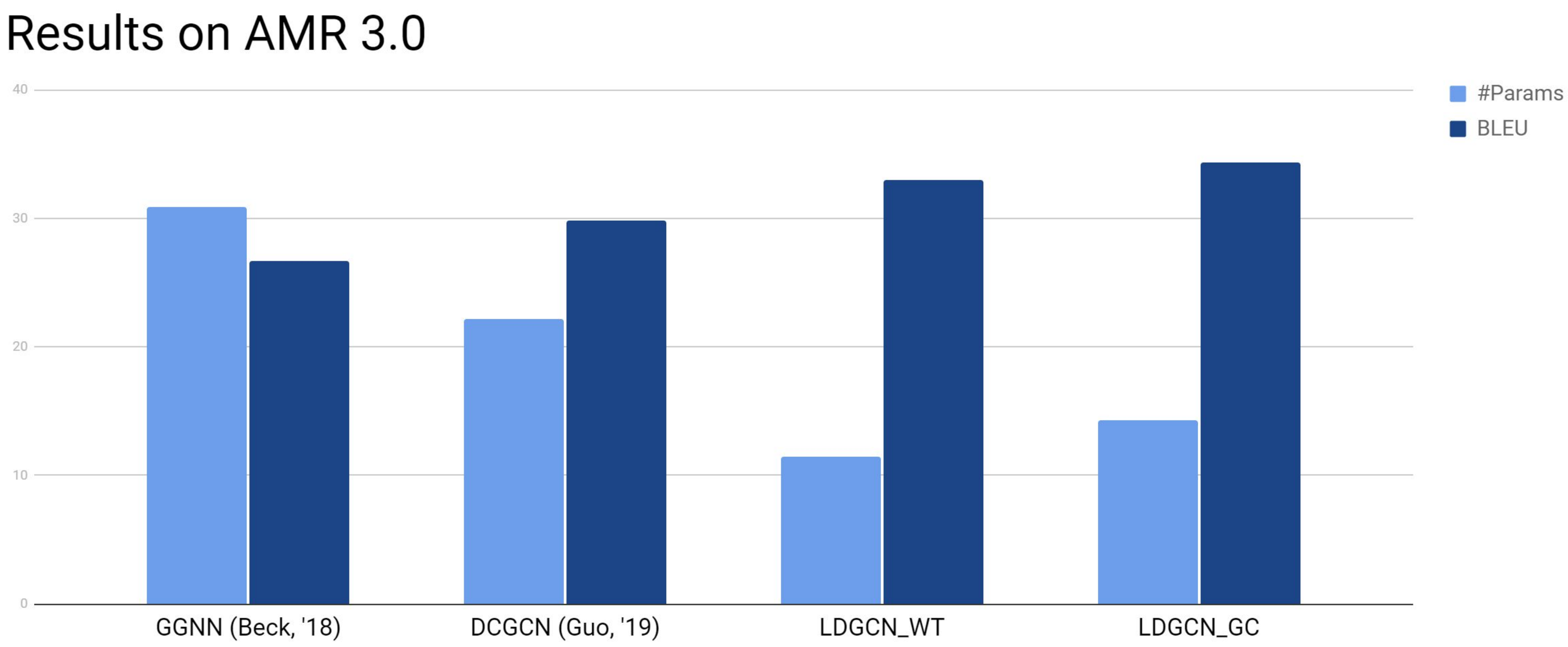
- **LDGCN_WT:** our model with weight tied convolution;
- **LDGCN_GC:** our model with group graph convolution;

Main Results



- **LDGCN_WT:** our model with weight tied convolution;
- **LDGCN_GC:** our model with group graph convolution;

Main Results



- **LDGCN_WT:** our model with weight tied convolution;
- **LDGCN_GC:** our model with group graph convolution;

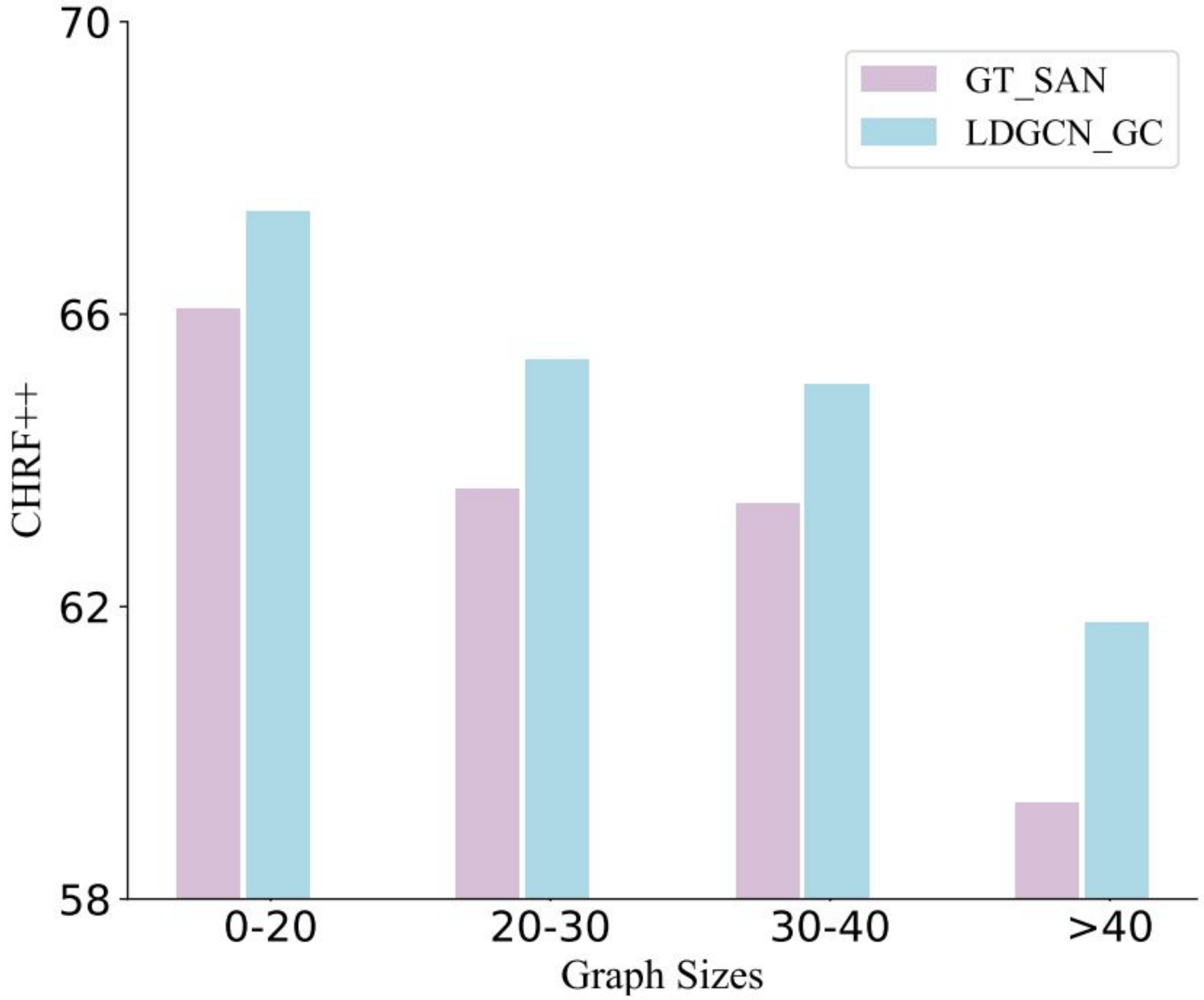
AMR 1.0-With External Training Data

Model	External	BLEU
Seq2Seq (Konstas et al., 2017)	2M	32.3
GraphLSTM (Song et al., 2018)	2M	28.2
Transformer (Wang et al., 2020)	2M	31.6
GT_Dual (Wang et al., 2020)	2M	36.4
Our LDGCN_GC	**0.5M**	36.0
Our LDGCN_WT	**0.5M**	**36.8**

Inference Speed

Model	Speed
Transformer	1.00x
DeepGCN	1.21x
Our LDGCN_GC	1.17x
Our LDGCN_WT	**1.22x**

Performance against Graph Size



Summary

- We propose the novel LDGCN model, which maintains a better balance between parameter efficiency and model capacity.
- Extensive experiments show that the LDGCN model outperforms state-of-the-art approaches with significantly fewer parameters.

Thank You

Code Available

https://github.com/yanzhangnlp/LDGCNs