Reducing Inference Latency with Concurrent Architectures for Image Recognition at Edge

IEEE Edge'23
July 2023

Ramyad Hadidi[§*]
Rain AI

Jiashen Cao[§]
Georgia Tech

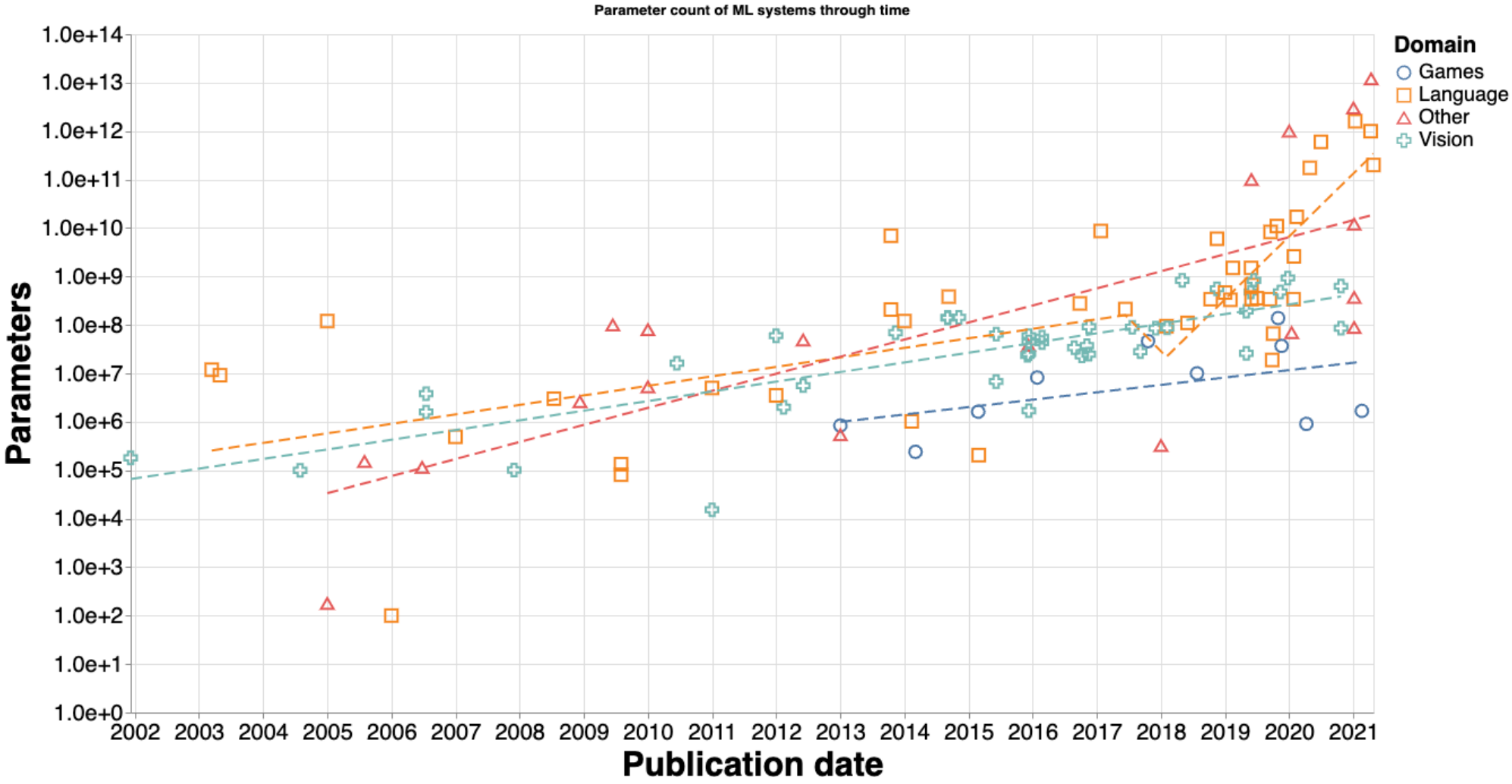
Michael S. Ryoo
Stony Brook University & Google

Hyesoon Kim
Georgia Tech

[§] Equal Contributions
* This work was done when the author was affiliated with Georgia Tech.

Georgia Tech

Modern AI



Parameter counts in Machine Learning, Jaime Sevilla,
Published in Towards Data Science Jul 2, 2021

Artificial Intelligence

GatesNotes THE BLOG OF BILL GATES

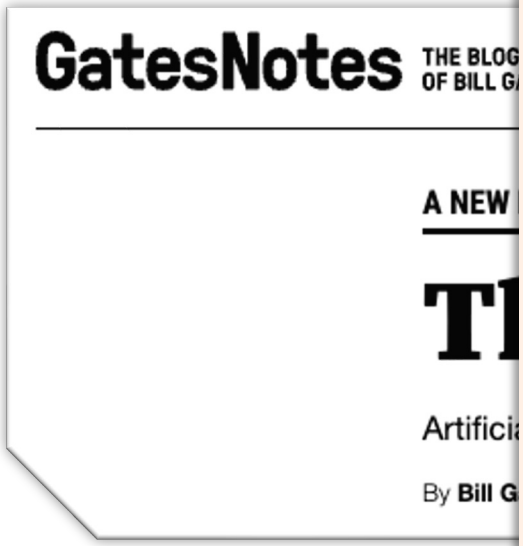
A NEW ERA

The Age of AI has begun

Artificial intelligence is as revolutionary as mobile phones and the Internet.

By **Bill Gates** | March 21, 2023 • 14 minute read

Artificial Intelligence


GatesNotes THE BLOG OF BILL G

A NEW

T

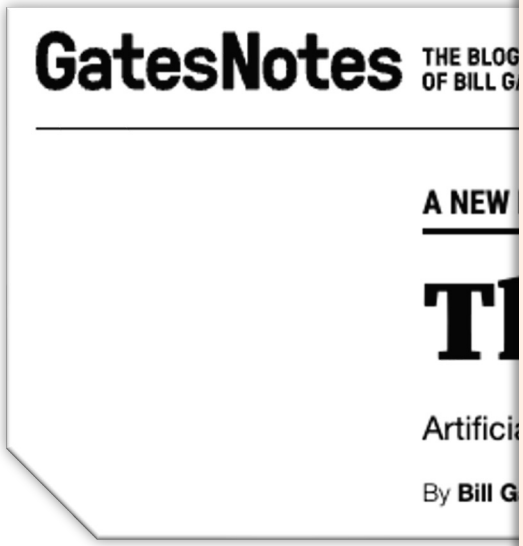
Artificia

By Bill G

	2012	2022
Compute used to train largest AI model	1e+16 FLOPS (10,000,000,000,000,000)	1e+24 FLOPS (1,000,000,000,000,000,000,000,000)
Data consumed by largest AI model	Imagenet: a dataset of 15mn labelled images (150GB)	The entire internet (45,000GB)
Capabilities of largest AI models	Can recognise images at "beginner human" level Superhuman at chess	Superhuman or high-human at a wide variety of games (Go, Diplomacy, Starcraft II, poker etc) Human-level at 150 reasoning & knowledge tasks Passes US Medical Licensing Exam, passes the Bar Exam Displays complex capabilities like power-seeking, deceiving humans Can self-improve by "reasoning" out loud Can write 40 per cent of the code for a software engineer

Hogarth, Ian. "We must slow down the race to God-like AI." Financial Times, 12 Apr. 2023

Artificial Intelligence


GatesNotes THE BLOG OF BILL G

A NEW

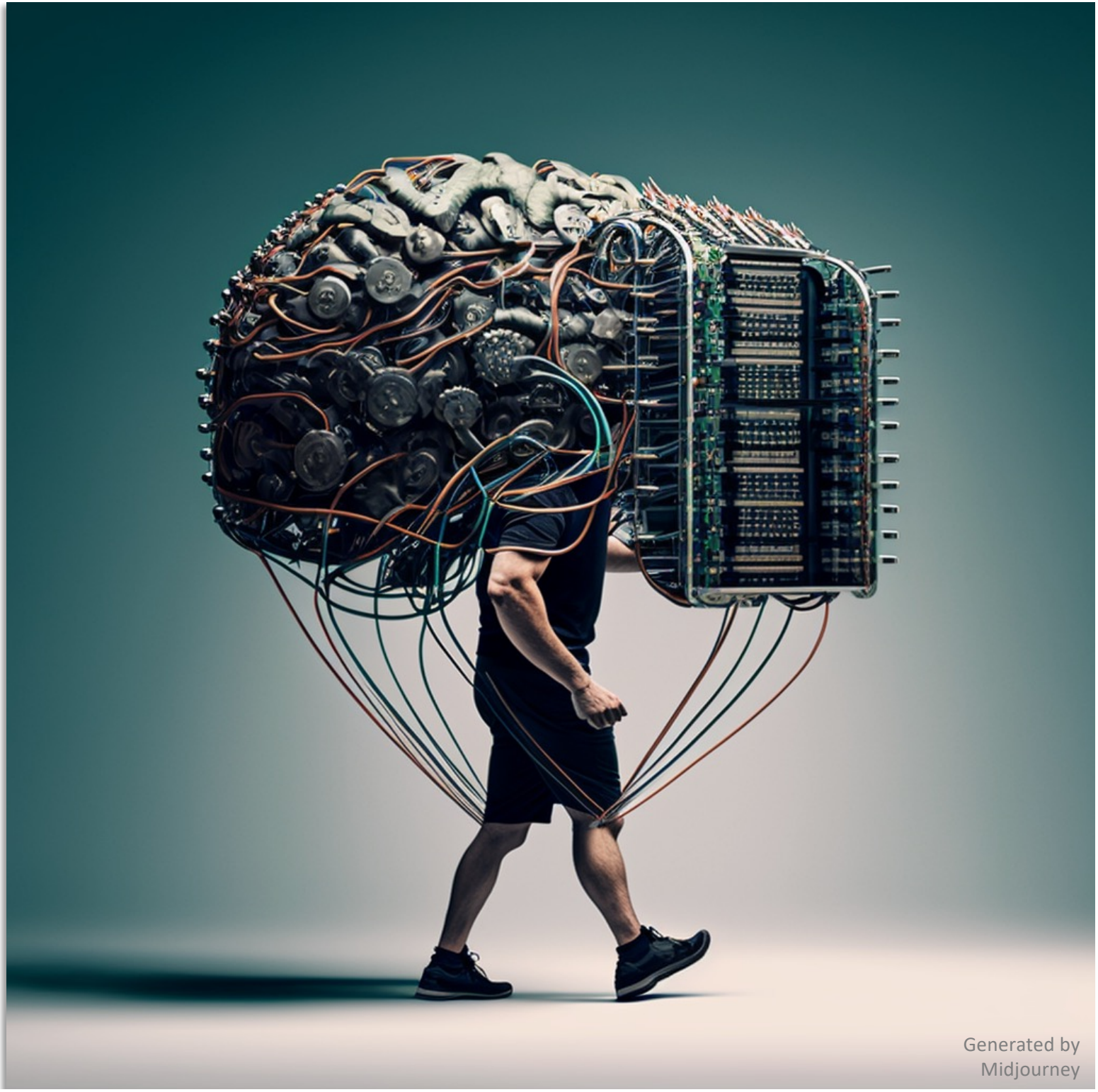
T

Artificia

By Bill G

	2012	2022
Compute used to train largest AI model	1e+16 FLOPS (10,000,000,000,000,000)	1e+24 FLOPS (1,000,000,000,000,000,000,000,000)
Data consumed by largest AI model	Imagenet: a dataset of 15mn labelled images (150GB)	The entire internet (45,000GB)
Capabilities of largest AI models	Can recognise images at "beginner human" level Superhuman at chess	Superhuman or high-human at a wide variety of games (Go, Diplomacy, Starcraft II, poker etc) Human-level at 150 reasoning & knowledge tasks Passes US Medical Licensing Exam, passes the Bar Exam Displays complex capabilities like power-seeking, deceiving humans Can self-improve by "reasoning" out loud Can write 40 per cent of the code for a software engineer

Hogarth, Ian. "We must slow down the race to God-like AI." Financial Times, 12 Apr. 2023


Generated by
Midjourney

The Challenge

DNNs are increasingly deeper and wider models with higher computational demands

Training is Hard

Training GPT-3 on
3584x H100s would take 46h*

* Nvidia, NVIDIA H100 GPUs Set Standard for Generative AI in Debut MLPerf Benchmark, 27 Jun. 2023

The Challenge

DNNs are increasingly deeper and wider models with higher computational demands

Training is Hard

Training GPT-3 on
3584x H100s would take 46h*

Fast Inference is Even Harder

Requiring optimizations at various levels of HW-SW and next-level of efficient production toolsets

* Nvidia, NVIDIA H100 GPUs Set Standard for Generative AI in Debut MLPerf Benchmark, 27 Jun. 2023

Speeding up Inference

Several techniques are employed for a faster inference:

- Reducing parameter size
  - Model compression (pruning)
  - Post-Training Quantization (PTQ)
  - Quantization-aware training (QAT)
- Exploiting **parallelism** in computation
  - Happens at several levels with several assumptions and end goals

Parallelizing Computations

Inside Model

Model Parallelism
Layer Scheduling
Per Layer
Per Tensor
Per Operation

Outside Model

Multiple Models
Data Parallelism

Parallelizing Computations

Inside Model

Model Parallelism
Layer Scheduling
Per Layer
Per Tensor
Per Operation

Model Itself!

No Explicit Parallelism

Blackhole of
ML Models

Outside Model

Multiple Models
Data Parallelism

Parallelizing Computations

Inside Model

Model Parallelism
Layer Scheduling
Per Layer
Per Tensor
Per Operation

Model Itself!

No Explicit Parallelism

Blackhole of
ML Models

Outside Model

Multiple Models
Data Parallelism

Current approaches in reducing the inference latency are always applied ***after*** a model architecture is defined

Model Parallelism

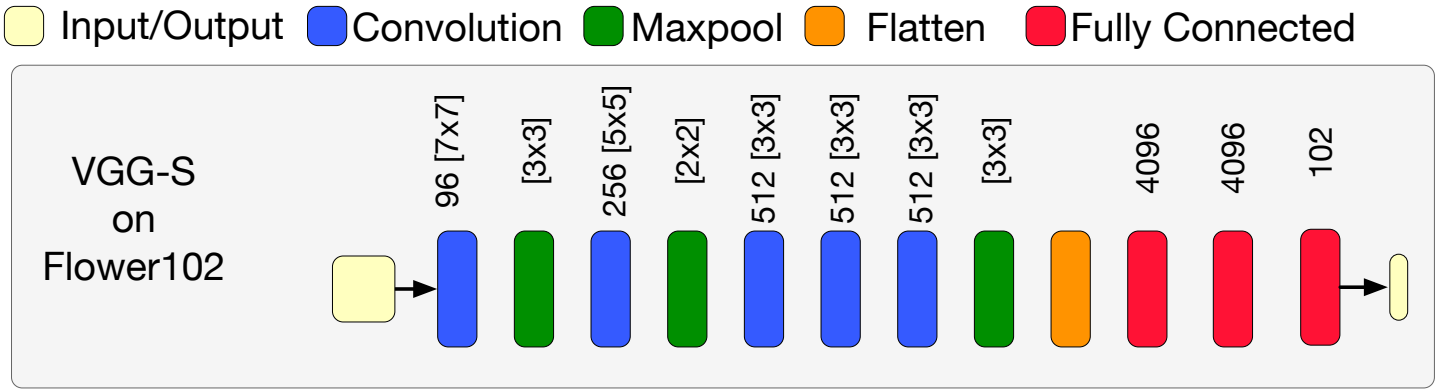
Model parallelism does not change the model

- Synchronization: Difficult to distribute
- Several Connections: High communication overhead

Model Parallelism

Model parallelism does not change the model

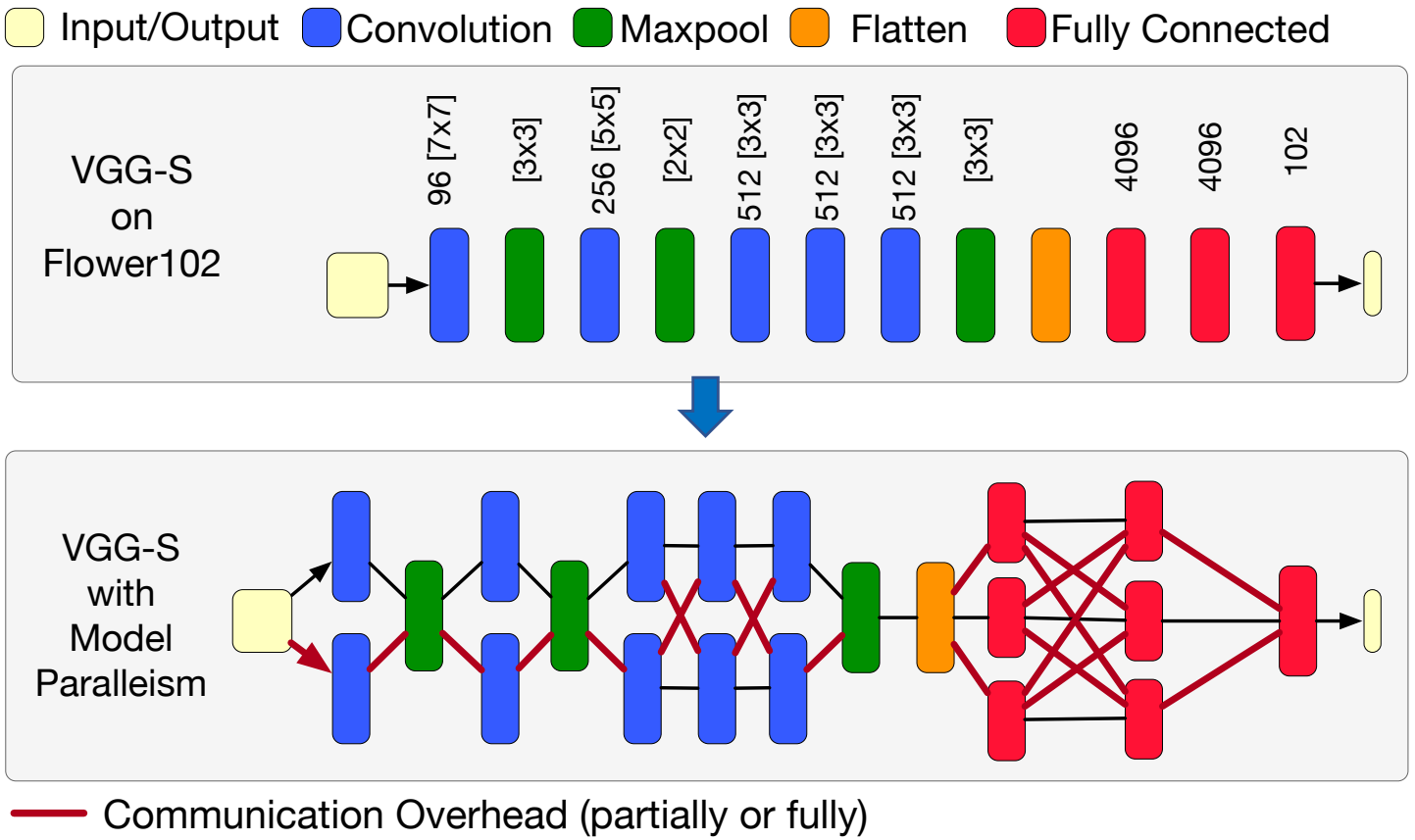
- Synchronization: Difficult to distribute
- Several Connections: High communication overhead



Model Parallelism

Model parallelism does not change the model

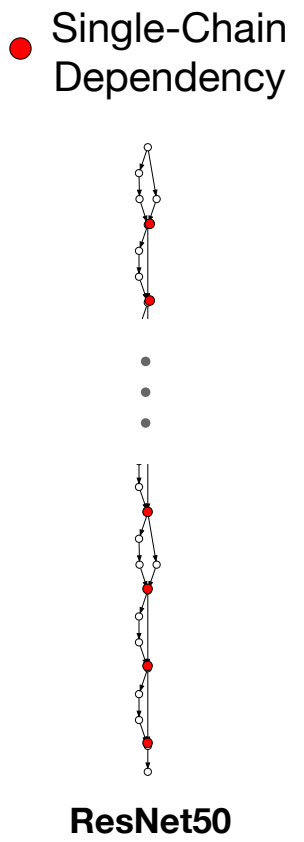
- Synchronization: Difficult to distribute
- Several Connections: High communication overhead



Single-Chain Data Dependency

The main performance barrier in model parallelism is **single-chain dependency**

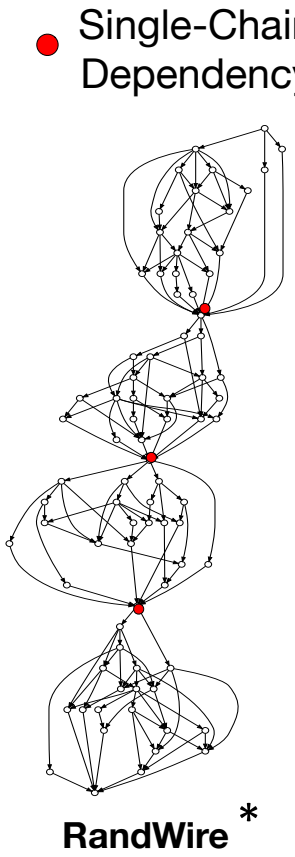
Cannot efficiently extend concurrency and distribution beyond current explicit parallelism exposed within intra-layer computations



Single-Chain Data Dependency

We discover that this bias also exist in

- well-known architectures and,
- neural architecture search (NAS) studies

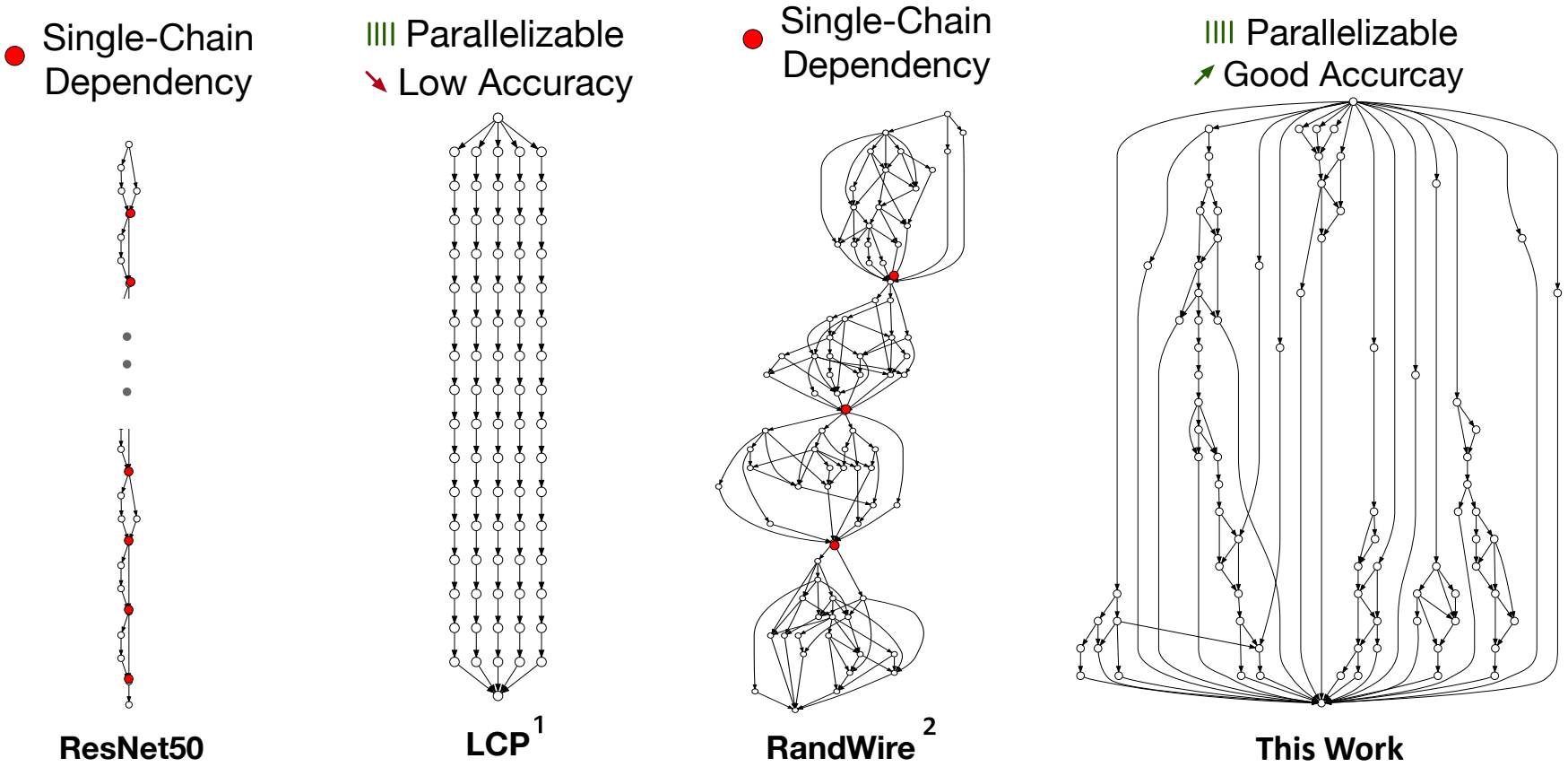


For instance, RandWire, a NAS study trying to search all possible models, has also this **single-chain dependency**

* Xie et al. "Exploring randomly wired neural networks for image recognition." ICCV'19

Our Solution

In this paper we **search** for models efficient for distribution, while providing a good accuracy!



[1] **Hadidi** et al. "LCP: A Low-Communication Parallelization Method for Fast Neural Network Inference in Image Recognition." *Accepted, CSCE 2023*

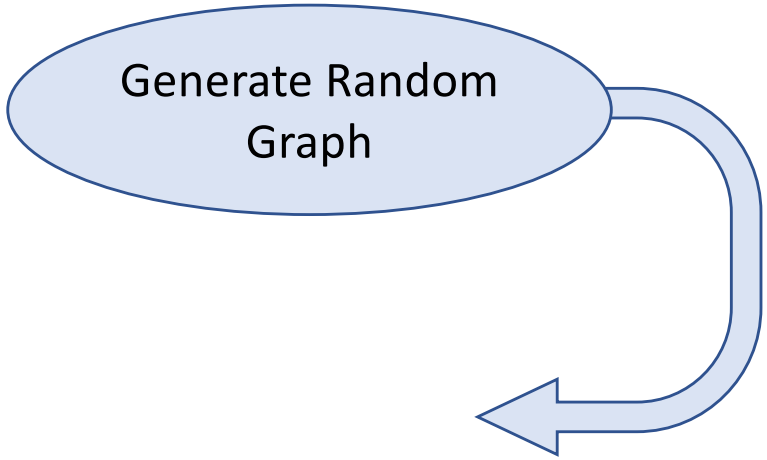
[2] Xie et al. "Exploring randomly wired neural networks for image recognition." ICCV'19

Searching for a Model

Formulating as neural architecture search (NAS) problem

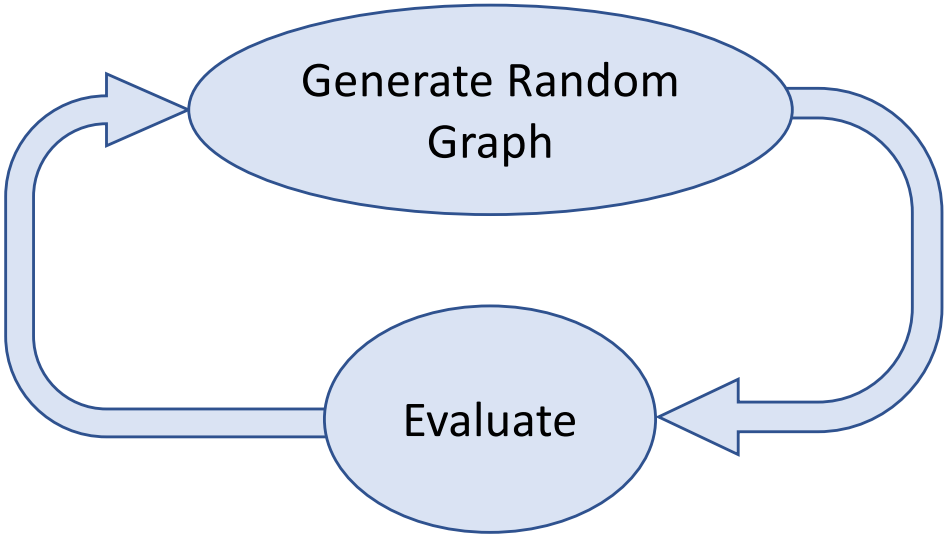
Searching for a Model

Formulating as neural architecture search (NAS) problem



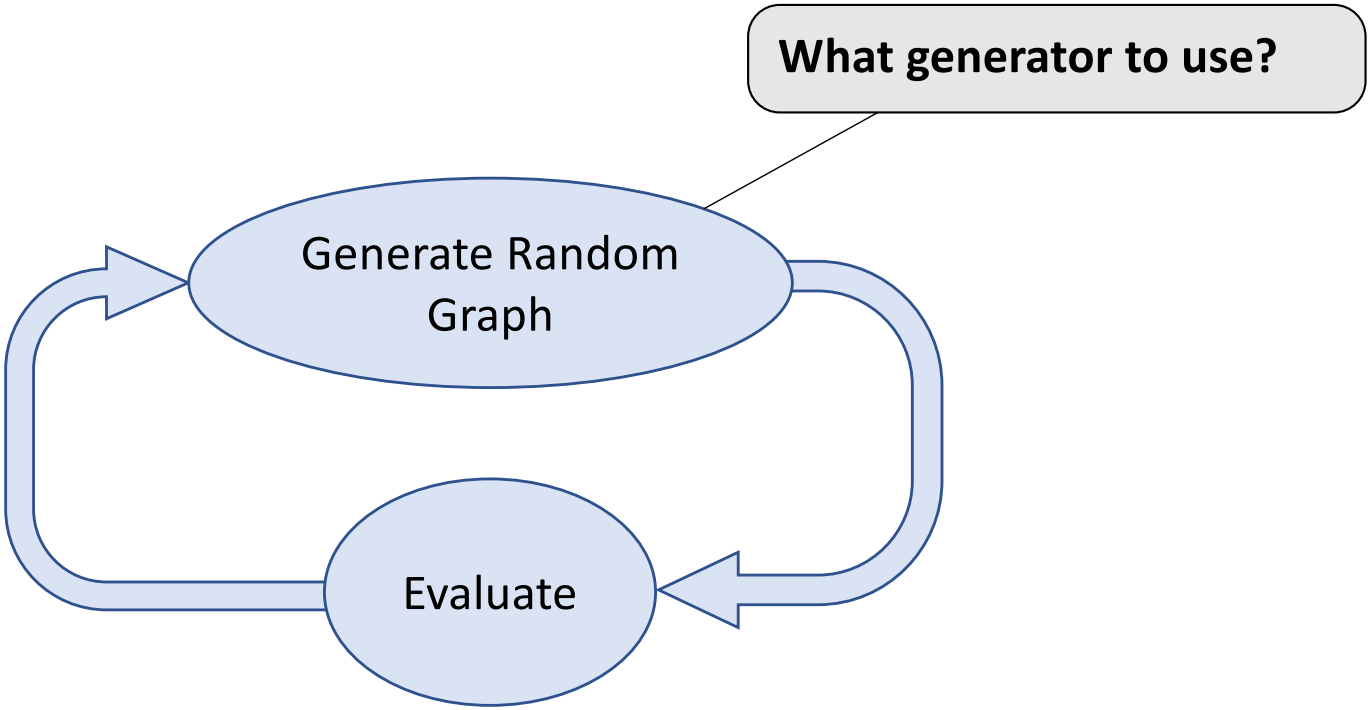
Searching for a Model

Formulating as neural architecture search (NAS) problem



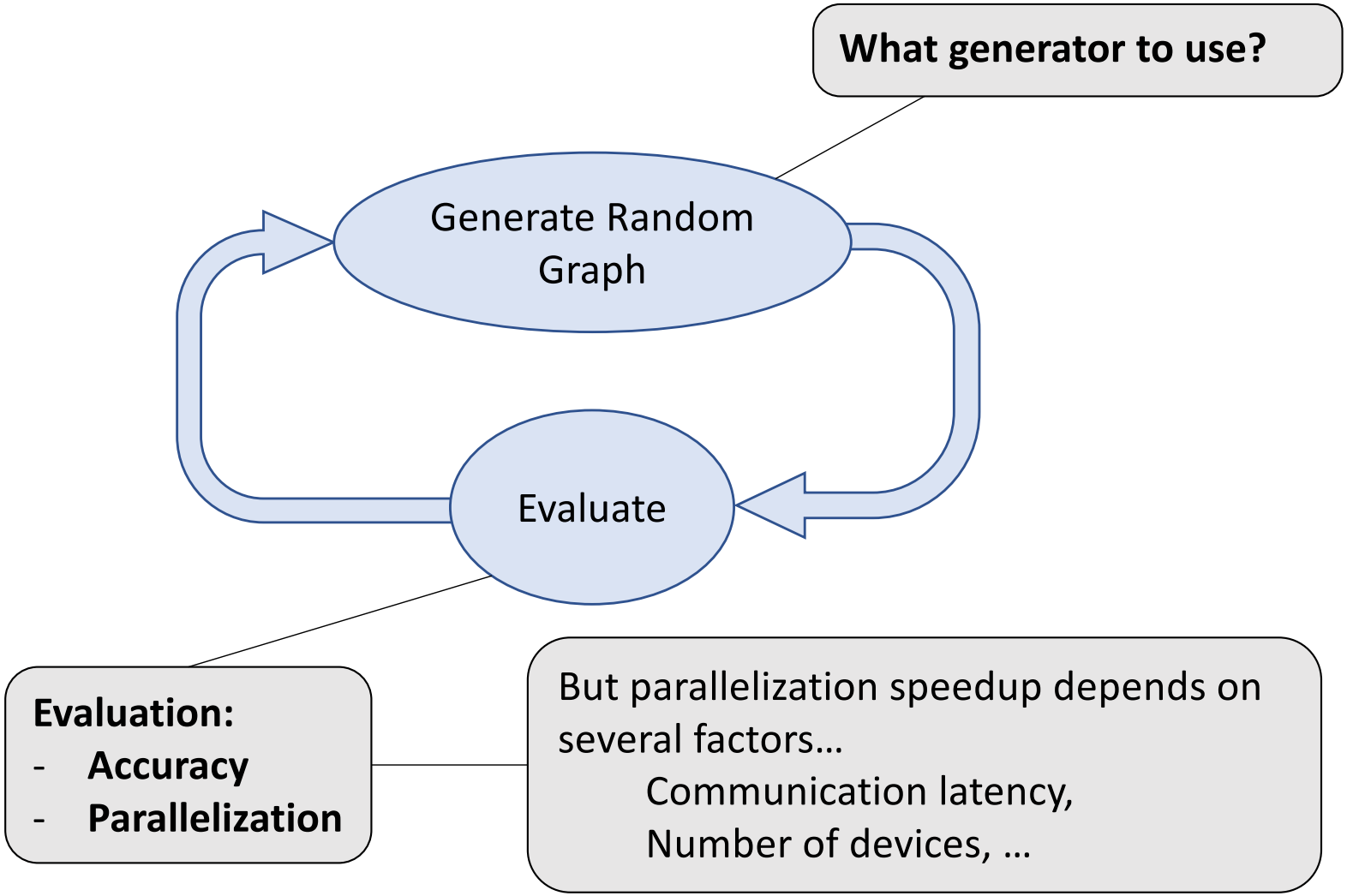
Searching for a Model

Formulating as neural architecture search (NAS) problem



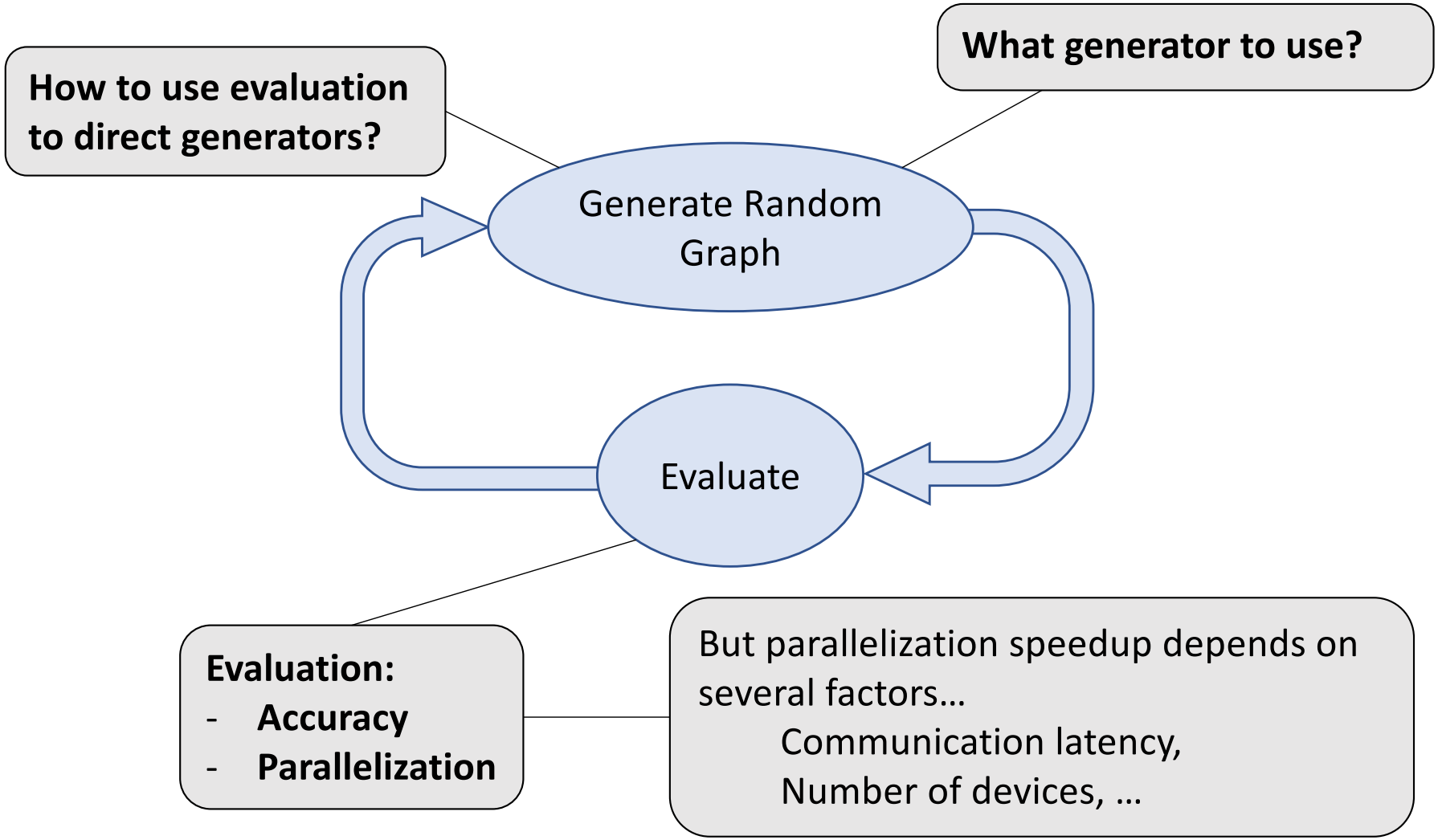
Searching for a Model

Formulating as neural architecture search (NAS) problem



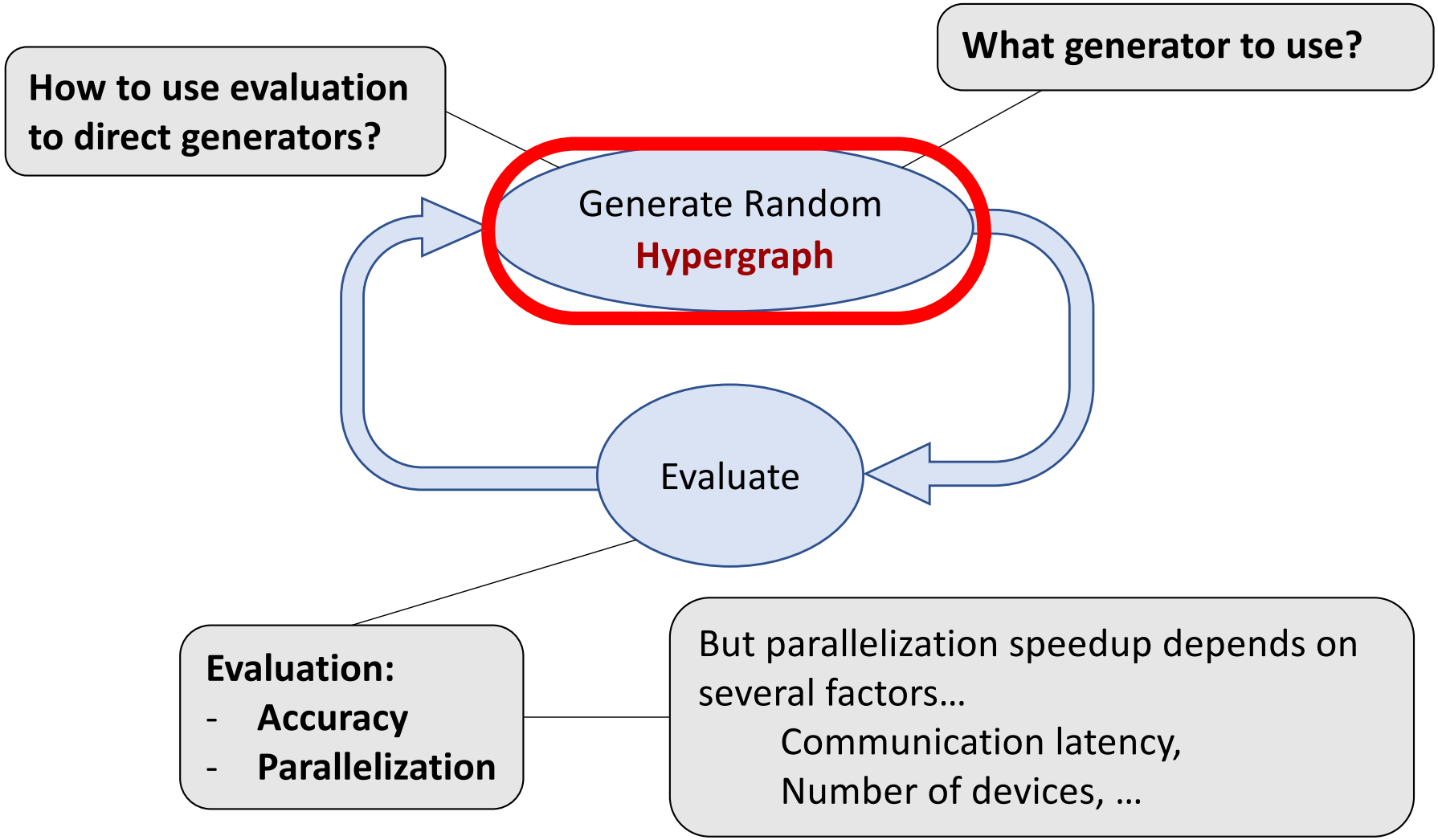
Searching for a Model

Formulating as neural architecture search (NAS) problem



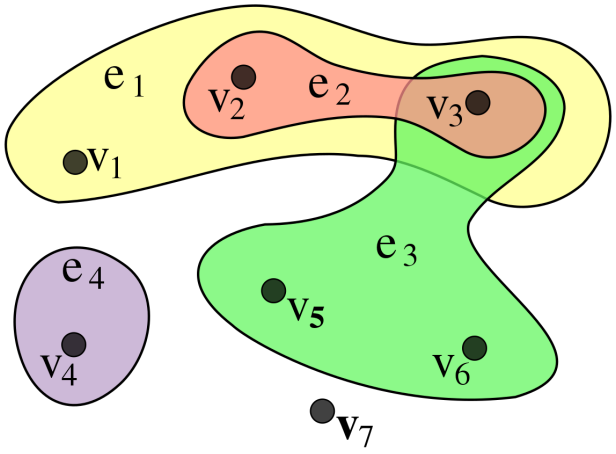
Searching for a Model

Formulating as neural architecture search (NAS) problem



Hypergraph Theory

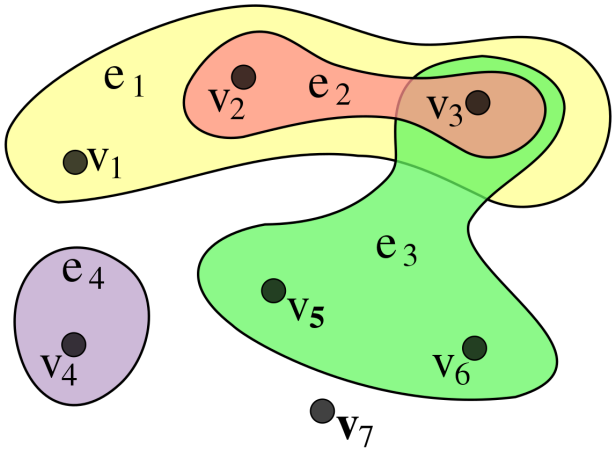
- Each edge can join any number of vertices
  - Better to represent communication



© Wikipedia

Hypergraph Theory

- Each edge can join any number of vertices
  - Better to represent communication



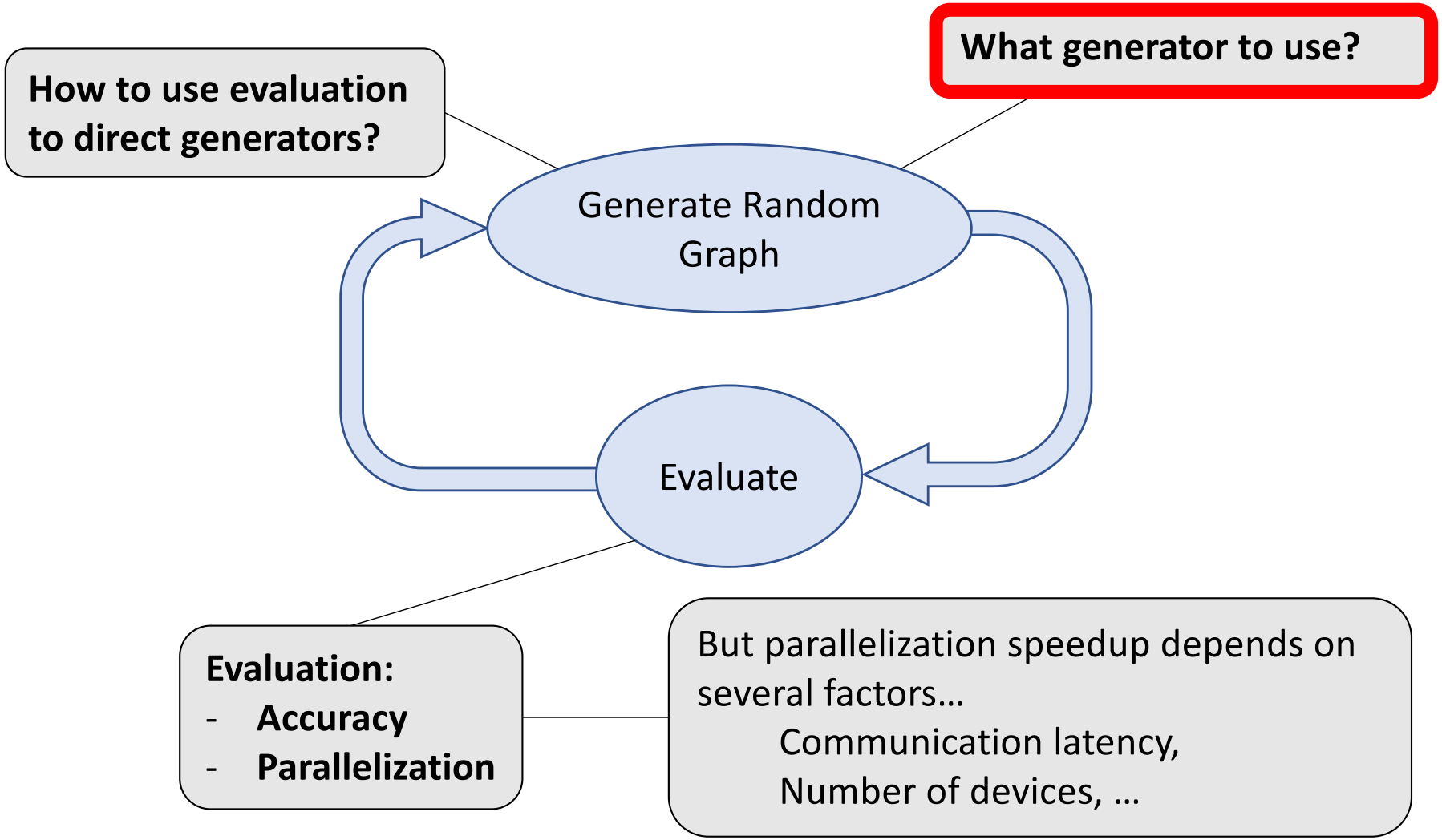
© Wikipedia

- Hypergraph partitioning
  - To find a load balanced partitioning, with minimum communication overhead on n processors
  - Common in data centers
  - Solve with METIS or PaToH [*Catalyurek et al.*]

Devine, K. D., Boman, E. G., Heaphy, R. T., Bisseling, R. H., & Catalyurek, U. V. (2006, April). Parallel hypergraph partitioning for scientific computing. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium* (pp. 10-pp). IEEE.

Searching for a Model

Formulating as neural architecture search (NAS) problem

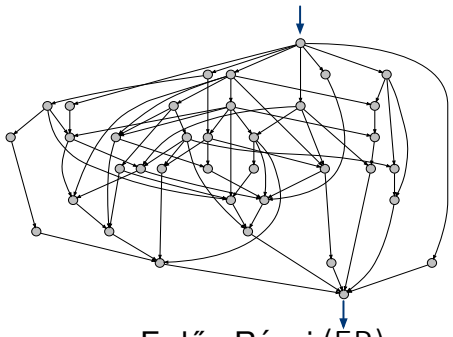


Graph Generators

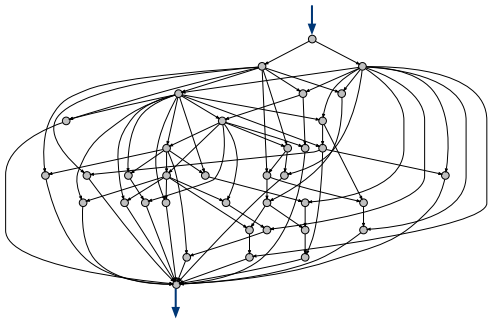
Use random graph generators to create models

Graph Generators

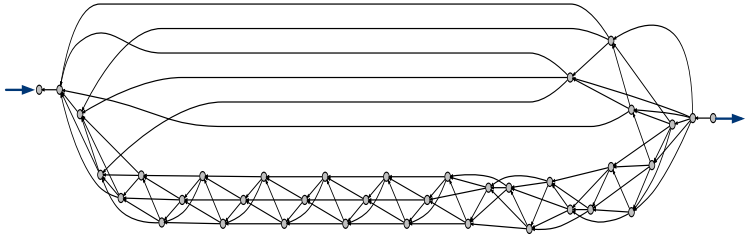
Use random graph generators to create models



Erdős-Rényi (ER)



Barabási-Albert (BA)

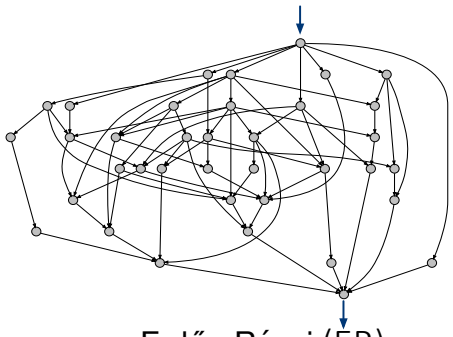


Watts-Strogatz (WS)

[1] Xie et al. "Exploring randomly wired neural networks for image recognition." ICCV'19

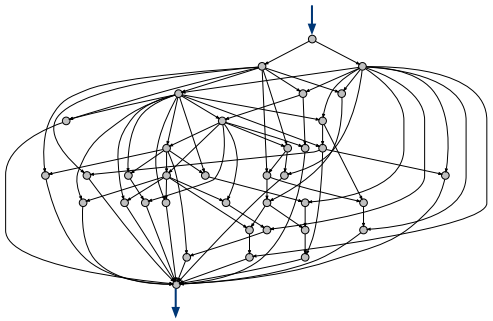
Graph Generators

Use random graph generators to create models



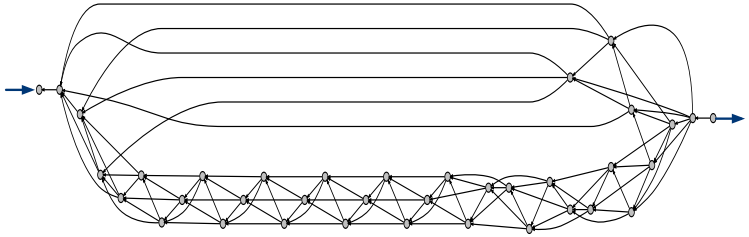
Erdős-Rényi (ER)

Not easy to parallelize



Barabási-Albert (BA)

Not easy to parallelize



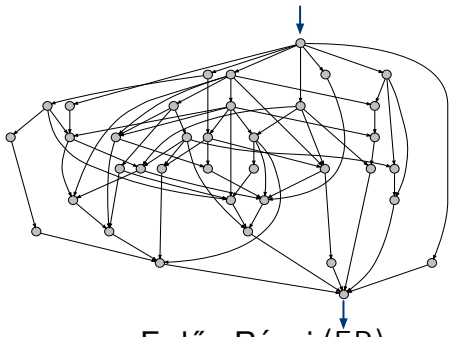
Watts-Strogatz (WS)

Not load balanced

[1] Xie et al. "Exploring randomly wired neural networks for image recognition." ICCV'19

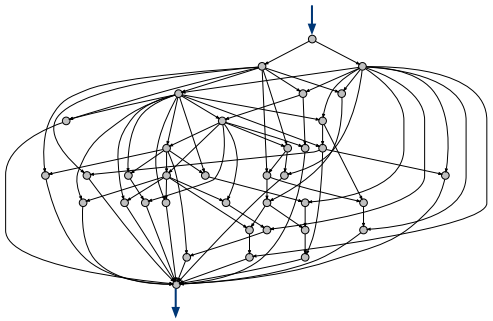
Graph Generators

Use random graph generators to create models



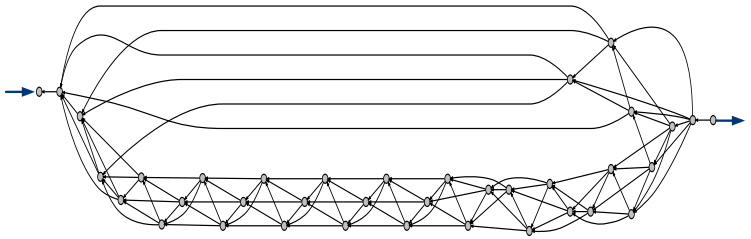
Erdős-Rényi (ER)

Not easy to parallelize



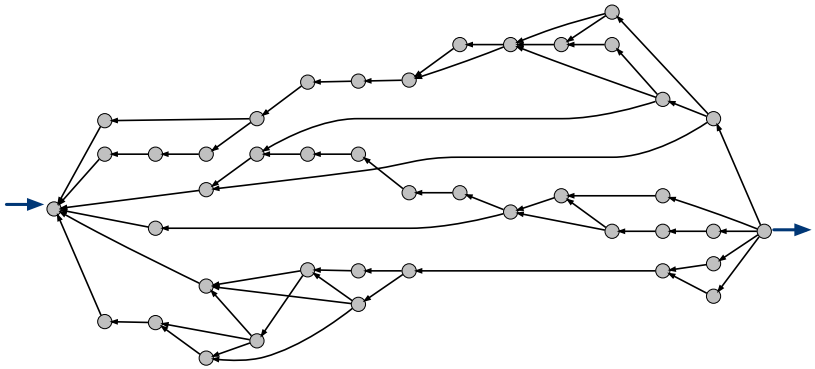
Barabási-Albert (BA)

Not easy to parallelize



Watts-Strogatz (WS)

Not load balanced



Exponential Distance Based (poposed)

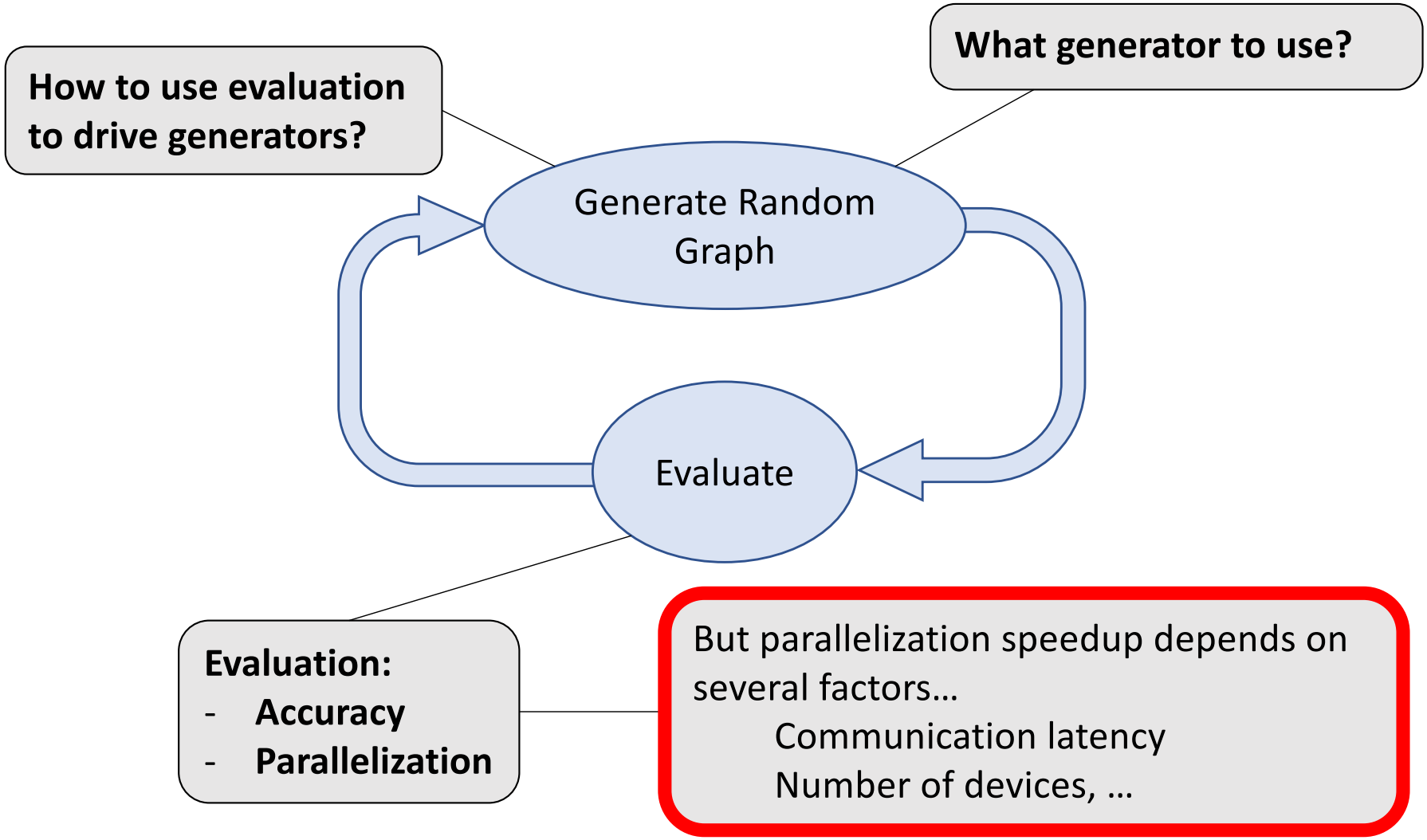
Easier to parallelize

Better in load balancing

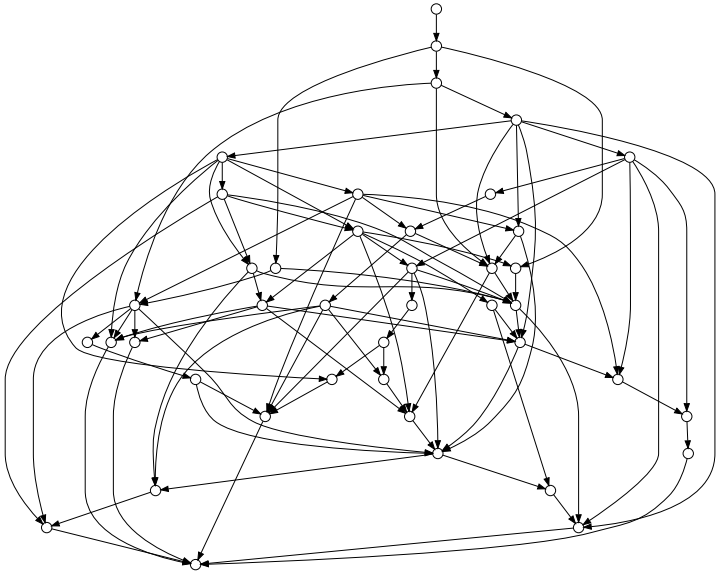
[1] Xie et al. "Exploring randomly wired neural networks for image recognition." ICCV'19

Searching for a Model

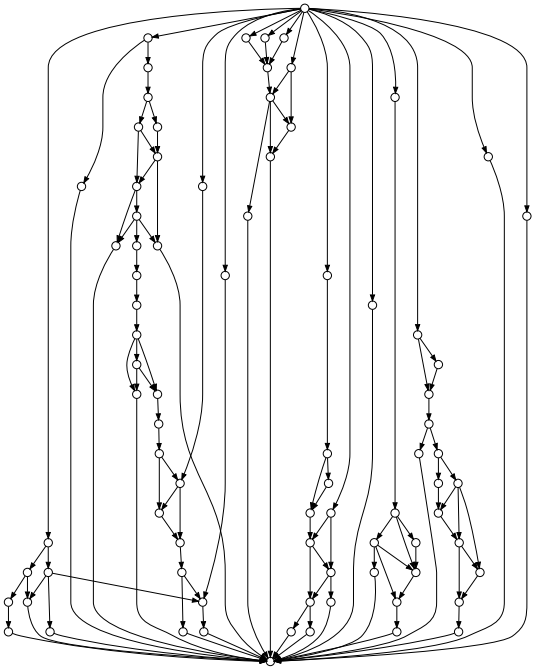
Formulating as neural architecture search (NAS) problem



Parallelization Score

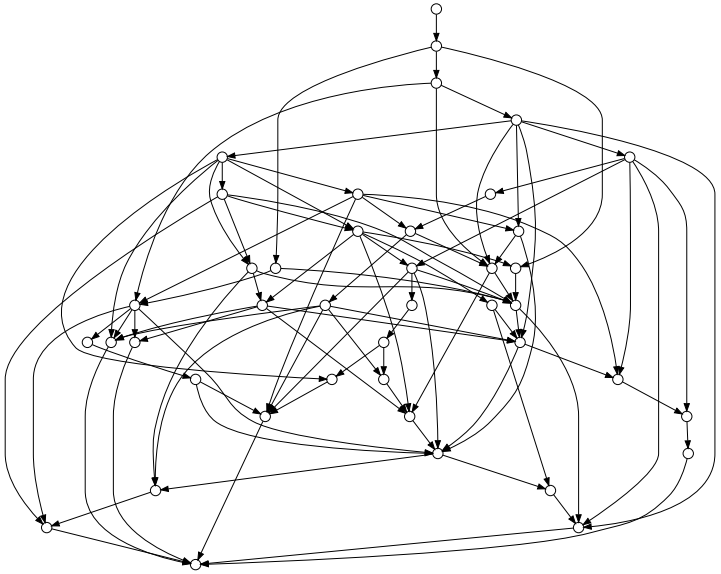


?

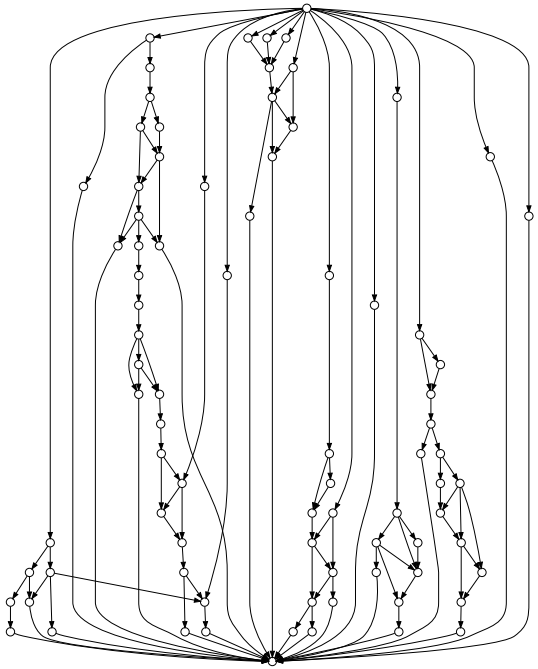


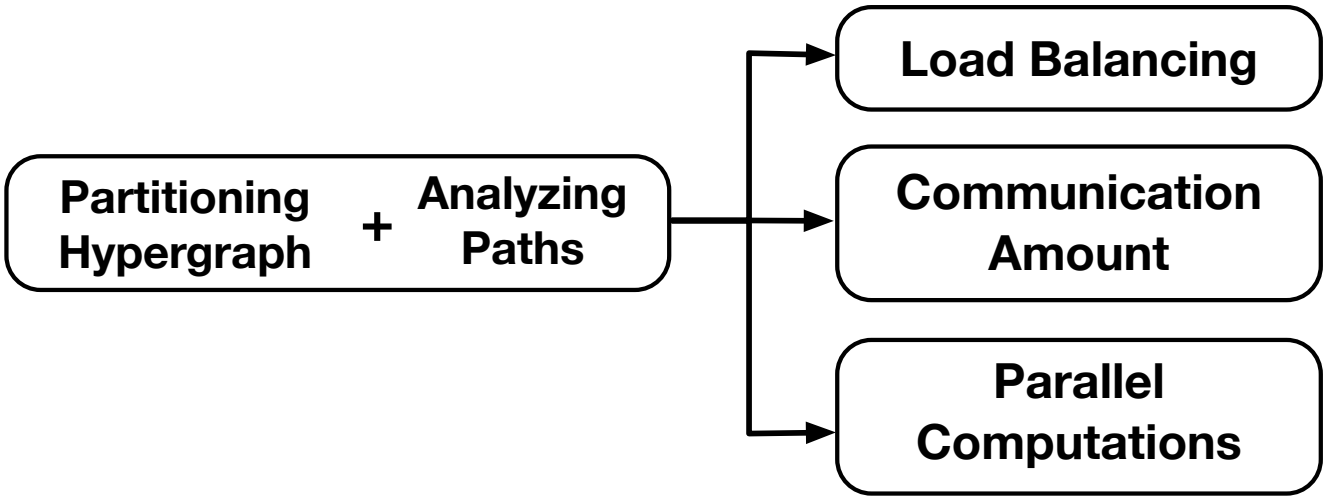
Partitioning Hypergraph + **Analyzing Paths**

Parallelization Score

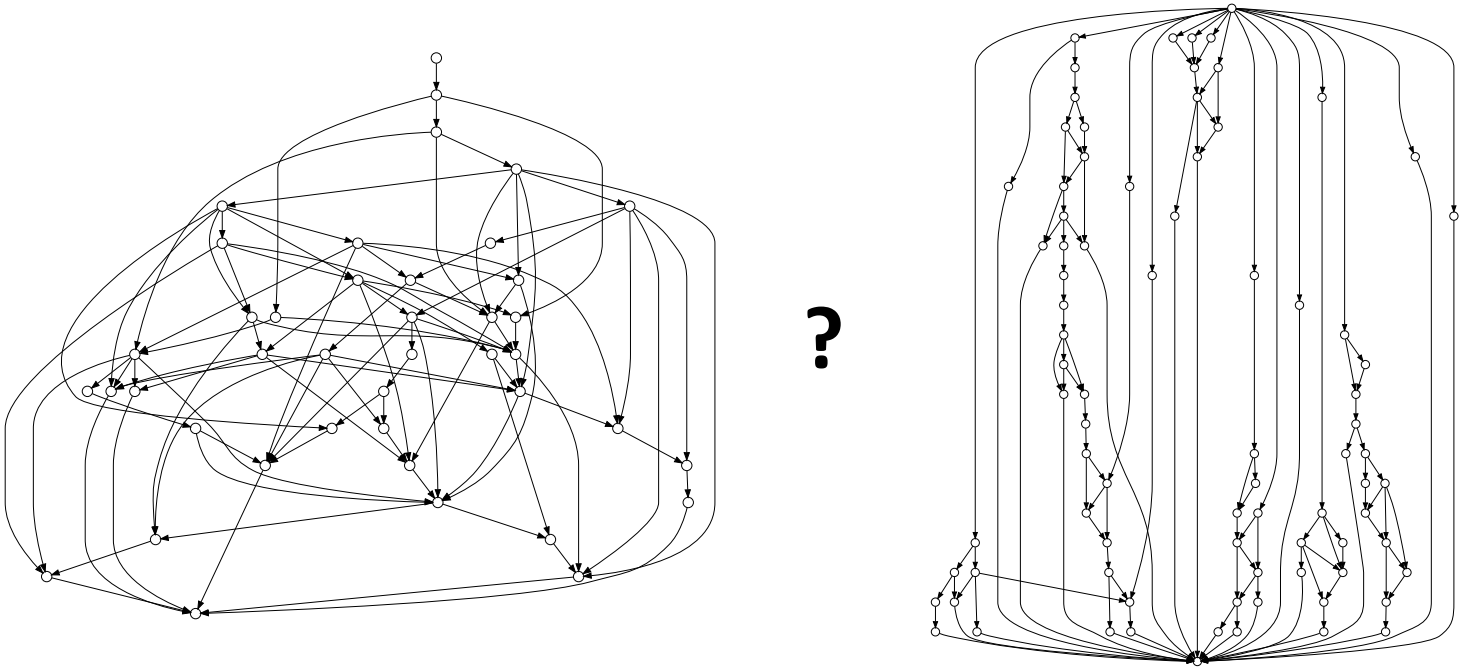


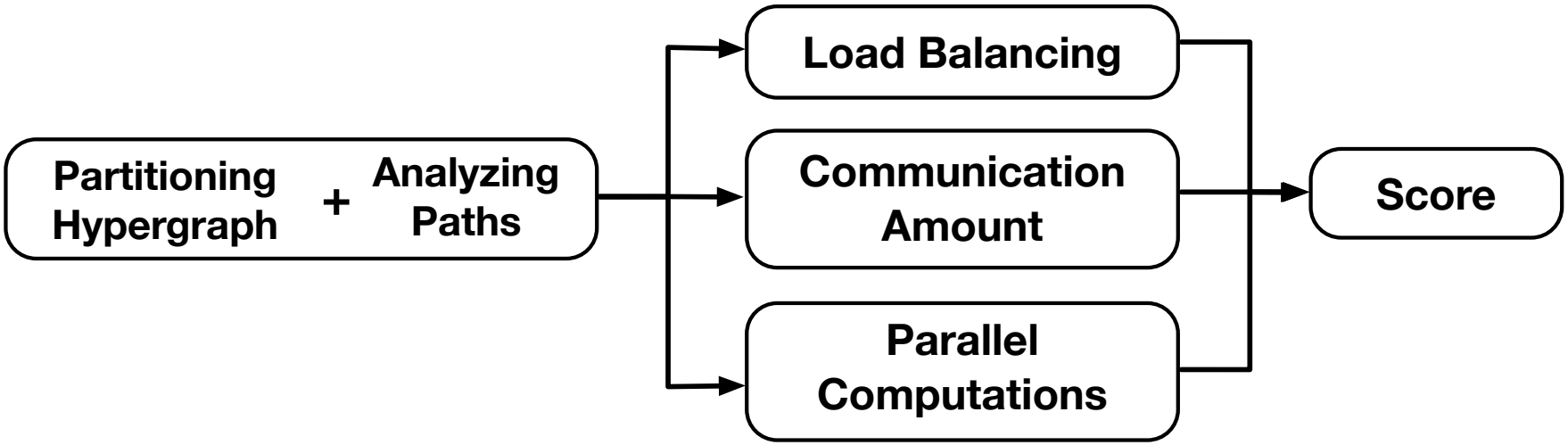
?



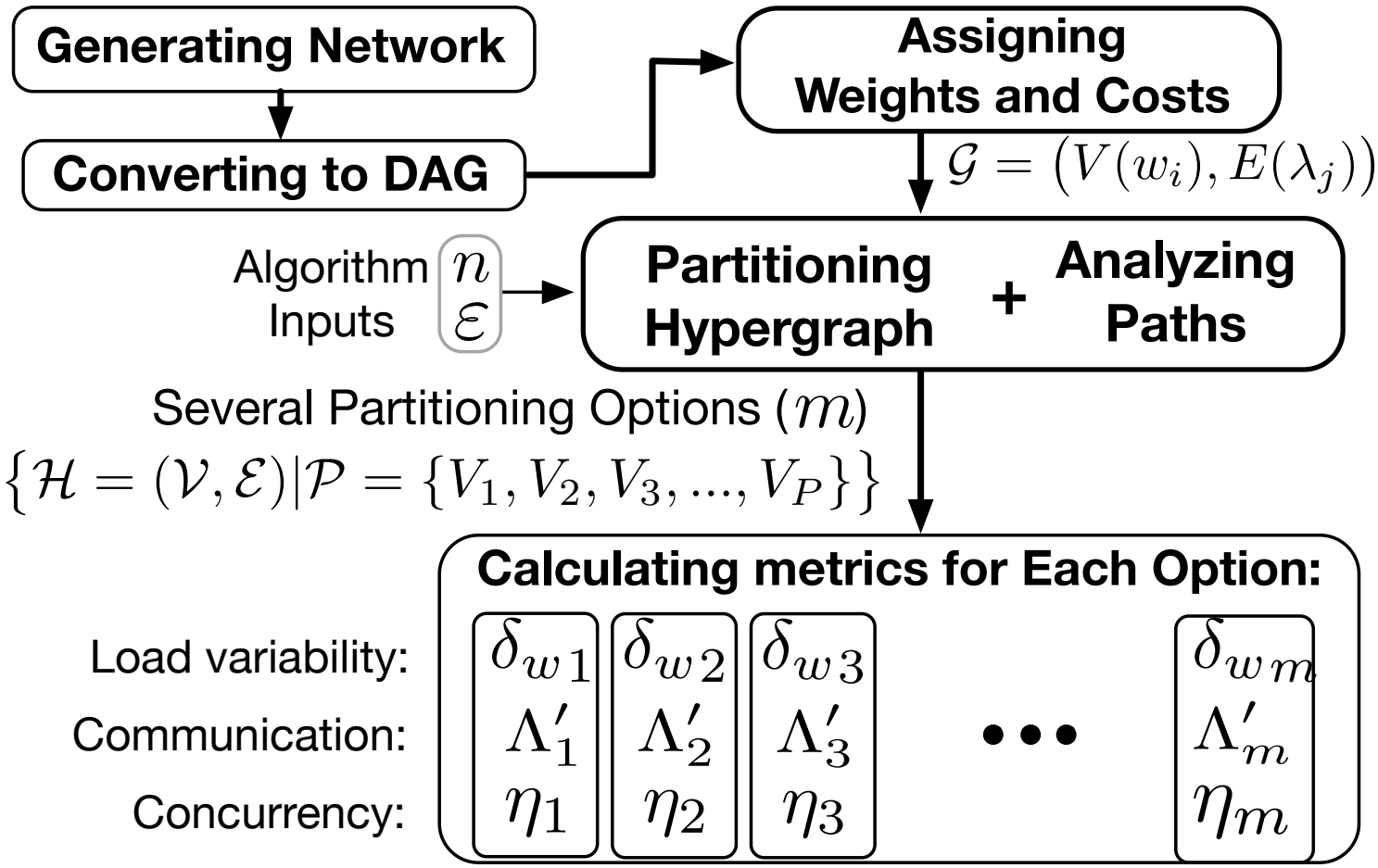


Parallelization Score

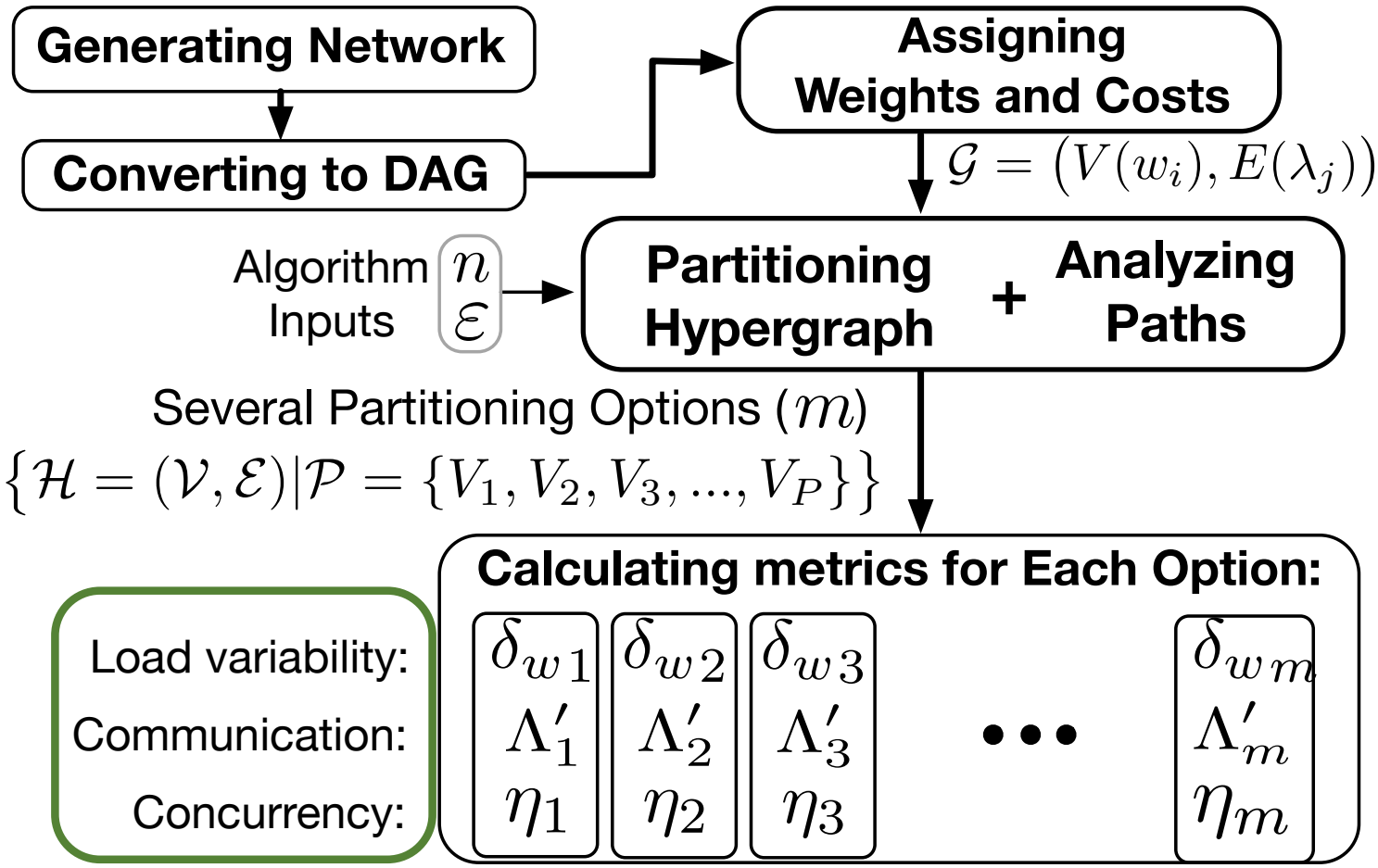




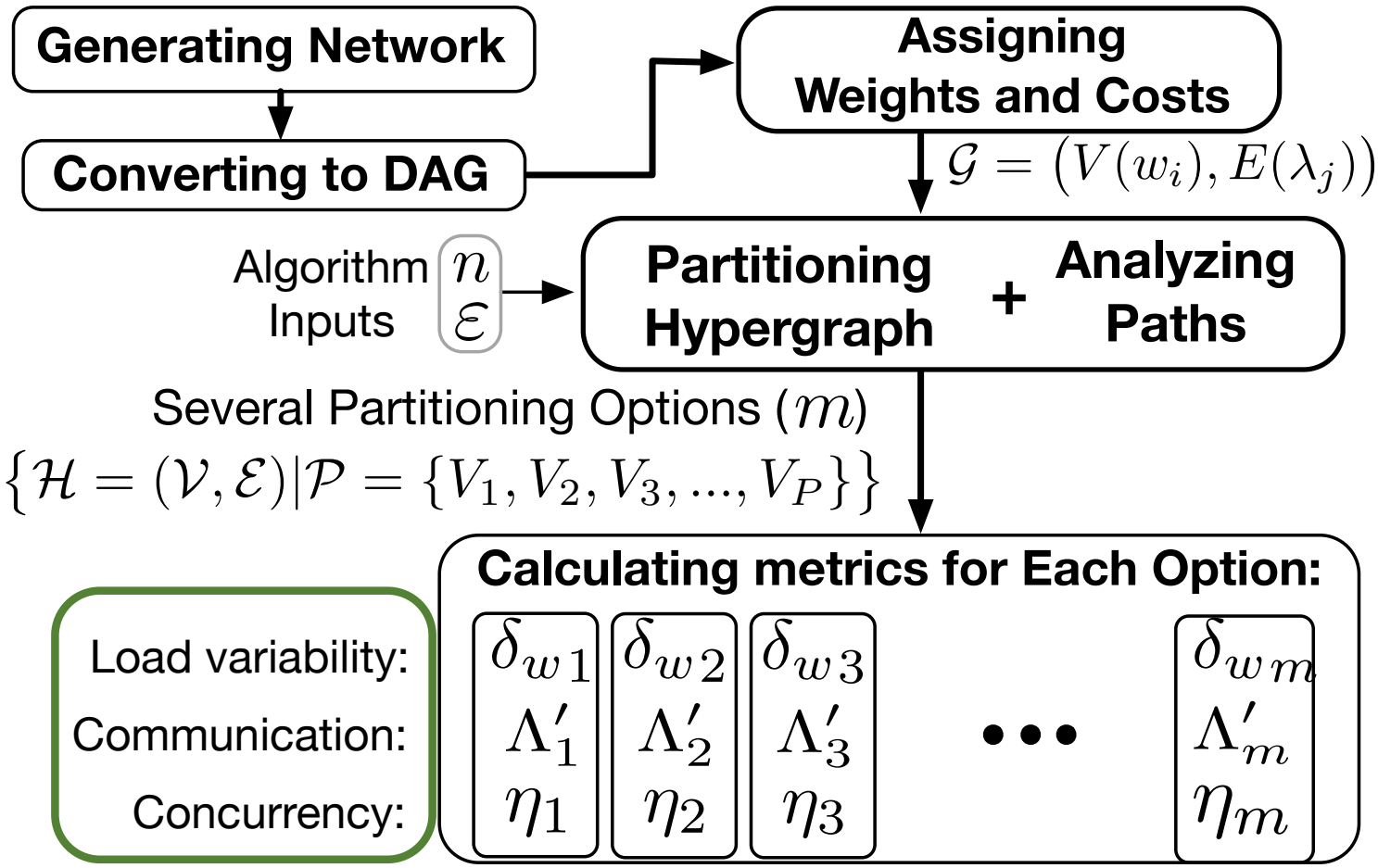
Parallelization Score



Parallelization Score



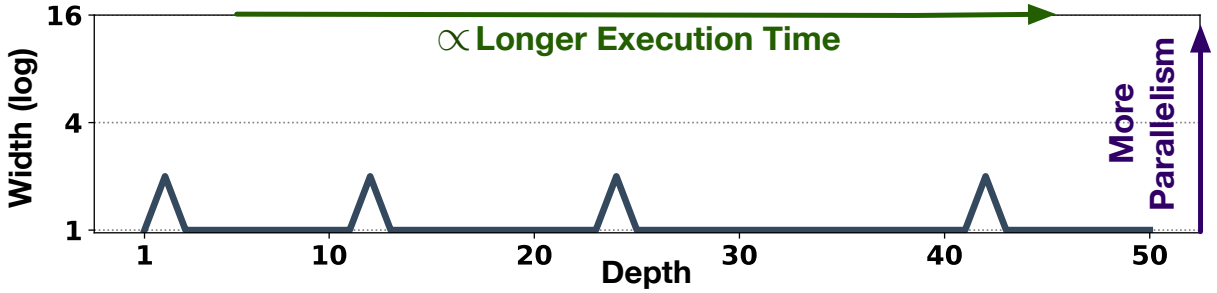
Parallelization Score



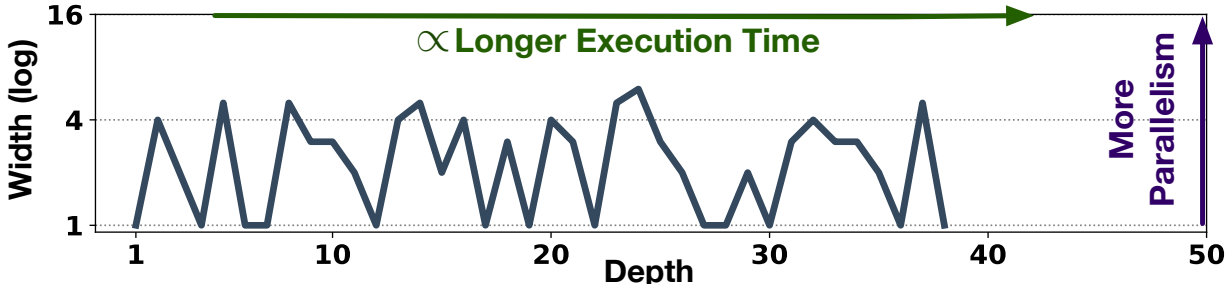
Parallelization Score: $\mathbf{PS} = \sqrt[1/3]{\delta_w^a \Lambda'^b \eta^c}$

Width vs. Depth Graphs

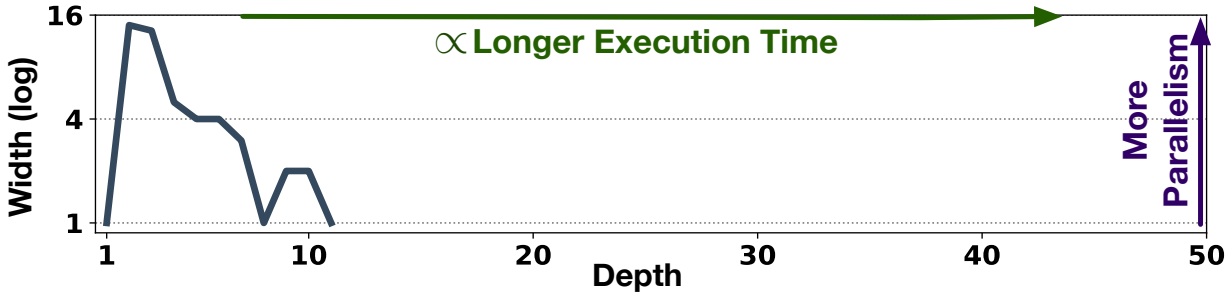
Width vs. depth of a model represents how well parallelizable a model is beyond the computations within a single layer



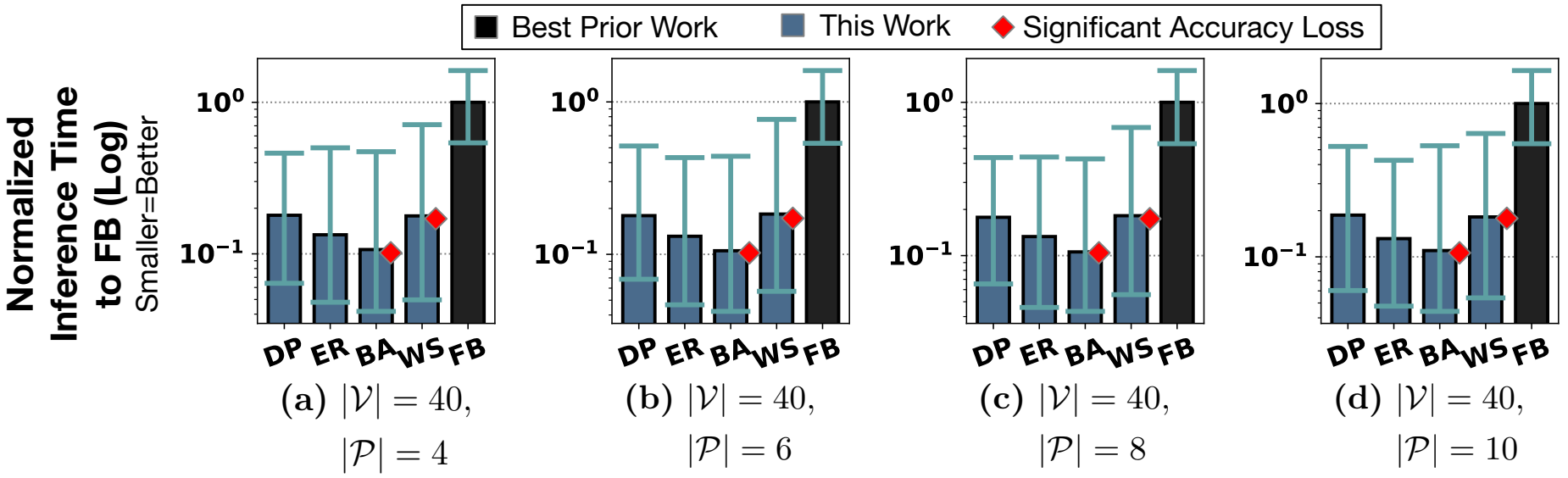
(a) ResNet50



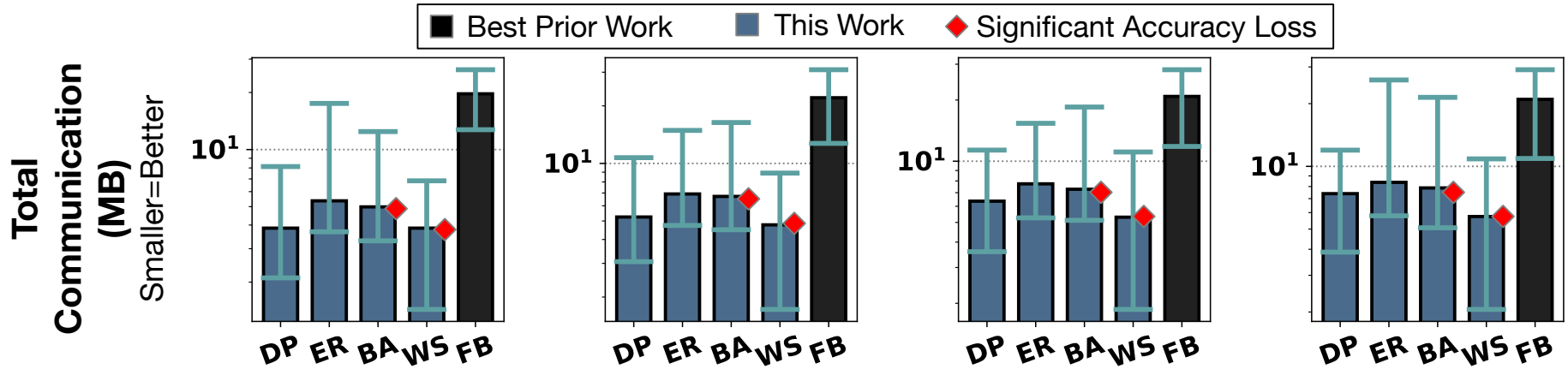
(b) Prior work



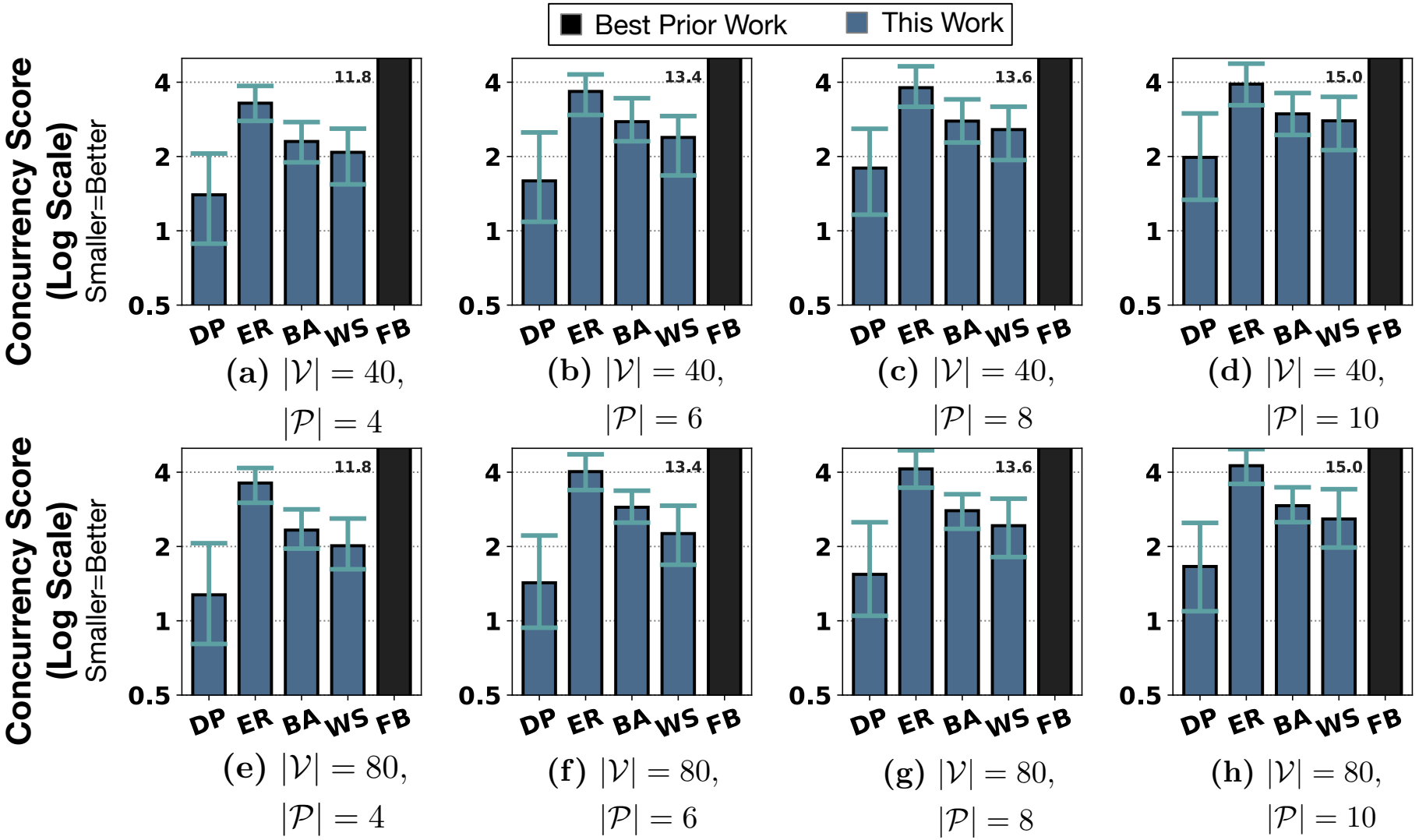
(c) This Work



(a) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 4$ (b) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 6$ (c) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 8$ (d) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 10$



(a) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 4$ (b) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 6$ (c) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 8$ (d) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 10$

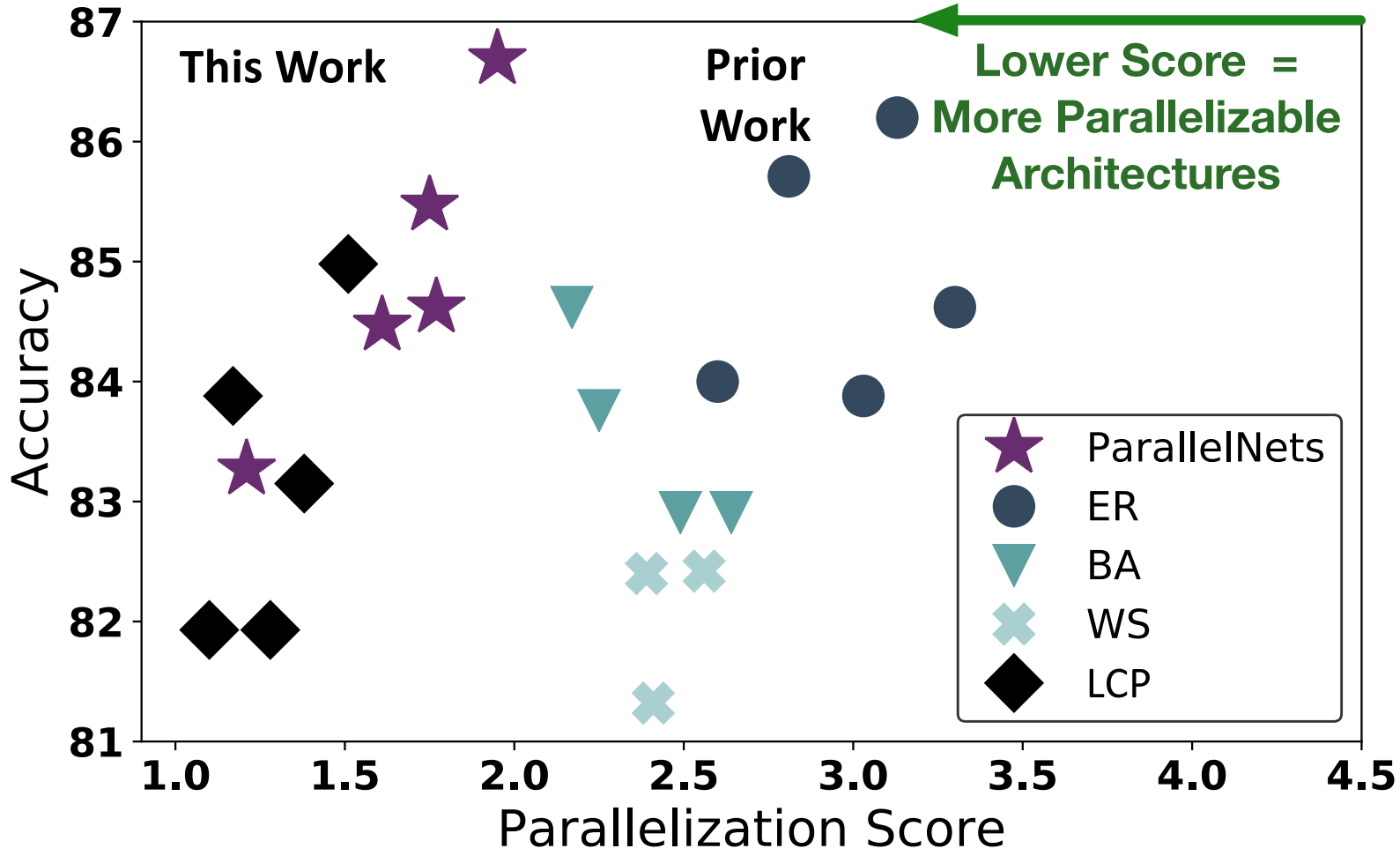


(a) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 4$ (b) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 6$ (c) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 8$ (d) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 10$

(e) $|\mathcal{V}| = 80$, $|\mathcal{P}| = 4$ (f) $|\mathcal{V}| = 80$, $|\mathcal{P}| = 6$ (g) $|\mathcal{V}| = 80$, $|\mathcal{P}| = 8$ (h) $|\mathcal{V}| = 80$, $|\mathcal{P}| = 10$

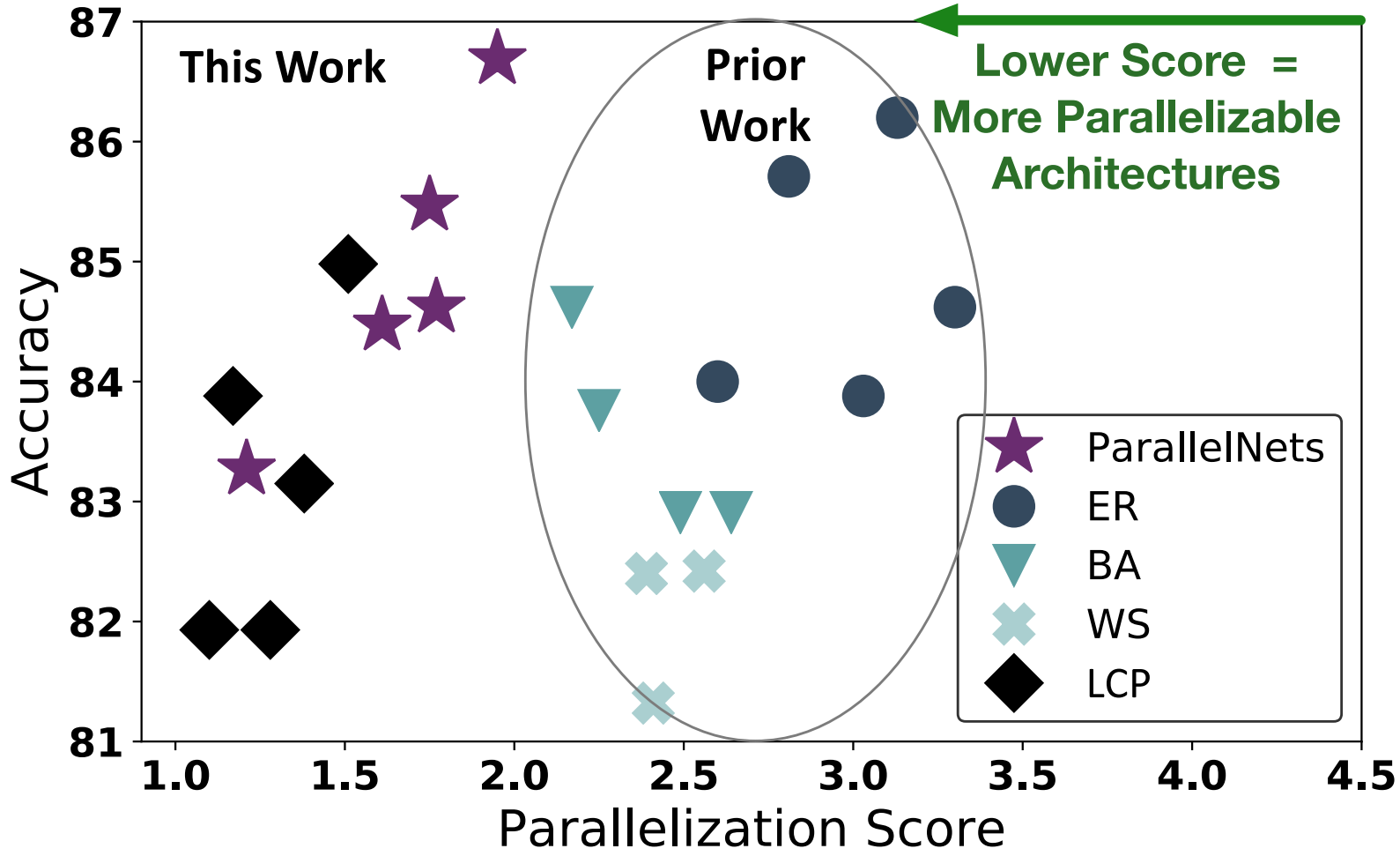
Parallelization Score Results

Results on small datasets



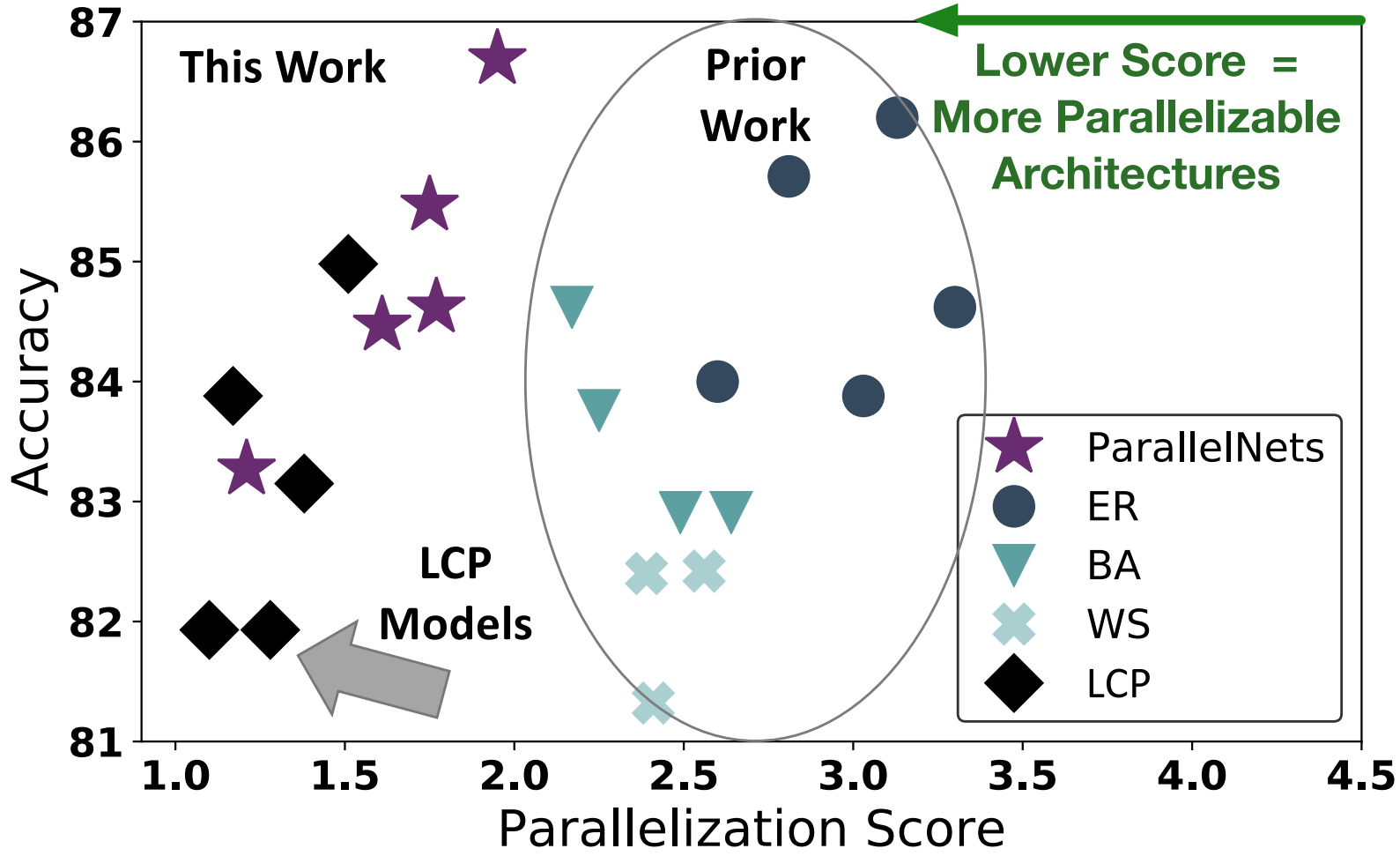
Parallelization Score Results

Results on small datasets



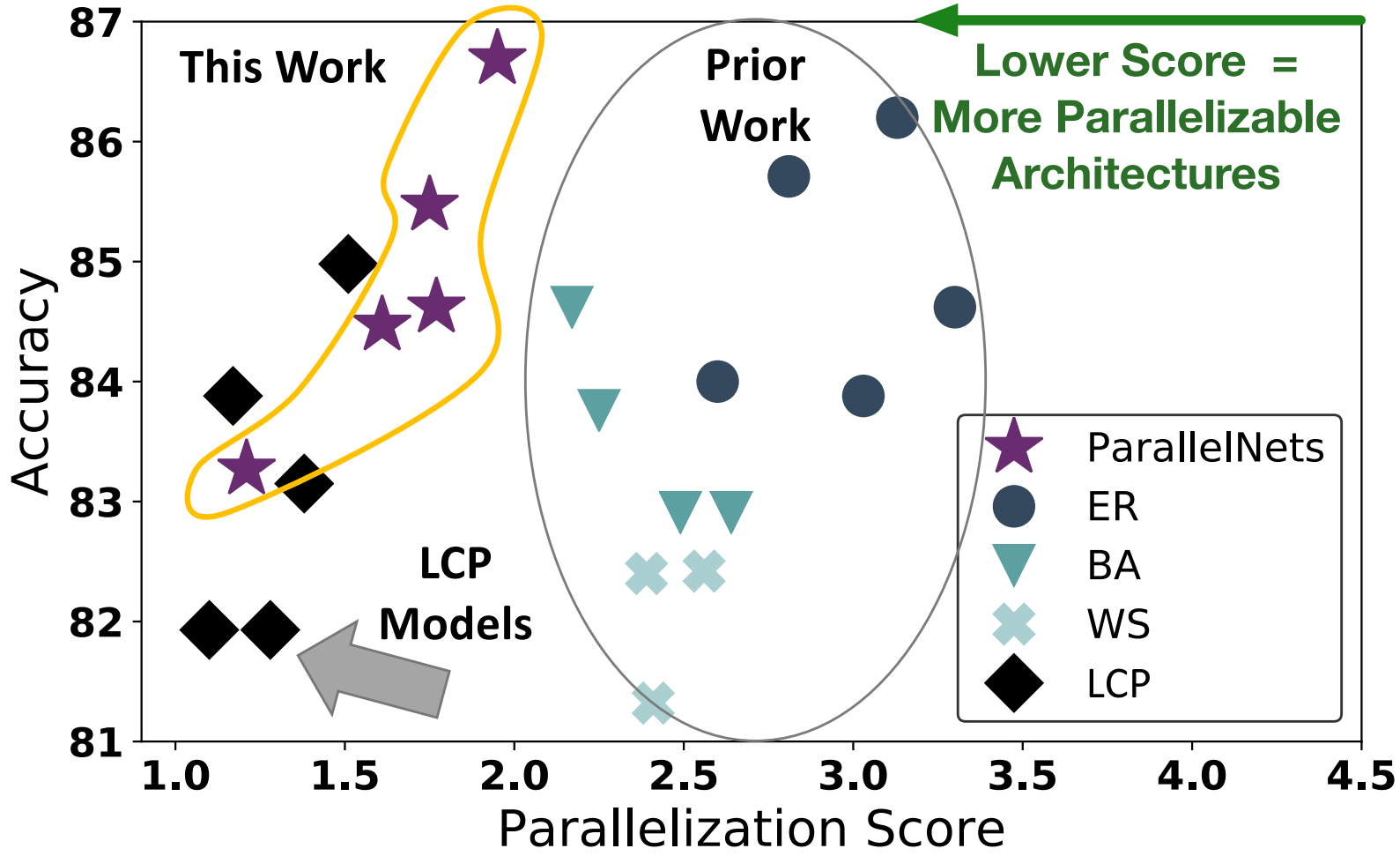
Parallelization Score Results

Results on small datasets



Parallelization Score Results

Results on small datasets



Please check the paper for more details on

- Distribution
- Scaling blocks & strategy
- Example deep dive
- Code
- Parallelization score
- Results on speedup and communication

Reducing Inference Latency with Concurrent Architectures for Image Recognition at Edge

Ramyad Hadidi[§*] — *Rain AI* — ramyad@rain.ai

Jiashen Cao[§] — *Georgia Tech* — jiashenc@gatech.edu

Michael S. Ryoo — *Stony Brook University and Google* — mryoo@cs.stonybrook.edu

Hyesoon Kim — *Georgia Tech* — hyesoon.kim@gatech.edu

***Abstract*—Satisfying the high computation demand of modern deep learning architectures is challenging for achieving low inference latency. The current approaches in decreasing latency only increase parallelism within a layer. This is because architectures typically capture a single-chain dependency pattern that prevents efficient distribution with a higher concurrency (*i.e.*, simultaneous execution of one inference among devices). Such single-chain dependencies are so widespread that even implicitly biases recent neural architecture search (NAS) studies. In this visionary paper, we draw attention to an entirely new space of NAS that relaxes the single-chain dependency to provide higher concurrency and distribution opportunities. To quantitatively compare these architectures, we propose a score that encapsulates crucial metrics such as communication, concurrency, and load balancing. Additionally, we propose a new generator and transformation block that consistently deliver superior architectures compared to current state-of-the-art methods. Finally, our preliminary results show that these new architectures reduce the inference latency and deserve more attention.**

***Index Terms*—Edge AI, Neural Architecture Search, Distributed and Collaborative Edge Computing, IoT, Collaborative Edge & Robotics**

I. Introduction & Motivation

Increasingly deeper and wider convolution/deep neural networks (CNN/DNN) [1]–[3] with higher computation demands are continuously attaining higher accuracies. Nevertheless, the high computation and memory demands of these DNNs hinder achieving low inference latency [4]. Although current platforms exploit parallelism, we discover that, since most architectures capture a *single-chain dependency pattern* [5]–[7], shown in Figures 1a & b, we cannot efficiently extend concurrency and distribution beyond current explicit parallelism exposed within intra-layer computations (*i.e.*, matrix-matrix multiplications) to reduce the latency of an inference. In other words, distribution and concurrency, if any, are implemented at data level [8], which only increases the throughput.

The status quo approaches in reducing the inference latency are always applied *after* an architecture is defined (*e.g.*, reducing parameters with weight pruning [9], [10] or reducing computation with quantization or compression [11]–[13]). Additionally, for extremely large architectures, limited model parallelism is applied on final layers (*i.e.*, large fully-connected layers that do not fit in the memory of edge devices [14]–[16]). However, since model-parallelism methods do not change the architecture, distributing all layers with such methods adds several synchronization/merging points, incurring high communication overheads (Figure 1a & b). We discover that the single-chain inter-layer dependency pattern, common in all the well-known architectures and even in state-of-the-art neural architecture search (NAS) studies [17], prevents the efficient model distribution for reducing inference latency.

This visionary paper addresses the single-chain data dependency in current architecture designs and endeavors to inspire discussion for new concurrent architectures for at-edge distribution. To do so, first, we analyze architectures generated by recent unbiased NAS studies [17] and discover that *scaling/staging* blocks implicitly enforce dependencies. Then, we generate new architectures with prior and our new distance-based network generators using our new probabilistic scaling block. Then, for quantitatively comparing generated architectures, we propose a *concurrency score* that encapsulates important metrics such as communication, load balancing, and overlapped computations, by reformulating the problem as a hypergraph partitioning problem [18], [19]. Based on the scores and experiments, our generated architectures have higher concurrency and are more efficient for distribution

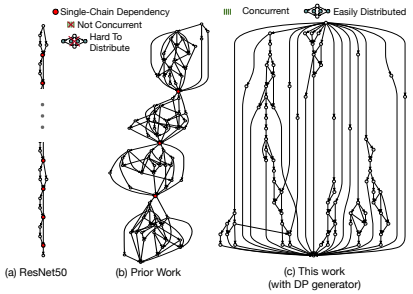


Fig. 1: **Sampled Architectures Overview** – (a) & (b) Limited concurrency and distribution due to single-chain dependency. (c) Improved concurrent architecture.

This work was partially supported by the NSF grant number 2103951 and Institute of Information and Communications Technology Planning and Evaluation grant funded by the Korea government (No. 2021-0-00766).

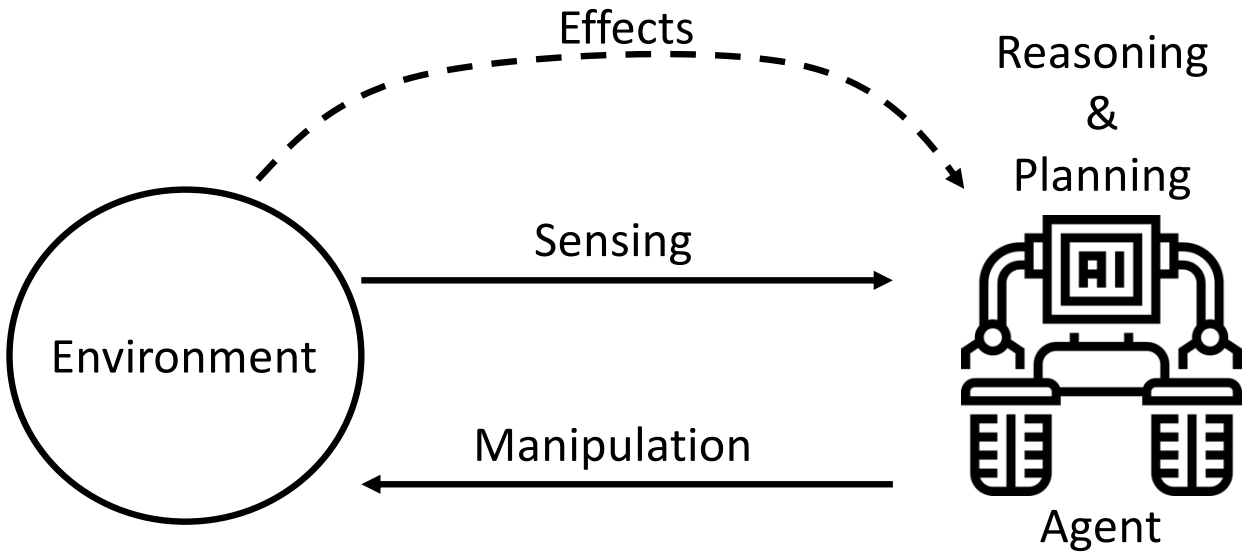
[§]Equal contribution

[*]This work was done when the author was affiliated with Georgia Tech.

Intelligence Requires Computation

Intelligent edge devices

For instance, robots need to sense, manipulate, and reason about their environment, all of which imposes heavy computations



Heavy Computations

Heavy computations are usually in the form of deep neural networks (DNNs) inference

- Allowing to function in diverse situations
- Requiring to perform inference computation locally in the edge on the device:

i.e., in-the-edge inference

©Tesla

©Skydio

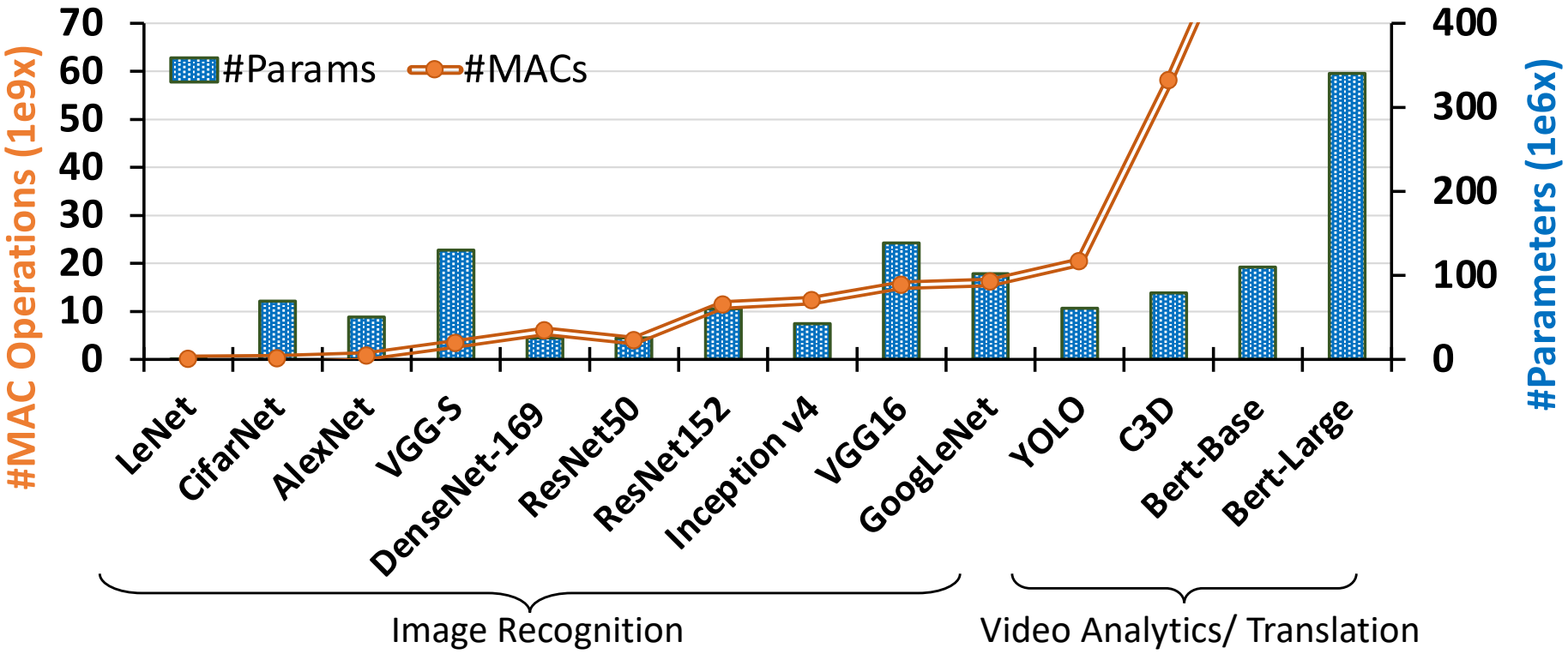
In-the-Edge Inference

- In-the-edge applications
  - Intelligence in self-driving cars, smart homes/cities
- Sometimes is the only option
  - No Internet connectivity
  - Intermittent connectivity
- Privacy preserving
  - Straightforward way to preserve privacy and security
  - Personalization
- Even faster
  - No cost associated with communication latency
- Sometimes cost($) efficient

In-the-Edge Inference (Challenge)

Edge devices cannot handle such heavy computations due to lack of resources

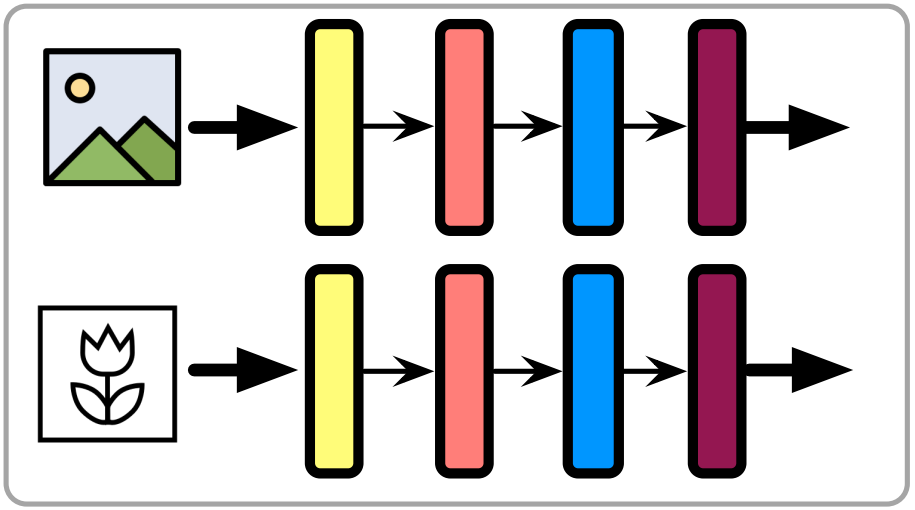
Newer DNNs are heavier for better understanding



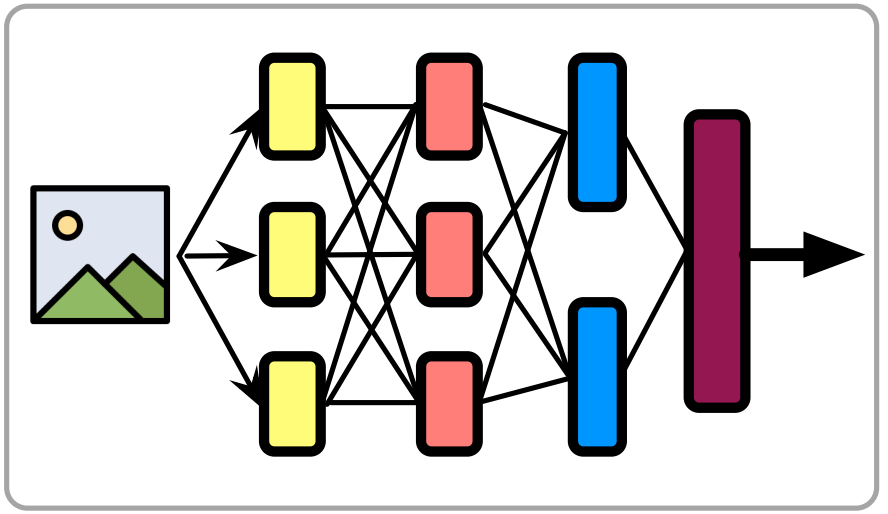
Model vs. Data Parallelism

- Data Parallelism – **Throughput Oriented**
  - Requires several input
  - High computation and memory footprints per device
  - Does not break down heavy layers
  - No adjustable work per device
- Model Parallelism – **Latency Oriented**
  - Requires one input
  - Exploits parallelism within a layer
  - Breaks down heavy layers
  - Adjustable work per device

Data & Model Parallelism



Data Parallelism



Model Parallelism

Data parallelism provides the next input to the next devices in a network

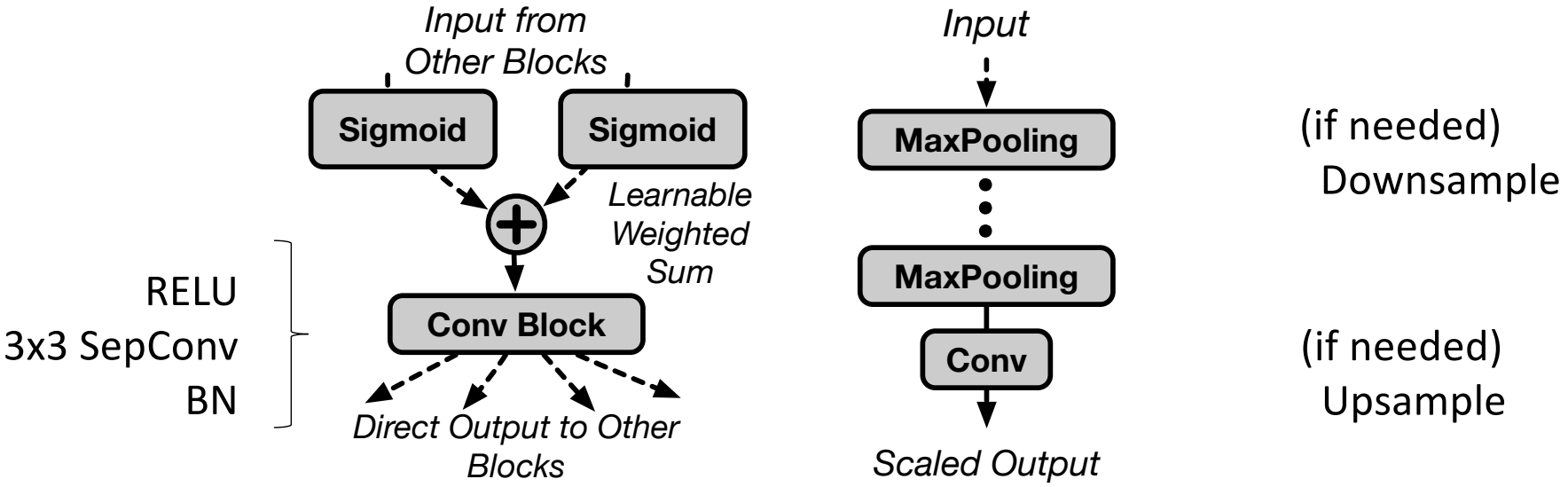
Model parallelism splits layers over multiple devices, working on the same input

Uniform Channels Do Not Scale

TABLE I: **Accuracy of Uniform Channels** – The mean accuracy comparison between sampled group architectures with uniform channel *vs.* handcrafted without any advanced optimizations. (baselines Cifar-10 and Flower-102 are vanilla CifarNet and ResNet-50, respectively).

Dataset	**Baseline**	**DNNs with Uniform Channels**
Cifar-10 32×32	80.70	81.13
Flower-102 224×224	87.80	74.73 **(Fails to Scale!)**

Blocks



(a) Basic Building Block. (b) Scaling Building Block.

Staging (1)

- Greedy-based Staging:
  - We set an upper limit for channel size.
  - As long as channel sizes have not reached the upper bound, we conduct staging (*i.e.*, down-sample the input & upsample the channel)
  - However, this design raises an issue that intermediate outputs are quickly squeezed through the maxpooling layer, which discards important features.
  - This approach hurts the accuracy to some extent.

Staging (2)

- Probabilistic-based staging
  - In this design, although the channel size may have not reached the limit, staging is done with a fixed probability of 0.5 to avoid discarding features too quickly.

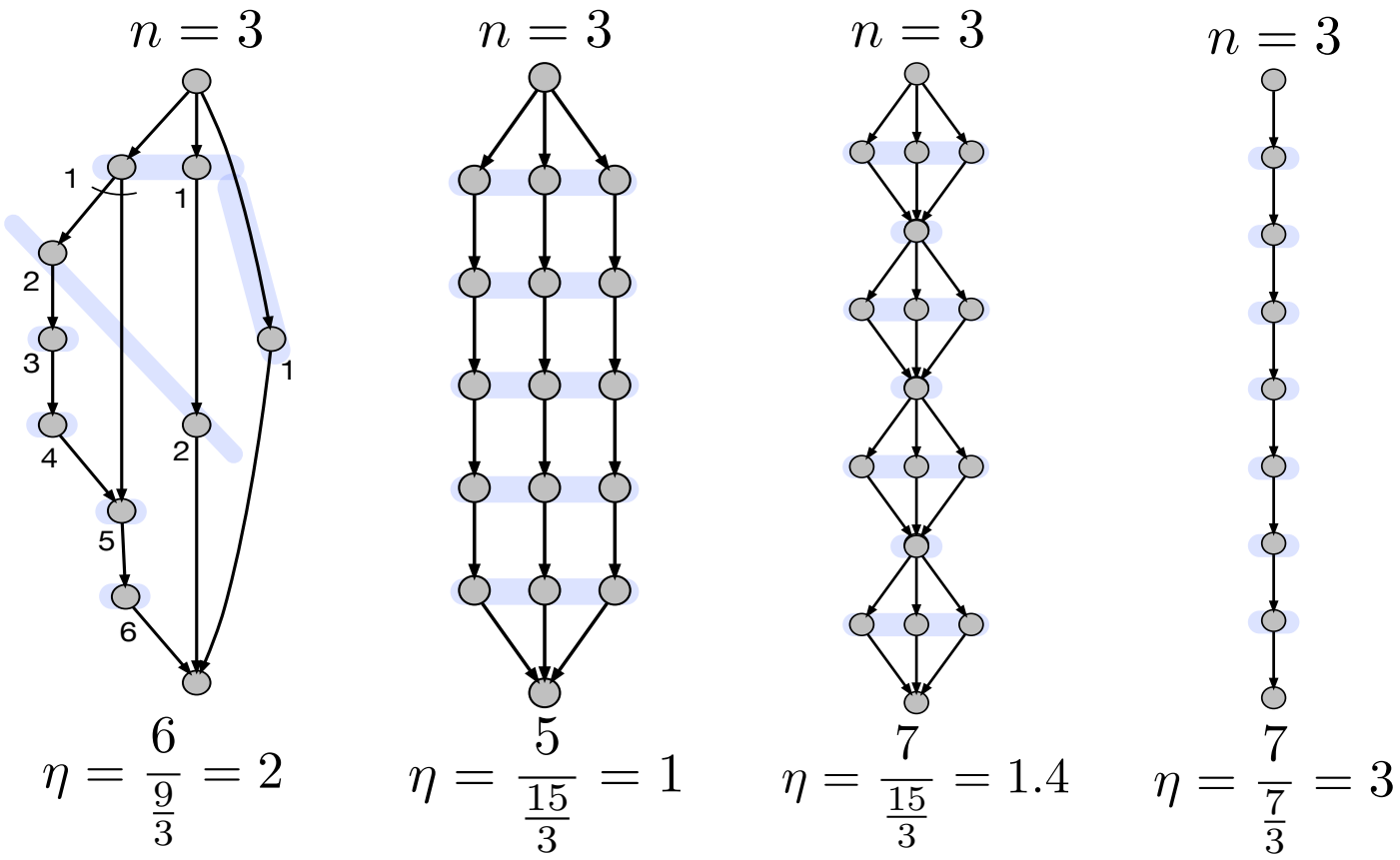
TABLE II: **Average Accuracy** – Comparison of randomly sampled group of generated architectures with different staging choices (trained on Flower-102).

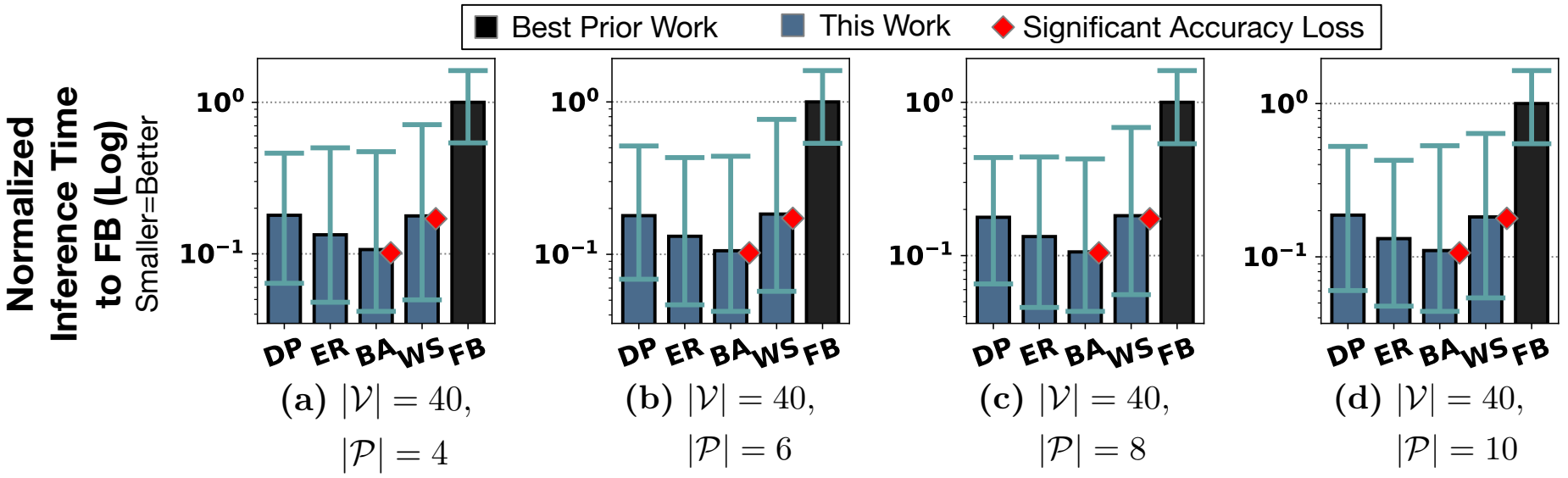
Staging/Samples	A	B	C	Overall Mean
Greedy	82.30	81.32	82.42	82.01
Probabilistic	82.42	86.69	84.62	**84.58**

TABLE III: **Average Accuracy/Parameters Ratio** – Comparison of randomly sampled generated architectures with different staging choices (trained Flower-102).

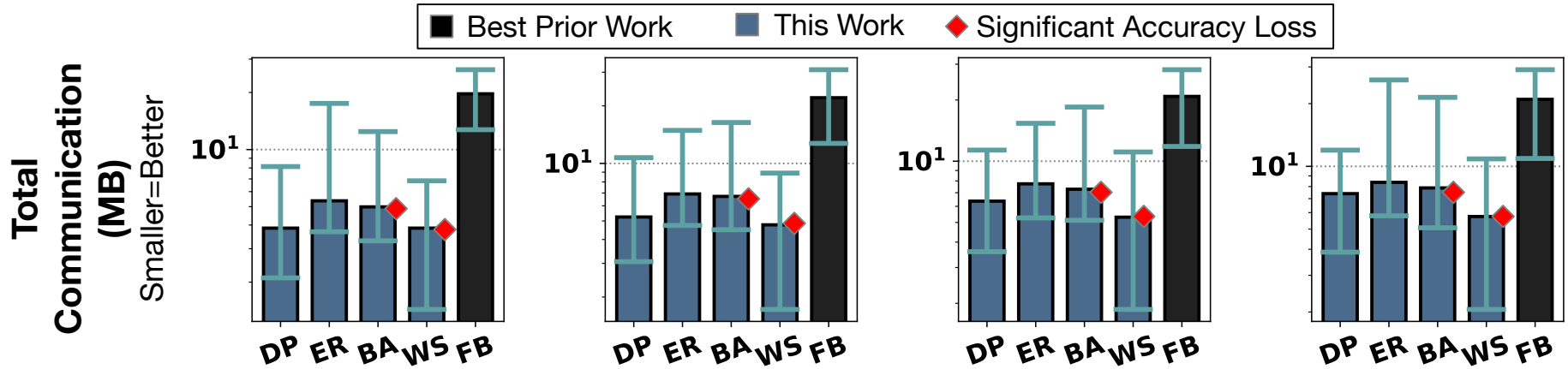
Staging/Samples	A	B	C	Overall Mean
Greedy	2.31	2.27	2.63	2.40
Probabilistic	3.00	3.28	3.58	**3.29**

Width of Concurrent Computations at Same Depth

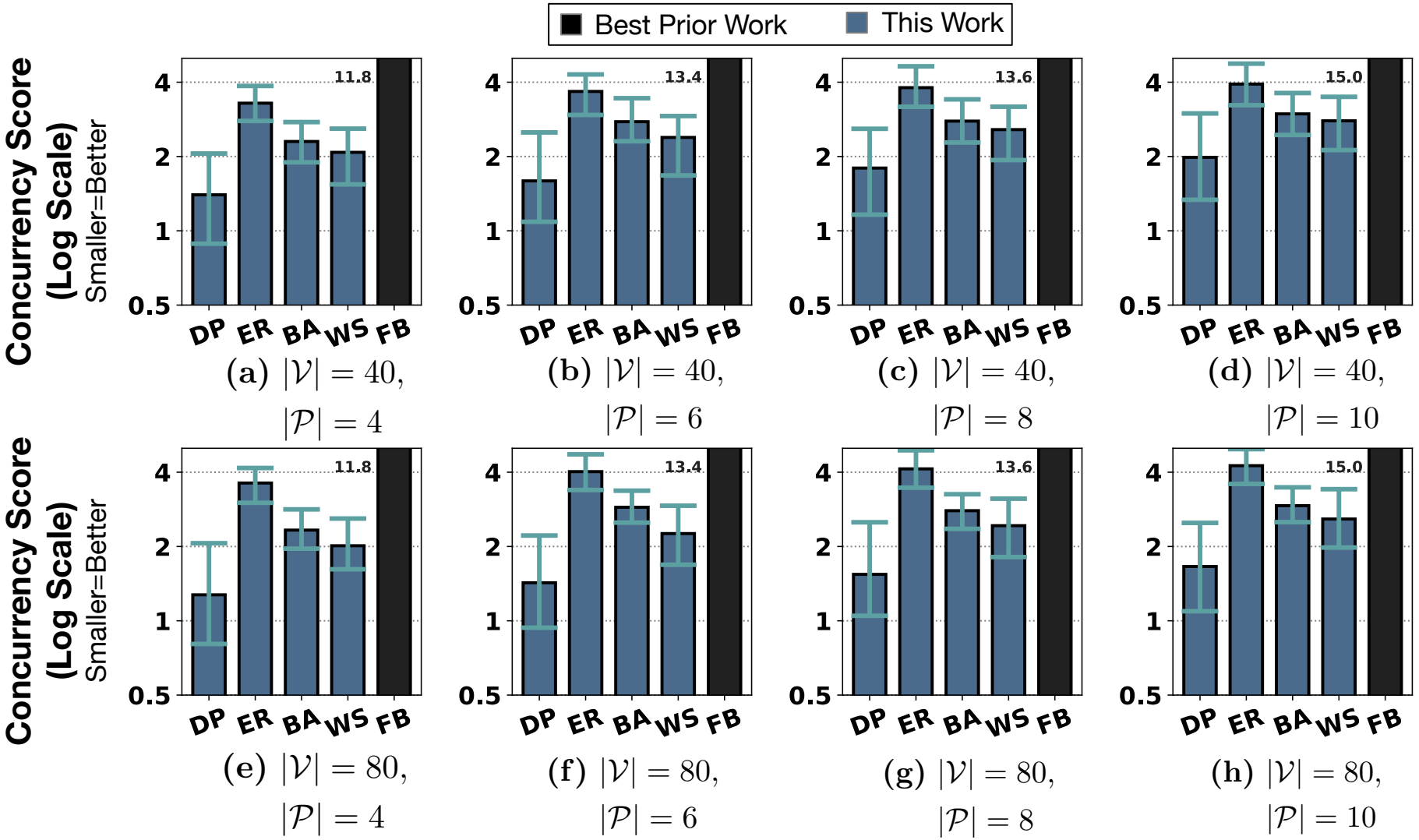




(a) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 4$ (b) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 6$ (c) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 8$ (d) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 10$



(a) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 4$ (b) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 6$ (c) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 8$ (d) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 10$



(a) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 4$ (b) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 6$ (c) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 8$ (d) $|\mathcal{V}| = 40$, $|\mathcal{P}| = 10$

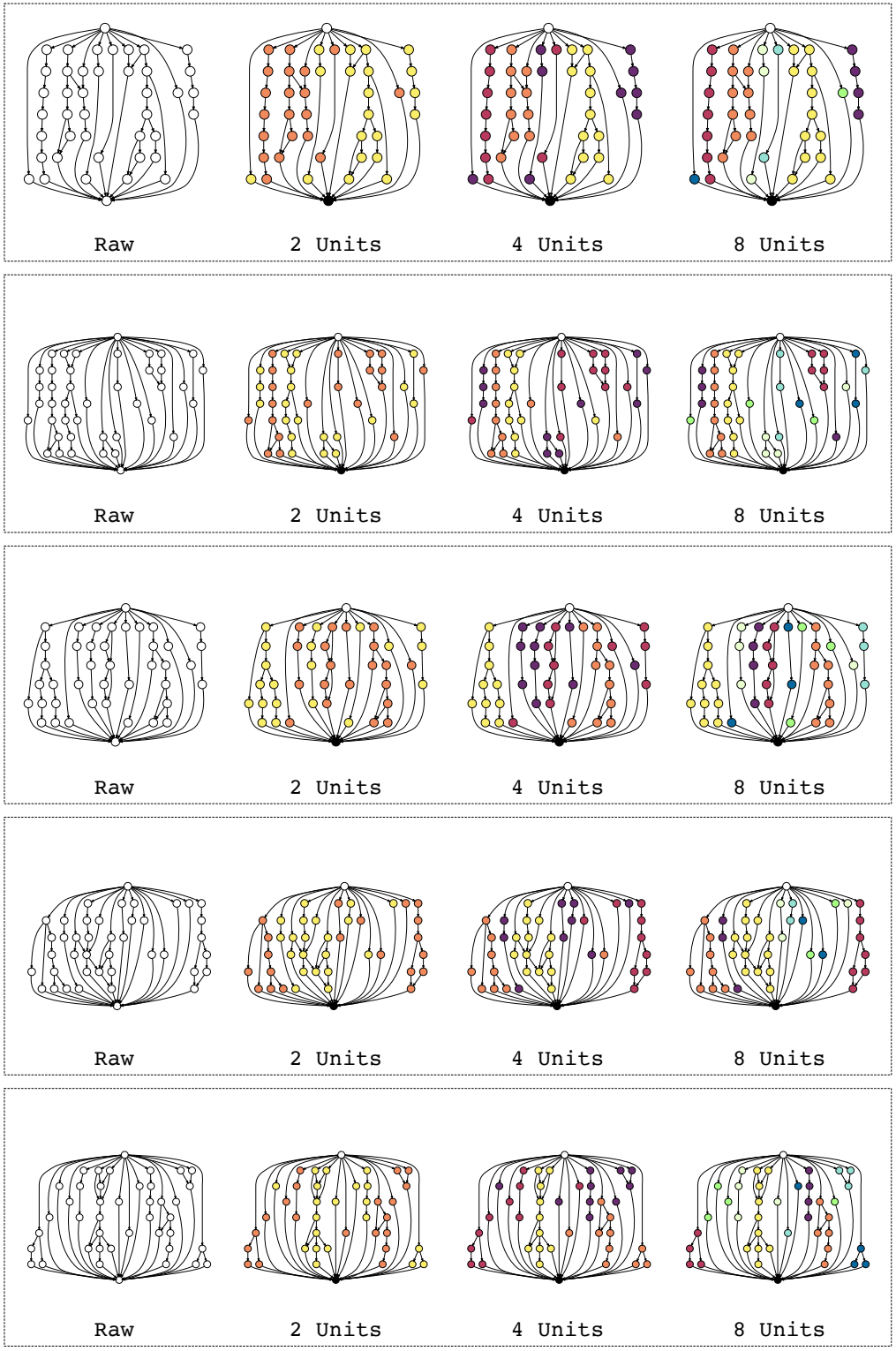
(e) $|\mathcal{V}| = 80$, $|\mathcal{P}| = 4$ (f) $|\mathcal{V}| = 80$, $|\mathcal{P}| = 6$ (g) $|\mathcal{V}| = 80$, $|\mathcal{P}| = 8$ (h) $|\mathcal{V}| = 80$, $|\mathcal{P}| = 10$

TABLE V: **Concurrent Architectures on Cifar-10** – Overall sampled metrics.

	Mean Acc.	**Best Acc.**	**Mean Acc./Param.**	**Best Acc./Param.**
CifarNet	80.70	80.70	5.38	5.38
ER	81.33	81.81	4.94	5.03
BA	80.29	81.66	4.81	4.92
WS	79.89	81.45	4.75	4.84
DP	80.87	82.47	4.81	4.90

TABLE VI: **Concurrent Architects on Flower-102** – Overall sampled metrics.

	Mean Acc.	**Best Acc.**	**Mean Acc./Param.**	**Best Acc./Param.**
ResNet-50	87.80	87.80	3.43	3.43
ER	84.88	86.20	2.11	2.43
BA	82.91	84.62	2.41	2.91
WS	81.46	86.57	3.17	3.10
DP	84.66	86.69	3.19	3.28


Raw
2 Units
4 Units
8 Units
Raw
2 Units
4 Units
8 Units
Raw
2 Units
4 Units
8 Units
Raw
2 Units
4 Units
8 Units
Raw
2 Units
4 Units
8 Units