Paper Review: GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism

Abdullah Mamun

February 1, 2021

About this paper

GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, Zhifeng Chen
{huangyp,ylc,ankurbpn,orhanf,miachen,dehao
hyouklee,jngiam,qvl,yonghui,zhifengc}
@google.com

Abstract

Scaling up deep neural network capacity has been known as an effective approach to improving model quality for several different machine learning tasks. In many cases, increasing model capacity beyond the memory limit of a single accelerator has required developing special algorithms or infrastructure. These solutions

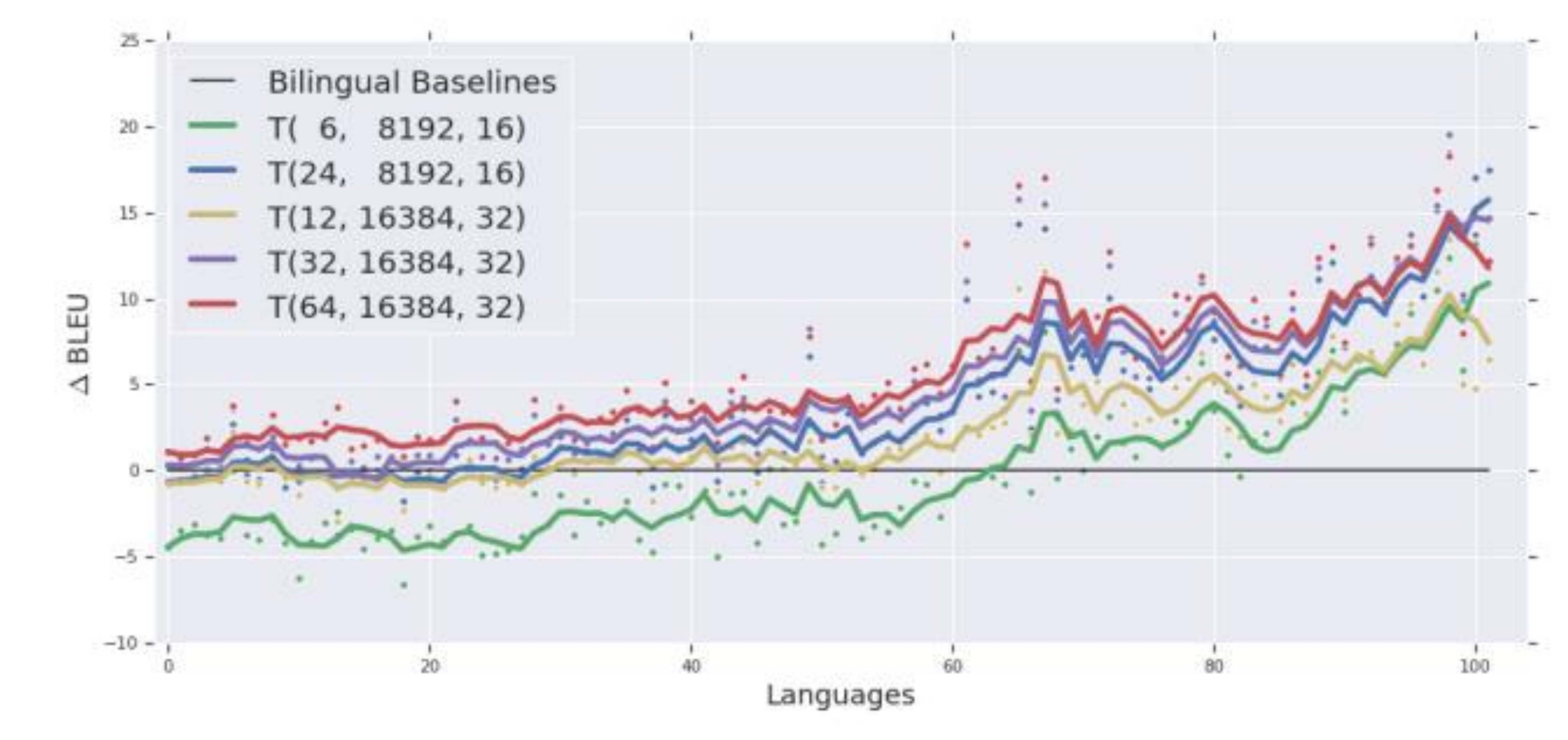
About this paper

- Published by a team of Google in 33^{rd} NeurIPS conference in 2019
- Cited by 340 as of February 1, 2021
- **Let's have a look at their accomplishments at first and then see what is the underlying technique.**

GPipe's accomplishment: Image classification with AmoebaNet

Dataset	# Train	# Test	# Classes	Accuracy (%)	Previous Best (%)
ImageNet-2012	1,281,167	50,000	1000	**84.4**	83.9 [12] (85.4* [27])
CIFAR-10	50,000	10,000	10	**99.0**	98.5 [26]
CIFAR-100	50,000	10,000	100	**91.3**	89.3 [26]
Stanford Cars	8,144	8,041	196	94.6	**94.8*** [26]
Oxford Pets	3,680	3,369	37	**95.9**	93.8* [29]
Food-101	75,750	25,250	101	**93.0**	90.4* [30]
FGVC Aircraft	6,667	3,333	100	92.7	**92.9*** [31]
Birdsnap	47,386	2,443	500	**83.6**	80.2* [32]

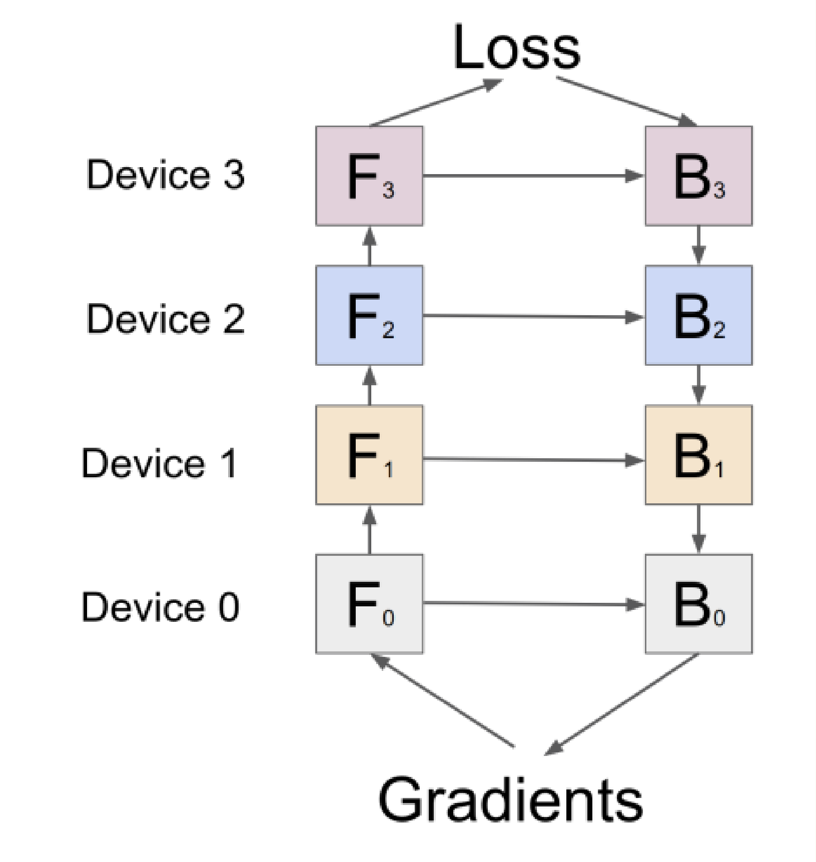
GPipe's accomplishment: Machine translation with Transformer



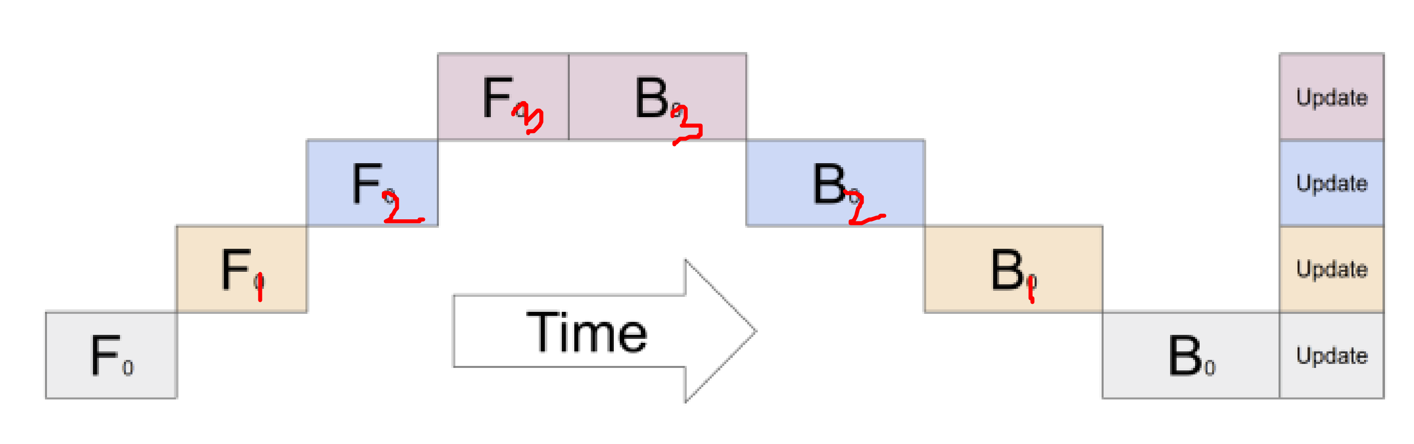
Behind this success

- Pipeline Parallelism: Divide the model layers into different devices and utilize them parallelly.
- Reduce the activation memory requirement so that a Giant Neural Network with billions of parameters can be supported.

(a) Sequential partitioning, (b) Naïve parallelism (severely under-utilized)



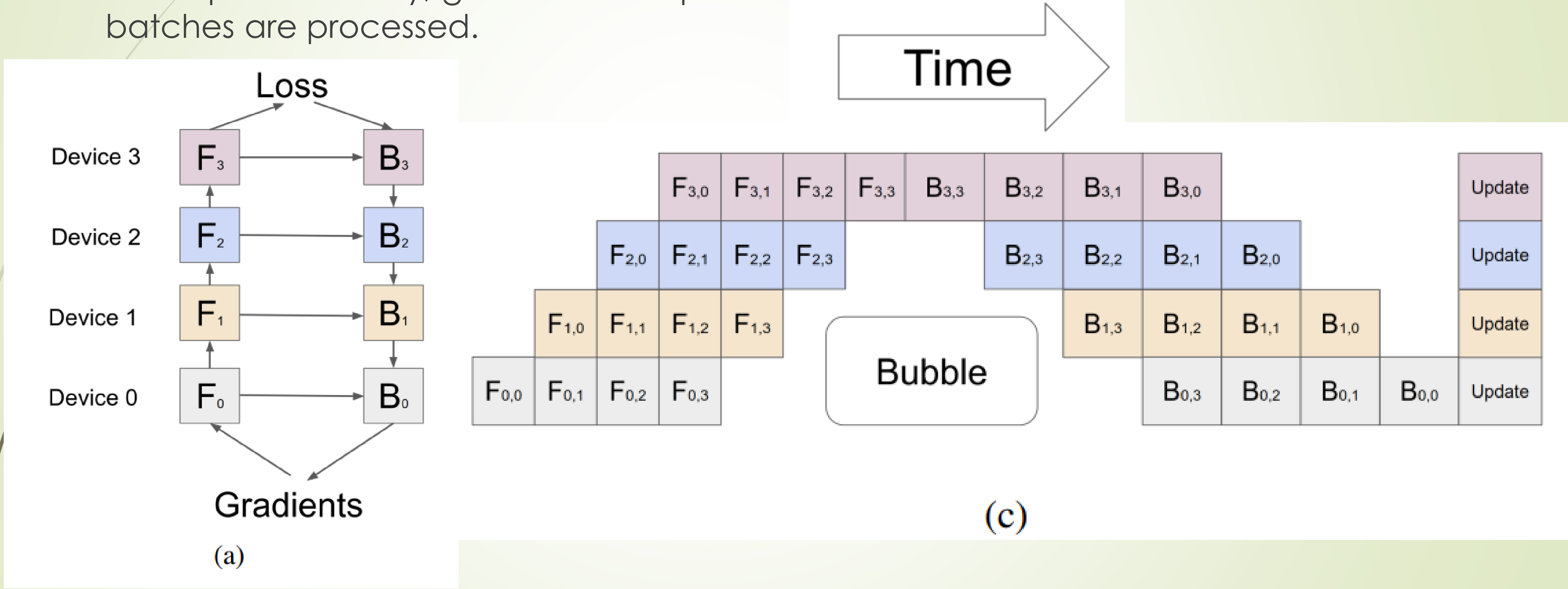
(a)



(b)

Enhancing throughput with pipeline parallelism

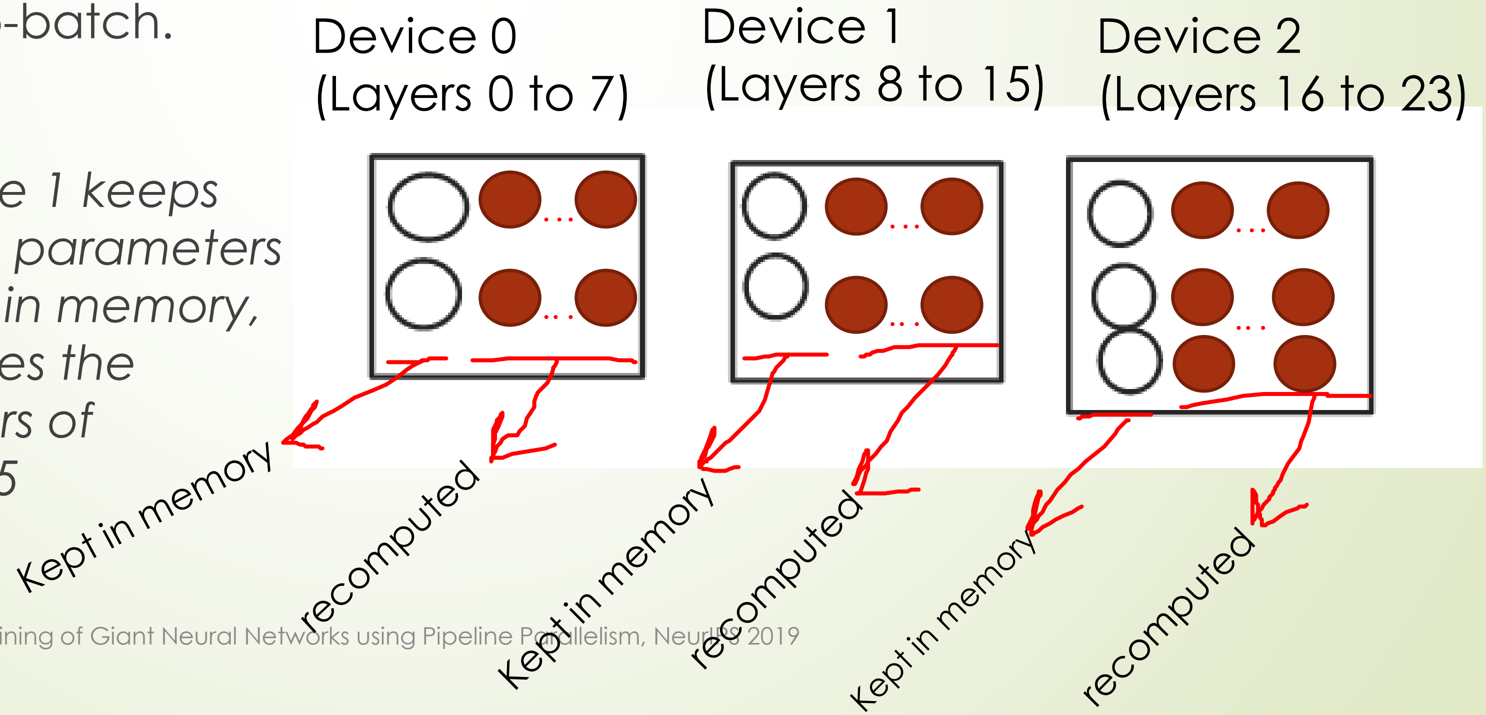
- A minibatch of size N will be divided into **M micro-batches** of size N/M each.
- To keep consistency, gradients are updated at the end after all micro-batches are processed.



(a) (c)

Reducing activation memory requirement

- Instead of keeping information of all layers' activations in the memory for the whole time, **a partition keeps only its border layer's activations** and **recomputes the others** during the backpropagation.
- Instead of keeping activations for a mini-batch, we are keeping activations of a micro-batch.
- *E.g. Device 1 keeps activation parameters of Layer 8 in memory, recomputes the parameters of Layers 9-15*



Maximum model size supported (Naive vs GPipe)

- **Activation** Memory requirement: O(N + L/K * N/M) per partition
- Without partition and micro-batch, it would be O(N * L), i.e. size of minibatch x #layers
- *Note that model parameter memory is not reducing.
- N = size of the minibatch, L= #layers, K= #partitions, M= #microbatches

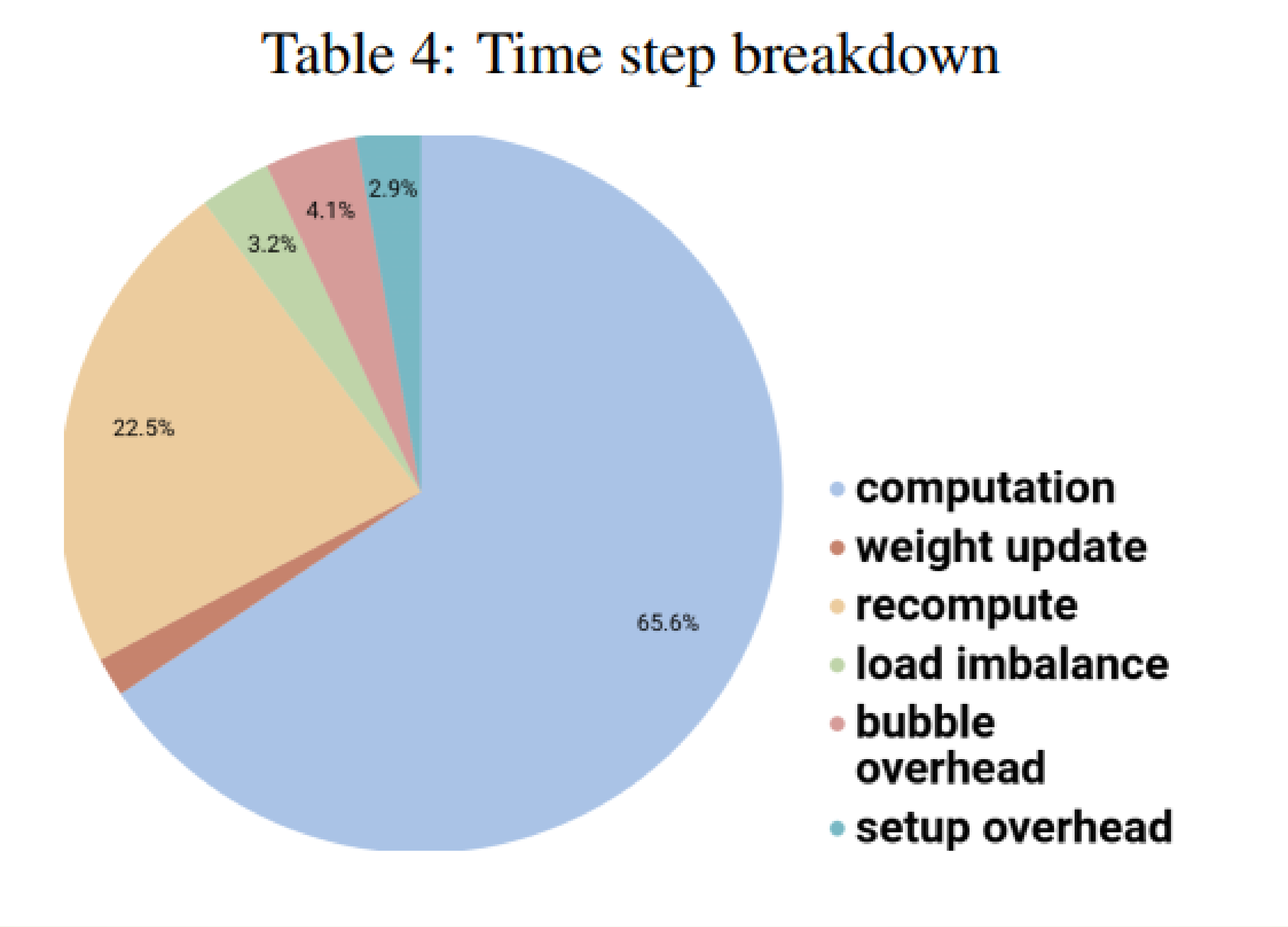
NVIDIA GPUs (8GB each)	Naive-1	Pipeline-1	Pipeline-2	Pipeline-4	Pipeline-8
AmoebaNet-D (L, D)	(18, 208)	(18, 416)	(18, 544)	(36, 544)	(72, 512)
# of Model Parameters	82M	318M	542M	1.05B	1.8B
Total Model Parameter Memory	1.05GB	3.8GB	6.45GB	12.53GB	24.62GB
Peak Activation Memory	6.26GB	3.46GB	8.11GB	15.21GB	26.24GB
Cloud TPUv3 (16GB each)	**Naive-1**	**Pipeline-1**	**Pipeline-8**	**Pipeline-32**	**Pipeline-128**
Transformer-L	3	13	103	415	1663
# of Model Parameters	282.2M	785.8M	5.3B	21.0B	83.9B
Total Model Parameter Memory	11.7G	8.8G	59.5G	235.1G	937.9G
Peak Activation Memory	3.15G	6.4G	50.9G	199.9G	796.1G

Improving the training throughput

- In the Transformer model, for M=32 (# micro-batch per minibatch), **the speed up from k=2 to k=8 is 3.5x.** The improvement is almost linear. (k= #partitions)
- In AmoebaNet, the improvement is sublinear due to imbalanced computation distribution.
- * Training throughputs were **not** evaluated on Giant networks

TPU	AmoebaNet			Transformer		
$K =$	2	4	8	2	4	8
$M = 1$	1	1.13	1.38	1	1.07	1.3
$M = 4$	1.07	1.26	1.72	1.7	3.2	4.8
$M = 32$	1.21	1.84	3.48	1.8	3.4	6.3

There are some overheads



Summary

- Training time is reduced by **parallelism and pipelining** to minimize bubble time.
- **Activation memory requirement** is reduced by micro-batching and re-computing the forward activations.
- GPipe with 8 partitions of Nvidia 8GB GPU each, can support a AmoebaNet-D of up to **1.5B parameters.**
- **Training throughput** can be increased by tuning #partitions and #micro-batches.
- While powerful and giant models can be supported, the **overheads** are not negligible.

Thank You!

Contact: abdullahal.mamun1@wsu.edu

Read the paper [here]