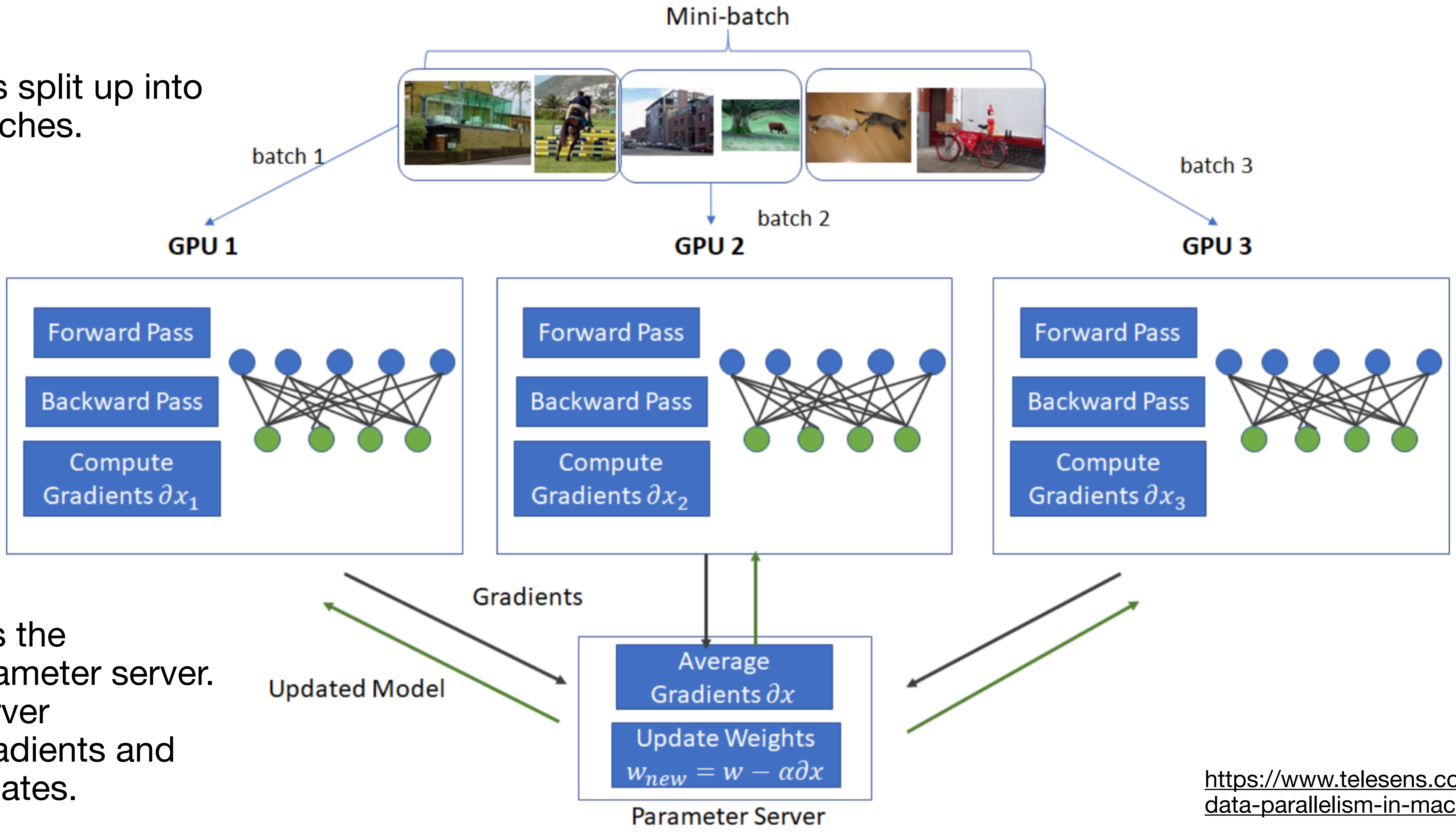
Pipeline Input and Layer Split

Table of Contents

Recap

Data Parallelism

1. A mini-batch is split up into smaller sized batches.



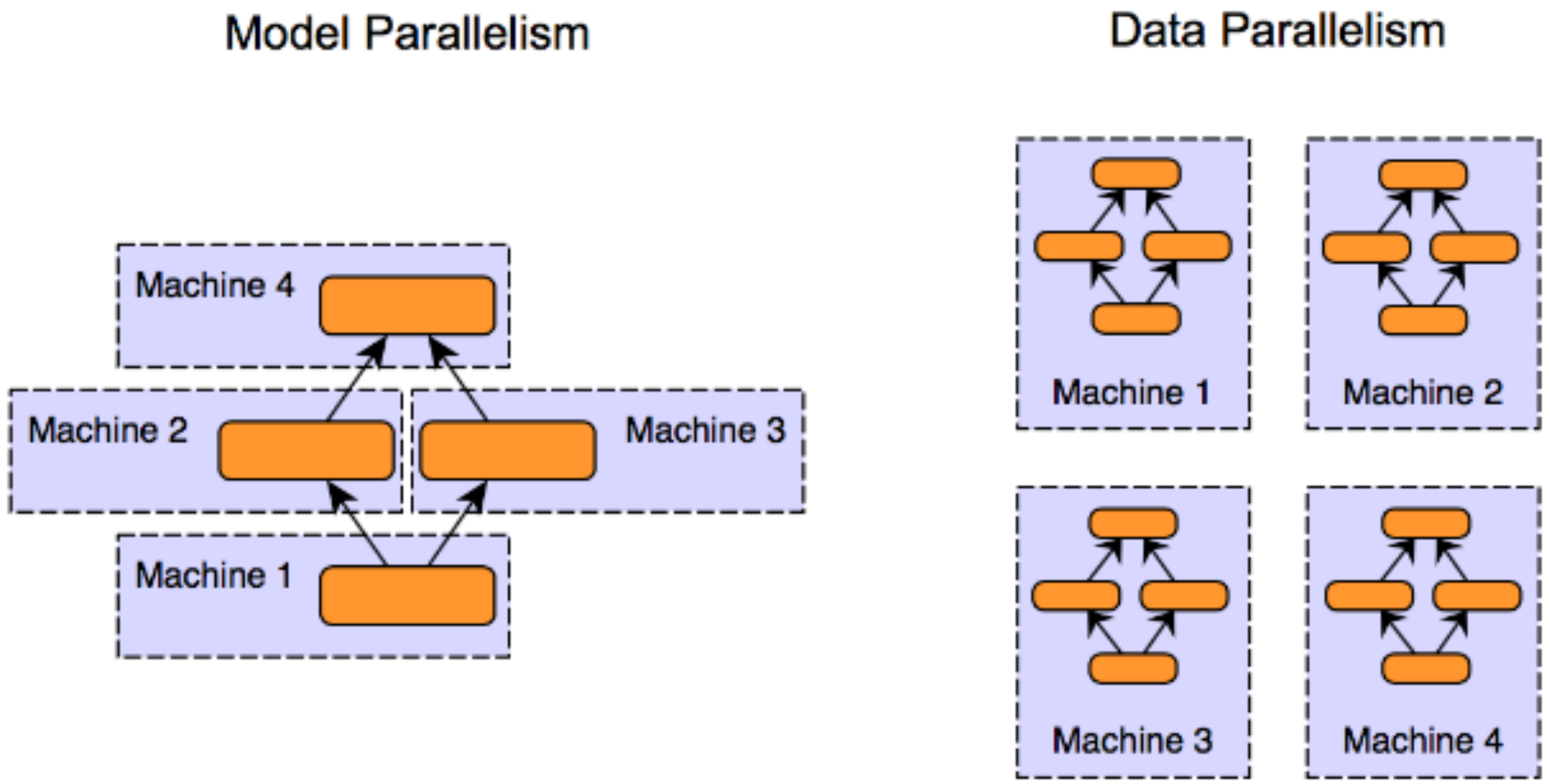
2. Each gpu holds an identical copy of the network parameters and runs the forward and backward pass.

3. Each gpu sends the gradients to a parameter server. The parameter server aggregates the gradients and computes the updates.

https://www.telesens.co/2017/12/25/understanding-data-parallelism-in-machine-learning/

Overview

Model Parallelism vs Data Parallelism



https://xiandong79.github.io/Intro-Distributed-Deep-Learning

- Model Parallelism
  - 각각의 machine이 모델 전체가 아닌 일부만을 담당
- Data Parallelism
  - 각각의 machine이 전체 모델에 대한 copy를 가지고 연산

Vanilla Model Parallelism

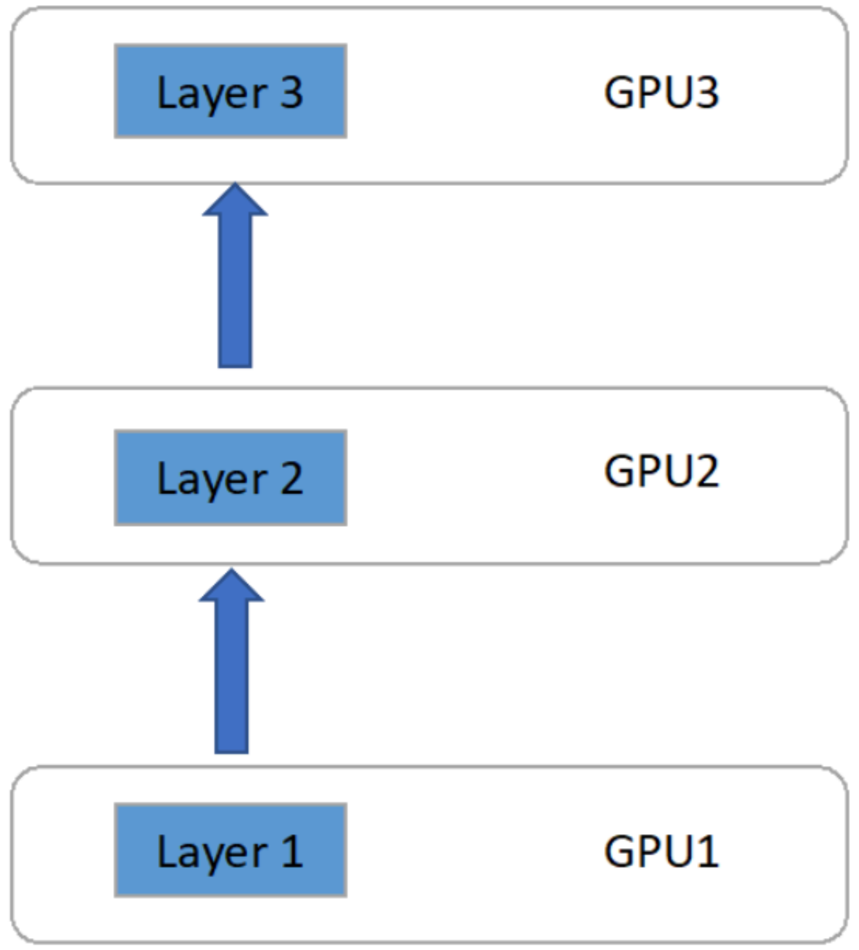
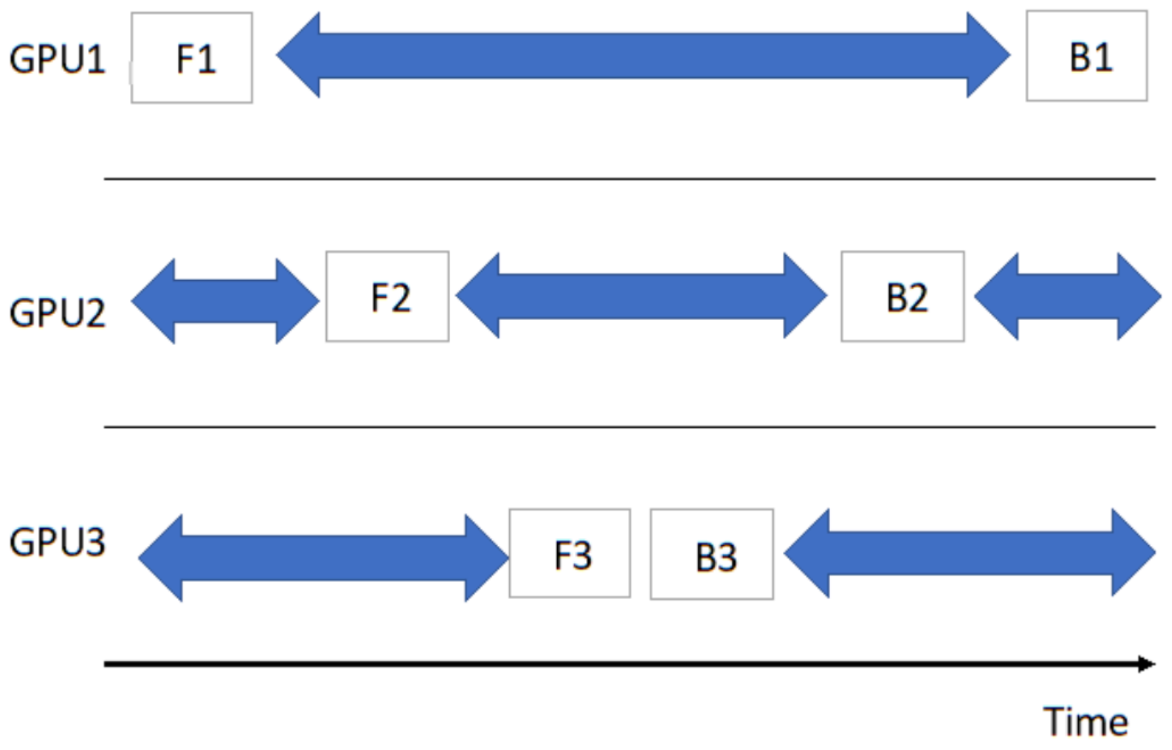


Figure 6.2 – Model partition on three GPUs

GPU's idle time
in blue arrows

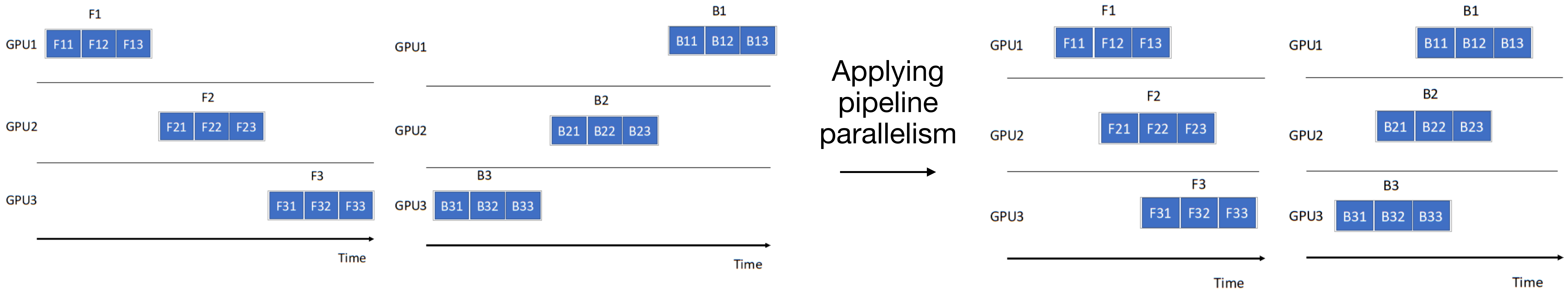


GPU utilization is only around 33%

- Sequential layer dependency -> inefficiency (low GPU utilization rate)
- System inefficiency gets worse as more GPUs are involved
  - $GPUutil = \frac{GPUwork}{total\ time} = \frac{2}{2*N} = \frac{1}{N}$
  - GPUwork: each GPU works for one forward and one backward computation
  - Total time: GPU work * total number of GPUs

Pipeline Paralleism

- Pipeline Parallelism breaks each batch of training input into smaller micro-batches and conducts data pipelining



- Left: takes 9 time slots for forward and backward pass
- Right: takes 5 time slots for forward and backward pass

Pipeline Paralleism

Pros and Cons

- Advantages
  - Reduces overall training time and GPU idle time
  - Easier implementation
  - Can be adapted to any kind of DNN model
- Disadvantages
  - GPU needs to send more instructions to GPUs
  - GPU idle time still exists
  - Introduces more frequent GPU communications (networking communication overhead)

Intra-layer Parallelism

$$y = X * A$$

Input Matrix (batch size = 4)

x(0,0)	x(0,1)	w(0,2)	w(0,3)
x(1,0)	x(1,1)	x(1,2)	x(1,3)
x(2,0)	x(2,1)	x(2,2)	x(2,3)
x(3,0)	x(3,1)	x(3,2)	x(3,3)

Layer 1 Matrix

w(0,0)	w(1,0)	w(2,0)	w(3,0)
w(0,1)	w(1,1)	w(2,1)	w(3,1)
w(0,2)	w(1,2)	w(2,2)	w(3,2)
w(0,3)	w(1,3)	w(2,3)	w(3,3)

$$y_{01}, y_{23} = [X * A_{01}, X * A_{23}]$$

Input Matrix (batch size = 4)

x(0,0)	x(0,1)	w(0,2)	w(0,3)
x(1,0)	x(1,1)	x(1,2)	x(1,3)
x(2,0)	x(2,1)	x(2,2)	x(2,3)
x(3,0)	x(3,1)	x(3,2)	x(3,3)

Layer 1 Matrix Splits (Column-wise)

w(0,0)	w(1,0)
w(0,1)	w(1,1)
w(0,2)	w(1,2)
w(0,3)	w(1,3)

A_01

w(2,0)	w(3,0)
w(2,1)	w(3,1)
w(2,2)	w(3,2)
w(2,3)	w(3,3)

A_23

- Intra-layer parallelism achieves model parallelism without communication among the model partitions on each GPU
- For one split, it only introduce one All-Reduce in either forward or backward pass
- Mostly applicable to NLP (MLP layers)

Model Parallelism in Megatron

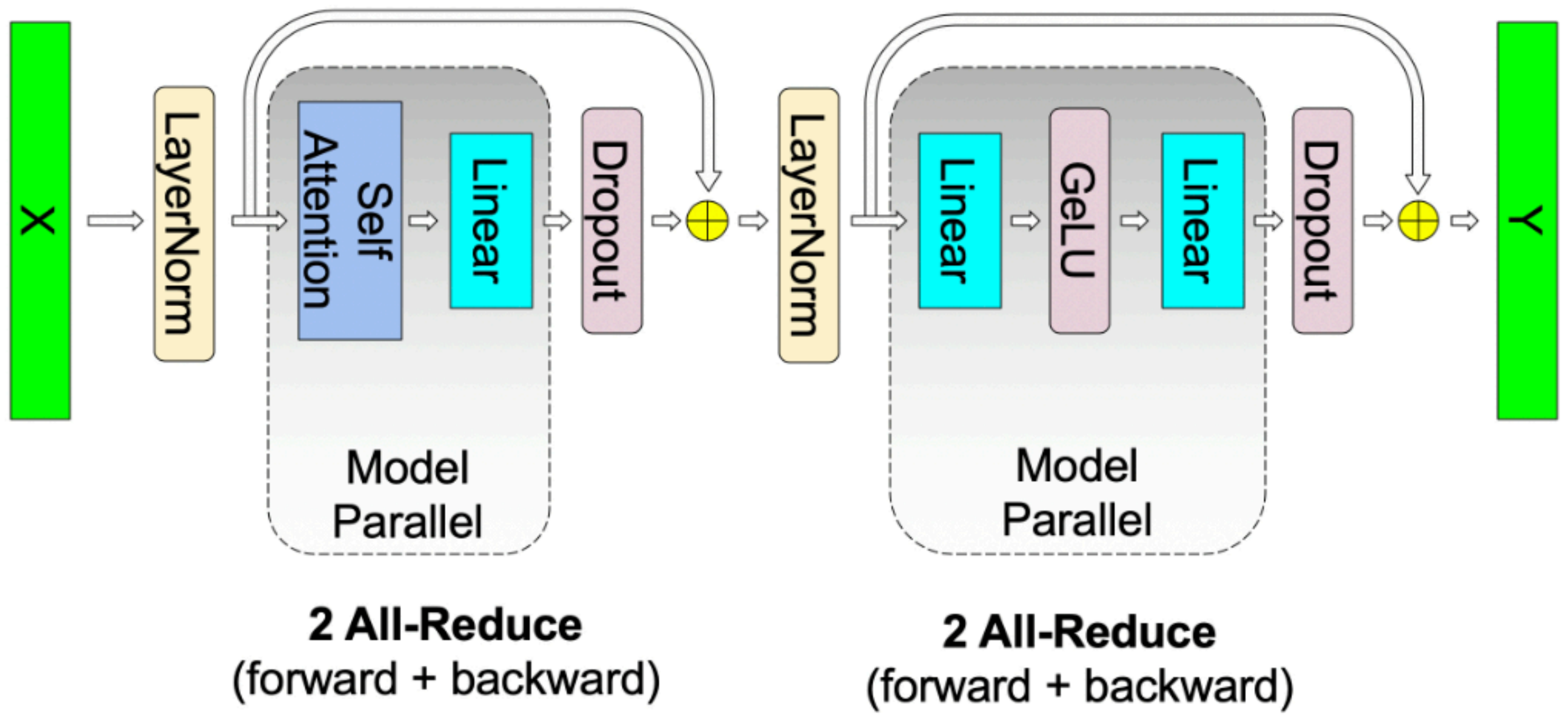


Figure 3: Model parallelism for a GPT-2 transformer layer.

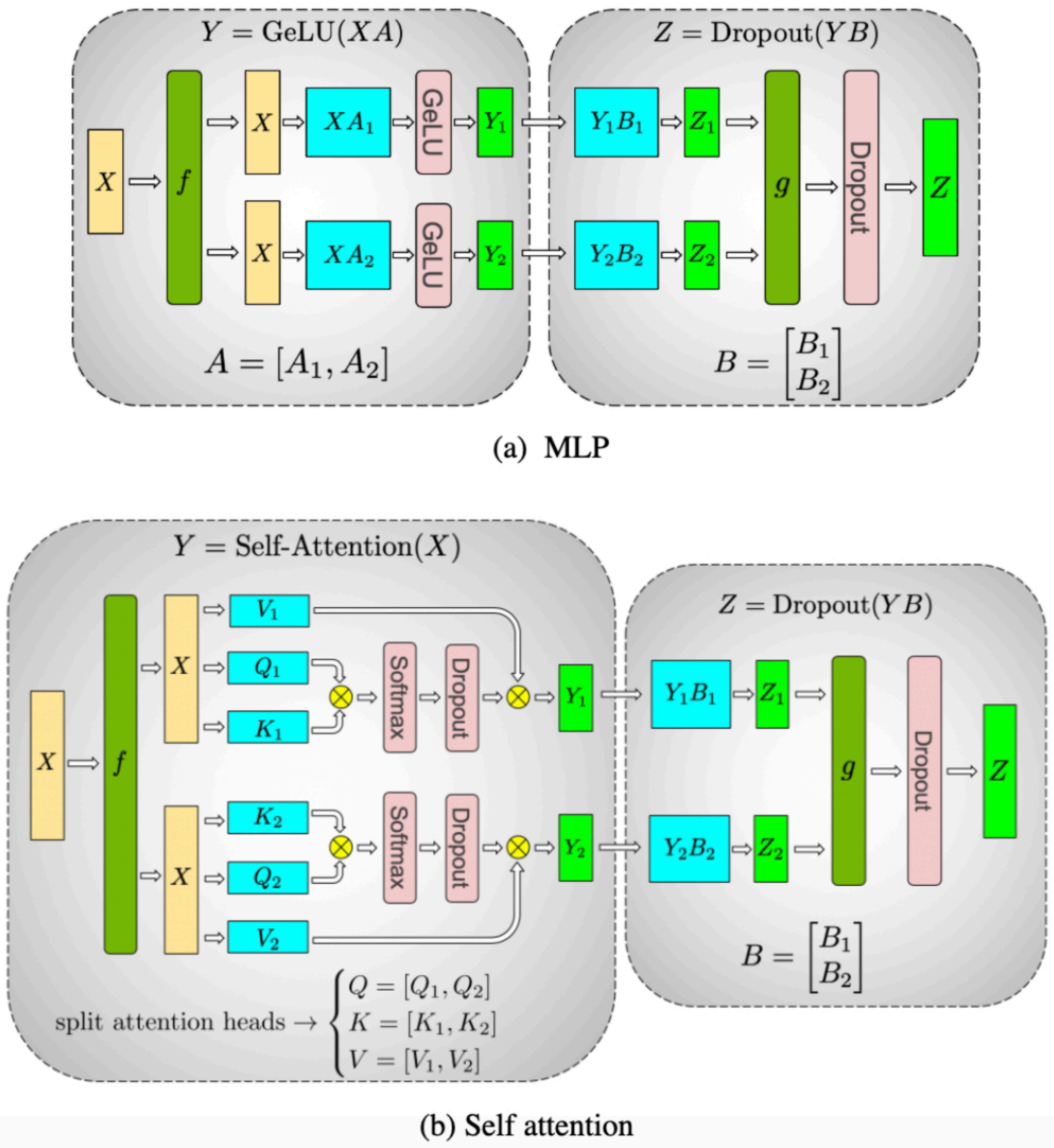


Figure 2: (a): MLP and (b): self attention blocks of transformer. ***f*** and ***g*** are conjugate, ***f*** is an **identity** operator in the forward pass and **all-reduce** in the backward pass while ***g*** is an **all-reduce** in forward and ***identity*** in backward.