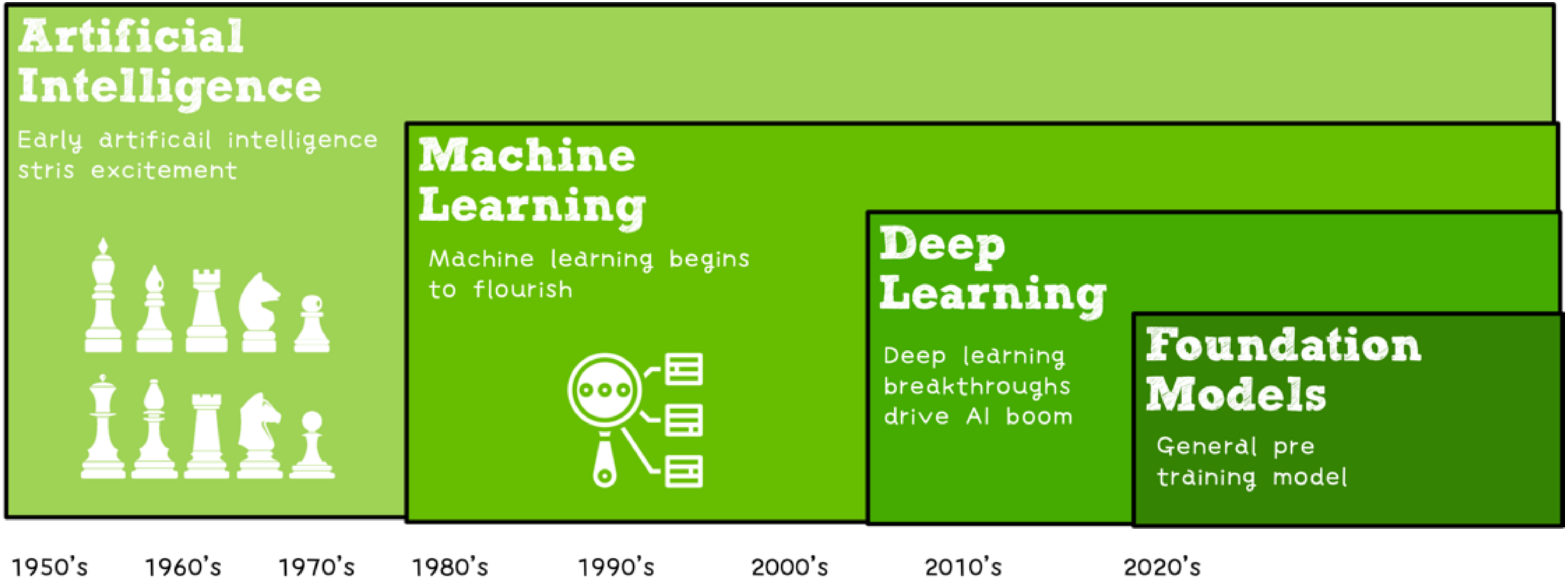
分布式训练系列

张量并行

ZOMI


Artificial Intelligence
Early artificail intelligence stris excitement
Machine Learning
Machine learning begins to flourish
Deep Learning
Deep learning breakthroughs drive AI boom
Foundation Models
General pre training model
1950's
1960's
1970's
1980's
1990's
2000's
2010's
2020's

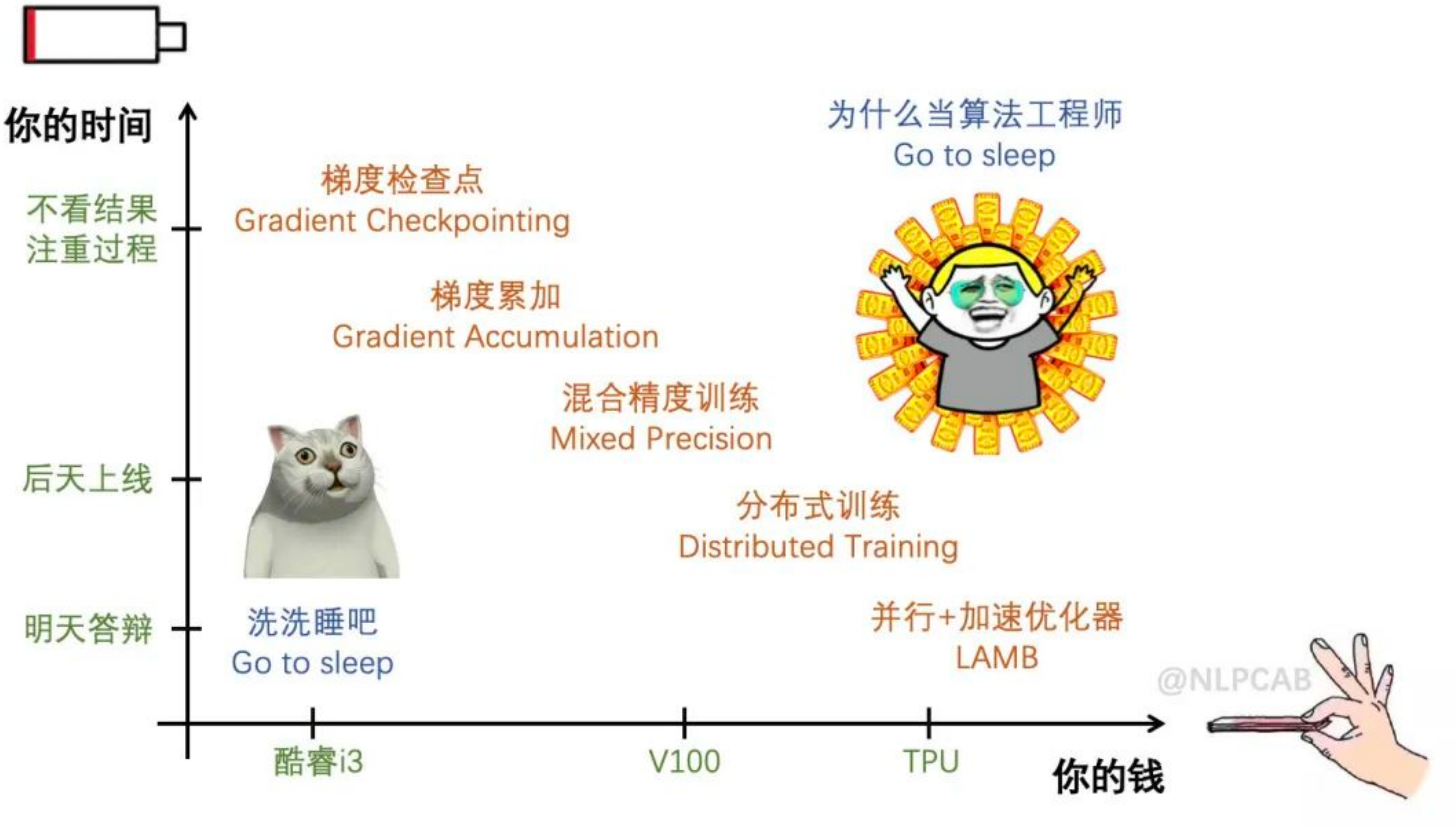
关于本内容

1. 内容背景

- 大规模分布式训练系统：串行到并行 – 并行处理体系 – 深度学习并行训练

2. 具体内容

- **大模型训练的挑战**：内存墙 – 性能墙 – 效率墙 – 调优墙
- **分布式训练系统**：并行处理硬件架构 – 业界分布式系统分析
- **分布式并行总体架构**：参数服务器模式 – 集合通讯模式
- **通信原语与协调**：通讯协调软硬件 - 通信实现方式 - 通信原语
- **大模型算法结构**：大模型算法发展 – NLP大模型 - CV大模型 – 多模态大模型
- **分布式并行**：数据并行 – 模型并行 – 流水并行 – 混合并行
- **内存和计算优化**：ZeRO内存优化 – 混合精度与计算优化


你的时间
不看结果
注重过程
后天上线
明天答辩
梯度检查点
Gradient Checkpointing
梯度累加
Gradient Accumulation
混合精度训练
Mixed Precision
分布式训练
Distributed Training
并行+加速优化器
LAMB
为什么当算法工程师
Go to sleep
洗洗睡吧
Go to sleep
@NLPCAB
酷睿i3
V100
TPU
你的钱

Model Parallelism, MP 模型并行

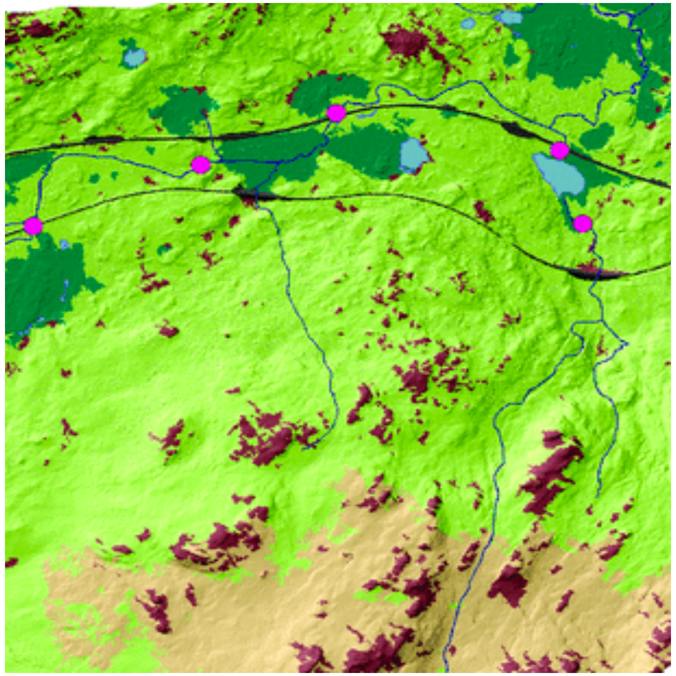
- Tensor Parallelism 张量并行
  - Principles 并行原理
  - Matmul 算子并行
  - Loss 损失并行
  - Transformer 算子并行
  - Tensor Redistribution 张量重排（MindSpore）
  - Stochastic Control 随机控制
- Pipeline Parallelism 流水线并行

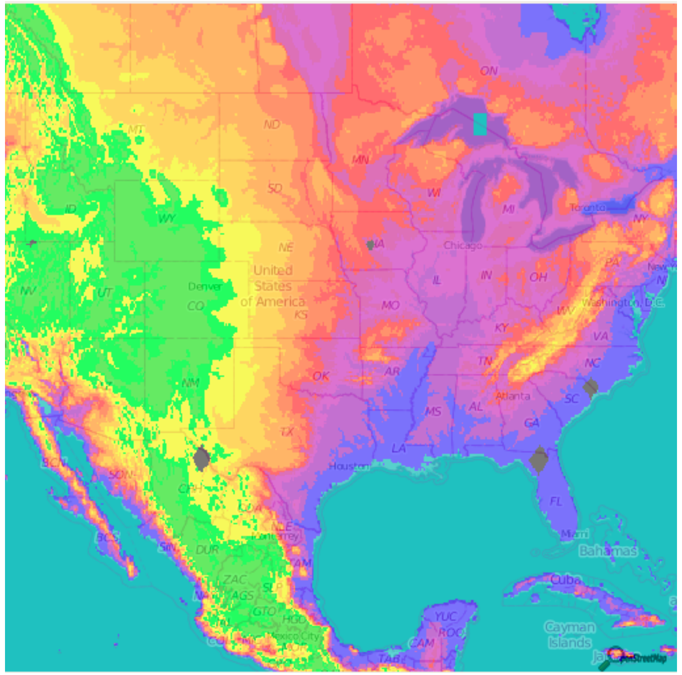
Data parallelism 数据并行

1. Data parallelism, DP
2. Distribution Data Parallel, DDP
3. Fully Sharded Data Parallel, FSDP

256 x 256 x 3

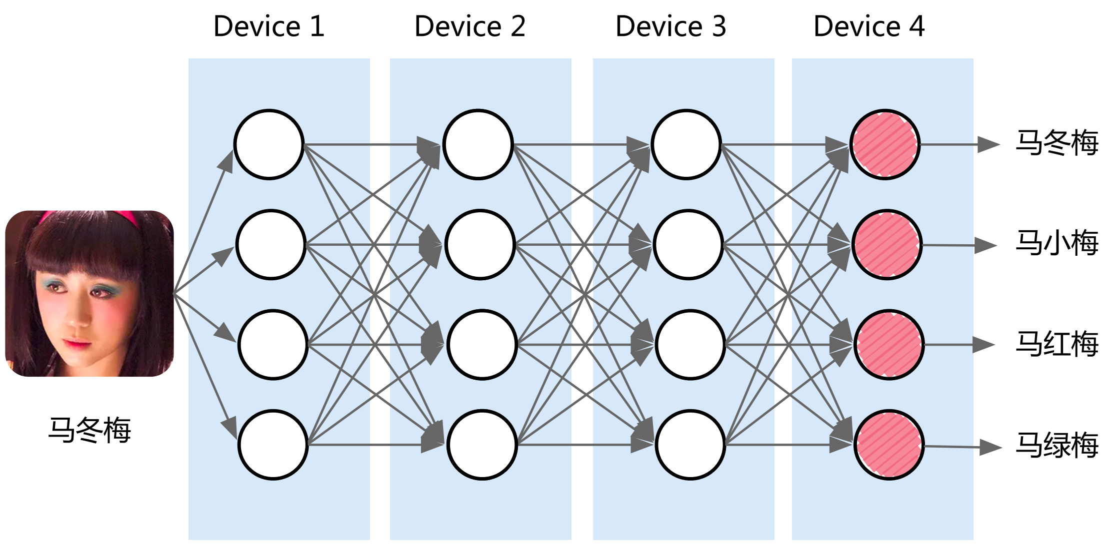
24599 x 35688 x 256



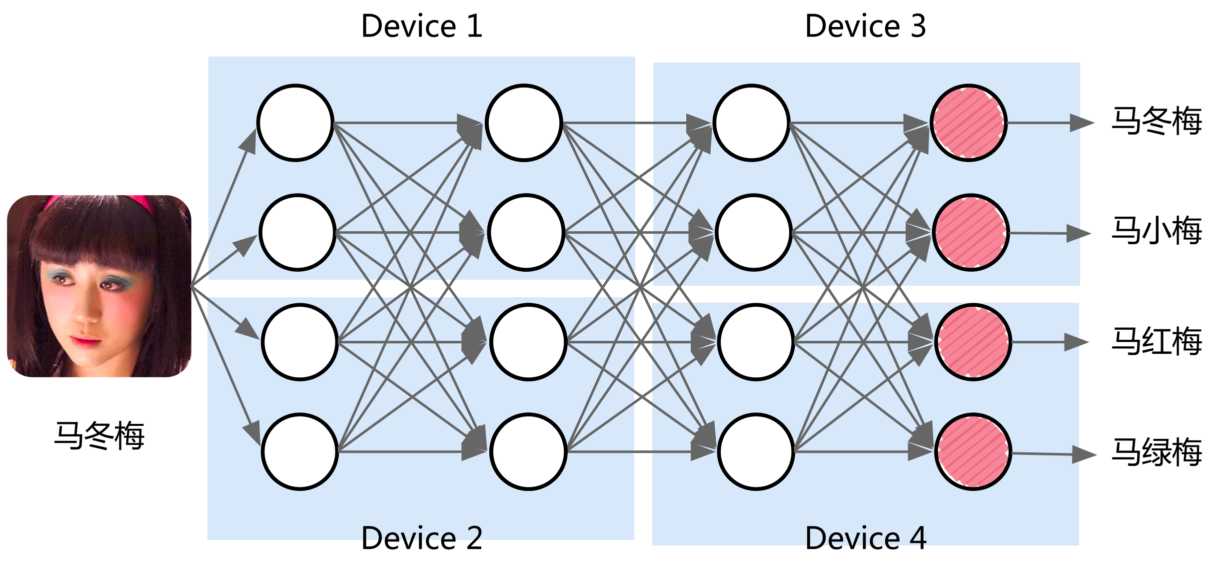


MP(I): Pipeline parallelism 流水线并行

- Model divided layers into different devices, which we called pipeline parallelism
- 流水线并行：按模型layer层切分到不同设备，即层间并行



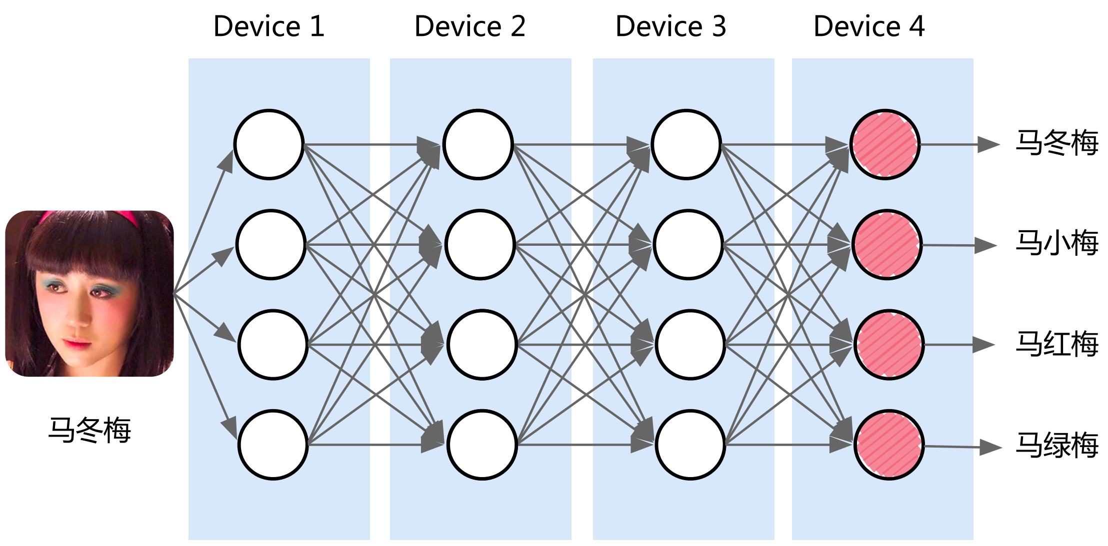
流水线并行（层间并行）



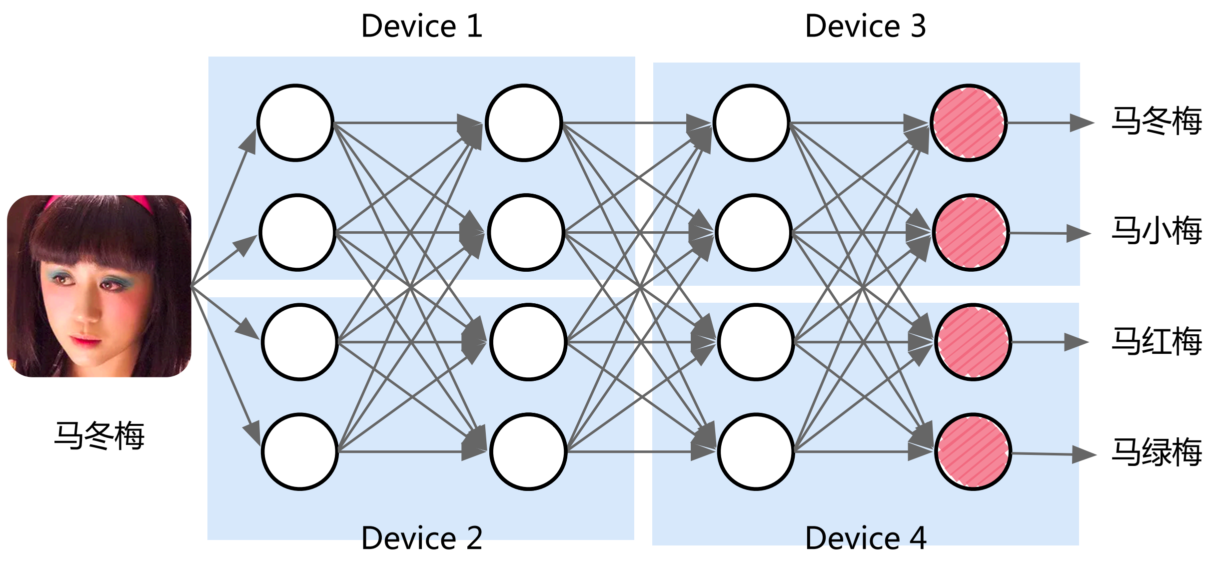
张量并行（层内并行）

MP(II): Tensor parallelism 张量并行

- Divide parameters in the layer into different devices, which we called tensor model parallelism
- 张量并行：将计算图中的层内的参数切分到不同设备，即层内并行



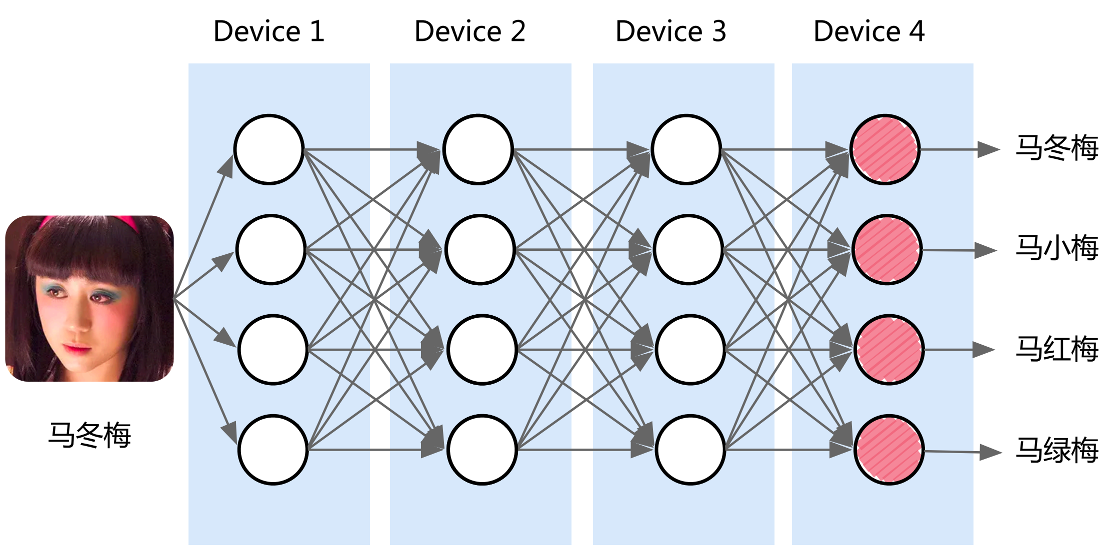
流水线并行（层间并行）



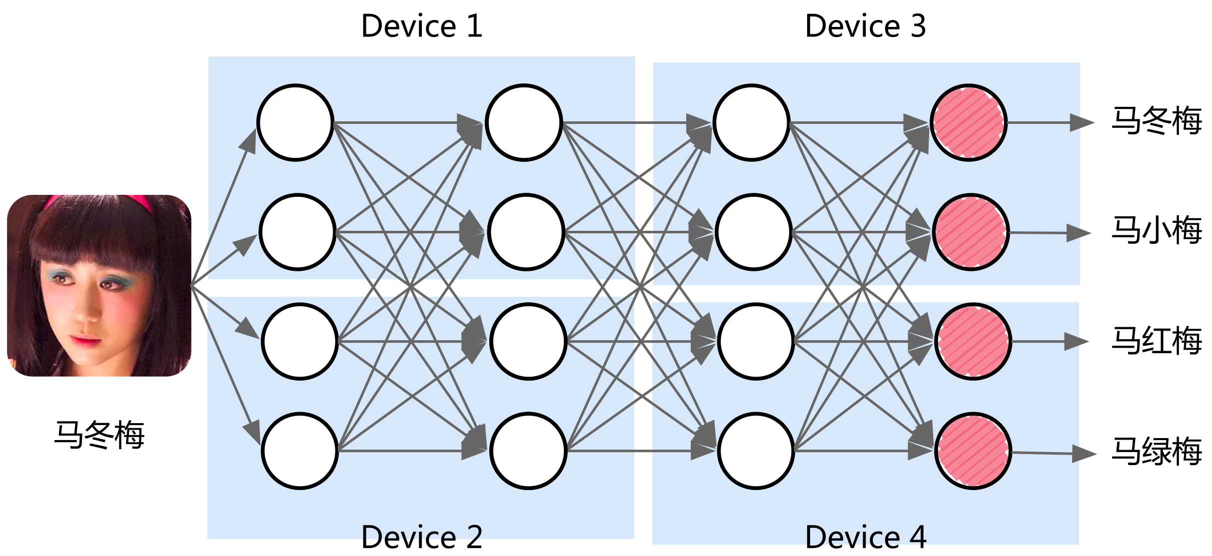
张量并行（层内并行）

MP(II): Tensor parallelism 张量并行

- Divide parameters in the layer into different devices, which we called tensor model parallelism.
  - How to segment the parameters to different devices, i.e., the segmentation mode? 如何切分？
  - How to ensure mathematical consistency after segmentation? 如何保证正确性？



流水线并行（层间并行）



张量并行（层内并行）

Mathematical Principles 数学原理

- 张量切分方式

$$[X] = \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}$$

Device1

$$\begin{bmatrix} X_{10} & X_{11} \end{bmatrix}$$

Device2

$$\begin{bmatrix} X_{00} & X_{01} \end{bmatrix}$$

行切分

$$[X] = \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}$$

Device1

$$\begin{bmatrix} X_{00} \\ X_{10} \end{bmatrix}$$

Device2

$$\begin{bmatrix} X_{01} \\ X_{11} \end{bmatrix}$$

列切分

$$[X] = \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}$$

Device1

$$\begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}$$

Device2

$$\begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}$$

复制

Mathematical Principles 数学原理

- 利用分块矩阵计算法则，将 A 分别做矩阵按列切分和按行切分

$$XA = Y$$

A 列切分 $$X \times [A_1 \quad A_2] = [XA_1 \quad XA_2] = Y$$

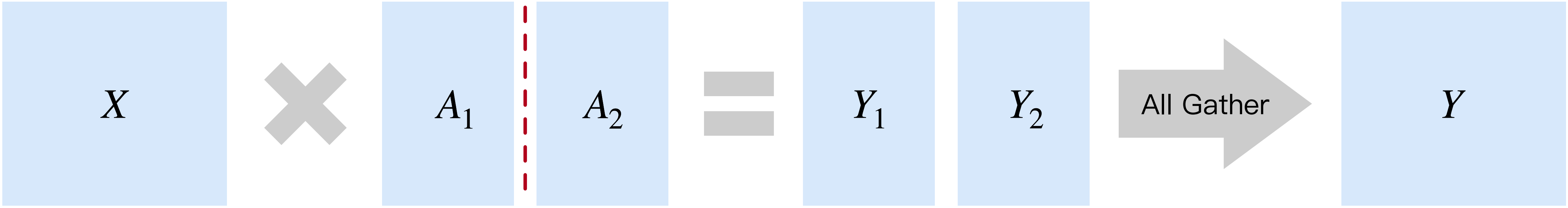
A 行切分 $$[X_1 \quad X_2] \times \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = [X_1A_1 + X_2A_2] = Y$$

MatMul 矩阵乘算子并行

- X作为激活输入， A作为算子权重，将 A 分按列切分

$$XA = Y$$

$$X \times [A_1 \quad A_2] = [XA_1 \quad XA_2] = Y$$



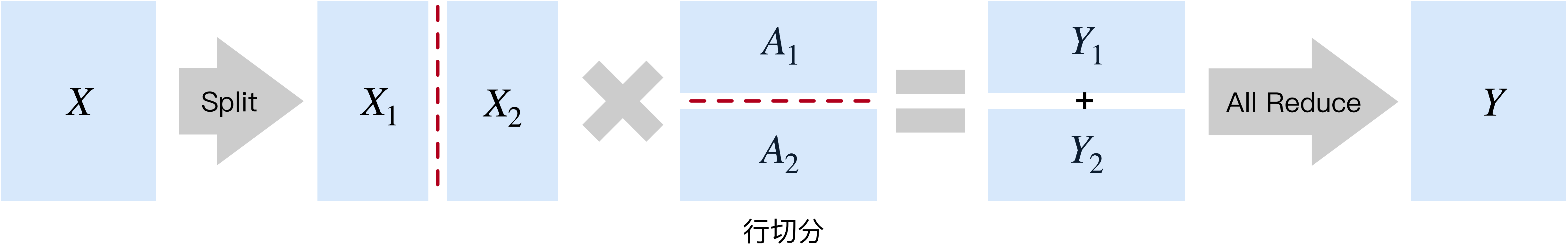
列切分

MatMul 矩阵乘算子并行

- X作为激活输入， A作为算子权重，将 A 分按行切分

$$XA = Y$$

$$[X_1 \quad X_2] \times \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = [X_1A_1 + X_2A_2] = Y$$



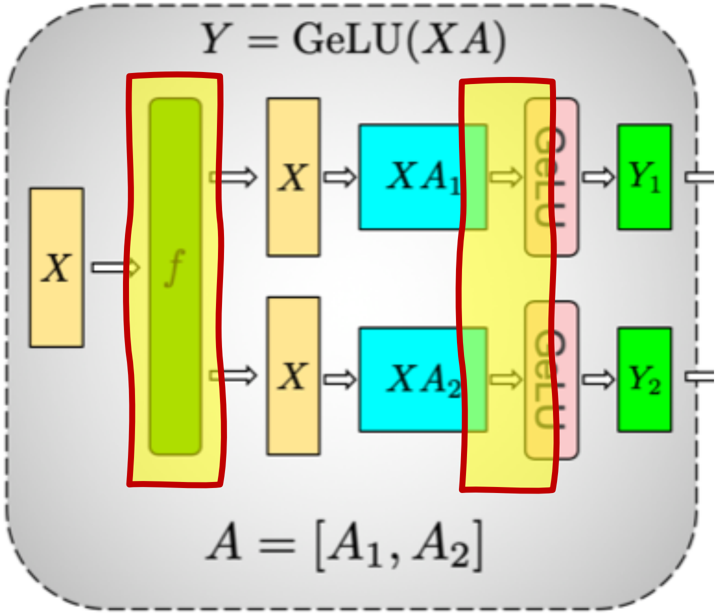
Transformer: MLP

$$Y = \mathrm{GeLU}(XA)$$

option I

$$\mathrm{X} = [X_1, X_2], A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$Y = \mathrm{GeLU}(X_1 A_1 + X_2 A_2)$$



Split

All Reduce

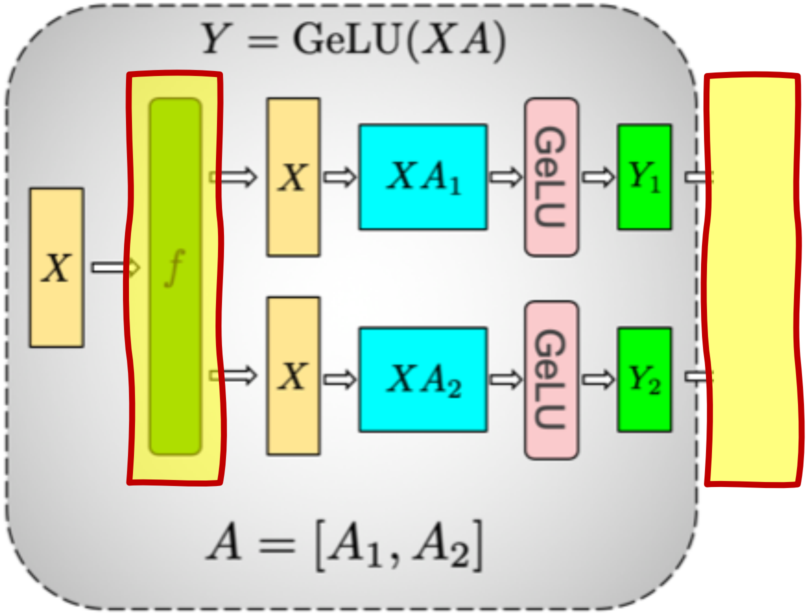
Transformer: MLP

$$Y = \mathrm{GeLU}(XA)$$

option 2

$$A = [A_1 \quad A_2]$$

$$[Y_1, Y_2] = [\mathrm{GeLU}(XA_1), \mathrm{GeLU}(XA_2)]$$

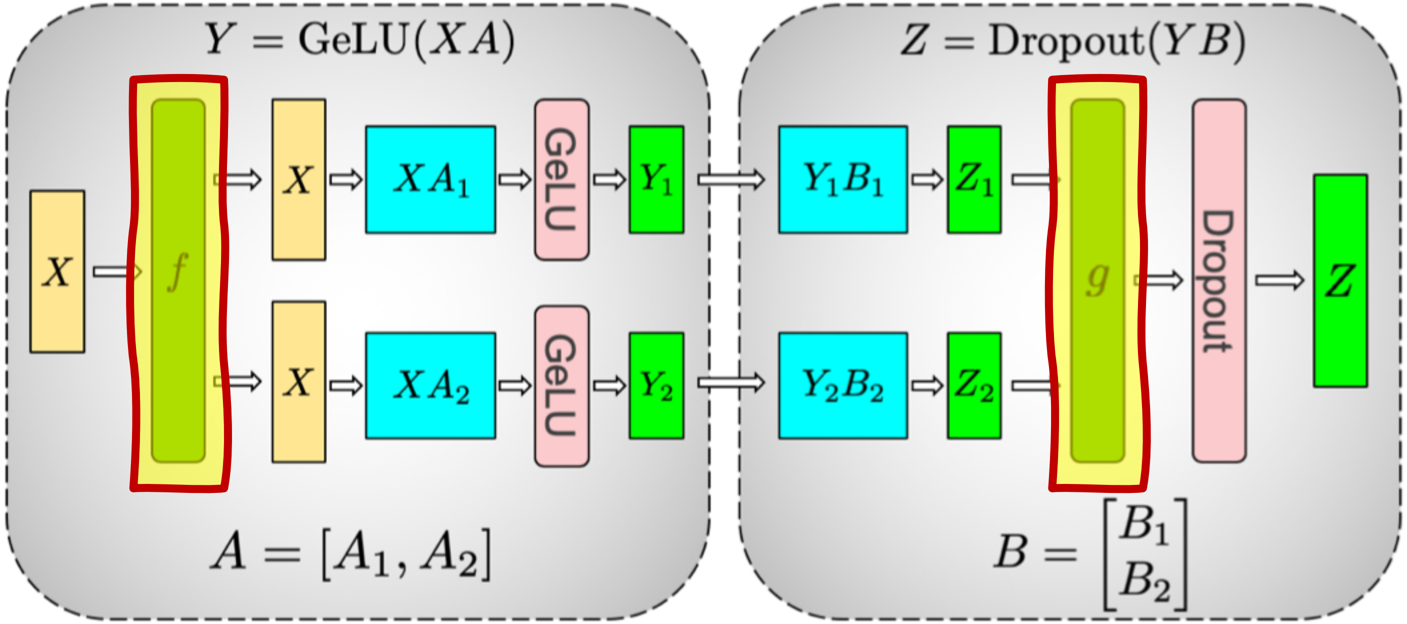


Copy

All Gather

Transformer: MLP

$$Z = \text{Dropout}(\text{GeLU}(XA), B)$$



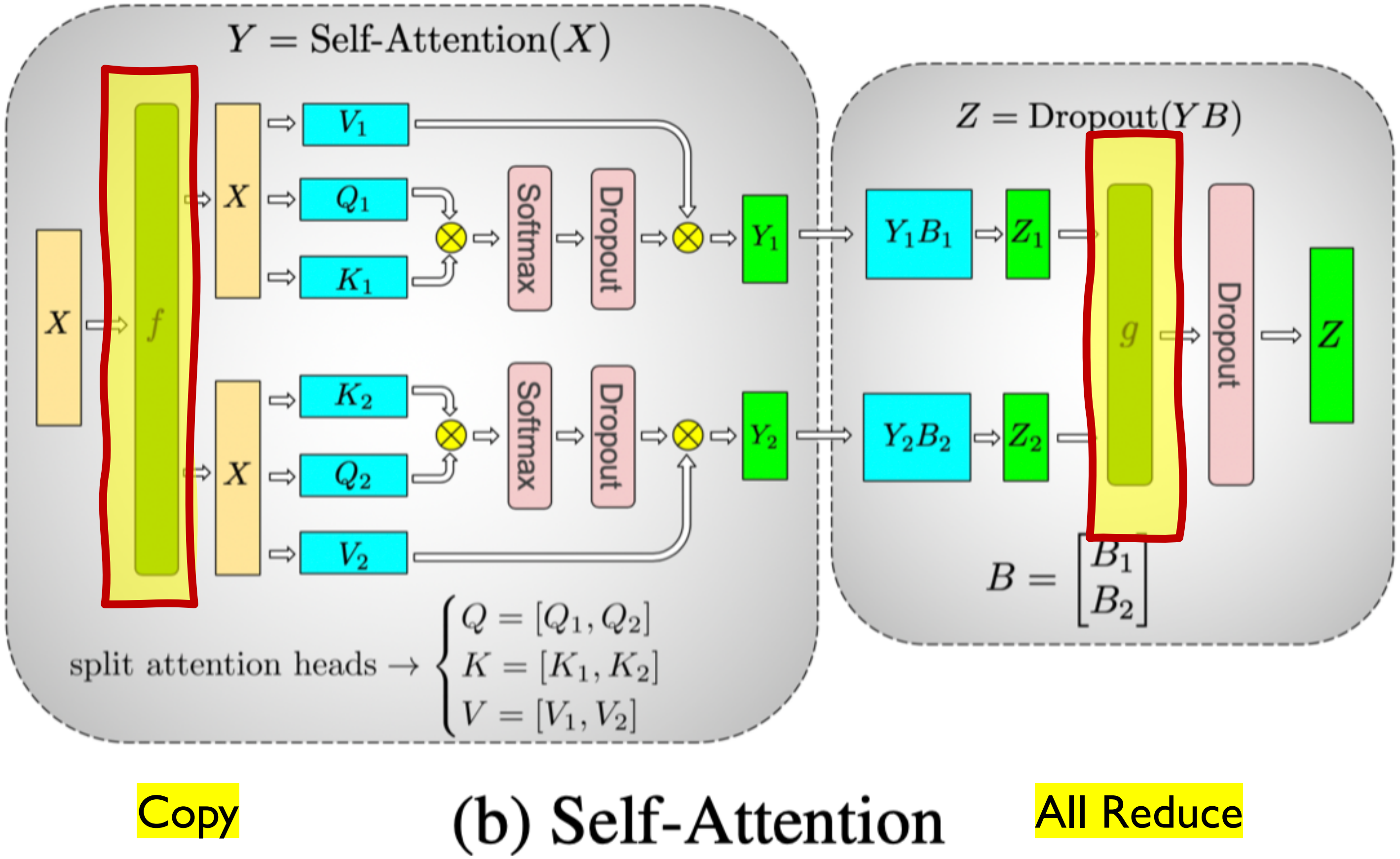
Copy

(a) MLP

All Reduce

Transformer: Self-Attention

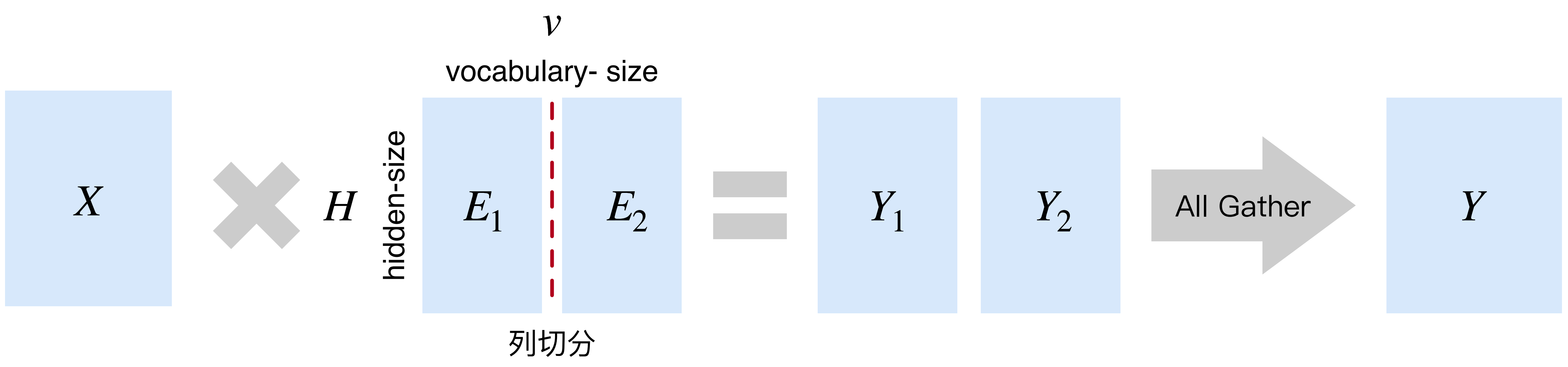
$$Z = \text{Dropout}(\text{Self-Attention}(XA), B)$$



(b) Self-Attention

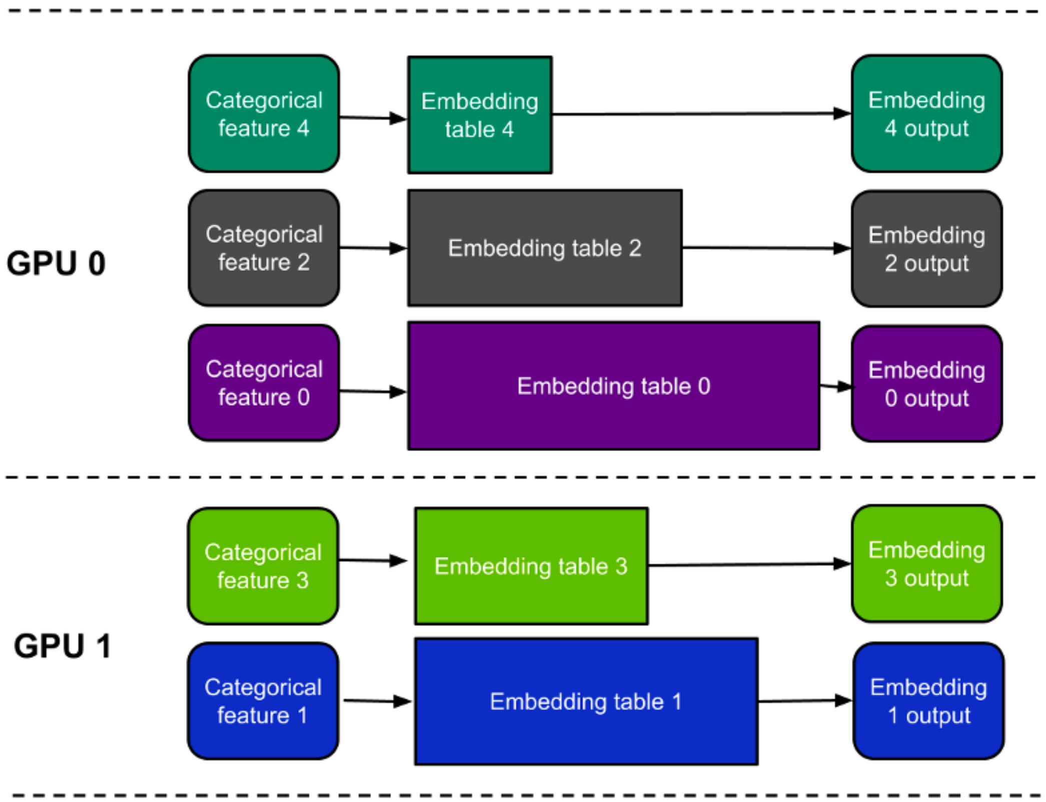
Transformer: Embedding

ie. GPT-2 used a vocabulary size of 50,257. 按照词的维度切分，即每张卡只存储部分词向量表，然后通过 All Gather 汇总各个设备上的部分词向量结果，从而得到完整的词向量结果

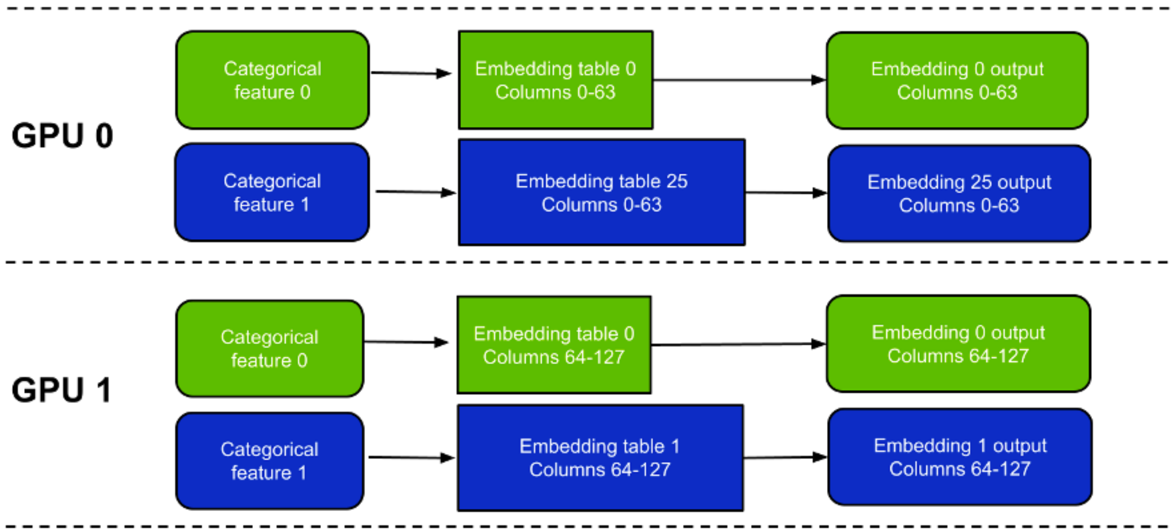


Recommender: Embedding

Table-wise split mode is when each GPU stores a subset of all the embedding tables



Column-wise split mode is when each device stores a subset of columns from every embedding table



Cross Entropy Loss

- 在大词表的语言模型中，logits 规模达到 [bs∗seq_len, vocab_size] 在单个设备上计算较为困难，那么就需要考虑将大词表拆分到多个设备上进行计算；
- 分类网络最后一层一般会选用 softmax 和 cross entropy 来计算损失。如果类别数量非常大，会导致单卡显存无法存储和计算 logit 矩阵，此时可以按照类别数维度切分；

$$L = Loss(logits, labels)$$

Cross Entropy Loss

原理介绍

二分类情况下，对于每个类别预测概率为 p 和 1−p ，此时表达式为：

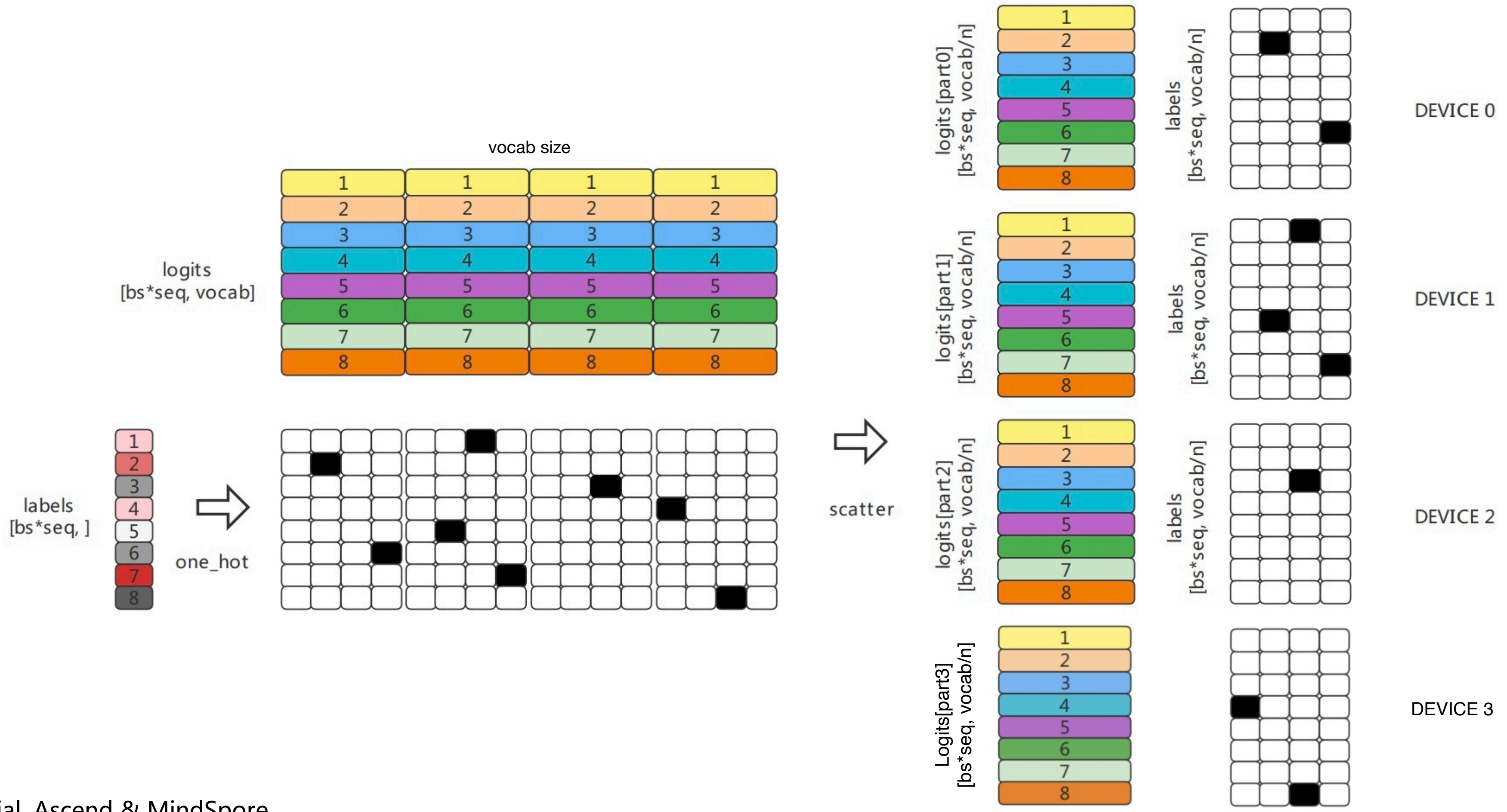
$$L = \frac{1}{N}\sum_{i} L_i = \frac{1}{N}\sum_{i} -\left[y_i \cdot \log\left(p_i\right) + \left(1 - y_i\right) \cdot \log\left(1 - p_i\right)\right]$$

多分类的情况实际上就是对二分类的扩展：

$$L = \frac{1}{N}\sum_{i} L_i = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic} \log\left(p_{ic}\right)$$

Cross Entropy Loss

1. Step1数据拆分：将 logits (input) 按照 vocab 维度进行拆分，同时将不同部分分发到各设备，labels (target) 需要先进行 one hot 操作，然后 scatter 到各个设备上



Cross Entropy Loss

1. **Step2 input(logits) 最大值同步**：input(logits) 需要减去其最大值后求 softmax，All Reduce (Max) 操作保证了获取的是全局最大值，有效防止溢出。

$$x_{\max} = MAX_p\left(MAX_k\left(x_k\right)\right)$$

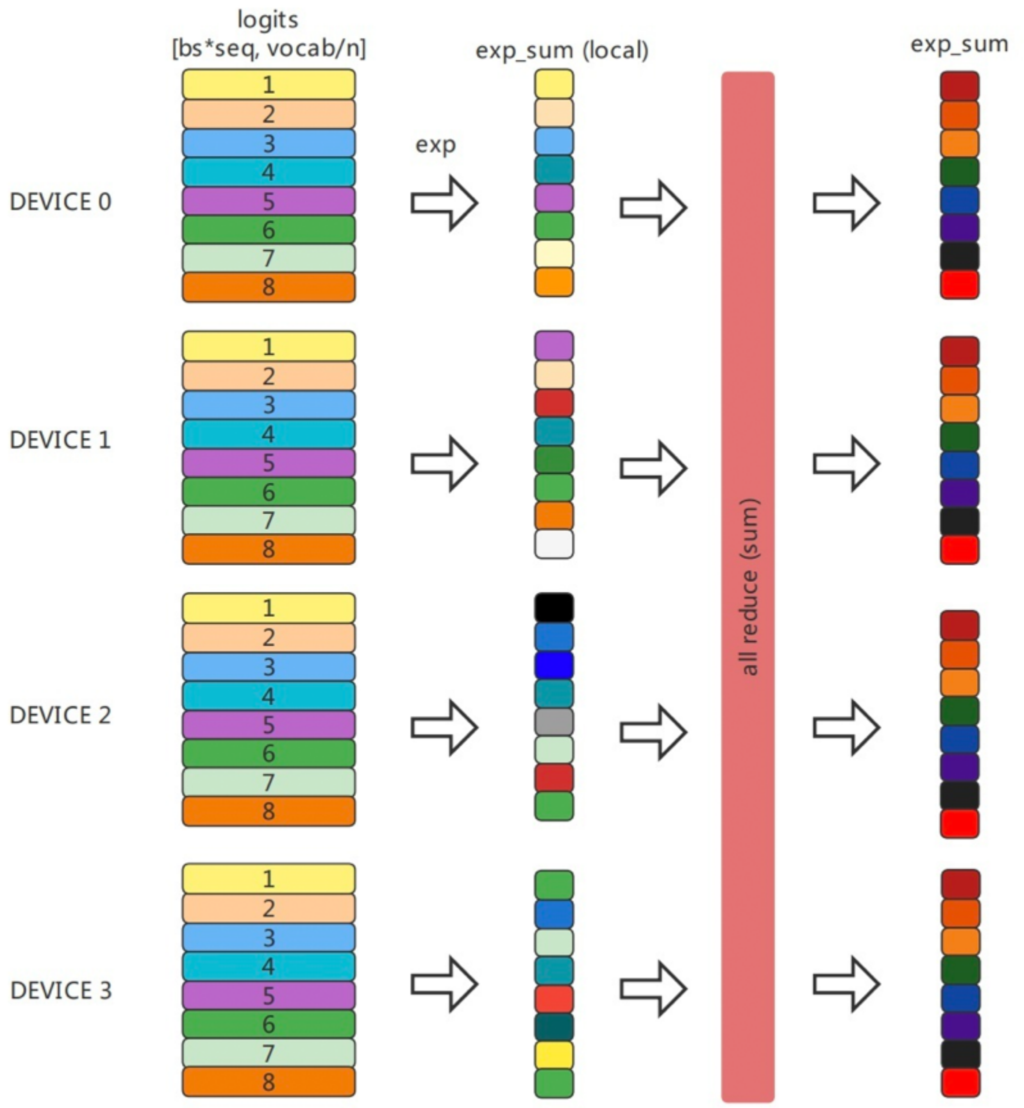
1. **Step 3 exp sum 与softmax 计算**：exp sum 即 softmax 计算中的分母部分， All Reduce (Max) 操作保证了获取的是全局的和。

$$\operatorname{softmax}\left(x_i\right) = \frac{e^{x_i}}{\sum_j e^{x_j}} = \frac{e^{x_i - x_{\max}}}{\sum_j e^{x_j - x_{\max}}} = \frac{e^{x_i - x_{\max}}}{\sum_p \sum_k e^{x_k - x_{\max}}}$$

Cross Entropy Loss

I. Step 3 exp sum 与softmax 计算：exp sum 即 softmax 计算中的分母部分， All Reduce (Max) 操作保证了获取的是全局的和。

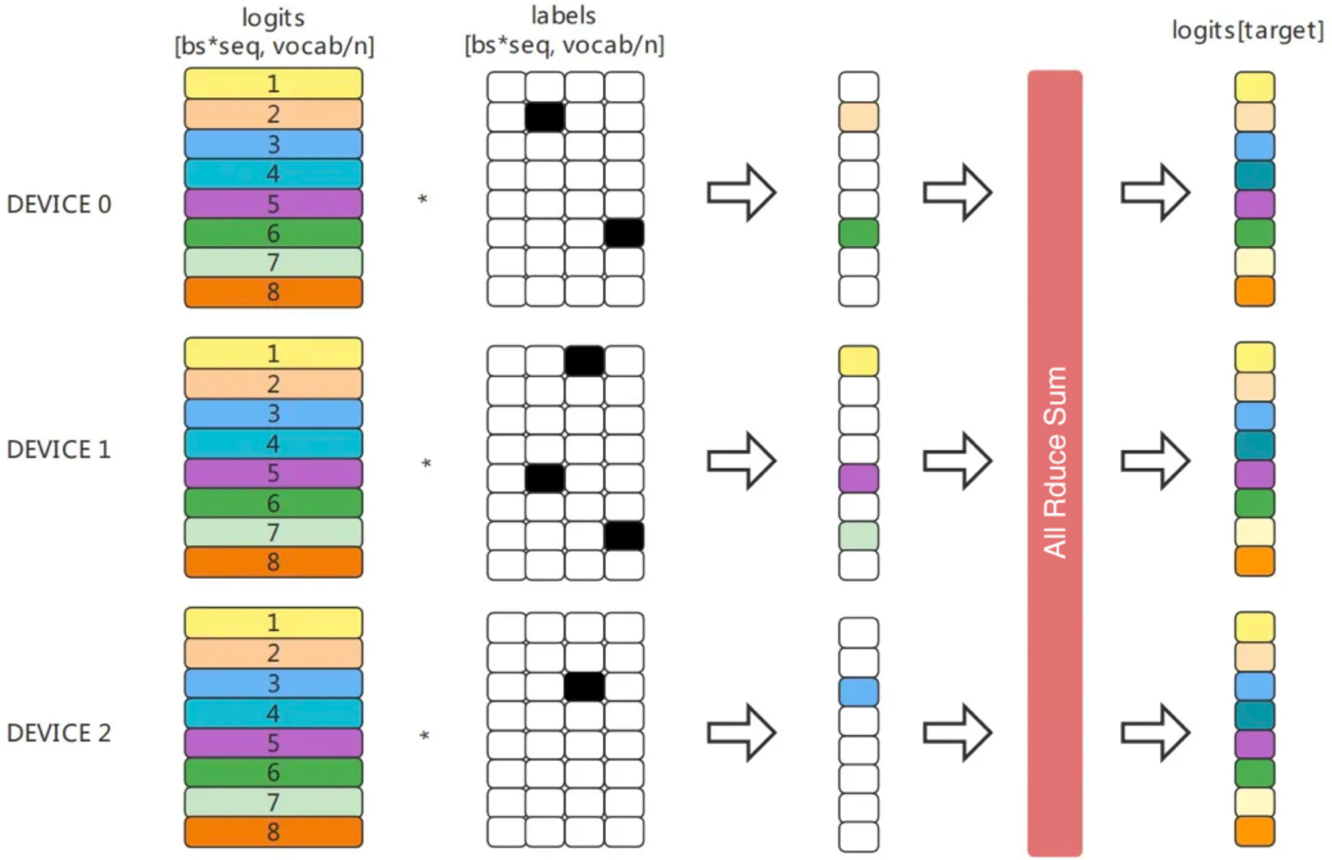
$$\mathrm{softmax}\left(x_i\right) = \frac{e^{x_i}}{\sum_j e^{x_j}} = \frac{e^{x_i - x_{\max}}}{\sum_j e^{x_j - x_{\max}}} = \frac{e^{x_i - x_{\max}}}{\sum_p \sum_k e^{x_k - x_{\max}}}$$



Cross Entropy Loss

I. Step 4 计算 Loss： input (logits) 与 one_hot 相乘并求和，得到 label 位置值 im ，并进行 all_reduce (Sum) 全局同步，最后计算 log softmax 操作并加上负号，得到分布式交叉熵的损失值 loss。

$$\log \sum_j e^j - i_m = -\log \frac{e^{i_m}}{\sum_j e^j}$$

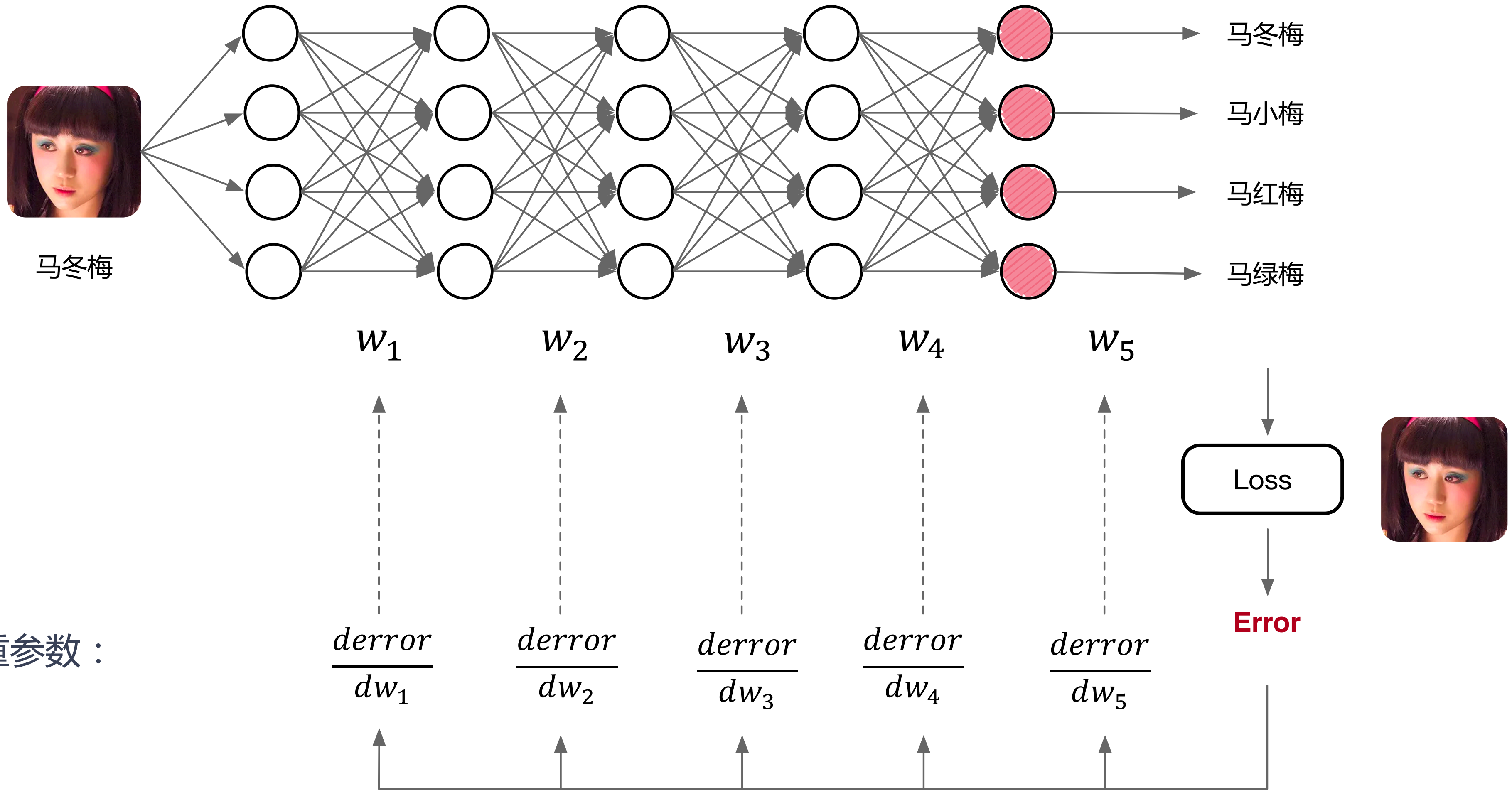


Review: Deep Learning Fundamentals

1. 定义一个神经网络:

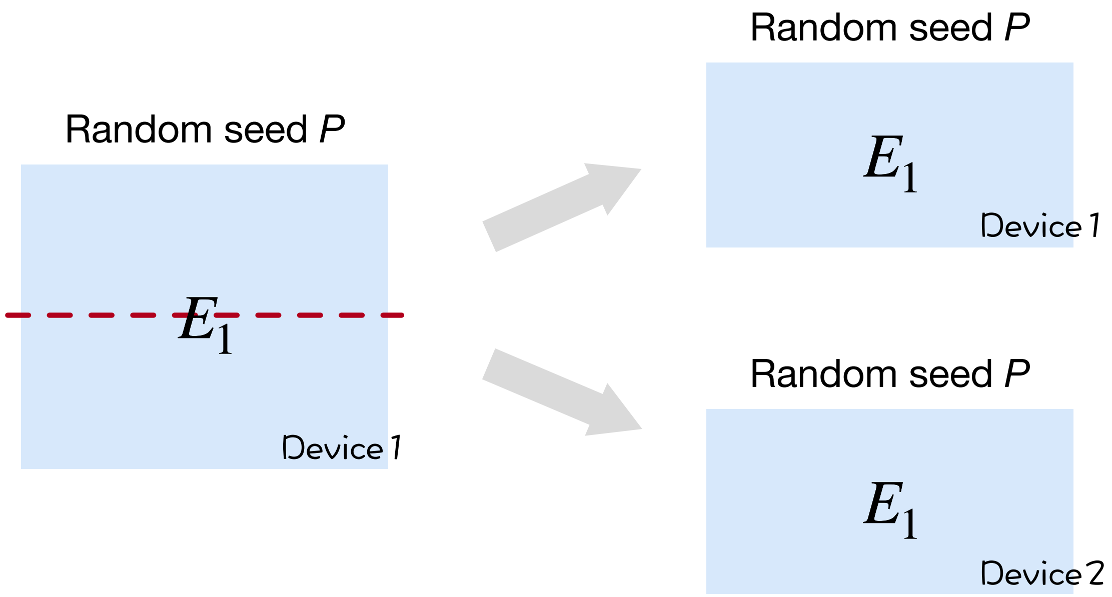
2. 定义优化目标：

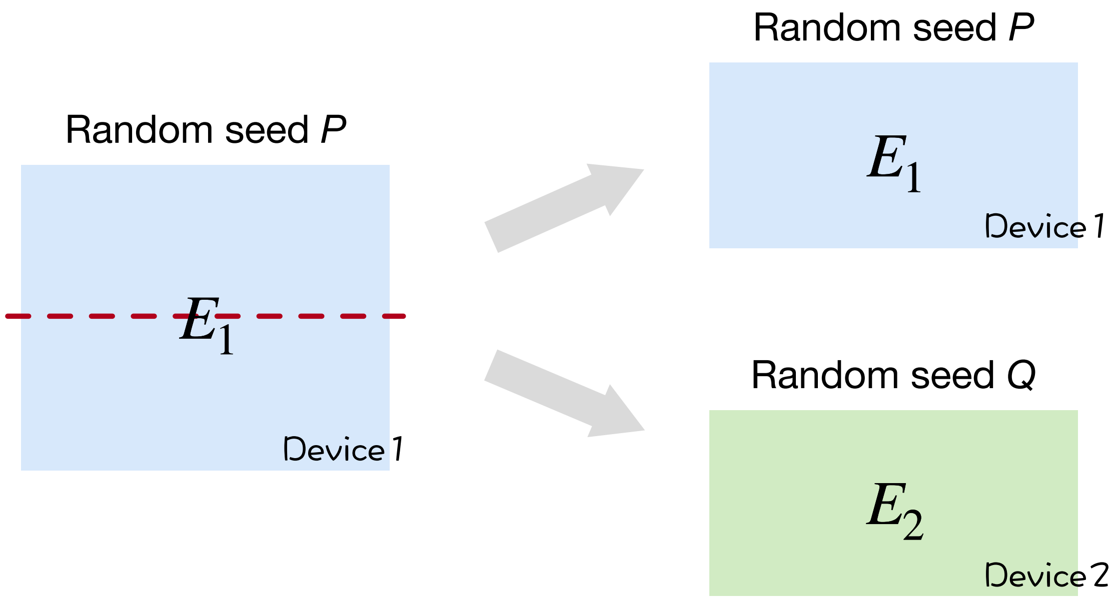
3. 计算梯度并更新权重参数：



Stochastic control(1): Computational 参数初始化随机性

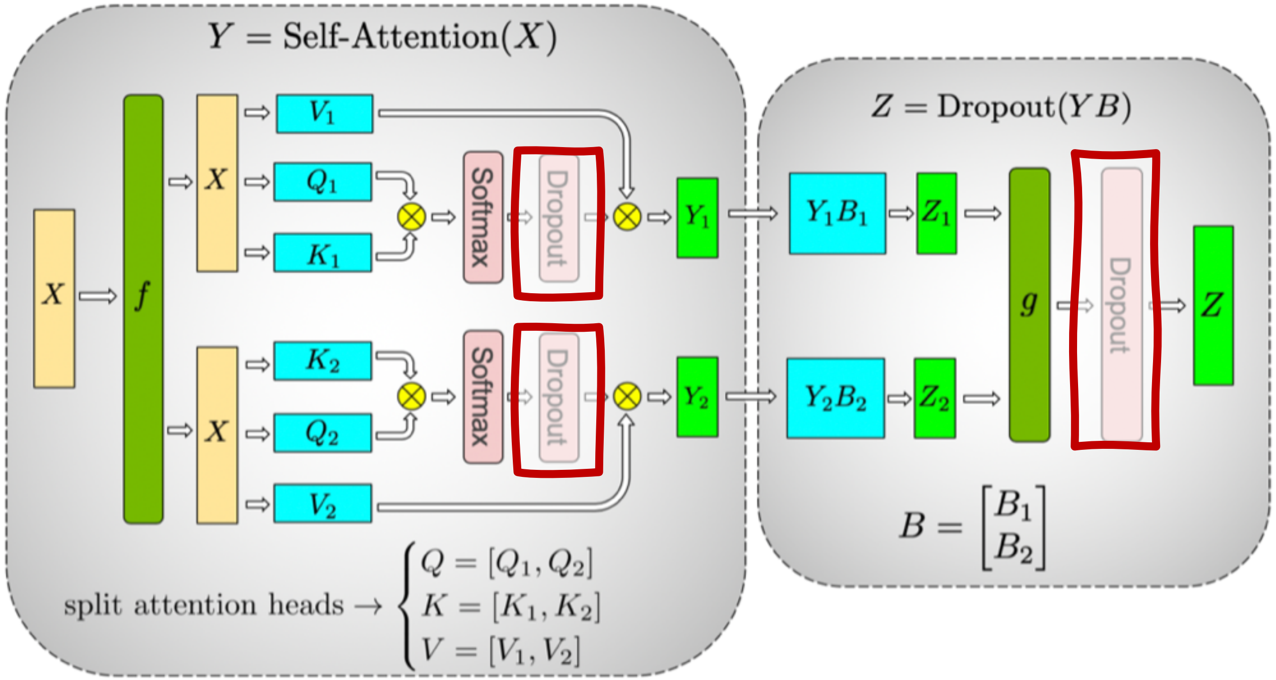
- 参数初始化后两个设备的参数将会初始化为相同的数值，和单设备上参数 E 数学不等价，失去了真正随机性
- 将参数切分到多个卡上后，再修改相应卡的随机性，保证各个卡的随机种子不同。多卡参数初始化随机性与单卡相同





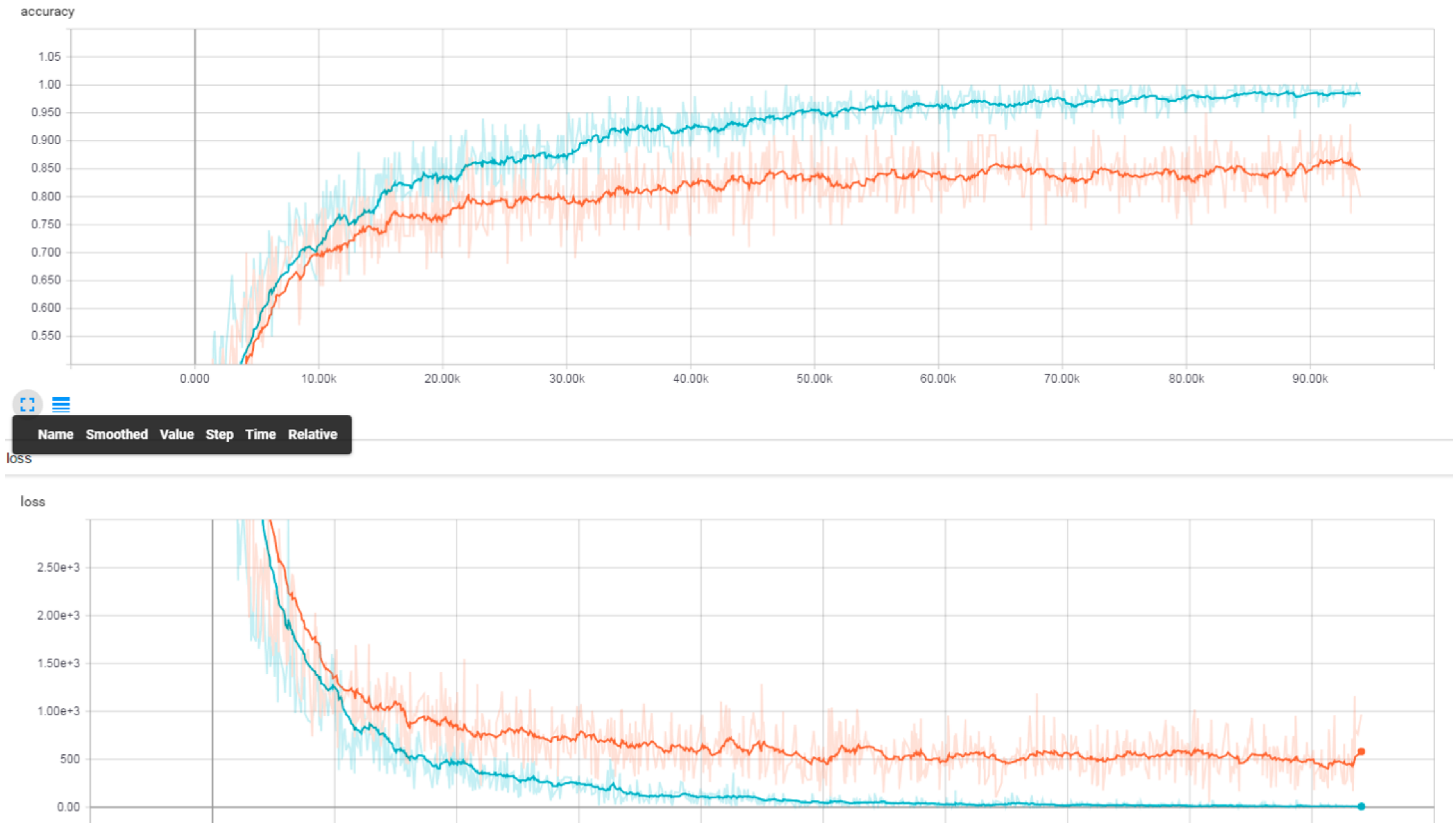
Stochastic control(1): Computational 算子计算的随机性

- 网络模型中的计算算子随机性，并行过程中的随机性不同
- i.e.: Dropout, TruncatedNormal, StandardNormal, Multinomial, StandardLaplace, Gamm. et. al.



(b) Self-Attention

Stochastic control(III): Effective



Summary 总结

1. 模型并行分为张量并行和流水线并行，张量并行主要层内并行、流水线主要层间并行，一般来说机内使用张量并行，机间使用数据并行；
2. 张量并行主要是对数据进行切分，切分方式有行（Row）切分和列（Col）切分，而通过复制组合可以形成多种通信形式；
3. 张量并行最常见的是 MatMul 算子并行，通过 MatMul 可以拓展到 Embedding、MLP、 Transformer等算子并行；
4. 张量并行的时候值得注意的是随机性问题，需要注意带有随机性算子的随机种子设置；

Inference

1. https://zhuanlan.zhihu.com/p/450854172 全网最全-超大模型+分布式训练架构和经典论文
2. https://developer.nvidia.com/blog/training-a-recommender-system-on-dgx-a100-with-100b-parameters-in-tensorflow-2/
3. https://developer.nvidia.com/blog/fast-terabyte-scale-recommender-training-made-easy-with-nvidia-merlin-distributed-embeddings/
4. https://www.mindspore.cn/docs/zh-CN/r1.7/design/operator_parallel.html
5. https://www.mindspore.cn/docs/zh-CN/r1.7/design/distributed_training_design.html
6. https://colossalai.org/zh-Hans/docs/features/2D_tensor_parallel/
7. https://zhuanlan.zhihu.com/p/507877303
8. https://zhuanlan.zhihu.com/p/450689346
9. https://zhuanlan.zhihu.com/p/497672789

BUILDING A BETTER CONNECTED WORLD

THANK YOU