



---

# 大模型训练与优化

---

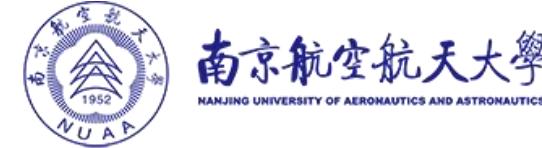
魏明强、宫丽娜  
计算机科学与技术学院

---

智周万物·道济天下

---

# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 训练数据准备



## □ 数据获取：

- 收集类别丰富的样本
- 筛选高质量的样本

## □ 数据增强：

- 扩充数据规模
- 提高数据多样性

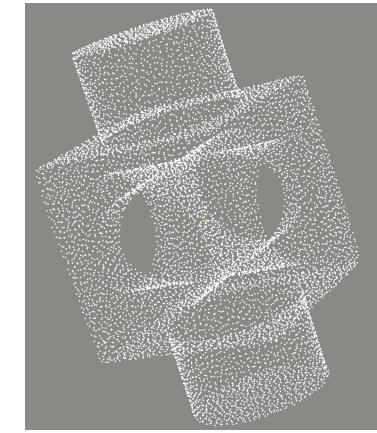
## □ 训练数据配比和课程设置：

- 确定数据的比例
- 编排数据使用顺序

In our paper, we have provided a formula to estimate the initial neighborhood size  $h$ , for arbitrary inputs, according to the bounding box size and points density.

In our experience, the initial neighborhood size should be small enough to capture fine-scale structures. But if it's too small, it would take more computational time and may produce unnatural branches, like the coral example shown in paper.

文本



图像

点云

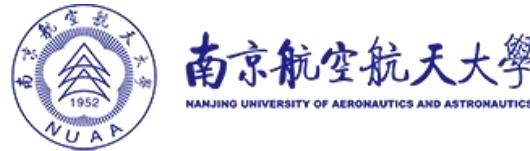
大模型展现出卓越性能的一个关键原因：海量的高质量训练数据

# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 训练数据准备—数据获取



## 1. 文本数据来源

### □ 通用文本：

- 来源：包括在线论坛、社交媒体、新闻、博客、书籍、期刊等
- 主题：涵盖社会、科技、娱乐、健康等
- 表达：囊括不同人群、地区和文化背景的表达方式

### □ 三个主要来源：

网页数据

对话数据

书籍数据

新闻文章

博客

百科数据

社交媒体

电子邮件对话

社交媒体对话

论坛帖子

科技类

社会人文类

历史类

小说

数据量大  
内容丰富

理解对话逻辑

表达规范  
长文本理解

# 训练数据准备—数据获取



## 1. 文本数据来源

### □ 专业文本：

- 数据占比较低
- 包含大量专业术语以及特定的语法句式

### □ 常见的专业文本数据：



建筑工程施工技术档案资料填写全套示范本（通用版）			
混凝土开盘鉴定			
质控（建）表4.1.6.8			共 1 页 第 1 页
工程名称	锦江花园1号楼	施工单位	福建××建筑工程有限公司
搅拌单位	××市宝洁混凝土有限公司	试配单位	××市建筑工程检测中心
试验配合比		试验配合比	

建筑工程施工技术档案资料填写全套示范本（通用版）			
混凝土工程（现场搅拌）施工记录			
质控（建）表4.1.6.9-1			共 × 页 第 1 页
工程名称	锦江花园1号楼	施工单位	福建××建筑工程有限公司
材料名称	水 泥	掺 合 料	外 加 剂
品牌、等级	中达牌 P.O 42.5	/	/
细度			
合格证明			

### 科学文本数据

学术论文    技术报告  
教材

赋予模型理解科学问题的能力

### 行业专业文本

法律法规    工程文档  
合同

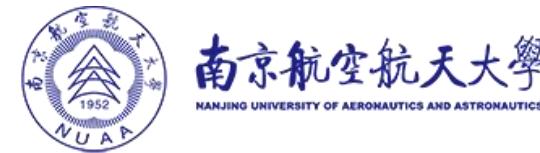
注重实际应用、业务操作和解决特定问题的需求

### 代码文本

开源代码仓库    开发者社区论坛  
编程竞赛和挑战平台

具有特定的语法规则及准确的执行逻辑

# 训练数据准备—数据获取



## 2. 图像数据来源

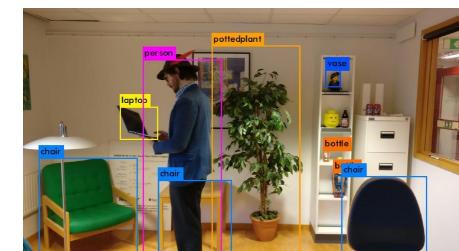
### □ 通用图像数据:

- 涵盖了人类日常生活中的各种场景
- 从互联网收集得到； 各种常规的便携设备，如手机、平板电脑、相机等拍摄获取
- 比文本的信息更加密集，包含丰富的视觉特征，如颜色、纹理、形状等



### □ 确保数据多样性需要考虑:

- **天气条件:** 收集图像时考虑不同的天气条件，包括晴天、阴天、雨天、雪天等
- **时间变化:** 收集一天不同时间段下拍摄的图像，这能够捕捉到光照、阴影等方面的变化
- **人群多样性:** 确保图像中包含不同人群的照片，考虑年龄、性别、种族等因素
- **物体类别:** 涵盖多个物体类别，包括不同的动植物、建筑物、交通工具等
- **场景多样性:** 需要包括常见的室内及室外场景，如办公室、卧室、城市街景
- **文化多样性:** 考虑在不同社会环境中收集图像，涵盖不同文化、习惯和社交活动

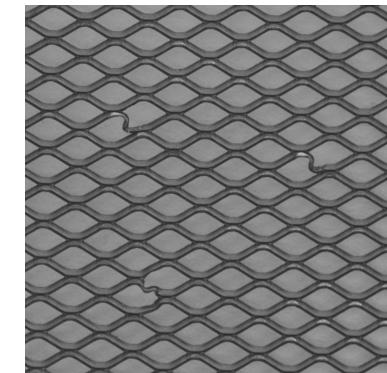
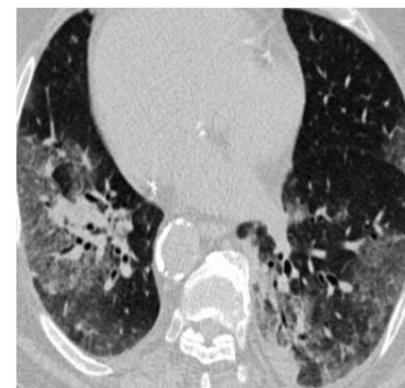


# 训练数据准备—数据获取

## 2. 图像数据来源

### □ 专业图像数据：

- 针对特定领域或专业需求采集的图像数据
- 使用专业设备或者在特定场景下采集
- 例如，通过X光机、CT扫描获得医学图像；
- 通过卫星或航空器获取的地球表面的遥感图像；
- 工业生产线上拍摄得到的产品缺陷检测图像；



# 训练数据准备—数据获取



## 3. 点云数据来源

- 常见的三维数据表示形式有：点云、三角网格、体素、隐式表达

- 点云定义：

- 三维点的数据集合

- 属性：

- 三维坐标
  - 强度
  - 颜色

- 采集设备：

- 激光扫描仪、深度相机、双目相机、光学相机多视角重建、结构光设备



# 目录

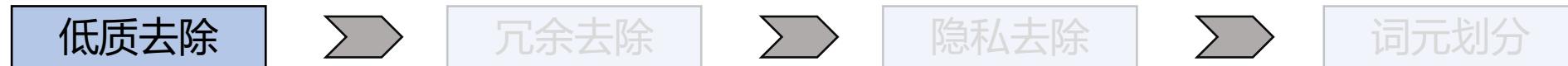


- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 训练数据准备—数据预处理



## 1. 文本数据预处理



### □ 低质去除：

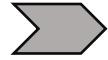
- 目标：去除那些质量较差，以及不符合标准的文本数据
- 基于分类器的方法：  
使用一组精选的文本（包括维基百科、书籍等），训练一个分类器用于判断文本的质量，将与训练数据类似的数据给定较高的分数。利用该分类器评估数据的内容质量
- 基于启发式的方法：  
自定义规则，对数据进行筛选  
例如：  
去除单词数量少于50个或者大于100000个的文档  
去除符号与单词的比例大于0.1的文件

# 训练数据准备—数据预处理

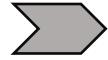


## 1. 文本数据预处理

低质去除



冗余去除



隐私去除



词元划分

### 冗余去除：

- 目标：去除文本数据中的冗余信息，精简数据集，防止模型在预测时陷入重复循环
- 句子级别：构建过滤方法，识别重复句子  
例如，提取并过滤文档间超过一定长度的相同字符串（公共子串匹配）
- 段落或者文档级别：基于文本之间的特征相似度来进行冗余去除  
例如，计算两个段落或者文档之间的13-gram的Jaccard相似度来判断它们是否重复

### 补充知识：

N-gram是一种文本特征表示方法，它将文本分解为连续的n个单词或字符序列。常用的是基于单词的n-gram，其中n表示连续的单词的数量。

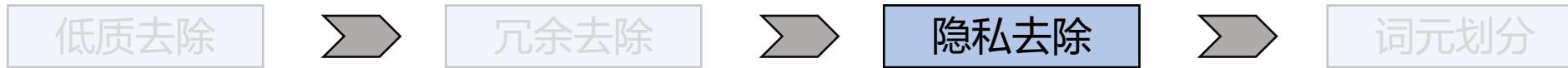
例如，在句子"This is an example"中，2-gram表示为[ "This is"， "is an"， "an example" ]。

对于文本，可以将其表示为n-gram序列，然后计算n-gram序列之间的Jaccard指数来比较两个文本的相似性

# 训练数据准备—数据预处理



## 1. 文本数据预处理



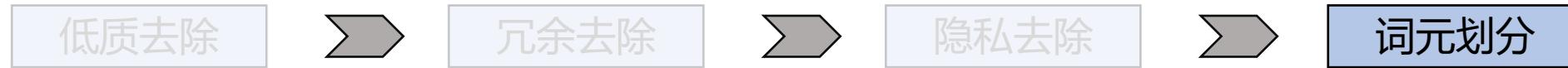
### □ 隐私去除：

- 目标：删除或替换文本数据中个人姓名、电话号码、电子邮件地址等敏感信息
- 基于规则的算法：  
例如，可使用命名实体识别算法，从文本中检测姓名、地址、电话号码等信息

# 训练数据准备—数据预处理



## 1. 文本数据预处理



### □ 词元划分:

- 目标：将连续的文本划分为有意义的词元 (tokens)

□ **词粒度划分：**将连续文本以单词为基本单元进行划分

#### □ 缺点：

- 只能处理预先定义的词表内的词
- 词表中可能存在常委分布，使得模型对稀有词的理解不佳
- 对于英语等语言，无法正确处理不同时态的单词

□ **字符划分：**将字符视为词元来构建词表

#### □ 缺点：

- 字符作为词元的语义表达不足
- 一个单词需要由多个字符来表示，计算成本增加

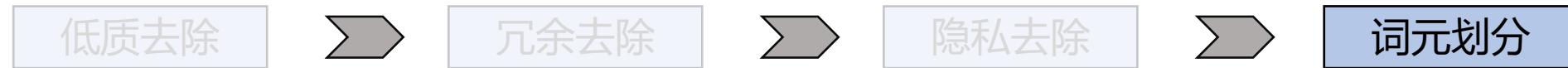
□ **子词划分：**基于某种规则对单词进行拆分，高频词保持原状，将低频词拆分成子词

例如，对于单词token不进行拆分，对于单词tokenization则拆分为token和ization。

# 训练数据准备—数据预处理



## 1. 文本数据预处理



### □ 词元划分:

- 常见的方法: Byte-Pair Encoding (BPE) 、 WordPiece、 Unigram Language Model (ULM)
- 基本流程: 1.构建词表 2.基于词表进行分词

#### Byte-Pair Encoding (BPE)

##### 构建词表:

1. 准备足够大的训练语料，并确定期望的Subword词表大小；

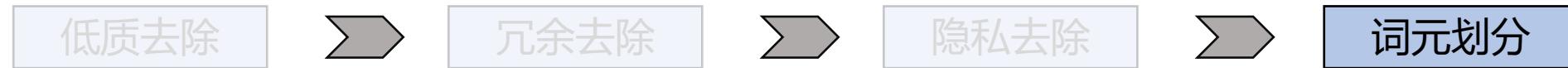
语料	词表
high </t>: 3次 oldest </t>: 6次 newest </t>: 3次	

</t>是插入到每个单词后的终止符，  
用于区分单词边界

# 训练数据准备—数据预处理



## 1. 文本数据预处理



### □ 词元划分:

- 常见的方法: Byte-Pair Encoding (BPE) 、 WordPiece、 Unigram Language Model (ULM)
- 基本流程: 1.构建词表 2.基于词表进行分词

#### Byte-Pair Encoding (BPE)

##### 构建词表:

1. 准备足够大的训练语料，并确定期望的Subword词表大小；
2. 将单词拆分为最小单元。比如英文中26个字母加上各种符号，这些作为初始词表；

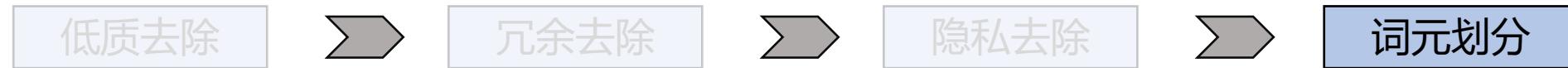
语料	词表
high </t>: 3次 oldest </t>: 6次 newest </t>: 3次	h,i,g,o,l,d,e,s,t,n,w, </t>

</t>是插入到每个单词后的终止符，  
用于区分单词边界

# 训练数据准备—数据预处理



## 1. 文本数据预处理



### □ 词元划分：

- 常见的方法：Byte-Pair Encoding (BPE)、WordPiece、Unigram Language Model (ULM)
- 基本流程：1.构建词表 2.基于词表进行分词

#### Byte-Pair Encoding (BPE)

##### 构建词表：

1. 准备足够大的训练语料，并确定期望的Subword词表大小；
2. 将单词拆分为最小单元。比如英文中26个字母加上各种符号，这些作为初始词表；
3. 在语料上统计单词内相邻单元对的频数，选取频数最高的单元对合并成新的Subword单元；

语料	词表
high </t>: 3次 oldest </t>: 6次 newest </t>: 3次	h,i,g,o,l,d,e,s,t,n,w, </t>,es

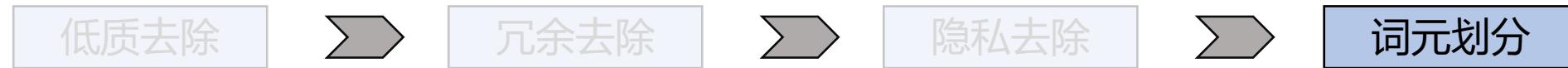
最高连续出现的子词对是 e+s，共出现了9次，进行合并

语料中不存在 s 了，需要删除

# 训练数据准备—数据预处理



## 1. 文本数据预处理



### □ 词元划分：

- 常见的方法：Byte-Pair Encoding (BPE)、WordPiece、Unigram Language Model (ULM)
- 基本流程：1.构建词表 2.基于词表进行分词

#### Byte-Pair Encoding (BPE)

##### 构建词表：

1. 准备足够大的训练语料，并确定期望的Subword词表大小；
2. 将单词拆分为最小单元。比如英文中26个字母加上各种符号，这些作为初始词表；
3. 在语料上统计单词内相邻单元对的频数，选取频数最高的单元对合并成新的Subword单元；
4. 重复第3步直到达到第1步设定的Subword词表大小或下一个最高频数为1

语料	词表
high </t>: 3次 oldest </t>: 6次 newest </t>: 3次	h,i,g,o,l,d,e,s,t,n,w, </t>,es

# 训练数据准备—数据预处理



## 1. 文本数据预处理

低质去除



冗余去除



隐私去除



词元划分

### □ 词元划分：

- 常见的方法：Byte-Pair Encoding (BPE)、WordPiece、Unigram Language Model (ULM)
- 基本流程：1.构建词表 2.基于词表进行分词

#### Byte-Pair Encoding (BPE)

##### 基于词表进行分词：

1. 将词表中的所有子词按照长度由大到小进行排序；
2. 对于给定单词，遍历词表。查看当前子词是否是该单词的子字符串，如果是，则输出当前子词，并对剩余单词字符串继续匹配；
3. 如果遍历完字典后，仍然有子字符串没有匹配，则将剩余字符串替换为特殊符号；
4. 使用上述遍历过程中得到的字词来表示单词

##### 解码过程：

如果相邻子词间没有终止符，则将两子词直接拼接，否则两子词之间添加终止符



## 2. 图像数据预处理

低质去除



冗余去除

### □ 低质去除：

- 目标：去除质量较低的图像，如模糊、过曝或者有严重噪声的图像
- 基于启发式的规则：
  - 图像损坏：  
通过检查数据集中数据是否能正常读取来判断文件是否损坏
  - 图像模糊：  
一种常用的方法是拉普拉斯算子。模糊图像通常具有较少的边界信息，其经过拉普拉斯算子计算后得到的图像像素值的方差也会较小，可以根据方差的大小来判断图像是否模糊
  - 图像格式：  
去除GIF等不符合常规图像数据集要求的格式
  - 图像分辨率：  
去除分辨率过小的图像
  - 图像内容质量：  
去除水印、广告

## 2. 图像数据预处理

低质去除



冗余去除

### □ 冗余去除：

- 目标：度量图像之间的相似度，去除数据集中内容相似的图像
- 基于直方图的方法：  
对于RGB图像，计算三个颜色通道的灰度直方图，使用某种距离或者相似度度量方法来比较图像的直方图

# 训练数据准备—数据预处理



## 2. 图像数据预处理

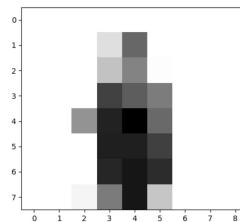
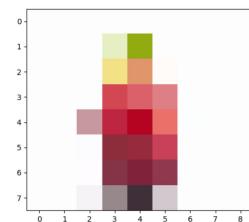
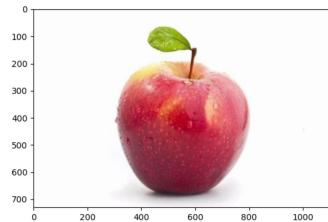
低质去除



冗余去除

### □ 冗余去除：

- 目标：度量图像之间的相似度，去除数据集中内容相似的图像
- 基于直方图的方法：  
对于RGB图像，计算三个颜色通道的灰度直方图，使用某种距离或者相似度度量方法来比较图像的直方图
- 基于哈希值的方法：  
将图像转换为一个固定长度的二进制字符串（哈希值），并且认为相似的图像会有相似的哈希值，包括：均值哈希、感知哈希和差值哈希  
以差值哈希为例：1. 缩小图像    2. 图像灰度化    3. 计算差异值    4. 生成哈希值    5. 哈希值比较



0000000000110000  
0011000000100000  
0111000000100000  
0011000001110000

0030302070203070

(2进制转16进制)



## 2. 图像数据预处理

低质去除



冗余去除

### □ 冗余去除：

- 目标：度量图像之间的相似度，去除数据集中内容相似的图像
- 基于直方图的方法：  
对于RGB图像，计算三个颜色通道的灰度直方图，使用某种距离或者相似度度量方法来比较图像的直方图
- 基于哈希值的方法：  
将图像转换为一个固定长度的二进制字符串（哈希值），并且认为相似的图像会有相似的哈希值，包括：均值哈希、感知哈希和差值哈希
- 基于余弦距离的方法：  
余弦相似度是一种常用的衡量向量之间相似度的方法。在图像相似度的计算中，可以先生成图像的特征向量，然后计算向量夹角的余弦值。

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}.$$

# 训练数据准备—数据预处理



## 2. 图像数据预处理

低质去除



冗余去除

### □ 冗余去除：

- 目标：度量图像之间的相似度，去除数据集中内容相似的图像
- 基于直方图的方法：  
对于RGB图像，计算三个颜色通道的灰度直方图，使用某种距离或者相似度度量方法来比较图像的直方图
- 基于哈希值的方法：  
将图像转换为一个固定长度的二进制字符串（哈希值），并且认为相似的图像会有相似的哈希值，包括：均值哈希、感知哈希和差值哈希
- 基于余弦距离的方法：  
余弦相似度是一种常用的衡量向量之间相似度的方法。在图像相似度的计算中，可以先生成图像的特征向量，然后计算向量夹角的余弦值。
- 基于结构相似度的方法：  
结构相似度是一种基于人眼视觉感知的图像相似度度量方法。它考虑了亮度、对比度和结构三个方面的差异，并通过比较图像的块、窗口和图像整体来计算相似度。结构相似度的取值范围为[ -1, 1 ]，越接近1表示图像越相似。

$$\mu_x = \frac{1}{N} \sum_{i=1}^N x_i \longrightarrow l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

(平均灰度作为亮度测量)

$$\sigma_x = \left( \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2 \right)^{\frac{1}{2}} \longrightarrow c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

灰度标准差作为对比度测量)

$$\frac{x - \mu_x}{\sigma_x} \longrightarrow s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}$$

(结构测量)

(结构对比函数)

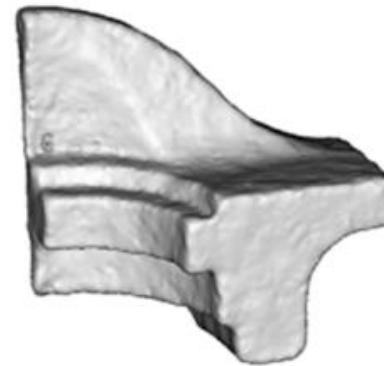
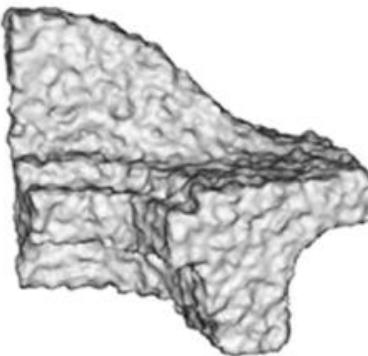
(SSIM函数)

$$S(x,y) = f(l(x,y), c(x,y), s(x,y)) \\ = \frac{(2\mu_x\mu_y + C_1)(2\sigma_x\sigma_y + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

## 3. 点云数据预处理

### □ 去噪:

- 去除采集到的点云的噪声，常见的点云去噪算法有统计滤波、高斯滤波、中值滤波、基于距离的滤波等。





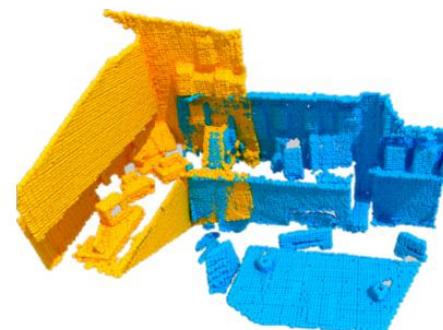
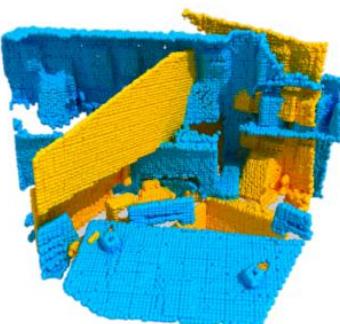
## 3. 点云数据预处理

### □ 去噪:

- 去除采集到的点云的噪声，常见的点云去噪算法有统计滤波、高斯滤波、中值滤波、基于距离的滤波等。

### □ 配准:

- 将来自不同视角或传感器的点云数据进行对齐。点云配准通常可分为粗配准和精配准两个步骤，粗配准方法有RANSAC、4PCS等，精配准方法有ICP、基于深度学习的方法等



# 训练数据准备—数据预处理



## 3. 点云数据预处理

### □ 去噪:

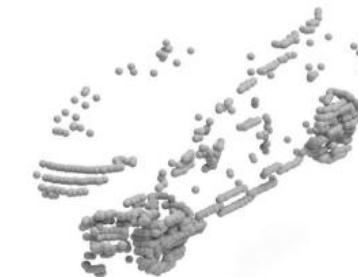
- 去除采集到的点云的噪声，常见的点云去噪算法有统计滤波、高斯滤波、中值滤波、基于距离的滤波等。

### □ 配准:

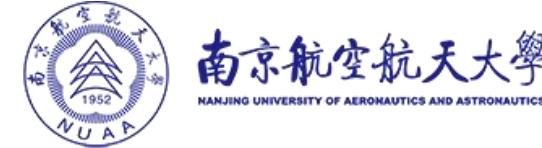
- 将来自不同视角或传感器的点云数据进行对齐。点云配准通常可分为粗配准和精配准两个步骤，粗配准方法有RANSAC、4PCS等，精配准方法有ICP、基于深度学习的方法等

### □ 补全:

- 补全由于遮挡或者设备限制等原因得到的缺失点云。常见的点云补全方法包括插值法、基于几何特征的方法，以及基于学习的方法等。



# 目录



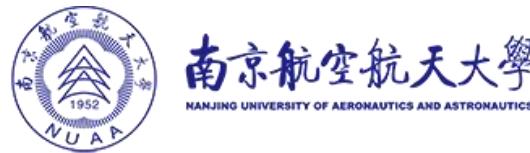
- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 训练数据准备—数据增强



- 数据增强是指通过对原始数据进行一系列变换和扩充，生成更多的训练样本的过程
- 一般可分为离线数据增强和在线数据增强：
  - 离线数据增强是指在训练前，对原始数据进行一系列变换和扩充，生成增强后的数据集。
  - 在线数据增强是指在每次模型训练的过程中，动态地对原始数据进行随机变换和扩充。这些变换和扩充可以在每个训练迭代中随机选择，并根据需求实时调整。

# 训练数据准备—数据增强



## 1. 文本数据增强

### □ 同义词替换:

- 将文本中的某些词替换为它们的同义词，以增加文本的多样性

### □ 随机插入:

- 在文本中随机插入一些额外的单词或短语，以增加文本的长度和多样性

### □ 随机删除:

- 随机删除文本中的某些单词或短语，以模拟文本的不完整或遗漏情况

### □ 随机交换:

- 随机交换文本中两个单词的位置

### □ 随机重排:

- 打乱文本中单词的顺序

### □ 文本生成:

- 使用文本生成模型生成新的文本样本，其与原始文本相似但不完全相同

### □ 回译:

- 将文本翻译成其他语言，再将翻译后的文本翻译回原语言，生成新的文本样本

### □ 掩码语言模型:

- 先利用预训练好的BERT、Roberta等模型，对原句子进行部分掩码，然后让模型预测掩码部分，从而得到新的句子

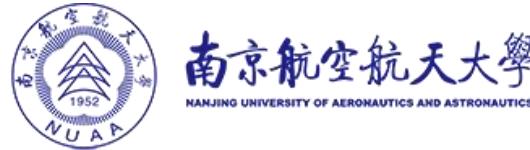
### □ 语法树增强:

- 使用自然语言处理工具分析句子的依存关系得到语法树，制定一些语法规则，如改变名词短语的结构、替换动词、调整从句的位置等。基于修改后的语法树，生成相应的文本

### □ 词混合:

- 对两个词嵌入或者句子以一定的比例进行混合

# 训练数据准备—数据增强



## 2. 图像数据增强

### □ 基于预定规则的方法:

- 事先定义好一些规则或变换方式，通常可分为单样本数据增强和多样本数据增强
- 单样本数据增强：进行数据增强时仅围绕当前样本进行各种变换
  - 几何变换：翻转、旋转、缩放、平移、裁剪等
  - 颜色变换：加噪、模糊、颜色扰动（对亮度、对比度、饱和度、色相的调整）
- 多样本数据增强：使用多个样本来生成新的样本
  - CutMix：在一幅图像上随机选择一个矩形区域，将选定的矩形区域剪切并粘贴到另一幅图像上
  - MixUp：将两幅图像按照一定比例融合
  - Mosaic：从训练数据中随机选择4幅图像，在待拼接的大图中随机选择一个中心点，将4幅图像按照选定的中心点进行缩放和平移，使它们围绕中心点排列

### □ 基于无监督的方法:

- 生成新样本：使用预训练的生成模型，例如对抗生成网络，生成与输入分布一致的图像
- 学习增强策略：自动学习数据增强的策略，该类方法能够推荐应该使用何种数据增强、推荐的数据增强方法的使用概率以及数据增强方法的超参数
  - 例如，AutoAugment、RandAugment



CutMix

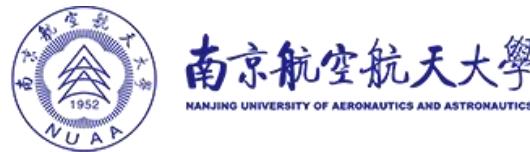


MixUp



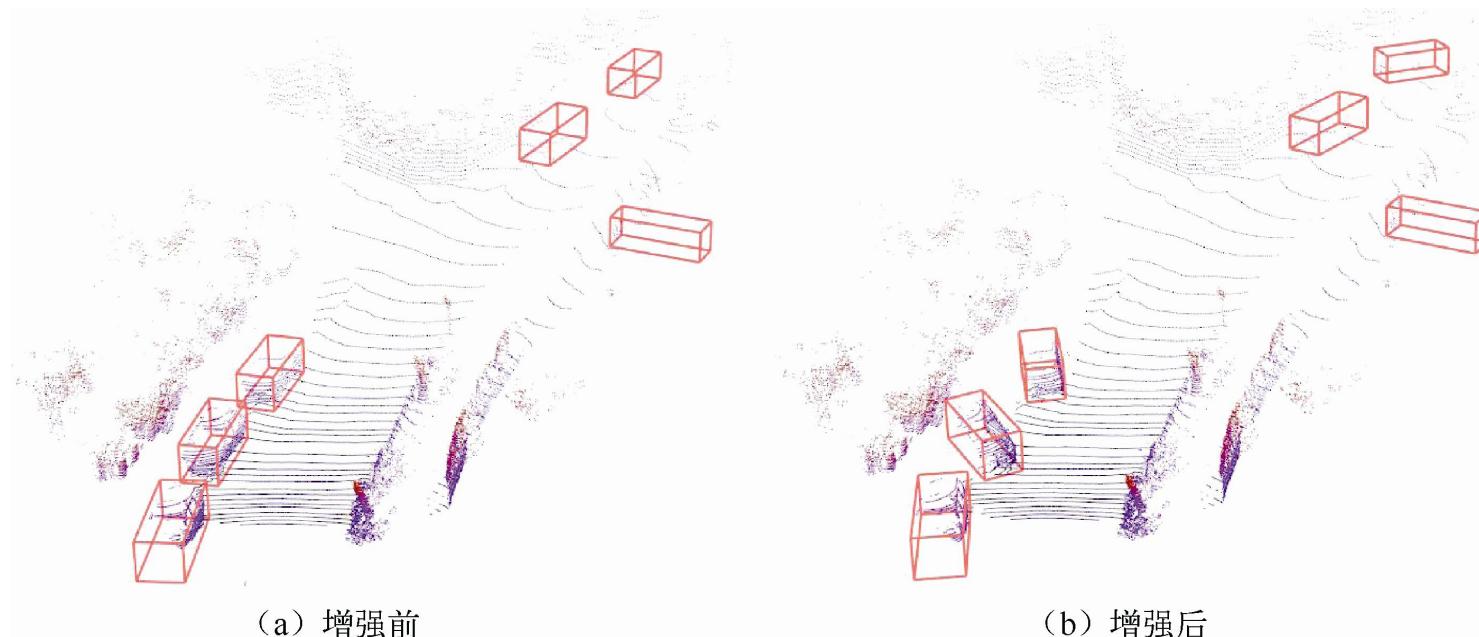
Mosaic

# 训练数据准备—数据增强



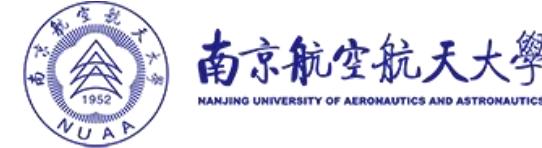
## 3. 点云数据增强

- 基于预定规则，通常包括旋转、平移、缩放、加噪、随机丢弃等方法
- 对于具有物体级别标签（对场景中的所有物体进行标注）的数据，可进行局部的数据增强



对包围框内的物体进行旋转和缩放

# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 训练数据准备—数据配比与课程设置



## 1. 数据配比

### □ 定义：

- 在大模型训练过程中设置不同类别和领域的数据样本的比例

### □ 为什么要进行数据配比：

- 不同类型的数据含有的信息量不同，合理的数据配比能够使得训练数据的信息含量最大化

-例如，对于文本数据而言，百科类文本信息密度大，普通的网页文本则包含一些长尾（出现频率较低）词汇且容易存在错误及冗余数据，通常信息密度较低

- 多样性的数据能够反映人类日常生活的整体概貌，合理的数据配比能够模拟真实的生活环境，帮助模型学习到符合人类习惯的知识

- 合理地混合不同难度的数据能够提升模型收敛速度及激发模型的潜能

-简单的数据样本，如日常交流用语构成的文本数据、仅包含单个物体的图像数据等，有助于模型在训练初期稳定收敛，而高难度的数据能够提升模型性能的上限

# 训练数据准备—数据配比与课程设置



## 1. 数据配比

### □ 如何设置数据配比：

- 在大模型训练的不同阶段，即预训练、二次预训练（如果说有的话）、微调，通常会设置不同的数据配比

- 预训练需要保证大模型能够学习到通用特征，具备一些通用功能，训练数据中的通用数据应该具有更高的比例

- 垂直大模型通常基于通用大模型进行二次开发，通过二次预训练给大模型注入领域知识，需要添加专业数据，但仍需让通用数据参与训练

- 在微调阶段，专业数据的比例可以适当提高

**无论是二次预训练阶段还是微调阶段，都需要混合通用数据，否则容易出现灾难性遗忘现象**

- 确定数据配比的策略：

- 使用不同的数据配比训练一系列小模型，选择令小模型达到最佳性能的数据配比来训练大模型

- 训练一个小型的代理模型来找到最佳的数据配比

# 训练数据准备—数据配比与课程设置



## 2. 课程设置

### □ 定义:

- 编排训练数据的使用顺序

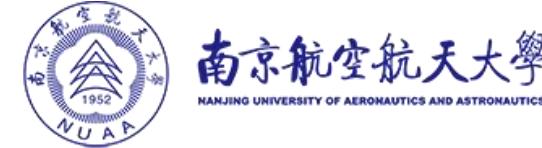
### □ 设置方式:

- 课程设置需要依据领域知识和实验经验

● 在训练初期可选择简单、易于理解的样本进行训练，有助于模型快速收敛，建立对任务的基本认知

- 随后可以不断增加样本难度，引入更多噪声，令模型适应复杂情况。

# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 训练数据准备—开源数据集



## 1. 开源文本数据集

### □ SQuAD:

- 用于机器阅读理解任务，从维基百科中选取了一系列文章段落，涉及历史、科学、人文等不同主题和领域。针对这些段落，创建了问题和与之相关的答案

---

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?  
**gravity**

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?  
**graupel**

Where do water droplets collide with ice crystals to form precipitation?  
**within a cloud**

---

# 训练数据准备—开源数据集



## 1. 开源文本数据集

### □ SQuAD:

- 用于机器阅读理解任务，从维基百科中选取了一系列文章段落，涉及历史、科学、人文等不同主题和领域。针对这些段落，创建了问题和与之相关的答案

### □ WMT:

- 用于机器翻译任务，包含源语言和目标语言的句子对，涵盖一些主流的国际用语：中英、英法等，以及一些小语种的翻译

Parallel Training Data:

File	EN-CS	EN-DE	EN-ES	EN-HI	EN-IS	EN-JA	EN-RU	EN-ZH	EN-UK	CS-UK
Europarl v10	✓	✓	✓		✓					
ParaCrawl v9	✓	✓				✓	✓	✓	✓	✓
Common Crawl corpus	✓	✓						✓		
News Commentary v18.1	✓	✓		✓		✓	✓	✓	✓	
Wiki Titles v3	✓	✓			✓	✓	✓	✓	✓	
Linguatools Wiki Titles			✓							
UN Parallel Corpus V1.0								✓	✓	
Tilde MODEL corpus	✓	✓	✓		✓		✓			✓

<https://www2.statmt.org/wmt24/mtdata/>

# 训练数据准备—开源数据集



## 1. 开源文本数据集

### □ SQuAD:

- 用于机器阅读理解任务，从维基百科中选取了一系列文章段落，涉及历史、科学、人文等不同主题和领域。针对这些段落，创建了问题和与之相关的答案

### □ WMT:

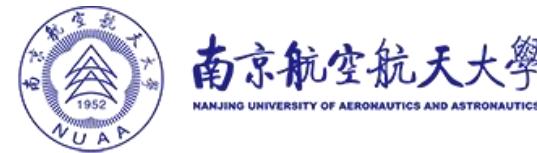
- 用于机器翻译任务，包含源语言和目标语言的句子对，涵盖一些主流的国际用语：中英、英法等，以及一些小语种的翻译

### □ THUCNews:

- 中文新闻文本分类数据集，包含大量的从互联网获取的新闻文本，涵盖政治、经济、体育、娱乐等多个主题。每篇新闻都被标记为一个或多个类别，使得该数据集适用于多标签分类任务

	label	content
0	体育	马晓旭意外受伤让国奥警惕 无奈大雨格外青睐殷家军记者傅亚雨沈阳报道 来到沈阳，国奥队依然没有...
1	体育	商瑞华首战复仇心切 中国玫瑰要用美国方式攻克瑞典多曼来了，瑞典来了，商瑞华首战求3分的信心也...
2	体育	冠军球队迎新欢乐派对 黄旭获大奖张军赢下PK赛新浪体育讯12月27日晚，“冠军高尔夫球队迎新...
3	体育	辽足签约危机引注册难关 高层威逼利诱合同笑里藏刀新浪体育讯2月24日，辽足爆发了集体拒签风波...

# 训练数据准备—开源数据集



## 1. 开源文本数据集

### □ SQuAD:

- 用于机器阅读理解任务，从维基百科中选取了一系列文章段落，涉及历史、科学、人文等不同主题和领域。针对这些段落，创建了问题和与之相关的答案

### □ WMT:

- 用于机器翻译任务，包含源语言和目标语言的句子对，涵盖一些主流的国际用语：中英、英法等，以及一些小语种的翻译

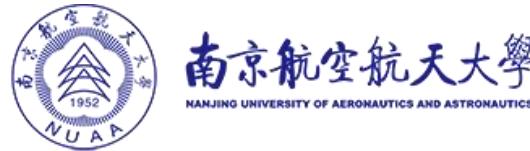
### □ THUCNews:

- 中文新闻文本分类数据集，包含大量的从互联网获取的新闻文本，涵盖政治、经济、体育、娱乐等多个主题。每篇新闻都被标记为一个或多个类别，使得该数据集适用于多标签分类任务

### □ WikiText:

- 用于语言建模任务的英文文本数据集，从维基百科的优质文章和标杆文章中提取得到，可用于评估语言模型的生成能力、上下文理解和长序列建模等方面的能力

# 训练数据准备—开源数据集



## 2. 开源图像数据集

## MNIST:

- 手写数字识别数据集，包含大约 70000幅28像素×28像素的手写数字图像，这些数字包括0到9

# 训练数据准备—开源数据集



## 2. 开源图像数据集

### □ MNIST:

- 手写数字识别数据集，包含大约 70000幅28像素×28像素的手写数字图像，这些数字包括0到9

### □ ImageNet:

- 大规模图像数据集，包含来自各种场景的上千万幅图像，具有两万多个类别



# 训练数据准备—开源数据集



## 2. 开源图像数据集

### □ MNIST:

- 手写数字识别数据集，包含大约 70000幅28像素×28像素的手写数字图像，这些数字包括0到9

### □ ImageNet:

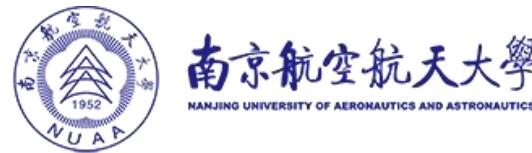
- 大规模图像数据集，包含来自各种场景的上千万幅图像，具有两万多个类别

### □ COCO:

- 用于多种计算机视觉任务的大型图像数据集，为目标检测、实例分割和关键点检测等任务提供了丰富且多样的标注



# 训练数据准备—开源数据集



## 2. 开源图像数据集

### □ MNIST:

- 手写数字识别数据集，包含大约 70000 幅 28 像素 × 28 像素的手写数字图像。这些数字包括 0 到 9



### □ ImageNet:

- 大规模图像数据集，包含来自各种场景的上千万幅图像，具有两万多个类别



### □ COCO:

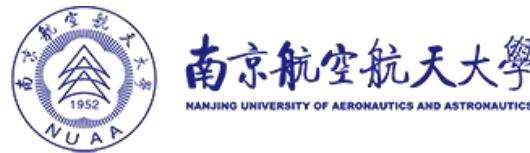
- 用于多种计算机视觉任务的大型图像数据集，为目标检测、实例分割和关键点检测等任务提供了丰富且多样的标注



### □ SA-1B:

- 图像注释数据集，包含约 11M 个具有多样性、高分辨率的图像，以及约 1.1B 个高质量的分割掩码

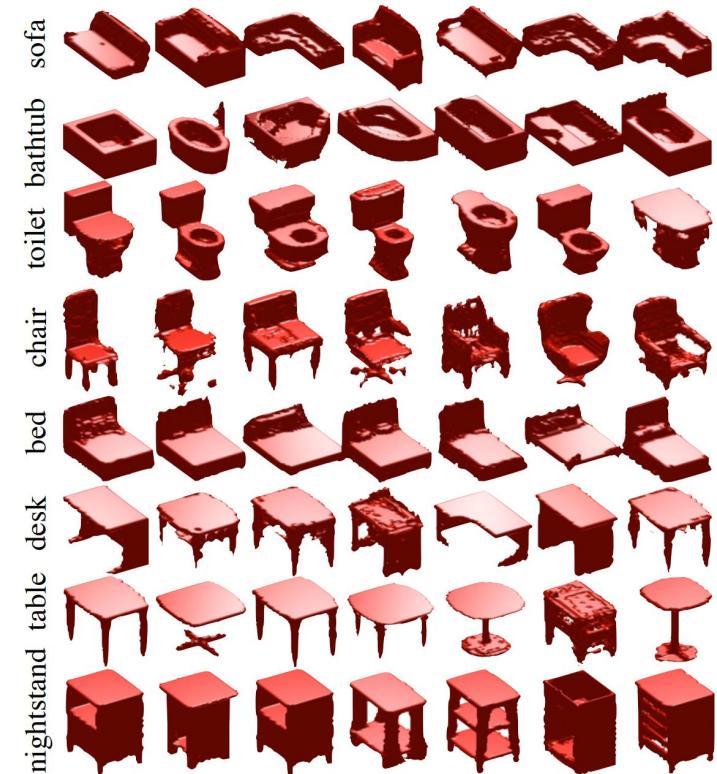
# 训练数据准备—开源数据集



## 3. 开源三维数据集

### □ ModelNet:

- 包含来自40个不同的物体类别，如椅子、桌子、汽车、飞机等的三维数据



# 训练数据准备—开源数据集



## 3. 开源三维数据集

### □ ModelNet:

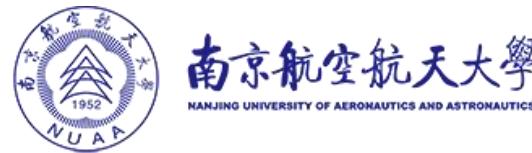
- 包含来自40个不同的物体类别，如椅子、桌子、汽车、飞机等的三维数据

### □ ShapeNet:

- ShapeNet数据集提供了丰富的标注信息，包括每个模型的类别、局部区域类别标签，以及与每个模型相关的2D图像



# 训练数据准备—开源数据集



## 3. 开源三维数据集

### □ ModelNet:

- 包含来自40个不同的物体类别，如椅子、桌子、汽车、飞机等的三维模型

### □ ShapeNet:

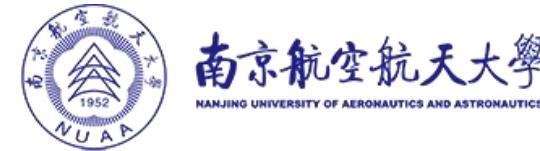
- ShapeNet数据集提供了丰富的标注信息，包括每个模型的类别、局部标签，以及与每个模型相关的2D图像

### □ ScanNet:

- 大规模室内场景三维扫描数据集，包含从多种室内场景中采集的三维扫描数据，包括住宅、办公室、商店等，该数据集提供了多种类型的数据，包括RGB图像、深度图像和三维点云



# 训练数据准备—开源数据集



## 3. 开源三维数据集

### □ ModelNet:

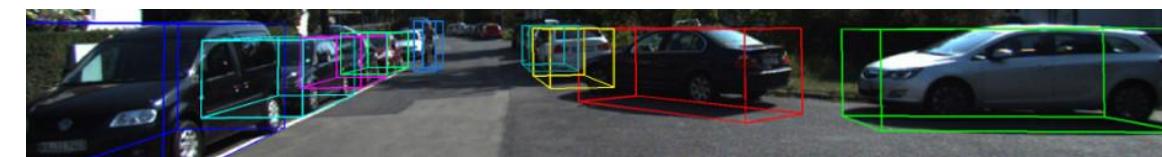
- 包含来自40个不同的物体类别，如椅子、桌子、汽车、飞机等的三维数据

### □ ShapeNet:

- ShapeNet数据集提供了丰富的标注信息，包括每个模型的类别、局部区域类别标签，以及与每个模型相关的2D图像

### □ ScanNet:

- 大规模室内场景三维扫描数据集，包含从多种室内场景中采集的三维扫描数据，包括住宅、办公室、商店等，该数据集提供了多种类型的数据，包括RGB图像、深度图像和三维点云



### □ KITTI :

- 用于自动驾驶和计算机视觉研究的开源数据集，数据集创建团队使用了一辆搭载了激光雷达、相机、惯性测量单元等多种传感器的车辆进行数据采集

# 目录

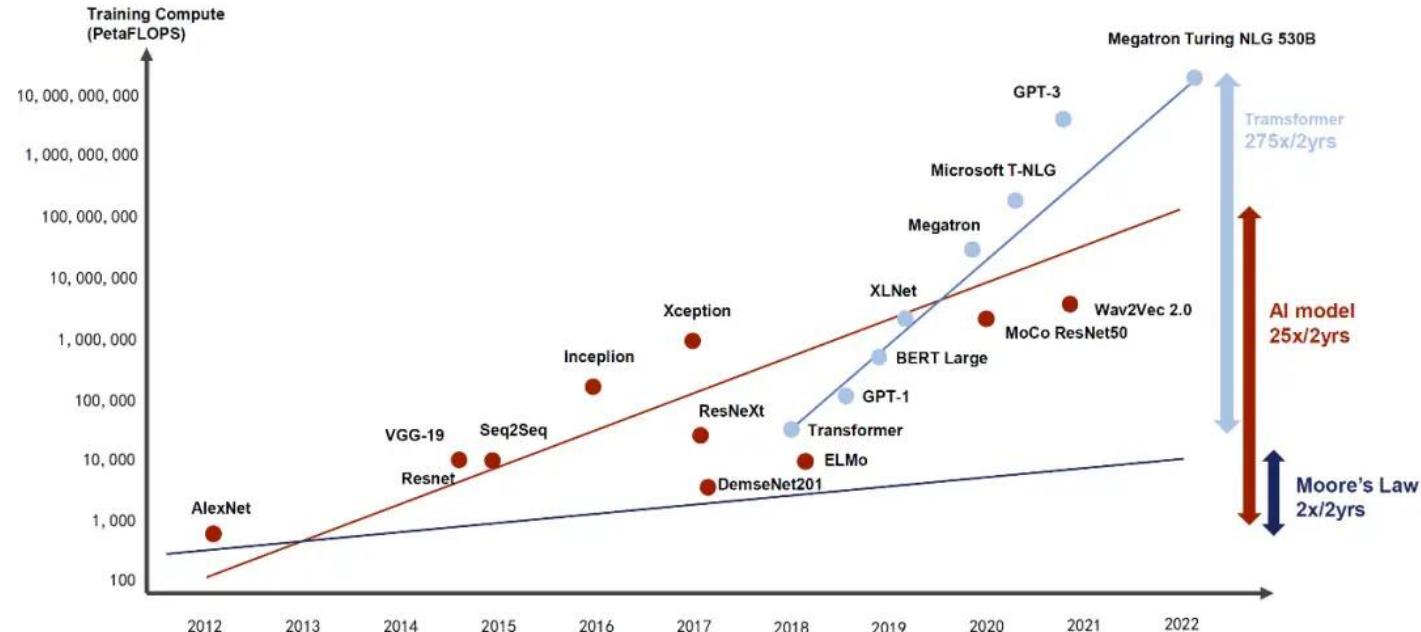


- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

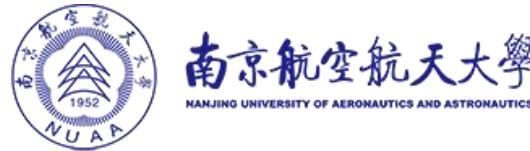
# 并行化与分布式训练——大模型训练的挑战



大模型时代算力需快速增长，增速快于摩尔定律



# 并行化与分布式训练——大模型训练的挑战



大模型时代算力需快速增长，增速快于摩尔定律

GPT-3：1750亿参数

用一块A100进行训练，按照半精度峰值计算性能，需要32年时间

80G显存无法训练GPT-3

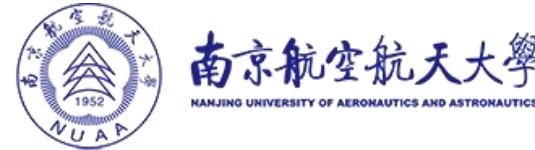
计算墙

显存墙

分布式训练

通信墙

# 并行化与分布式训练——大模型训练的挑战



- 分布式训练系统是由多个计算节点构成的网络，各个计算节点可由一台或多台主机构成
- 两个关键问题：
  - 如何有效地组织和划分训练数据和模型参数（并行化策略）
  - 如何在分布式环境中实现高效的数据通信，确保各个节点之间能够同步更新模型参数（节点间数据通信）

# 目录



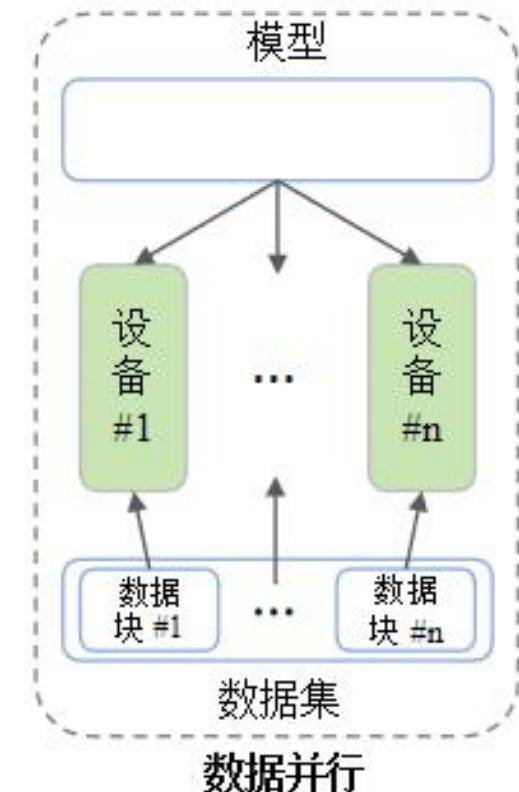
- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 并行化与分布式训练—并行化策略

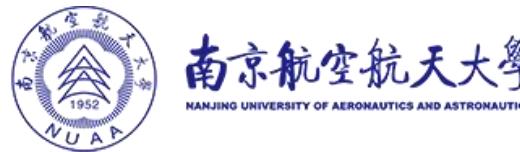
## 1. 数据并行

### □ 基本实现方式：

- 完整的模型被复制到每个计算节点上，对训练数据进行切分，并分发到不同的节点上进行计算
- 每个节点独立地使用被分配到的数据计算损失函数和梯度，并定期将梯度传递给特定节点进行聚合，获取全局梯度
- 利用全局梯度更新整个模型参数
- 可为基于样本的数据并行和基于样本维度的数据并行



# 并行化与分布式训练——并行化策略



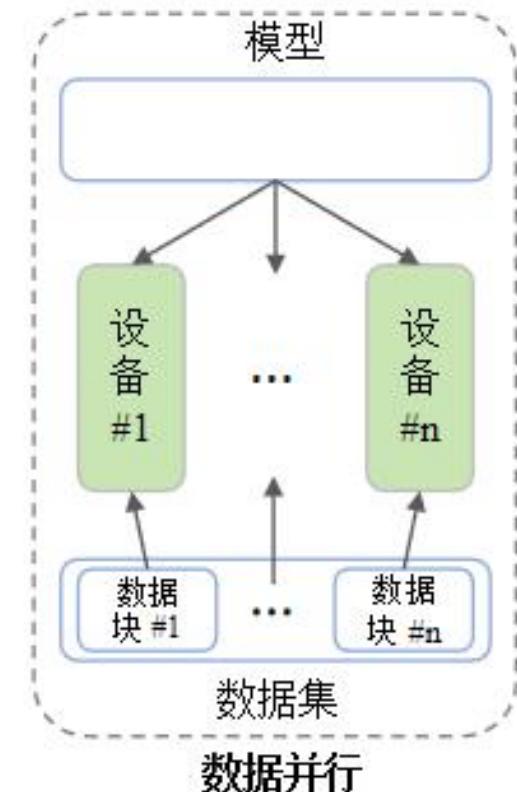
## 1. 数据并行

### □ 基于样本的数据并行：

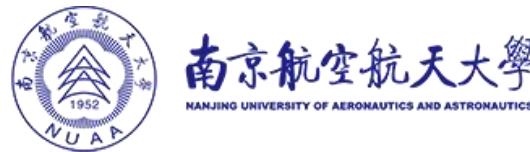
- 以样本为单位，按照一定的规则划分数据子集分配给各个节点
- 有放回地随机采样：  
假设当前有 $m$ 个数据样本， $n$ 个计算节点，执行 $m/n$ 次有放回的随机采样获得子集
- 局部（全局）置乱采样：  
将 $m$ 个样本打乱顺序并随机划分为 $n$ 个子集（不一定是等分），按照节点的计算能力进行分配

### □ 基于样本维度的数据并行：

- 对每个样本所具有的 $d$ 维的属性进行拆分，将某个属性维度分到对应计算节点



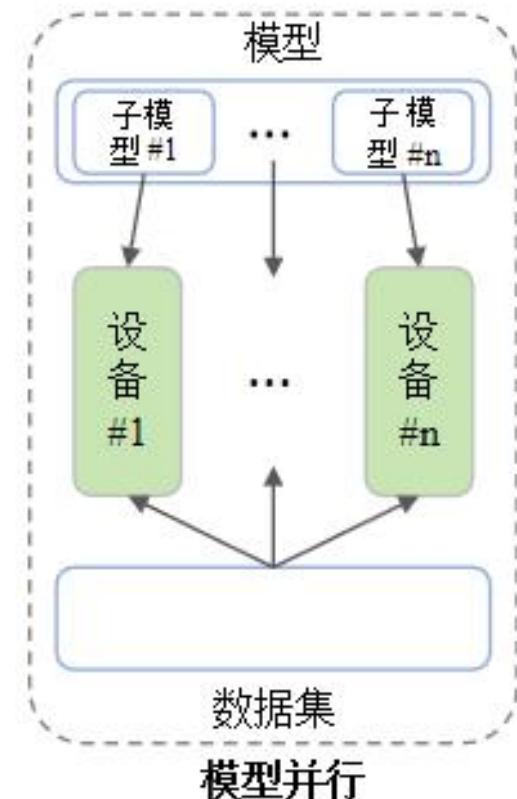
# 并行化与分布式训练—并行化策略



## 2. 模型并行

### □ 基本实现方式:

- 将原始模型划分为若干子模型，随后分配给每个节点
- 每个节点负责执行该子模型的前向传播和反向传播
- 模型并行除了需要在不同设备之间传递梯度，还需要传递子模型产生的中间结果
- 对模型的划分通常有垂直划分（张量并行）和水平划分（流水线并行）两种方式



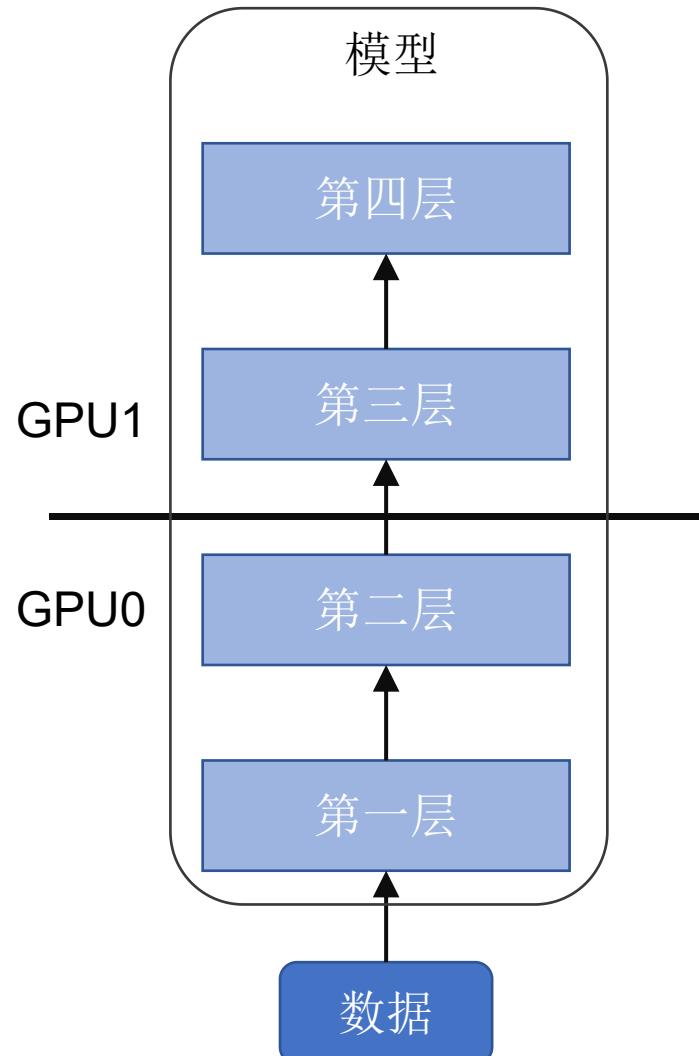
# 并行化与分布式训练—并行化策略



## 2. 模型并行

### □ 流水线并行：

- 对模型按层进行拆分
- 当使用朴素的流水线并行时，**中间层需要等待前一层的所有数据处理完成后才能开始计算**，大部分的计算节点都处于空闲状态
- 为了解决上述问题，目前使用最广泛的是微批次流水线并行：
  - 将传入的小批次（minibatch）分块为微批次（microbatch），并人为创建流水线来解决GPU空闲问题
  - 主流的流水线并行方法GPipe和PipeDream都采用微批次流水线并行方案
  - 基本思想：利用数据并行的思想，将minibatch的数据再进行划分，例如，假设共有有4个minibatch，将每个minibatch继续划分为4个microbatch，记为 $B_{i,j}$ ，第一个下标是第minibatch的索引，第二个是microbatch的索引。  
先将 $B_{0,0}$ 送入GPU0中，处理完后可送入GPU1，此时GPU0空闲，可处理 $B_{0,1}$ ，待GPU1处理完 $B_{0,0}$ ，可以立即处理 $B_{0,1}$



# 并行化与分布式训练—并行化策略



## 2. 模型并行

### □ 张量并行:

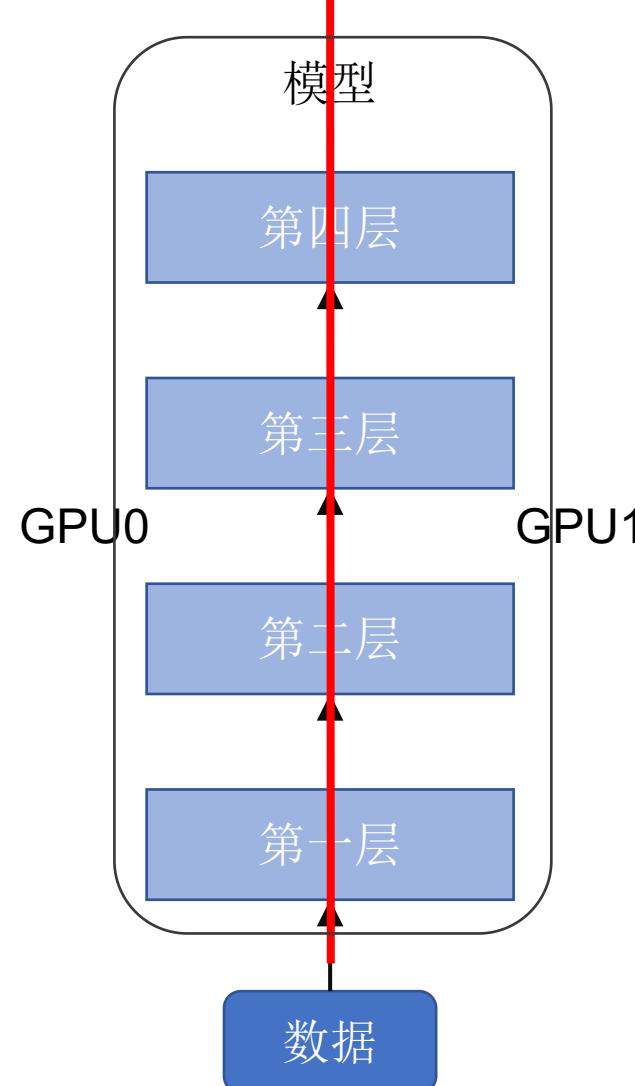
- 在同一层中，将层的参数张量划分为多个子张量
- 以矩阵乘法为例，神经网络中存在着大量的参数乘法，张量并行主要利用了矩阵的分块乘法的概念，将参数矩阵进行切分，随后部署到各个计算节点

假设有一个输入矩阵 $X$ （大小为 $m \times n$ ）、权重矩阵 $W$ （大小为 $n \times p$ ）、和偏置向量 $b$ 。计算过程可以表示为 $Y = X \cdot W + b$

将 $W$ 分为 $q$ 个子矩阵（如列分块）： $W = W_1, W_2, \dots, W_q$ ，每个 $W_i$ 是 $n \times (p/q)$ 大小的矩阵。随后可以将权重分配给各个节点，并行计算输出 $Y_i$ :

$$Y_i = X \cdot W_i + b$$

将所有的 $Y_i$ 汇总以获得最终的输出矩阵 $Y$

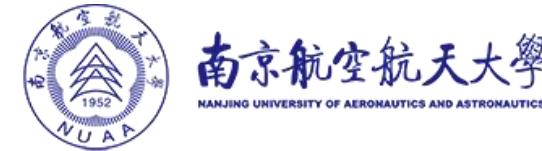


# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 并行化与分布式训练—节点间数据通信



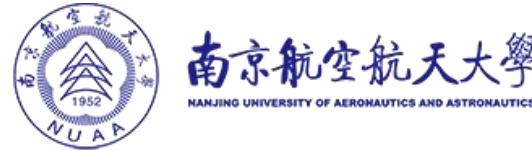
- 对于数据并行，需要在节点之间传递模型参数和梯度
- 对于模型并行则需要传递各个计算节点的中间结果（包括前向推理和反向传播）
- 在训练时，计算节点之间需要频繁地进行通信来传输大量的数据



需设计合理的通信机制来有效地管理和优化节点间的信息传递

- 节点间的通信机制主要涉及：
  - 计算节点的拓扑
  - 通信原语
  - 同步方式

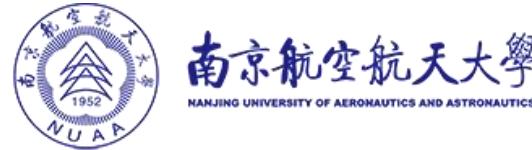
# 并行化与分布式训练—节点间数据通信



## 1. 计算节点的拓扑

- 指各个节点之间的连接方式，通常包含物理拓扑和逻辑拓扑两种概念
- **物理拓扑：**
  - 描述了计算节点在硬件层面上的连接方式和布局，涉及对服务器、交换机等设备的连接和部署
- **逻辑拓扑：**
  - 在分布式系统中，计算节点之间抽象的连接方式，其更关注节点间的通信模式、数据流和任务分配
  - 可分为中心化架构和去中心化架构

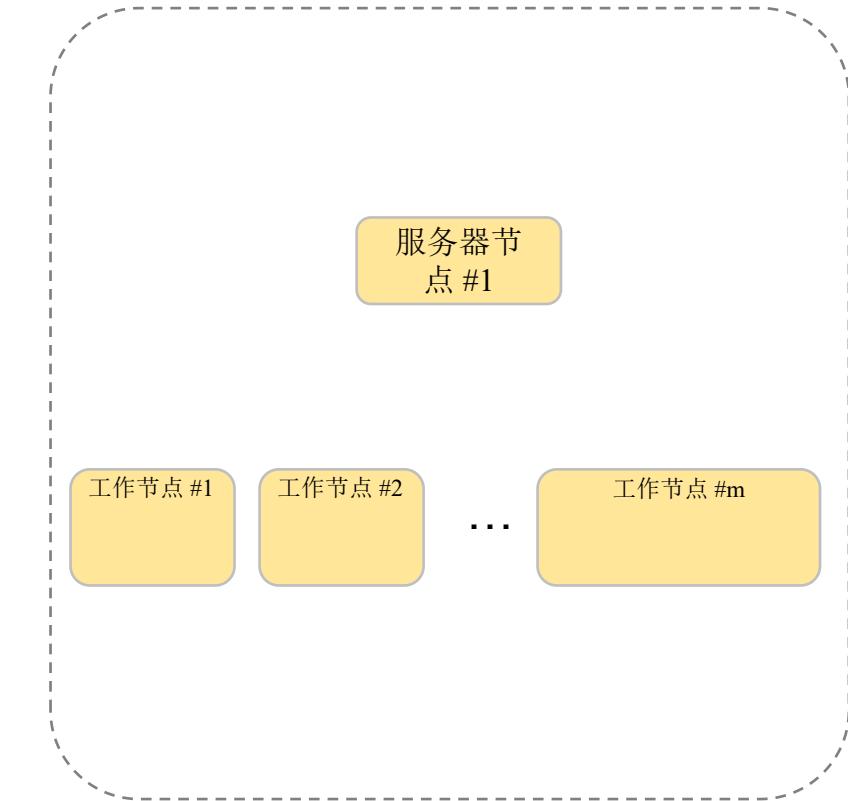
# 并行化与分布式训练—节点间数据通信



## 1. 计算节点的拓扑

- 中心化架构：
  - 存在若干中心节点负责协调和管理整个分布式训练过程
  - 典型代表：参数服务器架构
- 参数服务器架构：

- 计算节点被分为两类：服务器 (server) 节点和工作 (worker) 节点



# 并行化与分布式训练—节点间数据通信

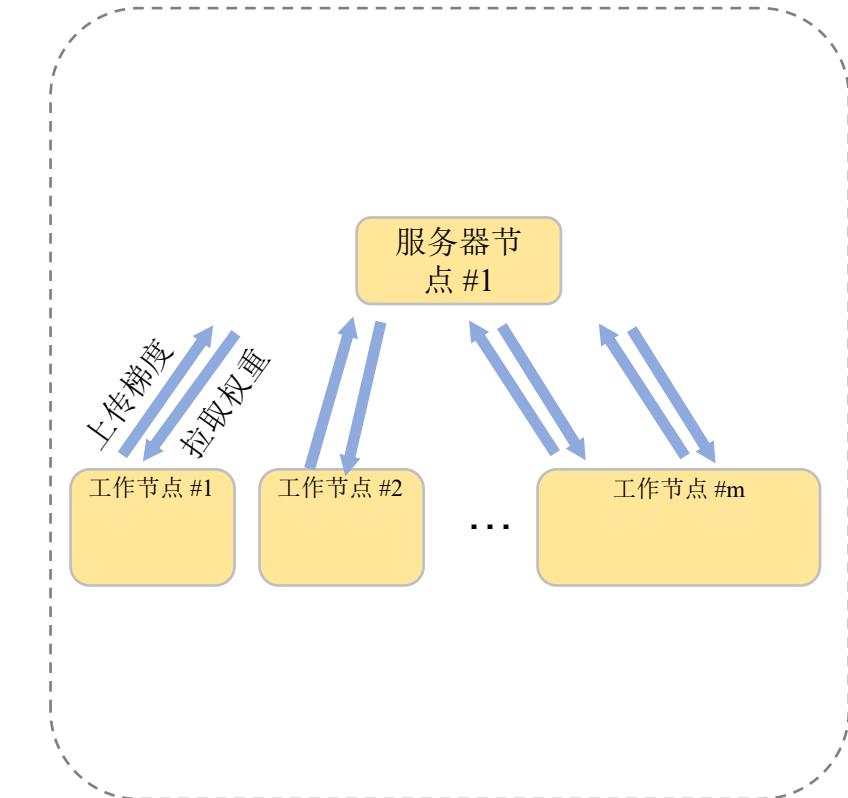


## 1. 计算节点的拓扑

- 中心化架构：
  - 存在若干中心节点负责协调和管理整个分布式训练过程
  - 典型代表：参数服务器架构

- 参数服务器架构：

- 计算节点被分为两类：服务器 (server) 节点和工作 (worker) 节点
  - 参数服务器节点负责存储和管理模型的参数，工作节点负责计算模型的局部梯度



# 并行化与分布式训练—节点间数据通信



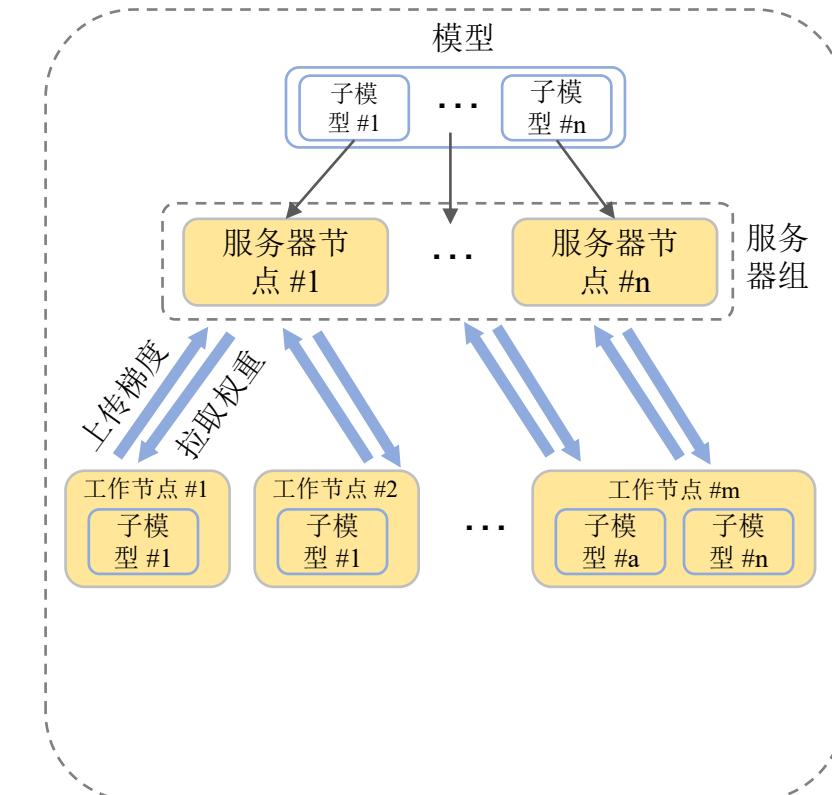
## 1. 计算节点的拓扑

### □ 中心化架构:

- 存在若干中心节点负责协调和管理整个分布式训练过程
- 典型代表: 参数服务器架构

### □ 参数服务器架构:

- 计算节点被分为两类: 服务器 (server) 节点和工作 (worker) 节点
  - 参数服务器节点负责存储和管理模型的参数, 工作节点负责计算模型的局部梯度
- 当使用模型并行时, 参数服务器架构会包含一个服务器组 (Server Group), 其内部包含多个服务器节点, 每个服务器节点维护一部分模型参数
  - 每个服务器节点所负责的模型参数可能被多个工作节点使用, 同时每个工作节点也可能使用多个服务器节点所负责的参数。



# 并行化与分布式训练—节点间数据通信



## 1. 计算节点的拓扑

### □ 中心化架构:

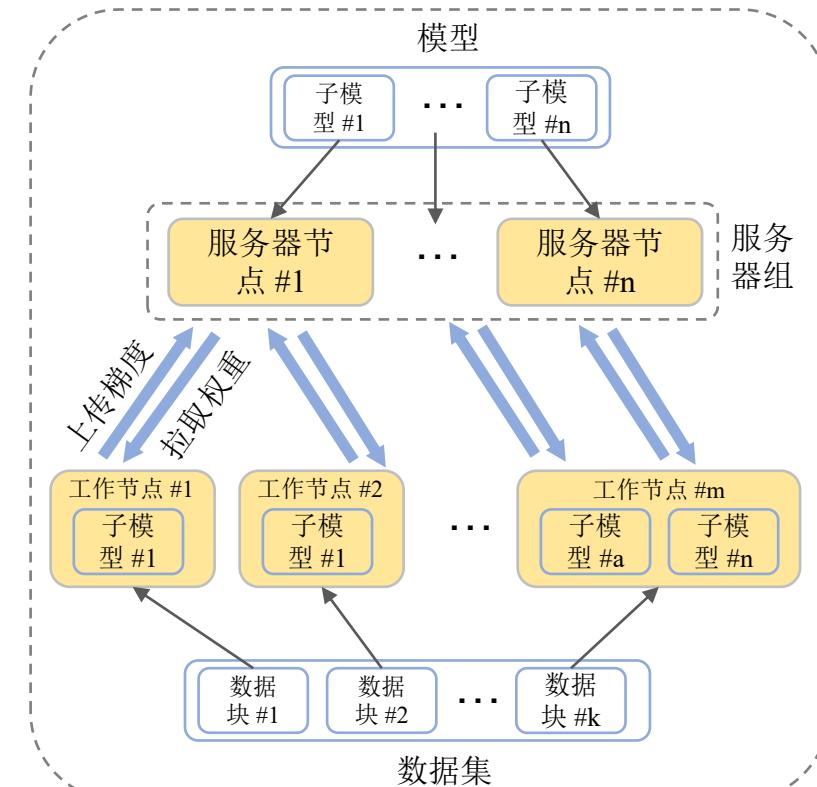
- 存在若干中心节点负责协调和管理整个分布式训练过程
- 典型代表: 参数服务器架构

### □ 参数服务器架构:

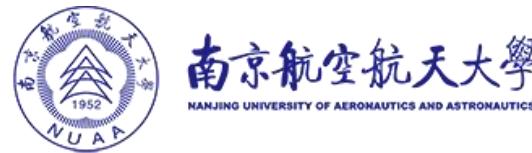
- 计算节点被分为两类: 服务器 (server) 节点和工作 (worker) 节点
  - 参数服务器节点负责存储和管理模型的参数, 工作节点负责计算模型的局部梯度
- 当使用模型并行时, 参数服务器架构会包含一个服务器组 (Server Group), 其内部包含多个服务器节点, 每个服务器节点维护一部分模型参数
  - 每个服务器节点所负责的模型参数可能被多个工作节点使用, 同时每个工作节点也可能使用多个服务器节点所负责的参数。
- 若使用数据并行策略, 可以将数据集划分后分配给各个工作节点。

右图:

1. 服务器组包含n个服务器节点, 用来管理原始模型的n个子模型。
2. 对于一号服务器节点, 其所管理的一号子模型被一号和二号工作节点使用, 因此一号服务器节点需要管理两个工作节点。
3. 第m个工作节点需要使用第a个和第n个子模型, 因此其被第a个和第 n 个服务器节点所管理。
4. 在该例子中, 数据集也被划分为k份, 分配给k个工作节点。



# 并行化与分布式训练—节点间数据通信



## 1. 计算节点的拓扑

### □ 参数服务器架构总流程：

- 初始化：服务器节点初始化模型参数，并将参数分发给所有工作节点，同时分发数据到每个工作节点

- 每个工作节点并行训练

### □ 工作节点流程：

- 初始化：载入分配的训练数据，从服务器节点拉取初始模型参数

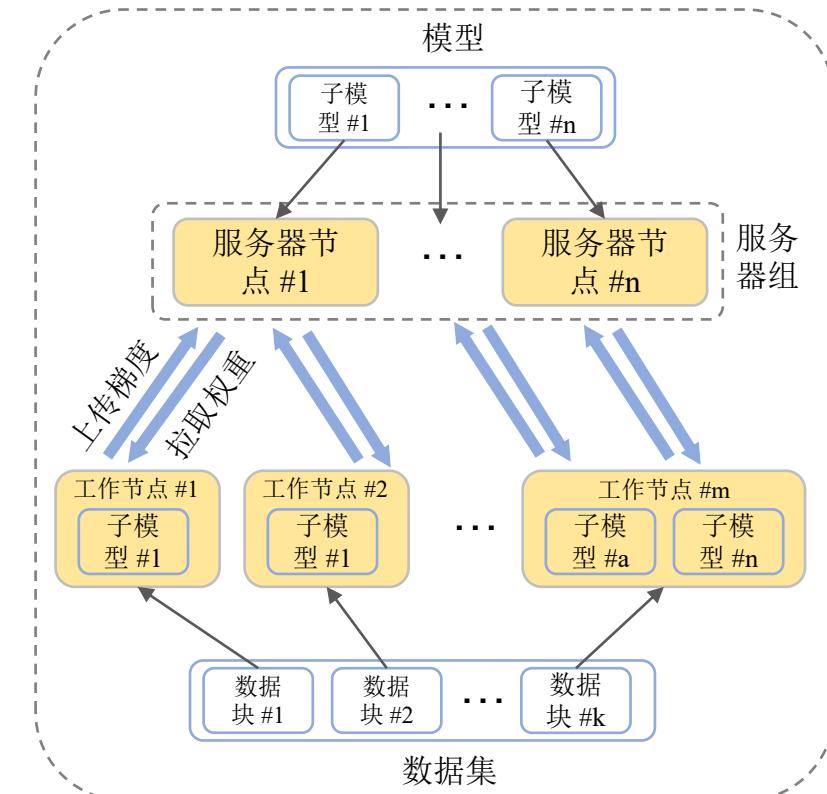
#### ● 迭代训练：

- 利用本节点的数据计算梯度
- 将局部梯上传到服务器节点
- 从服务器节点中拉取新的模型参数

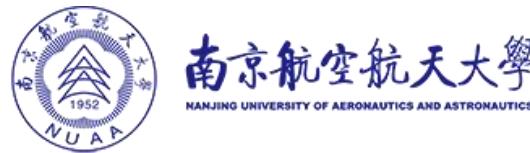
### □ 服务器点流程：

- 汇总来自工作节点的梯度

- 利用汇总的梯度更新模型参数



# 并行化与分布式训练—节点间数据通信

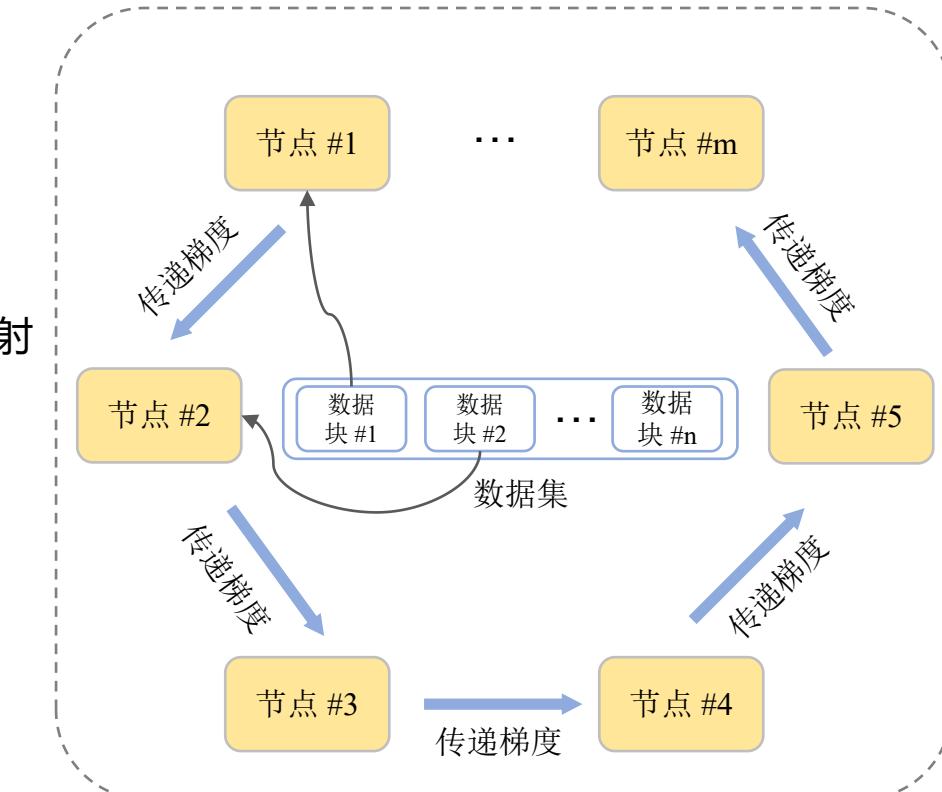


## 1. 计算节点的拓扑

- **去中心化架构:**
  - 没有中心节点，所有计算节点直接进行通信和协作
  - 典型代表: Ring Allreduce、Gossip、Ring-reduce

- **在Ring Allreduce架构下实现数据并行:**

- 每个节点从它的右邻居接收梯度数据，向左邻居发送梯度数据，通过散射  
规约和全局联集两个通信原语完成通信



# 并行化与分布式训练—节点间数据通信



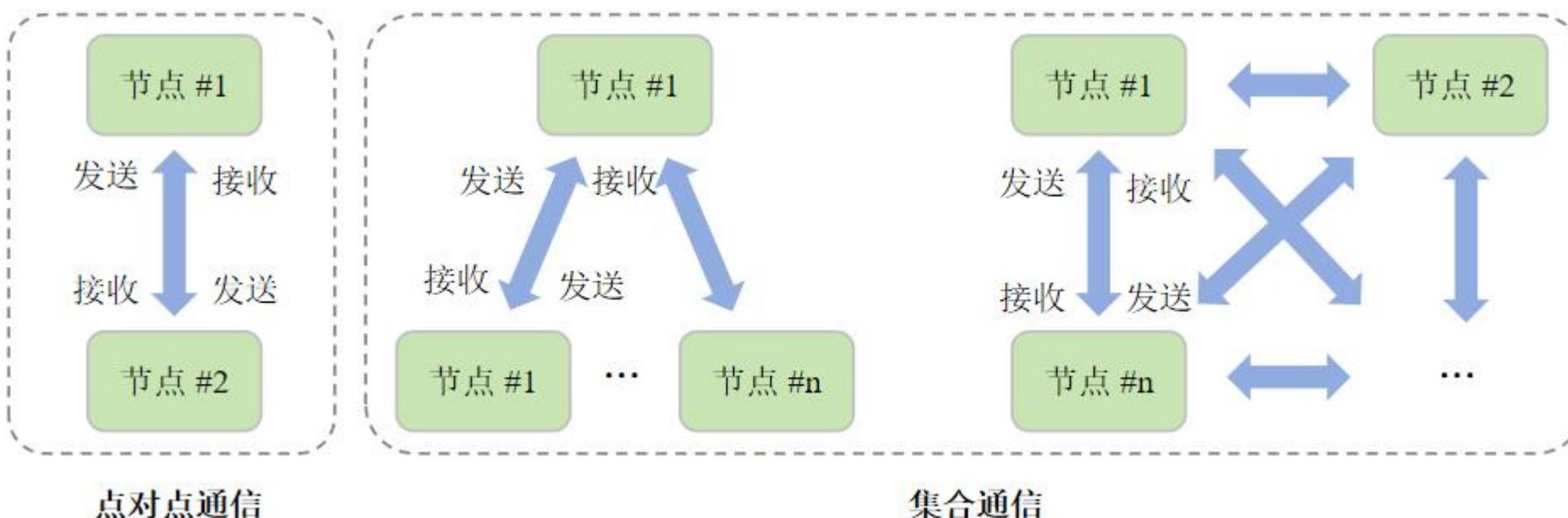
## 2. 通信原语

### □ 基本通信方式：

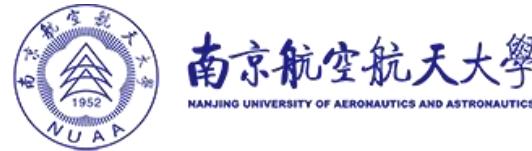
- 点对点通信 (P2P) 和集合通信 (CC)

- 点对点通信是两个节点之间进行通信，只有一个发送者和接收者（效率较低）

- 集合通信则是多个节点进行一对多或者多对多通信，有多个发送者和接收者

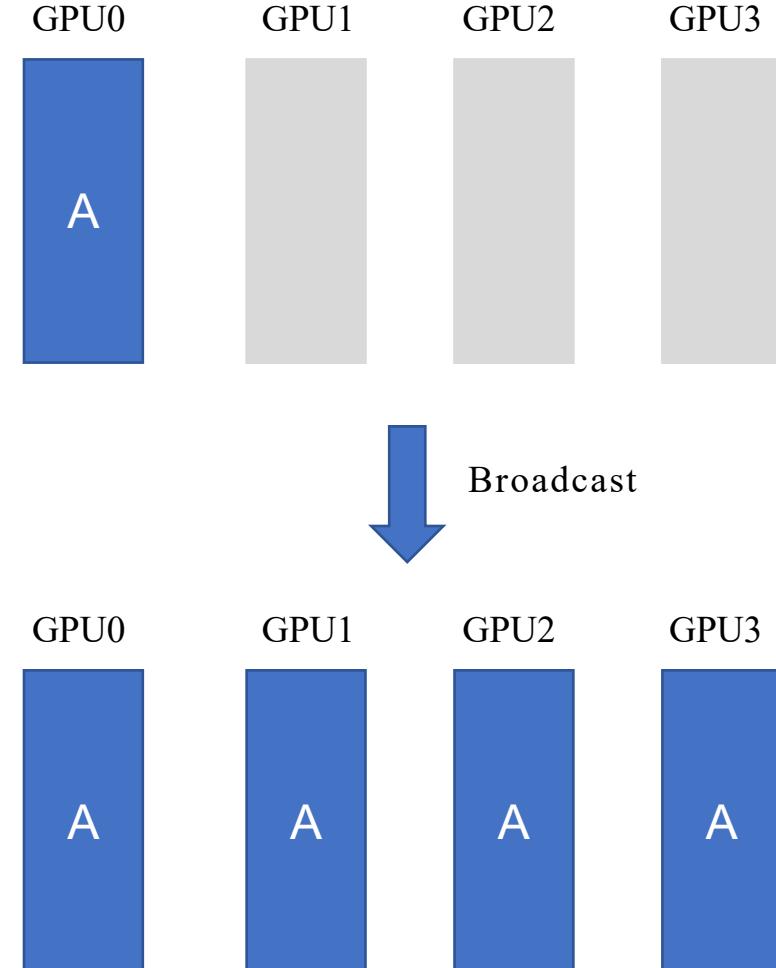


# 并行化与分布式训练—节点间数据通信

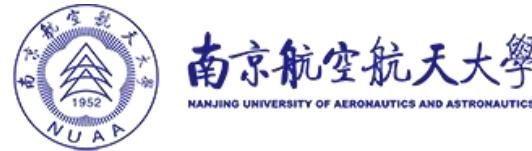


## 2. 通信原语：实现节点间通信的基本操作

- **发送 (Send) 和接收 (Receive) :**
  - 指节点之间消息的发送和接收，是最基础的通信原语
  
- **广播 (Broadcast) :**
  - 将某个节点的数据广播到其他节点上，属于一对多通信

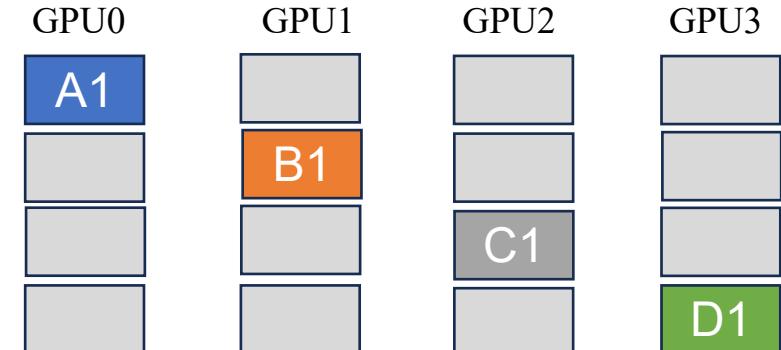
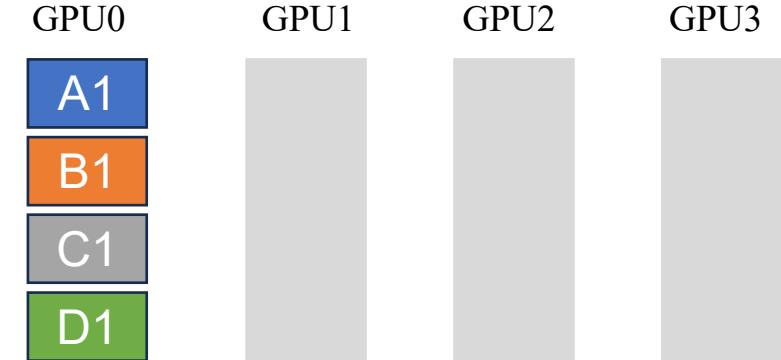


# 并行化与分布式训练—节点间数据通信

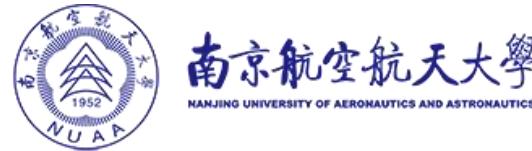


## 2. 通信原语：实现节点间通信的基本操作

- **发送 (Send) 和接收 (Receive) :**
  - 指节点之间消息的发送和接收，是最基础的通信原语
- **广播 (Broadcast) :**
  - 将某个节点的数据广播到其他节点上，属于一对多通信
- **散射 (Scatter) :**
  - 将数据划分之后再分发给指定节点，属于一对多通信

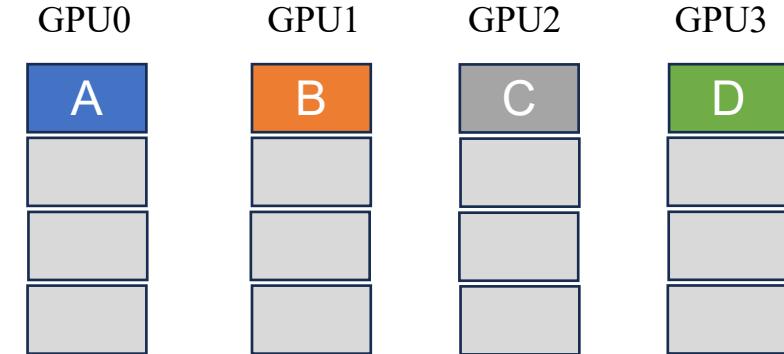


# 并行化与分布式训练—节点间数据通信

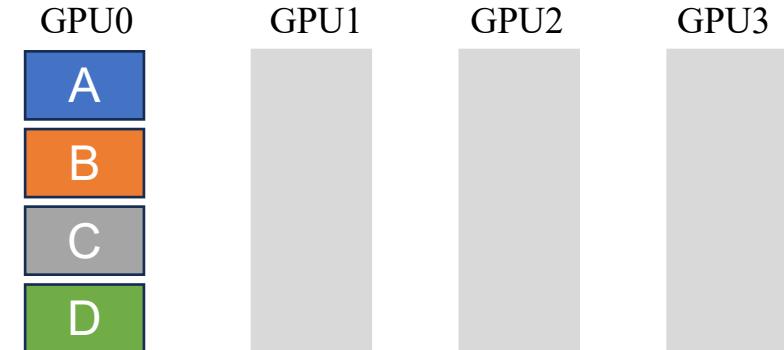


## 2. 通信原语：实现节点间通信的基本操作

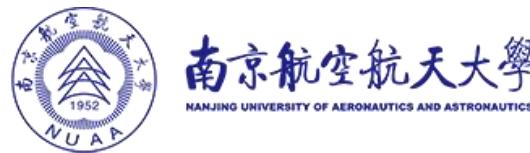
- **发送 (Send) 和接收 (Receive) :**
  - 指节点之间消息的发送和接收，是最基础的通信原语
- **广播 (Broadcast) :**
  - 将某个节点的数据广播到其他节点上，属于一对多通信
- **散射 (Scatter) :**
  - 将数据划分之后再分发给指定节点，属于一对多通信
- **聚集 (Gather) :**
  - 将多个节点的数据收集到一个节点上，属于多对一通信



Gather  
A large blue downward-pointing arrow is positioned between the two rows of GPUs, indicating the direction of data movement during the Gather operation.

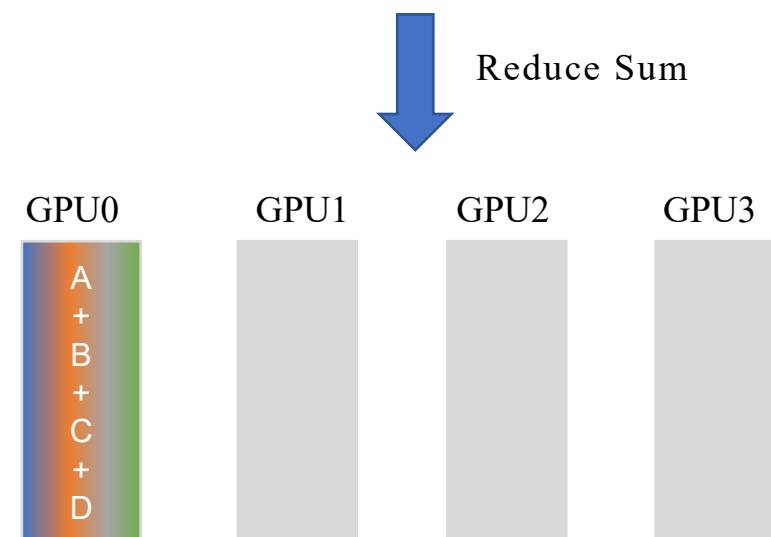
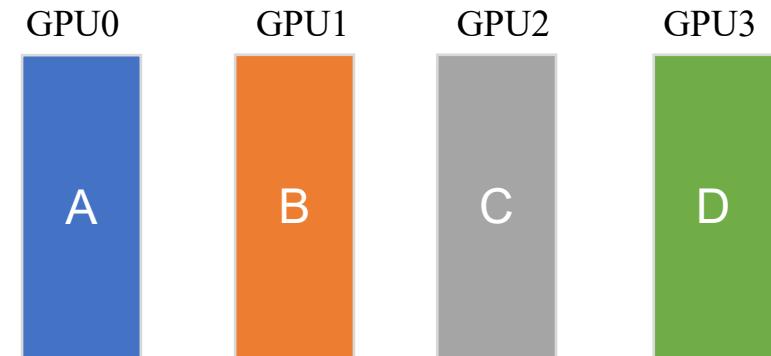


# 并行化与分布式训练—节点间数据通信

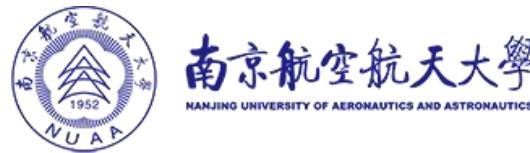


## 2. 通信原语：实现节点间通信的基本操作

- **发送 (Send) 和接收 (Receive) :**
  - 指节点之间消息的发送和接收，是最基础的通信原语
- **广播 (Broadcast) :**
  - 将某个节点的数据广播到其他节点上，属于一对多通信
- **散射 (Scatter) :**
  - 将数据划分之后再分发给指定节点，属于一对多通信
- **聚集 (Gather) :**
  - 将多个节点的数据收集到一个节点上，属于多对一通信
- **规约 (Reduce) :**
  - 把多个节点的数据规约运算到某个节点上，属于多对一通信
    - 常见的规约运算操作有求和 (SUM)、乘法 (PROD)、最小值 (MIN)、最大值 (MAX)、逻辑或 (LOR)、逻辑与 (LAND)、逻辑异或 (LXOR)、按位或 (BOR)、按位与 (BAND)、按位异或 (BOXR) 等



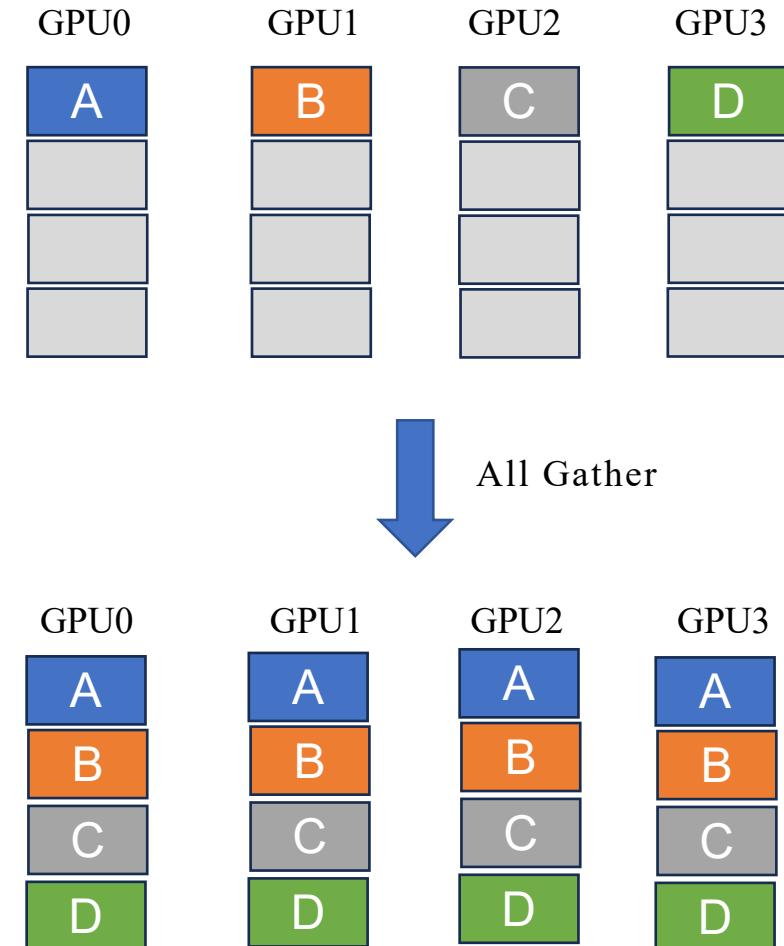
# 并行化与分布式训练—节点间数据通信



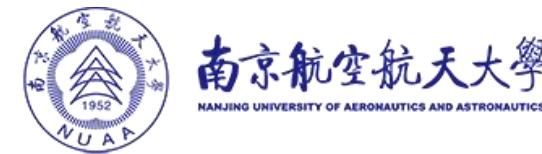
## 2. 通信原语：实现节点间通信的基本操作

### □ 全局聚集 (All Gather) :

- 收集所有节点的数据并分发到所有节点，其可以看作是先执行聚集操作，再执行广播操作，属于多对多通信



# 并行化与分布式训练—节点间数据通信



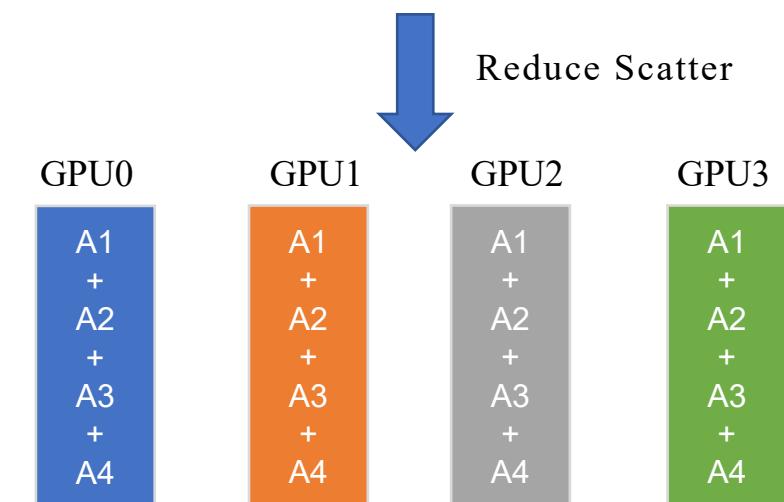
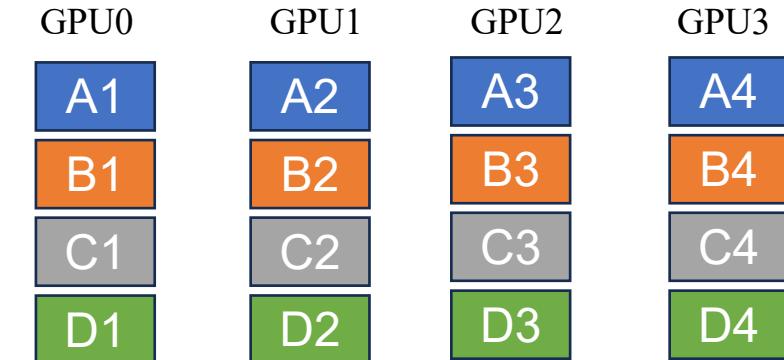
## 2. 通信原语：实现节点间通信的基本操作

### □ 全局聚集 (All Gather) :

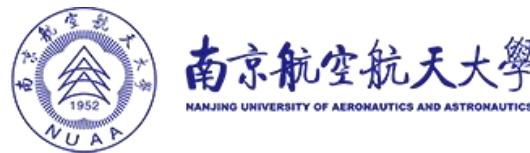
- 收集所有节点的数据并分发到所有节点，其可以看作是先执行聚集操作，再执行广播操作，属于多对多通信

### □ 散射规约 (Reduce Scatter) :

- 将所有节点的数据切分成若干块，每个节点对相同排序索引的数据块进行规约操作，属于多对多通信



# 并行化与分布式训练—节点间数据通信



## 2. 通信原语：实现节点间通信的基本操作

### □ 全局聚集 (All Gather) :

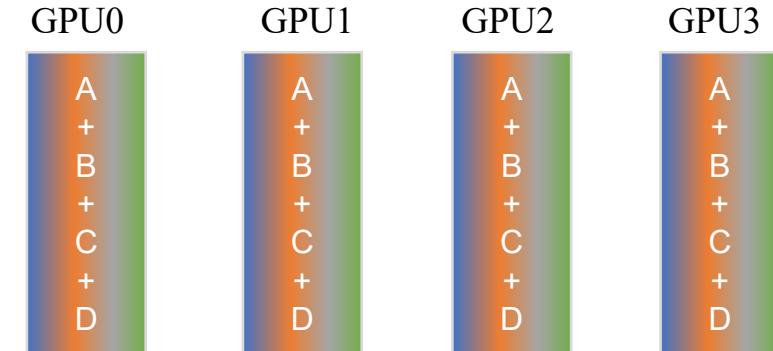
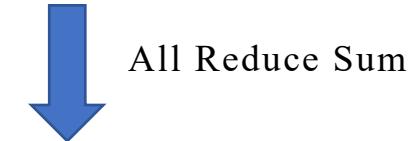
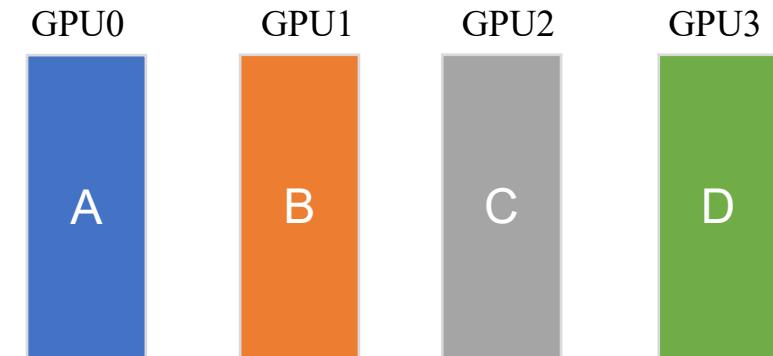
- 收集所有节点的数据并分发到所有节点，其可以看作是先执行聚集操作，再执行广播操作，属于多对多通信

### □ 散射规约 (Reduce Scatter) :

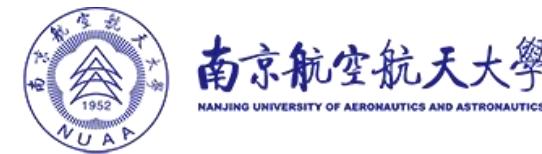
- 将所有节点的数据切分成若干块，每个节点对相同排序索引的数据块进行规约操作，属于多对多通信

### □ 全局规约 (All Reduce) :

- 将所有节点的数据进行规约，规约的结果将传递给所有节点，其可看作是先执行规约/散射规约操作，再执行广播/全局聚集操作，属于多对多通信



# 并行化与分布式训练—节点间数据通信



## 2. 通信原语：实现节点间通信的基本操作

### □ 全局聚集 (All Gather) :

- 收集所有节点的数据并分发到所有节点，其可以看作是先执行聚集操作，再执行广播操作，属于多对多通信

### □ 散射规约 (Reduce Scatter) :

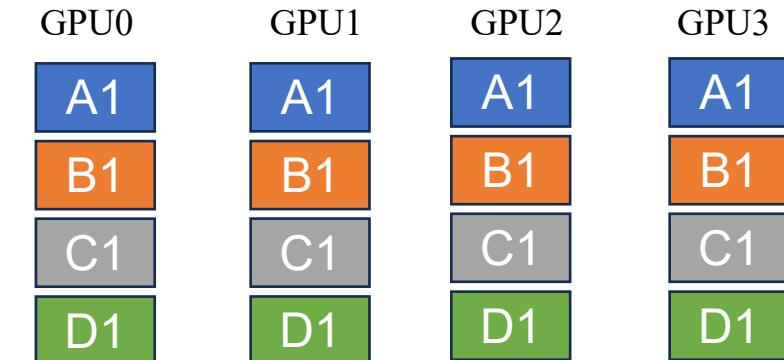
- 将所有节点的数据切分成若干块，每个节点对相同排序索引的数据块进行规约操作，属于多对多通信

### □ 全局规约 (All Reduce) :

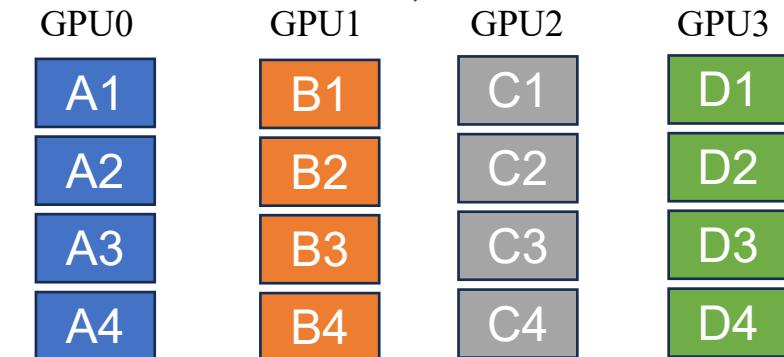
- 将所有节点的数据进行规约，规约的结果将传递给所有节点，其可看作是先执行规约/散射规约操作，再执行广播/全局聚集操作，属于多对多通信

### □ 全交换 (All to All) :

- 将所有节点的数据切分成若干块，每个节点获取相同排序索引的数据块



All to All

A large blue downward-pointing arrow is positioned between the initial state and the final state, indicating the direction of data exchange between all nodes.

# 并行化与分布式训练—节点间数据通信



南京航空航天大學  
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

## 3. 同步方式

### □ 目标：

- 维护不同节点上模型副本之间的一致性
- 现有的同步方式可分为同步算法和异步算法

### □ 同步算法：

- 各个节点在进行计算时需要等待其他所有节点完成计算，然后进行模型参数的更新
  - 参数服务器架构及去中心化架构都可使用同步算法进行通信

### □ 异步算法：

- 各个节点可以独立进行计算和参数更新，每个节点在上传梯度之后立即更新模型，而不需要等待其他节点

# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 并行化与分布式训练—分布式训练框架



## □ PyTorch—DP&DDP:

- PyTorch提供了分布式训练的相关模块，主要包括两种模式：Data Parallel (DP) 和Distributed Data Parallel (DDP)，均为数据并行
  - DP采用参数服务器架构，需要由主GPU来汇总梯度
  - DDP为每个GPU产生一个进程，采用全局规约模式进行通信

## □ TensorFlow-distribute.Strategy:

- 提供了tf.distribute. Strategy API，用于指定分布式训练策略，均为数据并行
- 目前所支持的策略有：
  - Mirrored Strategy**: 在每个GPU上都建立一个模型副本并使用全局规约进行节点间通信
  - MultiWorker Mirrored Strategy**与Mirrored Strategy的区别在于前者支持多机多卡训练
  - Parameter Server Strategy**采用参数服务器架构，支持多机多卡训练

# 并行化与分布式训练—分布式训练框架



## □ DeepSpeed:

- 专注于使用数据并行进行分布式训练
- 兼容多种深度学习框架，包括TensorFlow、PyTorch、MXNet等

## □ PaddlePaddle:

- 提供了多种并行策略的组合方式

    -对于十亿至百亿参数的模型，官方推荐使用分组参数切片的数据并行方式

    -对于更大规模的模型，提供了多维混合并行策略，融合了纯数据并行、分组参数切片的数据并行、张量并行、流水线并行、专家并行等多种并行策略

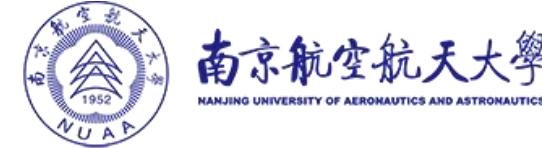
## □ Megatron:

- 专注于训练超大规模 Transformer模型，基于PyTorch实现，同时支持数据并行和模型并行

## □ DeepSpeed:

- 专注于提高深度学习模型在分布式训练环境中的性能，实现了零冗余优化器 (Zero Redundancy Optimizer, ZeRO) 支持的数据并行、张量并行和流水线并行

# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 模型压缩—量化



浮点数精度更高

模型训练时需要精确捕捉微小梯度变化

训练时使用浮点数

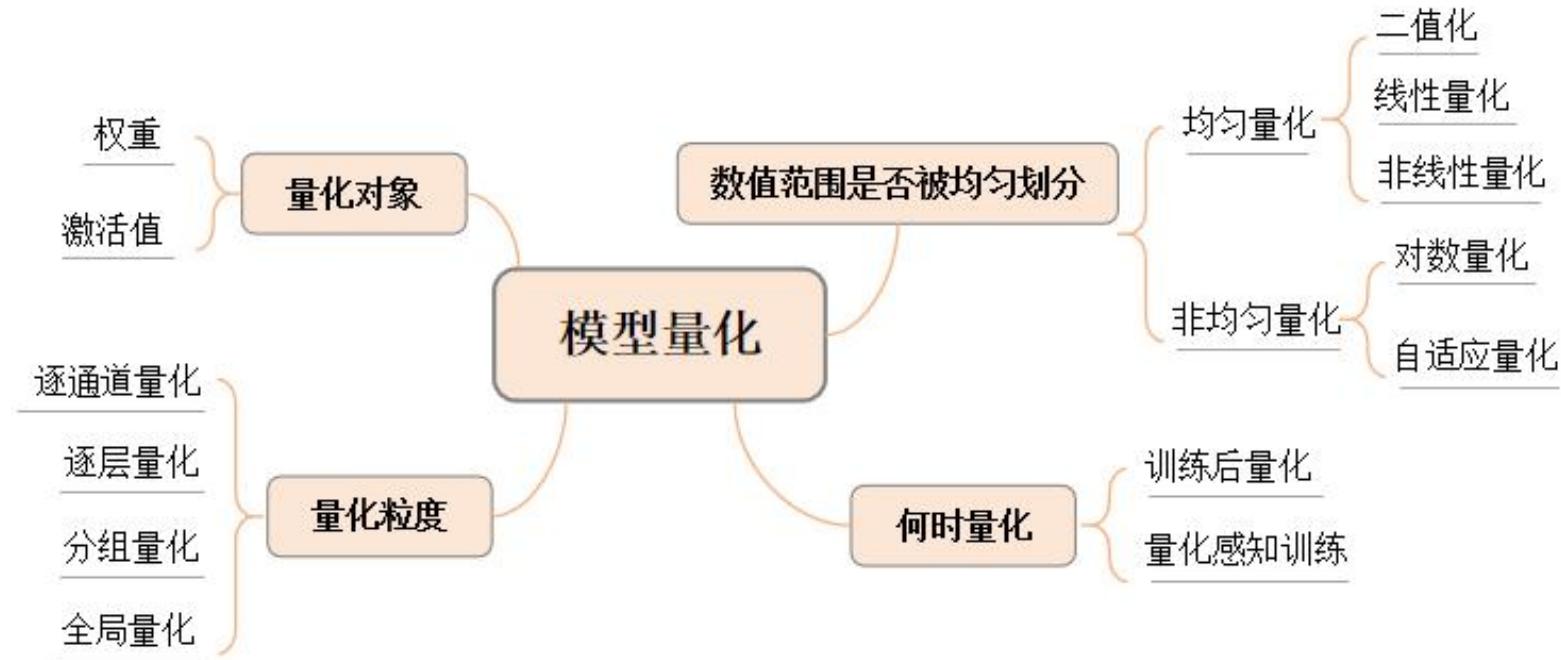
模型参数量大，推理速度慢

降低数值精度

模型性能下降

**模型量化的核心：在显著减小模型规模的同时，最大限度维持模型的性能水平**

# 模型压缩—量化



## 1. 浮点数与定点数

### □ 定点数：

- 使用固定的小数点位置来表示实数，其不包含指数部分
  - 在一个8位的计算机中，可规定前5位表示整数部分，后三位表示小数部分
    - 以十进制的20.125为例，其二进制表示中，整数部分20对应二进制的10100，小数部分0.125对应二进制的0.001，因此，整体可以使用10100001来表示20.125

### □ 浮点数：

- 采用科学计数法来表示实数，包括两个部分：尾数和指数
  - 例如，十进制的浮点数 1.23 可以表示为  $1.23 \times 10^0$ ，其中 1.23 是尾数，0 是指数
- 由于指数部分可以为负数、小数及整数，因此浮点数具有相比于定点数更广泛的表示范围
  - 使用 IEEE 754 标准表示，它将一个数分成三个部分：符号位、尾数（或称为有效数字）和指数

## 2. 量化基本原理

- 模型量化可以看作是一种建立定点数与浮点数之间映射关系的过程
  - 将数据的连续值转化为多个离散值，并以最小的代价保持模型原有的性能
  - 根据是否对连续的数值范围进行均匀的区间划分，量化可分为均匀量化和非均匀量化两种
- 均匀量化：
  - 将数值范围均匀地划分为相等的间隔，每个间隔映射到一个固定的离散值，常见的子类别包括二值化和线性量化
    - 二值化：将模型的权重和激活值量化为只有两个值，通常为+1和-1
    - 线性量化：将数值范围均匀地划分为等间距的离散值
- 非均匀量化：
  - 允许不同区间的间隔大小不同，以更加适应模型参数的实际分布情况，包括对数量化、自适应量化等
    - 对数量化：使用对数函数来划分数值范围

以线性量化为例，介绍量化的基本原理

# 模型压缩—量化



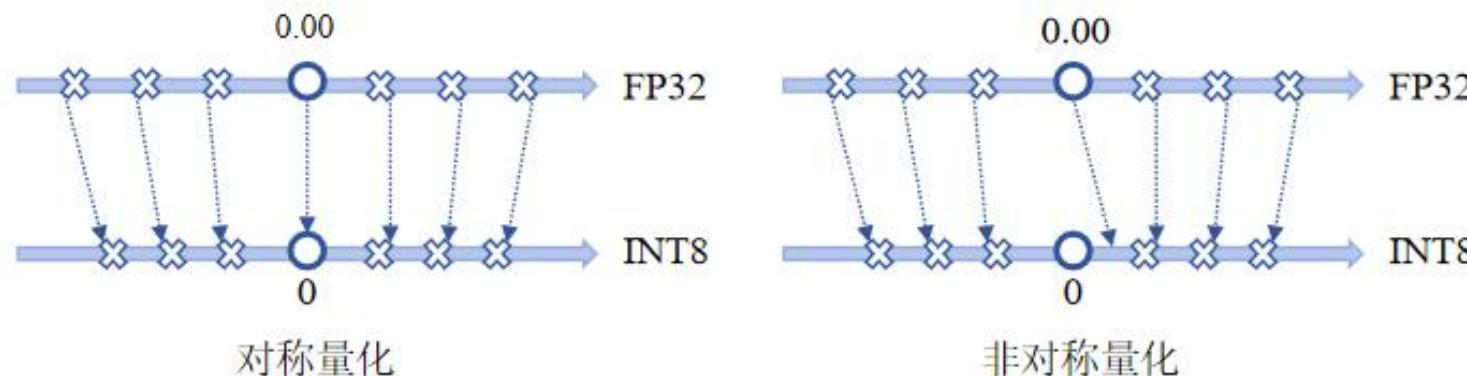
## 2. 量化基本原理

### □ 映射公式

$$Q = \text{round}(S \times R + Z)$$

其中，Q和R分别为量化之后和量化之前的数；S是缩放因子；Z称为Zero Point，表示原值域中的0经过量化后对应的值

- 参数S和Z决定了映射关系
- $Z=0$  (对称量化)  $Z \neq 0$  (非对称量化)



## 2. 量化基本原理

### □ 映射公式

$$Q = \text{round}(S \times R + Z)$$

- $Z$  可设置为：

$$\min_R \times S$$

- $\min_R$  是量化对象 R 的数值下界
- S 可设置为：

$$S = \frac{2^n - 1}{\max_R - \min_R}$$

- $\min_R$  和  $\max_R$  是量化对象 R 的数值下界和上界，n 是量化目标定点数的位数

## 2. 量化基本原理

### □ 映射公式

$$Q = \text{round}(S \times R + Z)$$

- $Z$  可设置为：

$$\min_R \times S$$

- $\min_R$  是量化对象 R 的数值下界

- S 可设置为：

$$S = \frac{2^n - 1}{\max_R - \min_R}$$

- $\min_R$  和  $\max_R$  是量化对象 R 的数值下界和上界,  $n$  是量化目标定点数的位数

- 对于两个主要的量化对象：权重和激活值
  - 在推理阶段，权重是固定值，容易确定上下界
  - 激活值随着输入数据而改变，是动态的数字



## 2. 量化基本原理

### □ 映射公式

$$Q = \text{round}(S \times R + Z)$$

- $Z$  可设置为：

$$\min_R \times S$$

- $\min_R$  是量化对象 R 的数值下界

- $S$  可设置为：

$$S = \frac{2^n - 1}{\max_R - \min_R}$$

- $\min_R$  和  $\max_R$  是量化对象 R 的数值下界和上界， $n$  是量化目标定点数的位数

- 对于两个主要的量化对象：权重和激活值
  - 在推理阶段，权重是固定值，容易确定上下界
  - 激活值随着输入数据而改变，是动态的数字

- 需进行校准，即使用额外的数据集进行采样来确定上下界



## 2. 量化基本原理

### □ 映射公式

$$Q = \text{round}(S \times R + Z)$$

- $Z$  可设置为：

$$\min_R \times S$$

- $\min_R$  是量化对象  $R$  的数值下界

- $S$  可设置为：

$$S = \frac{2^n - 1}{\max_R - \min_R}$$

- $\min_R$  和  $\max_R$  是量化对象  $R$  的数值下界和上界， $n$  是量化目标定点数的位数

- 直接选择量化对象的最大值和最小值作为上界和下界，则很容易受到离群点或者数值分布不均匀的影响

- 引入  $\text{clip}$  操作，将那些信息较为稀疏的区域切除，公式更新为：

$$Q = \text{round}(S \times \text{clip}(R, \alpha, \beta) + Z)$$

- $\alpha$  和  $\beta$  分别是设定的上界和下界； $\text{round}$  表示四舍五入操作



## 2. 量化基本原理

### □ 映射公式

$$Q = \text{round}(S \times \text{clip}(R, \alpha, \beta) + Z)$$

### □ 反量化

$$R = (Q - Z) / S$$

- 误差
- 量化与反量化不是完全互逆的过程，量化时的clip和round操作会带来量化误差

● **量化算法就是选取恰当的量化参数，如缩放因子、Zero Point、 $\alpha$ 、 $\beta$ 及数据映射方式，使得原始的浮点数映射到定点数后尽可能减少精度丢失**



## 2. 量化基本原理

### □ 映射公式

$$Q = \text{round}(S \times \text{clip}(R, \alpha, \beta) + Z)$$

- 对于上述映射关系，量化的关键在于确定 $\alpha$ 和 $\beta$
- 基于统计近似的方法：
  - 最大最小值法：直接使用浮点数的最大值和最小值来确定动态范围

-滑动平均最大最小值法：使用动量更新的方式来逐步更新动态范围的上  
下界， $c$ 是超参数，在PyTorch中默认设置为0.01

$$\alpha = \begin{cases} \min(R) & \text{if } \alpha = \text{None} \\ (1 - c) \times \alpha + c \times \min(R) & \text{otherwise} \end{cases} \quad \beta = \begin{cases} \max(R) & \text{if } \beta = \text{None} \\ (1 - c) \times \beta + c \times \max(R) & \text{otherwise} \end{cases}$$

-KL距离采样法：使用KL散度来度量量化前后的数据分布之间的差异

- 基于优化的方法、基于可微分的方法及基于贝叶斯的方法等

## 3.量化粒度

- 根据量化参数的共用范围，或者哪些量化对象使用相同的量化参数进行区分
- 逐通道量化：对张量的每个通道都配备一个独立的量化参数
- 逐层量化：每个张量具有独立的量化参数
- 分组量化：以组为单位，每组使用相同的量化参数，它的粒度介于逐通道量化和逐层量化之间
- 全局量化：整个网络模型使用相同的量化参数



## 4. 量化感知训练与训练后量化

### □ 量化感知训练

- 在模型训练期间引入伪量化算子，使得模型在训练过程中适应低精度表示
  - 伪量化算子：先进行量化，在进行反量化（引入量化误差）
- 从头训练：时间成本较高，需要充足的数据
- 微调：确保已经训练好的模型在量化为较低位的精度时仍然保持性能
- 一般流程：
  - 在数据集上以浮点数精度（如 FP32）训练模型，得到预训练的模型（若是从头训练的，则不需要此步骤）
  - 构建模拟量化网络：确定量化器、量化粒度，插入量化算子
  - 设置权重和激活值的动态范围的初始值
  - 训练模型
  - 训练完成后，根据量化参数对模型进行量化，同时删除伪量化算子
- Round操作梯度为0，使用Straight-Through Estimator (STE) 方法，让伪量化算子输出的梯度等于输入的梯度

## 4. 量化感知训练与训练后量化

### □ 训练后量化

- 在模型完成训练后应用量化
- 权重量化：只有模型的权重被量化，而激活值仍然保持浮点数
  - 模型的权重具有固定的上界和下界，因此权重量化不需要额外的校准数据集
- 全量化：对权重和激活值都进行量化
  - 使用校准数据集。校准数据集一般可以是来自训练集或者真实场景的数据集，通常只需要少量数据集即可
- PyTorch提供了两种训练后量化方式，分别为动态量化和静态量化
  - 动态量化根据对输入数据的观察动态地计算缩放因子和Zero Point，随后对当前网络层的输入张量进行量化
  - 静态量化同时对权重和偏置进行量化，其使用校准数据集，提前确定所有待量化的张量所需的参数

# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 模型压缩——剪枝

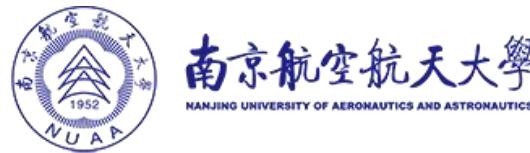


- 随着模型规模的增大，模型可能学到了一些对于任务并不重要的特征或者大量神经元激活值趋近于0

**模型剪枝的核心思想是从经过训练的深度学习模型中识别和删除不必要的连接、权重甚至整个神经元。**

删除这些冗余的组件能够显著减小模型的规模，同时能够提高模型的效率和泛化能力。

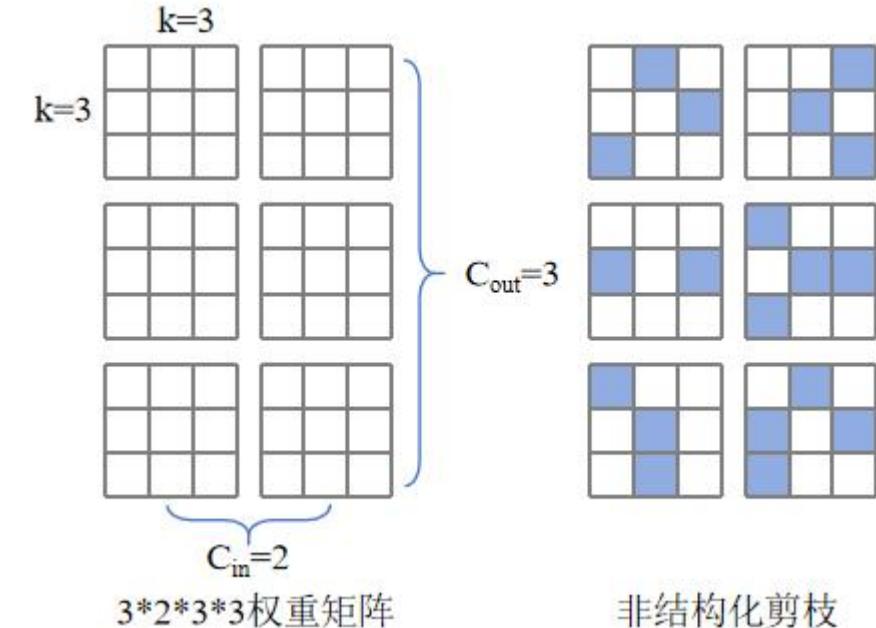
# 模型压缩—剪枝



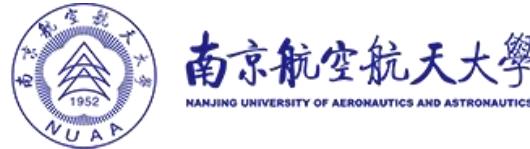
## 1. 剪枝粒度

### □ 非结构化剪枝

- 以模型的单个参数为单位进行剪枝操作
- 通过设定一个阈值或使用稀疏正则化等方法来判断参数的重要性，将权重或激活值较小的参数设置为零



# 模型压缩—剪枝



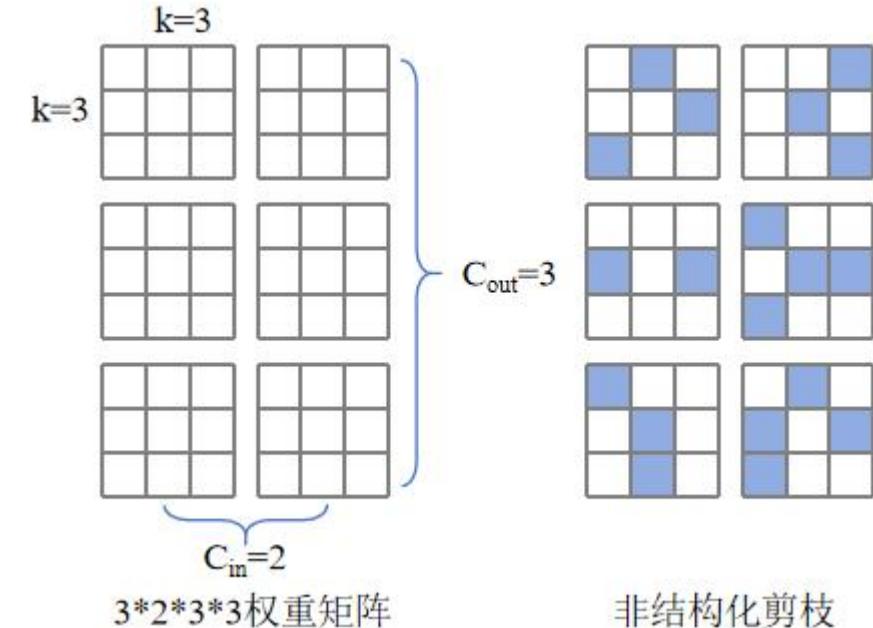
## 1. 剪枝粒度

### □ 非结构化剪枝

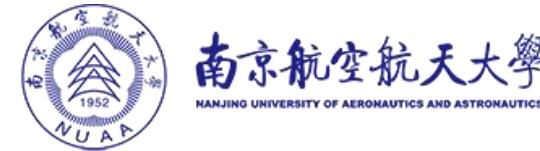
- 以模型的单个参数为单位进行剪枝操作
- 通过设定一个阈值或使用稀疏正则化等方法来判断参数的重要性，将权重或激活值较小的参数设置为零
- 优点：对模型的任意参数进行剪枝，不受模型的限制，并且理论上来说能够提供较高的压缩率
- 缺点：剪枝后的模型存在大量的零参数，引入了稀疏性，需要额外的处理与优化，严重依赖于计算库和硬件

-针对稀疏模型，其权重可以使用仅存储非零元素的位置和值的稀疏矩阵表示。在计算时，只需对非零元素进行计算，而忽略零元素，从而减少计算量

-特定的硬件可以设计稀疏计算单元，用于处理稀疏权重。稀疏计算单元可以检测和跳过输入中对应的零元素，只计算非零权重对应的乘法和累加操作。



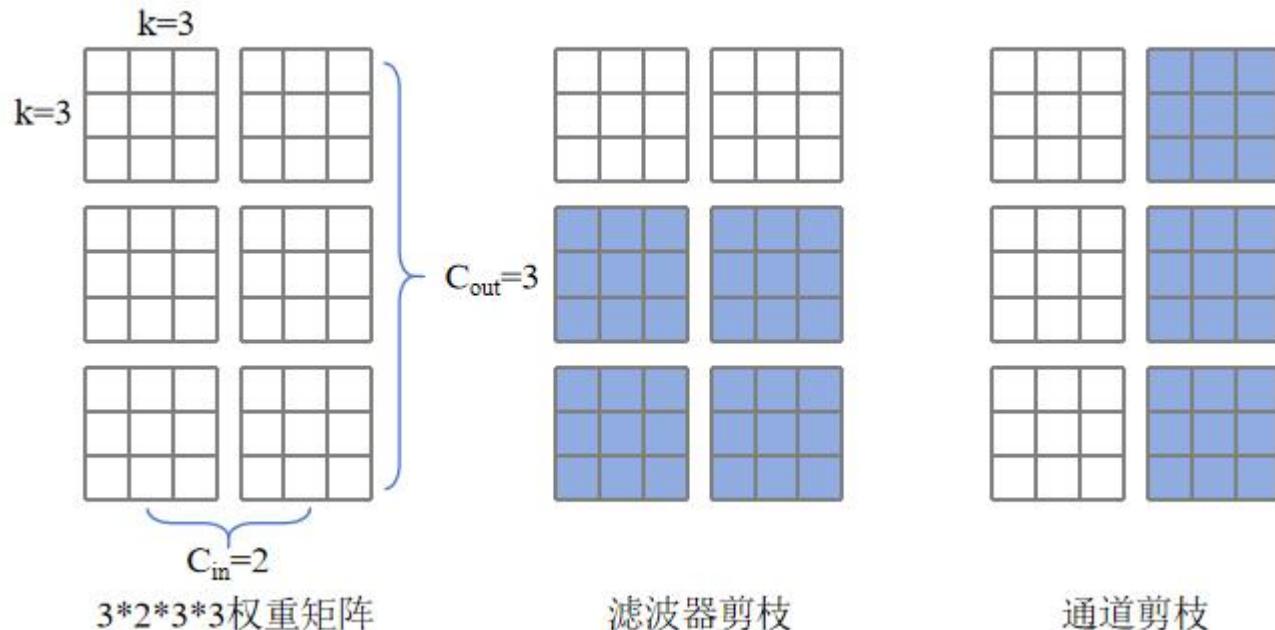
# 模型压缩—剪枝



## 1. 剪枝粒度

### □ 结构化剪枝

- 直接对权重矩阵进行剪枝，其在剪枝过程中保持了模型的整体结构
- 目前主流的结构化剪枝可分为滤波器/卷积核剪枝和通道剪枝
- **滤波器剪枝**：减少卷积层权重矩阵的滤波器数量来实现模型剪枝
- **通道剪枝**：改变滤波器的尺寸来减小模型的规模





## 2. 剪枝的重要性评估标准

- 剪枝的重要性评估标准通常都是启发式的
- 根据所使用的信息，重要性评估方法可以分为基于模型参数的方法和基于数据的方法两类
- **基于模型参数的方法**

- 不需要输入额外的数据，只需根据现有的模型参数来评估重要性
- 根据模型权重的大小评估重要性：一般认为绝对值接近于0的权重可以被剪去  
-对于结构化剪枝，可计算每个滤波器的权重的范数作为重要性分数
- 根据几何中位数评估：计算每个卷积层中的滤波器的几何中位数，丢弃接近几何中位数的滤波器
- 根据余弦相似度评估：权重向量相似的神经元具有更小的余弦距离，这表明它们对模型来说是冗余的
- 基于聚类算法：利用聚类算法对不同的过滤器进行分组，从一组中选择一个过滤器，并将其他过滤器删除
- 优点：不需要进行额外的计算或训练过程；评估过程相对简单，速度更快；不依赖与额外的数据集，具有更高的灵活性
- 缺点：需要设定剪枝阈值来作为重要性的判断依据  
-一般根据特定任务的需求以及网络结构来设定，降低模型剪枝的效率、剪枝算法的泛化性



## 2. 剪枝的重要性评估标准

### □ 基于数据的方法

- 需要输入额外的数据，通常从梯度、特征及模型输出结果等角度来分析剪枝对象的重要性
- 根据激活值评估：将滤波器之后的激活值大小作为评估标准，并丢弃激活值较小的滤波器
- 根据梯度值评估：将剪枝过程看作一个优化问题，其目的是最小化剪枝前后的模型在训练集上的代价差异
- 根据熵评估：将卷积层生成的特征转化为向量，然后计算每个滤波器所对应的特征向量的熵值，剪去熵值较低的滤波器
- 基于特征重建的方法：减少当前层输入特征的通道数，并让裁剪后的输入特征与原始特征更加接近
- 优点：
  - 不仅限于关注网络参数本身，还考虑了数据特征，能够更全面地评估重要性；
  - 基于数据的评估标准根据特定任务的需求选择，使得方法能够更好地适应不同的任务场景，选择更适合的评估标准
- 缺点：需要在数据集上进行额外的计算和评估，这可能会增加计算开销



# 模型压缩—剪枝

## 3. 剪枝算法的基本流程

### □ 训练后剪枝

- 先对模型进行一次训练，然后进行重要性评估并进行剪枝，最后微调剪枝后的模型使其性能接近剪枝前的模型

- 基本流程：

- 训练初始模型

- 剪枝：对剪枝对象（如单个参数、滤波器、通道等）进行重要性评估并将重要性低的对象剪去

- 微调：结构化剪枝会改变模型的结构，将剪枝后的模型在训练集上进行微调来恢复模型的表达能力

- 不断迭代剪枝与微调，直到模型能够满足剪枝的目标需求

### □ 训练期间剪枝

- 剪枝和模型训练同时进行

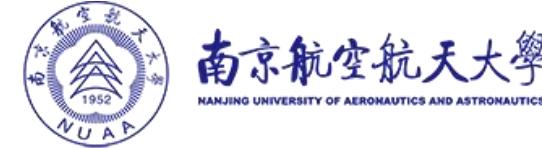
- Dropout

- 学习二进制掩码来选择性地修建某些连接或权重

### □ 训练前剪枝

- 使用预定义的架构和随机初始化的权重神经网络创建，然后直接对初始化权重进行重要性分析

# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片

# 模型压缩—知识蒸馏



## 1. 知识蒸馏概述

### □ 早期概念

- 利用“大模型”的输出知识来监督“小模型”的训练

- 输出知识通常为逻辑单元和类别概率，其中类别概率是逻辑单元经过 softmax 函数之后的输出结果

- 例如，设  $z$  是最后一层全连接层的输出结果，类别概率为：

$$P_i = \frac{\exp(z_i)}{\sum_{j=0}^k \exp(z_j)}$$

是  $z_i$  第  $i$  类的逻辑单元值； $P_i$  是类别概率； $k$  是类别数

- 直接使用逻辑单元作为知识传递给学生模型可能会受到标签噪声的影响
- 类别概率作为知识进行传递可以减小标签噪声带来的影响，但会忽略负标签的作用

- 例如，使用 Softmax 函数操作得到的概率分布对猫、狗、人这三个类别分别是 0.8、0.01、0.00001

- 此时，模型将狗错误地分类为猫的概率远高于分类为人的概率（模型已经具备区分狗和人的能力）（暗知识）

# 模型压缩—知识蒸馏



## 1. 知识蒸馏概述

### □ 早期概念

- 利用“大模型”的输出知识来监督“小模型”的训练

- 输出知识通常为逻辑单元和类别概率，其中类别概率是逻辑单元经过 softmax 函数之后的输出结果

- 例如，设  $z$  是最后一层全连接层的输出结果，类别概率为：

$$P_i = \frac{\exp(z_i)}{\sum_{j=0}^k \exp(z_j)}$$

是  $z_i$  第  $i$  类的逻辑单元值； $P_i$  是类别概率； $k$  是类别数

- 给类别概率加入了温度系数  $T$ ，称为软目标，进行知识蒸馏

$$P_i = \frac{\exp(z_i / T)}{\sum_{j=0}^k \exp(z_j / T)}$$

- 当  $T=1$  时，仍为类别概率

-  $T$  增大时，接近均匀分布，正确类别之外的概率值差异被放大的程度越大，有利于模型学习除正确类别之外的信息

- 数据的原始标签（硬目标）仍然具有独特的类别信息，因此通常会同时使用软目标和硬目标来构建知识蒸馏的损失函数



## 2. 知识的表示形式

### □ 模型输出知识

- 教师模型最后一层输出的逻辑单元或者类别概率

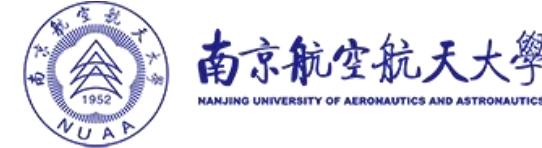
### □ 中间特征知识

- 大模型中间层特征所蕴含的知识
- 使用教师模型中多层的表示作为软目标的方法，其中包括浅层和深层的表示，能够提供更丰富的知识传递，使得学生模型更全面地学到教师模型的知识

### □ 关系特征知识

- 教师模型中不同层的特征之间及不同数据样本之间的关系知识。

# 目录



- 训练数据准备
  - 数据获取
  - 数据预处理
  - 数据增强
  - 数据配比与课程设置
  - 开源数据集
- 并行化与分布式训练
  - 大模型训练的挑战
  - 并行化策略
  - 节点间数据通信
  - 分布式训练框架
- 模型压缩
  - 量化
  - 剪枝
  - 知识蒸馏
- 华为昇腾芯片



## 1. Ascend AI芯片

□ Ascend 310和Ascend 910基于达芬奇架构

- 采用众核异构系统，即同时包含3D Cube矩阵计算单元、Vector向量计算单元、Scalar标量计算单元
- Ascend 310是一种低功耗、高性能的AI处理器，旨在为边缘计算和端侧设备提供强大的AI计算能力，
  - 适用于深度神经网络推理任务
  - 最大功耗仅为8W, FP16半精度算力为8TOPS, INT8整数精度算力为16TOPS
- Ascend 910侧重于对高性能和大规模训练任务的支持
  - Ascend 910的INT8整数精度算力为512TOPS, FP16半精度算力为256TOPS, 最大功耗为350W





- **Atlas系列硬件**: 面向多种应用场景提供了全面且丰富的解决方案，包含开发者套件、加速模块、推理和训练卡、智能小站、服务器和集群等
- **异构计算架构CANN**: 是Ascend芯片的核心软件层，是深度学习框架和AI处理器硬件之间的桥梁，功能类似于英伟达的CUDA+CuDNN
- **深度学习框架 MindSpore**: MindSpore支持多种主流的神经网络模型，同时提供了丰富的开发者资源和社区支持。该框架提供了函数式+面向对象融合的编程范式，内置了函数式自动微分功能
- **应用使能与行业应用**: 华为AI生态的应用使能层包含ModelZoo、MindX SDK、MindX DL、MindX Edge等子模块

谢谢!  
Thanks!

智周万物·道济天下