

# 分布式深度学习的梯度压缩方法研究及其应 用

作者姓名 卢天恩

指导教师姓名、职称 谢卫莹 教授

申请学位类别 工学硕士



学校代码 10701  
分 类 号 TP391

学 号 21011210016  
密 级 公开

# 西安电子科技大学

## 硕士学位论文

### 分布式深度学习的梯度压缩方法研究及其应用

作者姓名：卢天恩

一级学科：信息与通信工程

二级学科（研究方向）：通信与信息系统

学位类别：工学硕士

指导教师姓名、职称：谢卫莹 教授

学 院：通信工程学院

提交日期：2024 年 03 月



# **Research on Gradient Compression Method for Distributed Deep Learning and Its Application**

A thesis submitted to  
XIDIAN UNIVERSITY  
in partial fulfillment of the requirements  
for the degree of Master  
in Information and Communications Engineering

By  
Lu TianEn  
Supervisor: Xie WeiYing Title: Professor  
March 2024



## 西安电子科技大学 学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名：\_\_\_\_\_ 日 期：\_\_\_\_\_

## 西安电子科技大学 关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，结合学位论文研究成果完成的论文、发明专利等成果，署名单位为西安电子科技大学。

保密的学位论文在\_\_\_\_年解密后适用本授权书。

本人签名：\_\_\_\_\_ 导师签名：\_\_\_\_\_

日 期：\_\_\_\_\_ 日 期：\_\_\_\_\_



## 摘要

深度学习技术在大数据时代席卷了目标检测、图像分类、自然语言处理、自动驾驶、推荐系统等诸多应用领域，引领现在和将来的技术主流浪潮。然而，由于数据集规模和神经网络模型复杂度的迅速膨胀，模型训练需要越来越多的计算资源，导致单机训练速度缓慢，甚至无法成功部署。分布式计算技术能够组织、协同多个计算节点进行网络模型训练，是解决大规模数据场景下复杂模型训练的有效方案。但分布式训练的迭代过程中，网络拓扑中各节点之间高频、全尺寸全精度的数据通信交换了大量数据，严重拖慢了训练进程，成为分布式算法的关键瓶颈。针对这一通信瓶颈问题，本文从压缩通信数据和缩减通信频次两个角度出发，提出了基于近似质心的梯度压缩算法（Gradient Compression via Approximate Centroid, GCAC）和基于 $k$ -互异近邻的梯度压缩算法（Gradient Compression based on  $k$ -Reciprocal Nearest neighbors,  $k$ -RNGC），旨在减小训练过程中的通信开销，优化并加速了分布式训练流程。本文主要研究内容与创新点具体描述如下：

GCAC 算法首先利用加速计算技术得到网络模型中局部梯度张量的近似质心，然后根据梯度与近似质心之间的距离结合预设的梯度压缩比，筛选出合适的梯度，保留了模型参数优化所需的关键梯度信息，大幅减少了网络通信流量，加速了分布式训练速度。实验结果表明，GCAC 算法能在不同规模的任务场景中有效加速网络模型收敛、提高分布式训练效率且保证模型性能，尤其在小压缩比时，GCAC 算法还可以提高网络模型的最终收敛精度。但由于 GCAC 算法在实践过程采用了近似质心的加速计算技术，导致该算法在大压缩比场景下训练小模型时，模型收敛性和性能表现均有较大下降。

为了克服 GCAC 算法在大压缩比时的难题，本文从通信数据压缩和通信频次缩减协同的角度出发，提出了基于 $k$ -互异近邻的梯度压缩算法。 $k$ -RNGC 算法首先通过全局梯度计算感知算法获取局部梯度压缩比，然后基于该局部压缩比使用 $k$ -互异近邻梯度选择算法进行梯度优选，对原梯度张量进行压缩，大幅度减少了网络通信的数据量。该算法还融合了动量修正以及预热训练等组件，优化了分布式训练流程。高光谱目标检测和图像分类等实验结果表明，在 100 倍压缩比的情况下， $k$ -RNGC 算法不仅具有良好的模型和数据适应能力、克服了 GACA 所面临的大压缩比问题，还能够有效提高分布式训练效率。

**关键词：**分布式， 深度学习， 梯度压缩， 目标检测， 图像分类



## ABSTRACT

Deep learning technology has swept through various application fields including target detection, image classification, natural language processing, autonomous driving and recommendation systems in the era of big data. It will lead the wave of mainstream technology now and in the future. However, due to the rapid expansion of dataset size and neural network model complexity, model training requires more and more computational resources, which leads to slow training speed or even unsuccessful deployment on a single machine. Distributed computing technology can organize and collaborate with multiple computing workers, which is an effective solution for complex model training with large-scale dataset. However, during the iterative process of distributed training, the high-frequency, full-size and full-precision data communication between nodes in the network topology exchanges a large amount of data, which seriously slows down the training process and becomes a key bottleneck for distributed algorithms. To address this communication bottleneck, this paper proposes Gradient Compression via Approximate Centroid (GCAC) and Gradient Compression based on  $k$ -Reciprocal Nearest neighbors ( $k$ -RNGC) from the perspectives of compressing the communication data and reducing the communication frequency, aiming at reducing the communication overhead during the training process, and optimizing and accelerating the distributed training process. The main research contents and innovations of this paper are as follows:

The GCAC algorithm first uses accelerated computation techniques to obtain the approximate centroid of the local gradient tensor in the network model. Then, according to the distance between the gradient and the approximate centeroid and combined with the preset gradient compression ratio, it filters out the appropriate gradients which retains the key gradient information required for the optimization of the model parameters, dramatically reducing the network communication traffic and accelerating the distributed training speed. The experimental results show that GCAC algorithm can effectively accelerate the convergence of the network model, improve the efficiency of distributed training, and guarantee the model performance in different sizes of task scenarios, especially when the compression ratio is small, the GCAC algorithm can also improve the final convergence accuracy of the network model. However, the accelerated computational technique of approximating the centeroid is used in the practice of the GCAC algorithm, which leads to a large degradation of model

convergence and performance performance when the algorithm trains a small model in a large compression ratio scenario.

In order to overcome the difficulties of the GCAC algorithm with large compression ratios, this paper proposes Gradient Compression based on  $k$ -Reciprocal Nearest neighbors from the perspective of communication data compression and communication frequency reduction. The  $k$ -RNGC algorithm first obtains the local gradient compression ratio through a global gradient computation-aware algorithm, and then compresses the original gradient tensor based on this local compression ratio using the  $k$ -reciprocal nearest-neighbor gradient selection algorithm for gradient preference, which drastically reduces the amount of data in network communication. The algorithm also optimizes the distributed training process by incorporating momentum correction and warm-up training components. The experimental results of hyperspectral target detection and image classification show that with the compression ratio set to 100, the  $k$ -RNGC algorithm not only excels in the adaptability of the network model and dataset, overcomes the large compression ratio problem, but also effectively improves the distributed training efficiency.

**Keywords:** Distributed, Deep Learning, Gradient Compression, Target Detection, Image Classification

## 插图索引

图 1.1	Top-K 算法原理图.....	4
图 1.2	量化算法原理图 .....	4
图 1.3	分布式训练通信延迟算法原理图 .....	5
图 2.1	神经网络层级结构原理示意图 .....	9
图 2.2	常见的激活函数曲线 .....	10
图 2.3	单节点网络模型 SGD 训练流程 .....	12
图 2.4	数据并行、模型并行和流水线并行示意图 .....	14
图 2.5	数据并行策略下分布式训练场景中不同的通信拓扑架构 <sup>[67]</sup> .....	16
图 2.6	不同拓扑架构下同步更新的网络模型训练流程 .....	17
图 2.7	高光谱目标检测大致流程示意图 .....	20
图 2.8	对抗自编码网络结构示意图 <sup>[81]</sup> .....	21
图 2.9	VGGNet 和 ResNet 网络模型核心结构示意图 .....	22
图 2.10	CIFAR10 数据集标签及其部分图片样张.....	23
图 3.1	网络模型采用传统方法压缩前后的梯度分布 .....	28
图 3.2	基于近似质心的梯度压缩算法应用框图 .....	28
图 3.3	基于 GCAC 算法压缩前后梯度分布 .....	32
图 3.4	AeroRIT 数据集上，使用不同梯度压缩算法进行分布式训练的高光谱 目标检测可视化结果 .....	34
图 3.5	Xiongan 数据集上，使用不同梯度压缩算法进行分布式训练的高光谱 目标检测可视化结果 .....	35
图 3.6	CIFAR10 数据集上，不同网络模型采用 GCAC 算法进行分布式训练 的 $acc_1$ 与 loss 曲线 .....	38
图 3.7	CIFAR10 数据集上，不同网络模型采用不同梯度压缩算法进行分布式 训练的 $acc_1$ 与 loss 曲线 .....	39
图 4.1	通过计算感知算法以获取网络模型局部压缩比原理图 .....	45
图 4.2	$Epoch_{wm}$ 对不同网络模型性能的影响 .....	53
图 4.3	AeroRIT 数据集上，不同压缩比对 $k$ -RNGC 算法的 HTD 可视化结果 .....	54
图 4.4	Xiongan 数据集上，不同压缩比对 $k$ -RNGC 算法的 HTD 可视化结果 .....	55
图 4.5	CIFAR10 数据集上，不同压缩比对不同网络模型采用 $k$ -RNGC 算法的 分布式训练曲线 .....	56



## 表格索引

表 1.1	常见深度学习网络模型在 ImageNet 数据集上的性能表现 .....	1
表 2.1	常见大规模深度学习网络模型参数 .....	15
表 2.2	Allreduce 原语常见实现的通信时间 <sup>[29]</sup> .....	18
表 2.3	Xiongan 和 AeroRIT HSI 中目标和背景的样本数量以及空间占比 .....	23
表 3.1	下游任务、网络模型以及训练数据集相关设置 .....	33
表 3.2	GCAC 梯度压缩算法超参数设置 .....	34
表 3.3	不同训练数据集上，使用不同梯度压缩算法进行分布式训练的 HTD 实验结果 .....	36
表 3.4	CIFAR10 数据集上，GCAC 与其他算法的分布式训练 $acc_1$ 与 loss 收敛值 .....	40
表 3.5	CIFAR10 数据集上，不同网络模型采用不同梯度压缩算法的耗时比较 .....	41
表 3.6	CIFAR10 数据集上，“近似质心”加速计算组件对不同网络模型的影响 .....	42
表 4.1	$k$ -RNGC 算法相关超参数设置 .....	52
表 4.2	CIFAR10 数据集上，预热训练组件对不同网络模型精度的影响 .....	53
表 4.3	AAE 网络模型采用 $k$ -RNGC 算法在不同压缩比下的量化结果 .....	54
表 4.4	CIFAR10 数据集上，不同网络模型采用 $k$ -RNGC 算法进行分布式训练的量化结果 .....	57
表 4.5	CIFAR10 数据集上，不同模型采用 $k$ -RNGC 算法进行分布式训练的单次迭代耗时 .....	59



## 符号对照表

符号	符号名称
$\max(\cdot)$	最大值函数
$\min(\cdot)$	最小值函数
$\log_2(\cdot)$	以 2 为底数的对数函数
$\exp(\cdot)$	以自然常数 e 为底的指数函数
$\mathcal{L}(\cdot)$	损失函数
$\sigma(\cdot)$	非线性激活函数
$\delta(\cdot)$	单位函数
$\lceil \cdot \rceil$	向上取整
$ \cdot $	绝对值
$\ \cdot\ _2$	欧几里得范数
$\in$	属于
$\mathbb{R}$	数域
$\sum$	求和
$acc_k$	Top- $k$ 精度
$\arg$	变量取值
<b>J</b>	网络模型
<b>g</b>	梯度
<b>W</b>	参数矩阵
<b>b</b>	神经元偏置
$\Delta$	误差值
$\eta$	学习率
$\varphi$	Batch 大小
$\tau$	阈值
$\mathcal{C}$	质心
$\mathcal{G}^k$	$k$ -互异近邻梯度推荐子集



## 缩略语对照表

缩略语	英文全称	中文对照
AAE	Adversarial Autoencoder	对抗自编码网络
Adam	Adaptive Moment Estimation	自适应时刻估计
AE	Autoencoder	自编码网络
BP	Backforward Propagation	反向传播
CE	Cross Entropy	交叉熵
CNN	Convolutional Neural Network	卷积神经网络
DL	Deep Learning	深度学习
DNN	Deep Neural Network	深度神经网络
EO	Earth Observation	地球观测
FPR	False Positive Rate	虚警率
GPU	Graphics Processing Unit	图形处理单元
HSI	HyperSpectral Image	高光谱图像
HTD	HyperSpectral Target Detection	高光谱目标检测
HPC	High Performance Computing	高性能计算
ILSVRC	ImageNet Large Scale Visual Recognition Challenge	ImageNet 大规模视觉识别挑战
LR	Learning Rate	学习率
LSTM	Long Short-term Memory	长短期存储器
MAE	Mean Absolute Error	例如平均绝对误差
ML	Machine Learning	机器学习
MPI	Message Passing Interface	消息传递接口
MSE	Mean Square Error	均方误差
NCCL	NVIDIA Collective Communications Library	NVIDIA 集体通信库
RNN	Recurrent Neural Network	循环神经网络
ROC	Receiver Operating Characteristic	受试者工作特征曲线

缩略语	英文全称	中文对照
RPC	Remote Procedure Call	远程过程调用
RS	Remote Sensing	遥感
P2P	Point to Point	点对点
PS	Parameter Server	参数服务器
QSGD	Quantized Stochastic Gradient Descent	量化随机梯度下降
GCAC	Gradient Compression via Approximate Centroid	基于近似质心的 梯度压缩
$k$ -RNGC	Gradient Compression based on $k$ -Reciprocal Nearest neighbors	基于 $k$ -互异近邻的 梯度压缩

## 目 录

摘要.....	I
ABSTRACT.....	III
插图索引.....	V
表格索引.....	VII
符号对照表 .....	IX
缩略语对照表 .....	XI
<b>第一章 绪论.....</b>	1
1.1 研究背景与意义 .....	1
1.2 国内外研究现状 .....	3
1.3 本文研究内容和章节安排.....	5
1.3.1 本文研究内容 .....	5
1.3.2 本文章节安排 .....	6
<b>第二章 相关理论基础.....</b>	9
2.1 引言.....	9
2.2 深度学习基础.....	9
2.2.1 网络模型架构介绍 .....	9
2.2.2 激活函数简要介绍 .....	10
2.2.3 网络模型训练流程介绍 .....	11
2.3 分布式训练理论基础.....	13
2.3.1 分布式训练的并行方式 .....	14
2.3.2 分布式训练的拓扑架构 .....	15
2.3.3 分布式训练的同步方式 .....	18
2.3.4 现有分布式平台介绍.....	19
2.4 任务场景介绍.....	20
2.4.1 高光谱目标检测流程.....	20
2.4.2 相关网络模型介绍 .....	21
2.5 实验数据集与评价指标 .....	22
2.5.1 实验数据集 .....	22
2.5.2 评价指标 .....	24
2.6 本章小结.....	25
<b>第三章 基于近似质心的梯度压缩算法.....</b>	27

---

3.1	引言 .....	27
3.2	基于近似质心的梯度压缩算法 .....	27
3.2.1	网络模型中局部梯度分析 .....	27
3.2.2	基于近似质心的梯度压缩算法 .....	29
3.3	实验与结果分析 .....	33
3.3.1	实验环境以及相关参数配置 .....	33
3.3.2	高光谱目标检测实验结果分析 .....	33
3.3.3	图像分类实验结果分析 .....	37
3.3.4	耗时分析 .....	40
3.3.5	“近似质心”加速计算组件分析 .....	42
3.4	本章小结 .....	42
<b>第四章</b>	<b>基于 <math>k</math>-互异近邻的梯度压缩算法 .....</b>	<b>45</b>
4.1	引言 .....	45
4.2	基于 $k$ -互异近邻的梯度压缩算法 .....	45
4.2.1	计算感知获取局部梯度压缩比 .....	45
4.2.2	$k$ -互异近邻梯度选择算法 .....	47
4.2.3	动量修正 .....	48
4.2.4	预热训练 .....	50
4.3	实验与结果分析 .....	51
4.3.1	实验环境及其相关参数配置 .....	51
4.3.2	预热训练组件分析 .....	52
4.3.3	高光谱目标检测实验结果分析 .....	53
4.3.4	图像分类实验结果分析 .....	56
4.3.5	耗时分析 .....	58
4.4	本章小结 .....	59
<b>第五章</b>	<b>总结与展望 .....</b>	<b>61</b>
5.1	工作总结 .....	61
5.2	未来研究展望 .....	62
<b>参考文献 .....</b>	<b>63</b>	
<b>致谢 .....</b>	<b>71</b>	
<b>作者简介 .....</b>	<b>73</b>	

# 第一章 绪论

## 1.1 研究背景与意义

基于神经网络架构的深度学习 (Deep Learning, DL) 技术, 因其架构、网络模型以及可执行任务的灵活性, 展现出很强的高维数据表征能力。近年来, 深度学习技术凭借其超强的灵活性和数据表征能力席卷了目标检测<sup>[1][2][3]</sup>、图像分类<sup>[4][5][6]</sup>、自然语言处理<sup>[7][8]</sup>、视频理解<sup>[9][10][11]</sup>等诸多应用领域, 引领现在和将来的技术潮流。自从 ILSVRC (ImageNet Large Scale Visual Recognition Challenge) 比赛桂冠被 AlexNet<sup>[12]</sup>摘取以来, 神经网络模型的发展越来越快。VGGNet<sup>[5]</sup>、GoogLeNet<sup>[13]</sup>、ResNet<sup>[14]</sup>、Transformer<sup>[15]</sup>等网络模型的参数量和复杂度不断增加, 使得网络模型的灵活性和数据表征能力获得大幅提升、性能表现也同样更加优异。例如, 在高光谱目标检测 (Hyperspectral Target Detection, HTD) 任务中, 研究人员将更复杂的深度神经网络 (Deep Neural Network, DNN) 部署到超大规模的高光谱遥感数据集上, 从而推动了高光谱图像 (Hyperspectral Image, HSI) 数据解释方面的重大进展<sup>[16][17][18]</sup>。

表 1.1 常见深度学习网络模型在 ImageNet 数据集上的性能表现

网络模型	层数	模型规模 (MB)	参数量 (M)	Top-5 错误率 (%)
AlexNet <sup>[12]</sup>	8	233	60	16.4
VGGNet-19 <sup>[5]</sup>	19	548	143	7.32
GoogLeNet <sup>[13]</sup>	22	51	6.8	6.67
ResNet <sup>[14]</sup>	152	230	19.4	3.57

与此同时, 在许多现实世界的应用中, 训练数据集的规模一直呈现日益增长的趋势。例如, 在现在和将来的地球观测 (Earth Observation, EO) 任务场景中, 由 NASA's Jet Propulsion Laboratory 运行的 AVIRIS<sup>[19]</sup> 及其新一代 AVIRIS-NG 每小时就可获取将近 9GB 的数据, 同时 EO-1 Hyperion 传感器每小时大约采集 71.9GB 的数据, 这意味着每天将有超过 1.6TB 新数据的产生。同时, 由于传感器技术的蓬勃发展, 在采集到的数据规模持续增加的情况下, 数据质量也在不断提高, 使得对存储和处理能力等方面的要求呈现指数级增长<sup>[20][21][22][23]</sup>。

然而, 随着数据集规模、质量以及神经网络模型参数量、复杂度的不断增加, 模型所需的训练时间和资源开销越来越大, 严重拖慢了任务开发、迭代周期。具体来

说，在相同的任务和硬件设备环境下，由于网络模型的参数量和复杂度增加，每一次迭代过程将消耗更多的时间进行模型计算和信息更新；同时数据集规模的增加，也会将导致单个 Epoch 训练期间，每次数据集遍历操作花费更多时间。因此，为了充分利用、挖掘大规模数据集和高复杂度网络模型之间的协同优势，大规模和/或数据密集型的机器学习（Machine Learning, ML）、信号处理与控制等问题的解决通常需要足够并行度的并行/分布式算法<sup>[24]</sup>。也就是说，引入并行/分布式算法的目的是为了充分利用这些海量数据，并通过在多个节点之间分配计算负载，同时使用多个节点的计算资源，确保在网络模型快速收敛的情况下，解决大规模数据场景下高复杂度模型的训练难题<sup>[25][26][27][28]</sup>。

目前，在大规模数据集上采用分布式算法训练高复杂度网络模型过程中，各节点之间传输梯度、参数等模型信息的通信行为往往具有以下特点<sup>[29]</sup>：

- 1) **高频次**: 通信操作频繁；
- 2) **大流量**: 单次通信传输数据量巨大；
- 3) **全尺寸全精度**: 通信数据采用全尺寸全精度格式。

然而，这些高频次、大流量、全尺寸全精度的通信行为，导致了整个分布式训练的时间复杂度从“计算”转移到了“通信”，即网络通信开销成为了分布式算法的主要瓶颈。例如，在训练具有数百万个参数的深度神经网络模型（例如，AlexNet、VGGNet、ResNet 和 LSTM 等）时，通信时长占比（通信时长占比 = 通信总时长/训练总时长）竟然高达 80%<sup>[30][31][32]</sup>。因此，通过合理技术手段优化分布式训练过程中的网络通信效率是解决通信瓶颈问题的关键。许多分布式计算框架的开源实现都基于 MapReduce 模型<sup>[33]</sup>，而 Horovod<sup>[34]</sup>作为大数据场景下的分布式训练框架也受到越来越多人的欢迎。

应用于并行/分布式训练的梯度压缩算法可以在很大程度上减轻网络通信瓶颈问题，但同时也面临着众多挑战。第一，梯度压缩算法需要在减少通信量的同时，保持与未压缩梯度相当的模型准确度，但实践中压缩比越高，准确度下降的可能性就越大。第二，分布式训练所使用的计算设备可能存在异构性，即各个设备之间具有不同的计算能力、内存大小和网络带宽，难以找到一种适用于所有机器的通用压缩算法。第三，深度神经网络模型往往具有非常高维的梯度张量，梯度压缩操作过程可能导致内存消耗过大，限制分布式训练规模，增加了梯度压缩算法的设计难度。第四，随着分布式网络拓扑规模、数据量以及模型参数的不断增大，梯度压缩算法的计算延迟和通信延迟可能会影响分布式的整体训练性能。

综上所述，在诸如高光谱遥感图像等大规模数据集的场景中，采用并行/分布式算法训练高复杂度网络模型仍存在诸多亟待解决的挑战和问题。针对梯度压缩算法所面临的难点，本文在高光谱遥感图像和自然图像两种不同类型的数据集基础上，将

高光谱目标检测和自然图像分类两个典型任务作为本文的下游验证任务，开展相关的梯度压缩算法研究具有一定的创新性和应用价值，对于推动梯度压缩技术在大规模数据集和高复杂度网络模型场景下的分布式训练应用具有重要意义。

## 1.2 国内外研究现状

近年来，为了解决分布式训练中的通信瓶颈问题，即在分布式训练过程中，高频、大流量、全尺寸全精度数据格式的通信行为导致网络模型训练时间过长。因此，国内外研究人员提出了各种分布式优化算法，主要分为以下三种类型：1) 稀疏化（Sparsification）、2) 量化（Quantization）、3) 通信延迟（Communication Delay）。

1) **稀疏化：**梯度稀疏化算法是根据梯度的某种度量将模型中局部或全局梯度进行排序，然后基于排序结果，选取最高  $k$  项并与其他节点进行通信操作的方法<sup>[35][36]</sup>，原理如图 1.1 所示。Strom<sup>[37]</sup>在 2015 年提出 Scalable 算法，该算法在分布式训练过程中仅将大于某个预定阈值的梯度用于通信操作。但实际操作过程中，由于不同网络模型或同一网络模型的不同层之间的最佳阈值相差甚远，因此很难提前选取一个合适的阈值<sup>[38]</sup>。Aji 等人<sup>[39]</sup>在 2017 年以压缩比（压缩比 = 压缩前梯度规模/压缩后梯度规模）为目标，将梯度绝对值作为度量，并采用动态选取阈值取代了预定阈值，但实验结果表明这种方法在小压缩比的情况下，会导致网络模型的收敛速度以及最终准确率有所下降。Lin 等人<sup>[31]</sup>基于现有的梯度压缩方法，利用动量校正和局部梯度修剪的方案保证局部梯度的累计值处于一个安全的范围，改善了稀疏度过高时对模型收敛性的损伤，并且使用动量因子屏蔽和预热训练的方案减小了梯度陈旧效应带来的影响，但是该算法仍然使用梯度绝对值作为度量，因此存在梯度选择会受范数方差影响的问题。Cui 等人<sup>[40]</sup>在 2020 年提出了 ClusterGrad 算法，使用两次 K-Means 聚类算法进行梯度稀疏化，动态地将梯度分成不同的簇，并只对每个簇的梯度代表进行压缩和传输，从而减少通信开销。Peng Luo 等人<sup>[41]</sup>没有使用阈值方法对梯度进行分割，而是协同概率和梯度绝对值随机选取目标梯度用于通信操作。梯度稀疏化压缩算法的压缩比没有理论上限，但在实践中压缩比越高，网络模型准确度下降的风险就越大，甚至可能造成网络模型无法收敛。

2) **量化：**梯度量化算法是将高精度梯度值量化为较小位宽的低精度值（浮点型或整型），并将量化后的梯度应用于各节点之间的通信操作<sup>[42]</sup>，原理如图 1.2。Seide 等人<sup>[32]</sup>在 2014 年凭借实验证明，1-bit 量化方案仍然可以提供良好的性能，同时大大降低了分布式系统中的通信成本。Wen 等人<sup>[43]</sup>在 2017 年提出了 TernGrad（Ternary Gradient Compression）算法，该算法通过随机数生成器为该方案提供理论保证，以确保随机量化后的梯度（三个数值：-1, 0, 1）仍然是原梯度的无偏近似。TernGrad 不仅大大减少了通信时间，而且在梯度有界的前提下，其收敛性在数学上得到了证明，

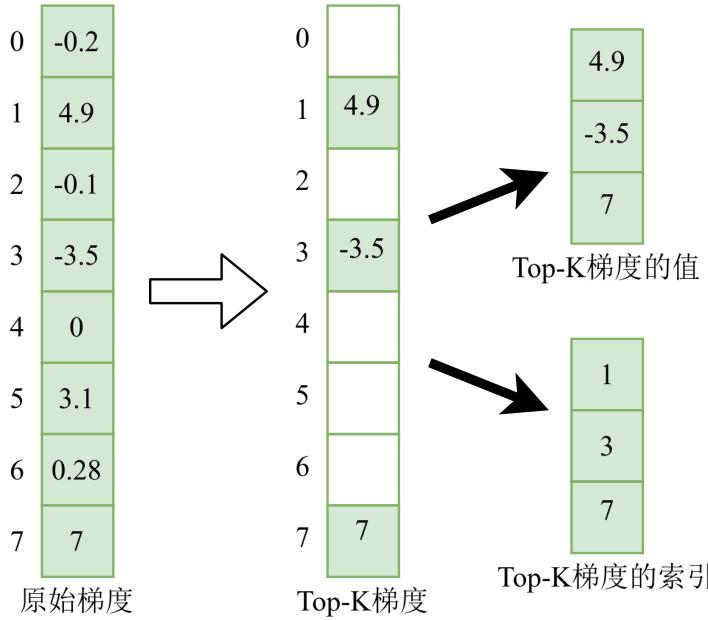


图 1.1 Top-K 算法原理图

但不幸的是，在大模型场景下，由于梯度方差极大膨胀，导致模型的准确度下降明显。Alistarh 等人<sup>[30]</sup>通过探索数据并行化 SGD 中“通信成本”和“收敛保证”之间的关联性，提出了一系列有损压缩量化算法 - 量化随机梯度下降（Quantized Stochastic Gradient Descent, QSGD）。在梯度数据方差的协助下，处理器可以通过该算法权衡每次迭代的通信数据比特数，但 QSGD 主要面向的是单机多 GPU（Graphics Processing Unit）的分布式训练场景。signSGD 算法由 Bernstein 等人<sup>[44]</sup>提出，是一种服务于分布式训练的量化方案，该算法将每个计算节点的梯度值量化为二进制符号，然后主节点通过多数表决的方法对量化梯度进行汇总。在神经网络模型分布式训练场景中，梯度通常用 32-bit 浮点数表示，因此用于梯度量化算法的压缩比上限为 32。

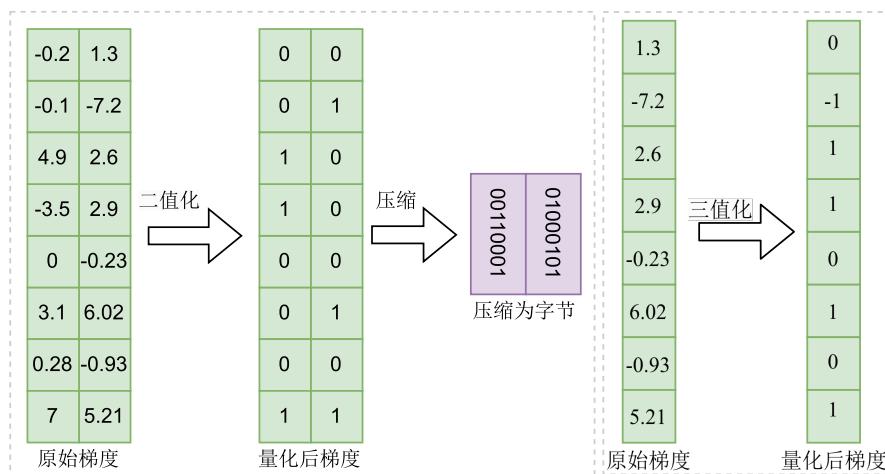


图 1.2 量化算法原理图

3) 通信延迟: 该类算法的思路是先在本地网络模型上进行数轮梯度更新, 然后再与其他节点进行通信操作以实现全局梯度更新, 从而降低各个节点之间的通信频率<sup>[45]</sup>, 原理如图 1.3 所示。McDonald 等人<sup>[46]</sup>和 Povey 等人<sup>[47]</sup>分别将局部迭代得到的平均梯度值应用于感知器和语音识别场景的 DNN 分布式训练。Zhang 等人<sup>[48]</sup>研究出了一种具有“弹性平均”的 SGD 异步方法。McMahan 等人<sup>[49]</sup>在 2017 年提出了联合平均算法 (Federated Averaging), 该算法在每个计算节点的多次迭代步骤中执行 SGD 算法, 并将多次迭代得到的局部梯度平均值用于网络通信, 而非每次迭代获得的梯度值, 从而大幅降低了各节点之间的通信频率。实验表明, 在不同的神经网络模型中, 例如卷积神经网络 (Convolutional Neural Network, CNN) 和循环神经网络 (Recurrent Neural Network, RNN), 局部迭代的通信延迟算法不仅没有显著降低网络模型的收敛速度, 而且可以将必要的通信次数降低 10~100 倍。

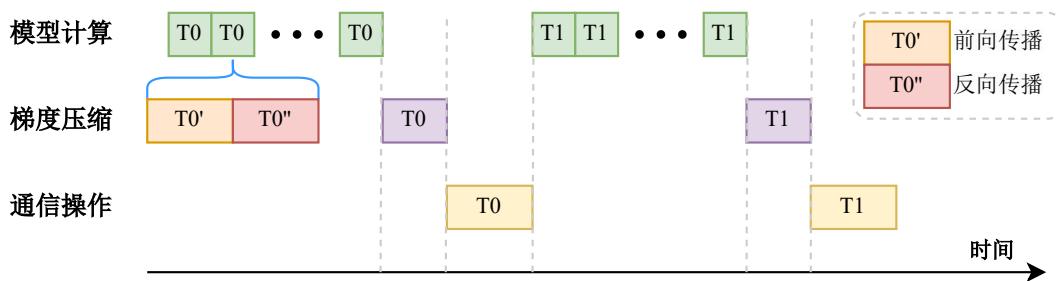


图 1.3 分布式训练通信延迟算法原理图

## 1.3 本文研究内容和章节安排

### 1.3.1 本文研究内容

本文主要以大规模数据场景下进行高复杂网络模型的分布式训练为背景, 深入学习和研究了各种梯度压缩算法, 并对这些梯度压缩算法所面临的挑战进行了分析。针对图像分类和高光谱目标检测等下游任务中分布式训练所面临的挑战和问题, 从梯度稀疏化以及通信延迟两个关键角度入手, 展开了相关梯度压缩技术的深入研究, 提出了以下两种高效的梯度压缩算法:

#### 1) 基于近似质心的梯度压缩算法

面对诸如高光谱目标检测等任务涉及到的大数据大模型场景时, 传统单节点的计算资源、存储资源有限, 难以满足部署、训练的要求。而现有的分布式训练技术尽管可以部署、训练, 但因高频、大流量、全尺寸全精度的通信行为, 导致各节点之间的通信花销成为其关键瓶颈, 从而大幅增加了训练时长。因此, 本文提出基于近似质心的梯度压缩算法 (Gradient Compression via Approximate Centroid, GCAC), 该算法基于网络模型反向传播时梯度的近似质心, 将网络模型中各梯度与近似质心的欧

几里得范数作为距离度量，对模型中的梯度进行排序，并在排序结果的基础上，利用 Top-K 算法选取合适的梯度用于通信操作，大幅压缩通信数据，显著降低了通信开销。

## 2) 基于 $k$ -互异近邻的梯度压缩算法

压缩比较小的分布式训练场景中，现有的梯度稀疏化算法在整体训练速度和模型准确度都取得了不错的效果。当期望分布式训练花费的时间进一步缩短时，自然而然的想法就是增加压缩比，进而大幅减少各节点之间通信操作数据量。但这时，一些常见的梯度稀疏化算法（包括 GCAC）都对模型的准确度产生了较大的影响，模型收敛性差，使得网络模型无法很好地发挥其优势。针对大压缩比导致的模型收敛性差的情况，本文提出基于  $k$ -互异近邻的梯度压缩算法（Gradient Compression based on  $k$ -Reciprocal Nearest neighbors,  $k$ -RNGC），该算法通过梯度计算感知与  $k$ -互异近邻选择算法相结合，优选出满足预设全局压缩比阈值的梯度，并协同动量修正和预热训练等组件，优化了分布式训练流程，显著降低了分布式训练过程中的网络通信数据量，大幅度加快了网络模型训练速度。

### 1.3.2 本文章节安排

围绕分布式训练和梯度压缩算法，本论文的具体结构组织以及各章节的主要内容描述如下：

第一章，绪论。本章首先介绍了诸如高光谱遥感图像等大规模数据集的特点以及分布式训练的背景和意义。然后，根据高光谱图像、超大规模自然图像等数据集的特点以及深度神经网络的发展现状，深入分析了分布式训练在大规模数据集场景下训练高复杂度网络模型时所面临的挑战和问题。接下来，对国内外现有的梯度压缩算法研究现状做了简要介绍和分析。最后，阐述了本文的主要研究内容和章节安排。

第二章，相关理论基础。本章介绍了深度学习基础、分布式训练基础、现有分布式平台介绍以及本文涉及的任务场景相关介绍。首先，简要介绍了深度学习基础，深入分析并阐述了将分布式训练应用于大数据场景的必要性。其次，简略介绍了分布式训练中的关键组件，包括分布式算法的实现方式、并行方式、同步方式等，同时介绍了现有的分布式平台。最后，详细介绍了实验任务场景相关理论，包括高光谱目标检测和自然分类两个下游任务及其相关数据集和评测指标，为后续的实验设计和结果分析提供支撑。

第三章，基于近似质心的梯度压缩算法。本章首先对网络模型中的局部梯度进行分析，深入探讨了局部梯度的分布特性，为后续算法设计打下基础。然后，详细介绍了 GCAC 算法的相关理论，并给出了该算法应用于分布式训练的具体流程。最后，在不同规模、不同类型的图像数据集上，通过在高光谱目标检测以及图像分类两大任务场景进行实验，验证了该算法在分布式训练中的优势与不足，即在减少通信开销、

提高训练效率等方面优势，以及面对大压缩比场景下的局限性。

第四章，基于  $k$ -互异近邻的梯度压缩算法。首先，剖析了 GCAC 算法所面临的大压缩比难题，并指出导致该问题产生的原因，深入分析 GCAC 算法在大压缩比时所面临的挑战。其次，详细介绍了  $k$ -RNGC 算法的核心组件，包括局部梯度计算感知算法、 $k$ -互异近邻梯度选择算法、动量修正以及预热训练，并给出了各组件在分布式训练中应用流程。然后，通过组件分析实验以及耗时分析实验，验证了  $k$ -RNGC 算法核心组件的有效性以及必要性。最后，在不同规模的数据集上实施了高光谱目标检测和图像分类实验，实验结果证实体本章所提出算法的有效性和泛化性，充分展示了  $k$ -RNGC 算法在实际应用中的潜力和优势。

第五章，总结与展望。本章一方面总结了论文的主要研究内容；另一方面，基于对梯度压缩算法的充分了解，提出了对未来工作的展望。



## 第二章 相关理论基础

### 2.1 引言

目前以及将来的高光谱遥感图像数据，是数据规模非常大的 3-D 数据体，包括了空间位置信息和光谱信息。HSI 日益增加的数据规模对 ML 技术提出了更高的要求，尤其是在数据存储和处理方面，因此必须依赖足够大规模的并行/分布式训练技术。值得注意的是，梯度压缩算法在大程度上推动了分布式训练技术的发展<sup>[50][51][52][53]</sup>。因此，为了便于后续研究本论文关注的梯度压缩算法，本章首先介绍深度学习相关理论基础，紧接着对分布式理论基础进行阐述，然后介绍高光谱目标检测、图像分类等下游任务场景，最后介绍与下游任务场景相关的实验数据集和性能评价指标。

### 2.2 深度学习基础

#### 2.2.1 网络模型架构介绍

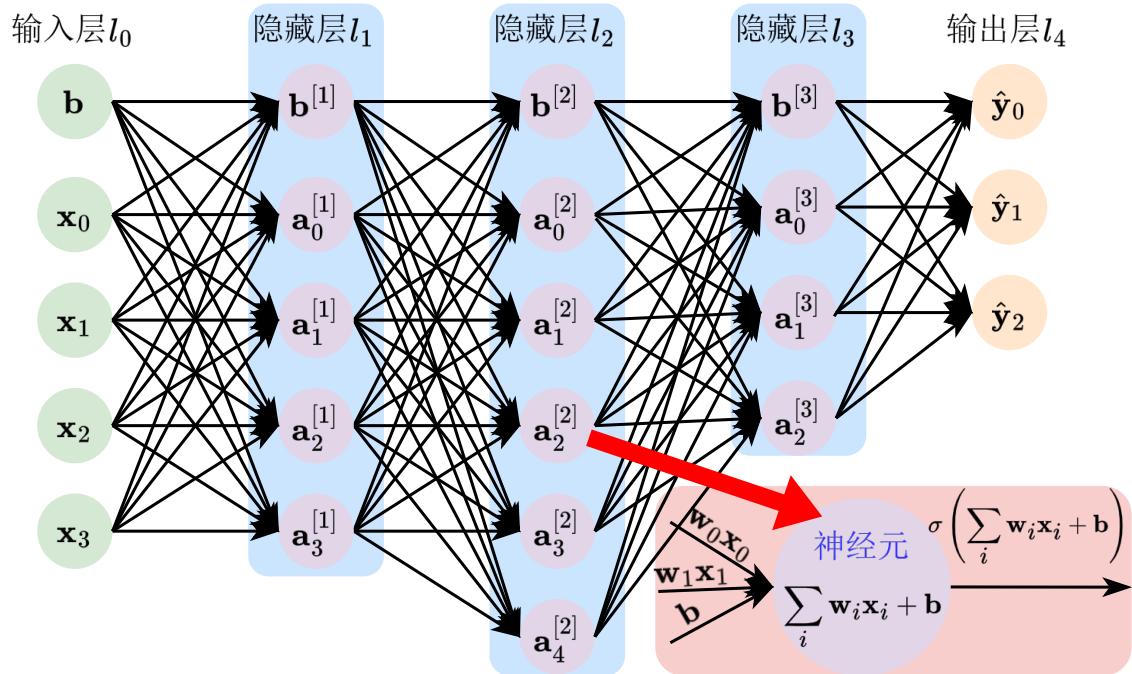


图 2.1 神经网络层级结构原理示意图

深度学习技术是机器学习领域的一个重要分支，层级人工神经网络模型通常依照输入层、输出层以及大量隐藏层的网络结构搭建而成，并堆叠了大量的非线性信息处理单元，凭借其网络结构对数据的表征、抽象能力强以及可执行任务灵活高的优

势，被广泛应用于计算机视觉、自然语言处理等众多领域。如图 2.1 所示，神经网络模型的每一层都包含大量相互连接的神经元（Neuron），而不同的神经网络架构正是体现在神经元连接方式和计算方式等方面的差异。当输入数据通过神经网络时，网络模型中每一层神经元上的权重将会进行适当调整，即神经元被激活或抑制，从而产生相应的输出。以卷积神经网络为例，与图 2.1 所示的全连接网络模型不同，CNN 以卷积层作网络架构核心组件，每一个卷积层都是由卷积计算单元、非线性计算单元以及可选的下采样变换单元构成，最后根据不同的下游任务选择合适的全连接层、分类器、检测器等后续组件。

在图 2.1 中， $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{n^{[l]}}\}$  表示输入数据， $\mathbf{b}$  为神经元偏置项（Bias）， $\mathbf{a}$  表示该神经元的激活值， ${}^{[l]}$  用于表明当前层位于该网络模型中第  $l$  层， $L$  表示整个网络模型的总层数， $n^{[l]}$  表示第  $l$  层的神经元数量， $\mathbf{w}_{i,j}^{[l]}$  表示第  $l-1$  层中第  $i$  个神经元与第  $l$  层中第  $j$  个神经元之间的参数， $\sigma(\cdot)$  表示非线性激活函数（Activation Function）。

## 2.2.2 激活函数简要介绍

神经网络技术中，激活函数定义了该神经元在给定的输入或输入集合情况下的输出，即定义了神经网络输出的数学方程式，如图 2.2 展示了一些常见激活函数的曲线分布。

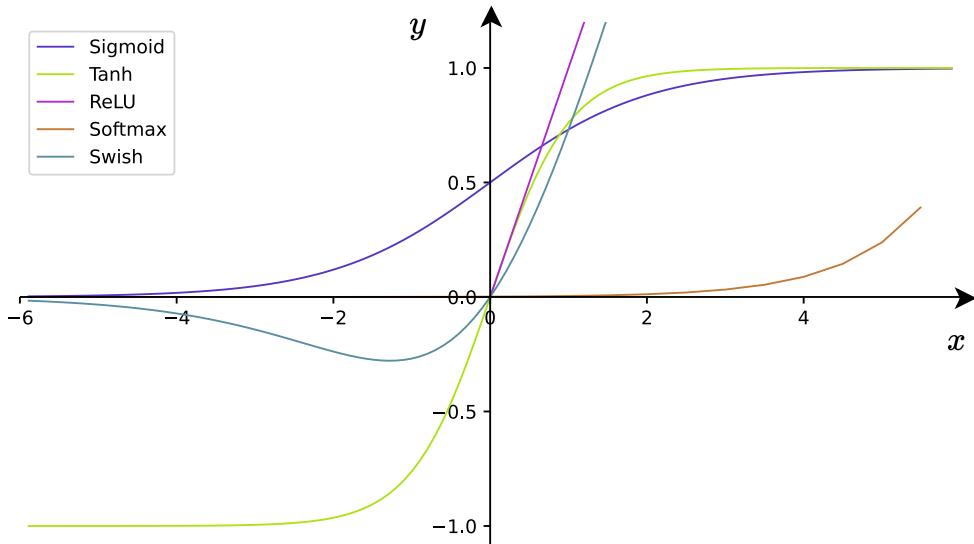


图 2.2 常见的激活函数曲线

Sigmoid 激活函数，数学描述如公式 (2-1) 所示，该函数的输出区间为  $(0, 1)$ ，相当于对每个神经元的输出进行归一化，常用于预测概率模型和神经网络的输出层。

$$\sigma_{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (2-1)$$

双曲正切（Tanh）激活函数，如公式 (2-2) 所示，输出区间为  $(-1, 1)$ ，且当输入

较大或较小时，保证输出平滑的同时，梯度较小，常用于隐藏层。

$$\sigma_{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2-2)$$

如公式(2-3)所示的ReLU激活函数，在深度学习领域中较为流行。当输入为正时，不存在梯度饱和的问题，且只存在线性关系，计算速度更快。但当输入为负，反向传播时梯度恒为零，ReLU激活函数完全失效。

$$\sigma_{ReLU}(x) = max(0, x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases} \quad (2-3)$$

Softmax激活函数，如公式(2-4)所示，该函数的输出范围为(0, 1)，且输出是总和为1的 $n^{[L]}$ 维实向量。该激活函数的分母结合了原始输出值的所有因子，因此可认为是argmax函数的概率版本。但该激活函数零点不可微，且当输入为负时梯度为零，常用于多分类问题。

$$\sigma_{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{t=1}^{n^{[L]}} e^{x_t}}, \quad i = 1, \dots, n^{[L]} \quad (2-4)$$

Swish激活函数，如公式(2-5)所示，其设计思路受到了长短期存储器(Long Short-term Memory, LSTM)和高速网络中Gating的Sigmoid函数使用的启发。当 $\rho$ 为1时，激活函数曲线如图2.2所示。该激活函数能够轻松替换以单个标量为输入的激活函数（例如ReLU），且无需修改隐藏容量或参数数量。Swish的“无界”特性有助于防止产生慢速训练期间梯度逐渐接近0的问题，并且该激活函数的平滑度在优化和泛化中起到了重要作用。

$$\sigma_{Swish}(x) = x * Sigmoid(\rho x) \quad (2-5)$$

### 2.2.3 网络模型训练流程介绍

深度学习技术尽管可以根据数据集是否具有标签信息划分为有监督和无监督两种训练范式，但是二者都对应数学上的凸优化问题。然而，无论是无约束条件的凸优化问题还是有约束条件的凸优化问题，都可以通过引入拉格朗日函数转换成无约束条件的凸优化问题<sup>[54]</sup>。因此，神经网络模型训练过程本质上就是求解一个无约束条件的凸优化问题的过程，如公式(2-6)所示：

$$\hat{\mathbf{W}} = \arg \min_{\mathbf{W}} \mathcal{L}(\mathbf{X}; \mathbf{W}) \quad (2-6)$$

其中， $\mathbf{X}$ 表示输入数据， $\mathcal{L}$ 通常被称为损失函数(Loss Function)，例如平均绝对误差(Mean Absolute Error, MAE)、均方误差(Mean Square Error, MSE)、交叉熵(Cross

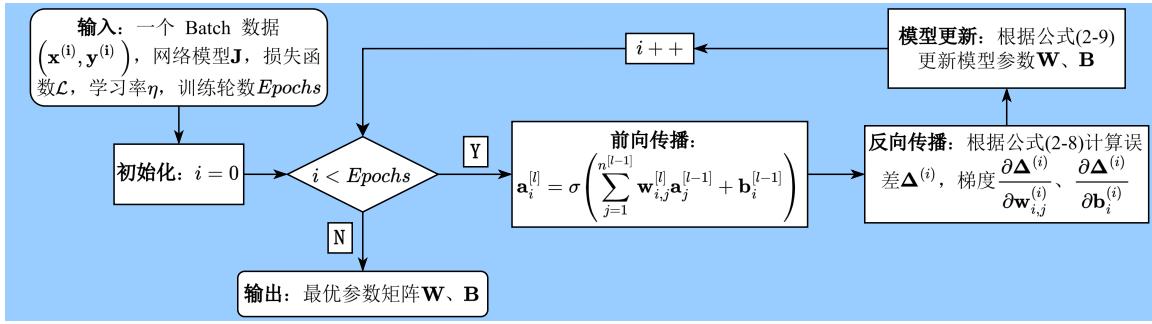


图 2.3 单节点网络模型 SGD 训练流程

Entropy, CE) 等。

整个训练过程不断通过前向传播 (Forward)、反向传播 (Backforward Propagation, BP) 等步骤的交替迭代进行，以实现模型参数矩阵  $\mathbf{W}$  的最优化求解，如图 2.3 所示，其中的核心步骤描述如下：

1) **初始化网络模型参数**: 对网络模型中的参数矩阵  $\mathbf{W}$  和偏置项  $\mathbf{b}$  进行初始化，可采用包括随机初始化、零值初始化、特征初始化等初始化方式。

2) **前向传播**: 将训练数据  $\mathbf{X}$  输入到网络中，并依据 **输入层**  $\Rightarrow$  **隐藏层**  $\Rightarrow$  **输出层** 的网络结构依次计算各层神经元的激活值  $\mathbf{a}_i^l$ ，并进一步求得实际输出  $\hat{\mathbf{Y}}$ ，如公式 (2-7) 所示：

$$\mathbf{a}_i^l = \sigma \left( \sum_{j=1}^{n^{[l-1]}} \mathbf{w}_{i,j}^l \mathbf{a}_j^{l-1} + \mathbf{b}_i^{l-1} \right) \quad (2-7)$$

3) **反向传播**: 与前向传播相比，该步骤数据流的方向恰恰相反，即依据 **输出层**  $\Rightarrow$  **隐藏层**  $\Rightarrow$  **输入层** 的方向，并根据预先定义的损失函数计算实际输出  $\hat{\mathbf{y}}$  与目标  $\mathbf{y}$  之间的误差值  $\Delta$ ，然后采用链式求导规则计算每一层的梯度  $\mathbf{g}$ ，如公式 (2-8) 所示：

$$\begin{aligned} \Delta &= \mathcal{L}_{MSE} (\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{M} \sum_{m=1}^M \sum_{i=1}^{n^{[L]}} (\hat{\mathbf{y}}_i^{[L]} - \mathbf{y}_i)^2 \\ \mathbf{g}(\mathbf{w}_{i,j}^l) &= \frac{\partial \Delta}{\partial \mathbf{w}_{i,j}^l} = \frac{\partial \Delta}{\partial \mathbf{a}_i^l} \frac{\partial \mathbf{a}_i^l}{\partial \mathbf{w}_{i,j}^l} = \left( \sum_{k=1}^{n^{[l+1]}} \frac{\partial \Delta}{\partial \mathbf{a}_k^{[l+1]}} \frac{\partial \mathbf{a}_k^{[l+1]}}{\partial \mathbf{a}_i^l} \right) \frac{\partial \mathbf{a}_i^l}{\partial \mathbf{w}_{i,j}^l} \\ \mathbf{g}(\mathbf{b}_i^l) &= \frac{\partial \Delta}{\partial \mathbf{b}_i^l} = \frac{\partial \Delta}{\partial \mathbf{a}_i^l} \frac{\partial \mathbf{a}_i^l}{\partial \mathbf{b}_i^l} = \left( \sum_{k=1}^{n^{[l+1]}} \frac{\partial \Delta}{\partial \mathbf{a}_k^{[l+1]}} \frac{\partial \mathbf{a}_k^{[l+1]}}{\partial \mathbf{a}_i^l} \right) \frac{\partial \mathbf{a}_i^l}{\partial \mathbf{b}_i^l} \end{aligned} \quad (2-8)$$

其中， $M$  为目标种类数量， $\mathbf{g}(\mathbf{w}^l)$  和  $\mathbf{g}(\mathbf{b}^l)$  表示第  $l$  层神经元相关梯度，为了方便后续描述以及公式清晰化，相关梯度将简化表示为  $\mathbf{g}^l$ 。

4) **网络模型参数更新**: 基于公式 (2-8) 求得的梯度，结合预设的优化算法（如 SGD、Adagrad、Adam 等）完成各层神经元的参数更新。以 SGD 优化算法为例，网

络模型参数更新如公式 (2-9) 所示:

$$\begin{aligned}\mathbf{w}_{i,j}^{[l]} &= \mathbf{w}_{i,j}^{[l]} - \eta \frac{\partial \Delta}{\partial \mathbf{w}_{i,j}^{[l]}} \\ \mathbf{b}_i^{[l]} &= \mathbf{b}_i^{[l]} - \eta \frac{\partial \Delta}{\partial \mathbf{b}_i^{[l]}}\end{aligned}\quad (2-9)$$

其中,  $\eta$  表示模型训练过程中的学习率 (Learning Rate, LR)。

5) 迭代: 重复步骤 2) 至步骤 4), 直至满足预设条件或模型收敛, 输出最优参数矩阵。

## 2.3 分布式训练理论基础

结合大数据时代以及神经网络模型训练过程, 不难发现, 整个训练过程涉及到的数据量、计算量都非常大。在理论层面再次表明, 现在以及将来的深度学习训练场景必然对设备存储、计算等要求暴增。无论是总线拓扑 (单机多 GPU) 还是网状/星型拓扑 (多机多 GPU) 的多设备并行/分布式训练, 不仅满足存储和计算要求, 而且在理论上不借助其他算法就可以加速网络训练速度, 可以极大程度地发挥深度学习的优势, 必将成为主流训练范式。

分布式训练是一种采用“分而治之”思想的训练技术。该训练技术将一个训练任务分配到多个计算节点上进行并行计算, 旨在提高模型训练速度和模型质量。该技术已被广泛应用于深度学习领域, 尤其是面对大规模数据集和复杂的网络模型, 例如超大规模自然图像处理、高光谱目标检测、自然语言处理等众多任务领域<sup>[55]</sup>。另外, 目前常见的 DL 框架<sup>[56][57][58][59][60]</sup>均已支持分布式训练, 可见其发展势头之猛。

分布式训练过程通常会协同多张 GPU, 以提供足够的显存、计算能力以及并行度, 极大程度提高训练速度。分布式算法的核心思路是将一个任务划分为多个小型任务进行并行计算。详细地说, 分布式训练是首先基于分布式网络拓扑架构, 将一个大型的计算任务划分为许多小型任务, 并交由相应的节点完成并行计算, 然后在各计算节点之间采用同步算法进行聚合、更新, 以确保本次迭代过程中有效节点的计算结果都成功用于模型梯度、参数更新, 重复迭代整个训练过程直至模型收敛。需要注意的是, 计算节点对应的是计算机中的进程, 而各个进程并不要求与硬件设备一一对应, 也就是说多个节点可以部署在同一台设备上 (即单机多 GPU 拓扑), 也可以部署在多台设备上 (即多机多 GPU 拓扑)。

接下来本章各小节将对分布式训练的并行方式、设备拓扑架构、各节点之间的同步更新方式以及现有分布式平台等方面进行深入介绍。

### 2.3.1 分布式训练的并行方式

目前，从并行方式的角度出发，现有的分布式训练技术主要有以下几种实现方式：1) 数据并行 (Data Parallelism)<sup>[61]</sup>、2) 模型并行 (Model Parallelism)<sup>[62]</sup>、3) 流水线并行 (Pipeline Parallelism)<sup>[63]</sup>。

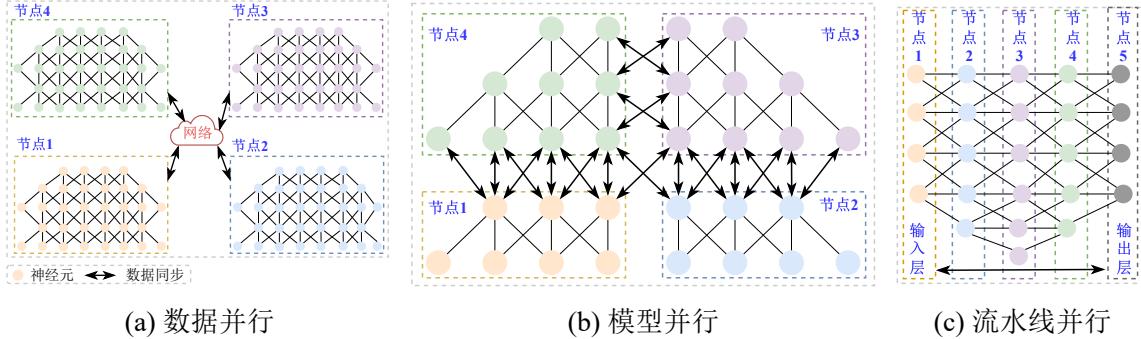


图 2.4 数据并行、模型并行和流水线并行示意图

1) **数据并行**: 众所周知，用于训练的数据量越大，神经网络模型的效果越好，但如今单计算节点显存往往不足以存储日益增加的数据。自然而然的解决方案就是将海量数据进行分割并分配到各个计算节点，让每个节点只负责训练本地的子数据集，这种划分子数据集进行并行计算的训练模式就是数据并行，如图 2.4 (a) 所示。后来逐渐发展成为一种广泛应用于深度学习技术领域的学习范式。在网络模型被完整部署到每个计算节点的前提下，数据并行将超大规模的数据集进行随机且均匀地分割成多个子数据集，每个计算节点只需负责训练被分配到的本地子数据集，然后与其余计算节点进行网络模型信息的同步以及更新。

由 2.2.3 小节介绍可知，网络模型的整个训练过程是一个不断迭代执行的过程，也就意味着整个训练过程将会进行大量迭代，同时模型的参数矩阵正是在每次迭代中逐渐优化，而每次迭代都需要在节点之间传输大量数据（模型权重和/或梯度），势必会拖慢网络模型的整体训练节奏。总之，数据并行在实际应用中也面临着巨大的挑战：第一，现在以及将来的神经网络模型体积越来越大，网络参数爆炸式增长，导致分布式训练梯度同步过程中的通信数据量越来越多，逐渐成为限制模型训练的关键因素。第二，近几年，GPU 和其他神经网络加速器等设备的计算能力取得了飞跃式的进步，但用于各计算节点之间的网络设备带宽的发展相对迟缓<sup>[64]</sup>。这意味着，算力提升与网络带宽之间的发展不匹配，导致计算节点的计算能力无法完全用于提升网络模型的训练速度，最终将会使得网络通信的瓶颈问题暴露愈加明显。

2) **模型并行**: 大型深度学习网络模型，通常包含数百万甚至数十亿个参数，如表 2.1 所示。当使用 GPU 进行网络模型训练加速时，由于单个计算节点上的 GPU 显存总是受限，往往无法将网络模型中的参数矩阵和梯度矩阵以及用于前向传播、反向

传播时的辅助变量全部加载到显存中。例如，Nvidia 公司推出的 GeForce RTX 2080Ti GPU 采用 Turing 架构，有效显存只有 11GB，显然易见，当神经网络模型体积、参数量过大时，单节点单 GPU 的情形下可能需要数天或数周时间，甚至连模型都可能无法顺利部署。

表 2.1 常见大规模深度学习网络模型参数

模型	ResNet-50	VGGNet-19	ELMo	BERT-Base	BERT-Large	GPT-2 Large	GPT-2 XL	GPT-3
参数量（亿）	0.256	1.43	0.078	1.1	3.4	7.75	15	1750

为了解决该问题，将大规模网络模型进行划分的训练模式被提出，让每一个计算设备仅仅负责管理、训练被分配到的部分模型，为大规模神经网络的训练提供一种可行方案，也就是模型并行，如图 2.4 (b) 所示。模型并行根据训练集群中的计算节点数量将整个模型进行切分，每一个单独的计算节点将部署一个部分模型，并负责训练整个数据集。多个计算节点并发执行，并定期与其他节点交换信息以更新模型。通过使用模型并行的训练策略，不仅能够使超大规模的网络模型如期部署，而且还可以大幅缩短分布式训练时间。

3) 流水线并行：Google 最初在 GPipe<sup>[63]</sup> 提出流水线并行。该并行方式将分布式训练过程中的计算任务划分为多个阶段，然后将这些阶段任务分配给不同的计算节点进行并行训练，以此来减少模型训练时的显存占用，如图 2.4 (c) 所示。PipeDream<sup>[65]</sup> 由 MicroSoft 提出，通过模型参数的异步更新以及 1F1B 调度策略减少并行气泡。由于 1F1B 尽早释放占用内存，因此在内存占用方面优于 GPipe，但同样可能存在模型不收敛以及内存膨胀的问题。近年来，DAPPLE<sup>[66]</sup> 在于 1F1B 之基础上采用模型参数同步更新，巧妙地避开了这些问题。另外，分布式并行计算训练场景中，由于不同的计算节点之间需要交换数据进行同步，增加了通信开销，而通信开销大小取决于训练数据集的规模和网络模型的复杂程度，流水线并行策略对于小规模分布式训练可能不会带来明显的性能提升。

尽管分布式训练的三种并行策略各有长短，但在深度学习技术领域的训练场景中，数据并行策略可保证神经网络模型快速收敛，实现简单，额外的性能开销相对不大且扩展性极强，应用范围最为广泛，逐渐成为分布式训练的主流范式。因此，本文将重点关注数据并行策略下的梯度压缩技术研究。

### 2.3.2 分布式训练的拓扑架构

本文 2.3.1 小节介绍了数据并行策略，规定了输入数据的划分方式，但没有描述分布式训练场景中的各个计算节点之间的网络拓扑。根据网络中各计算节点的通信拓扑结构，数据并行策略下的分布式训练架构主要分为以下两种：1) 中心化的参数

服务器 (Parameter Server, PS) 拓扑架构, 如图 2.5 (a) 所示; 2) 去中心化的 Collective 架构, 如图 2.5 (b) 所示。

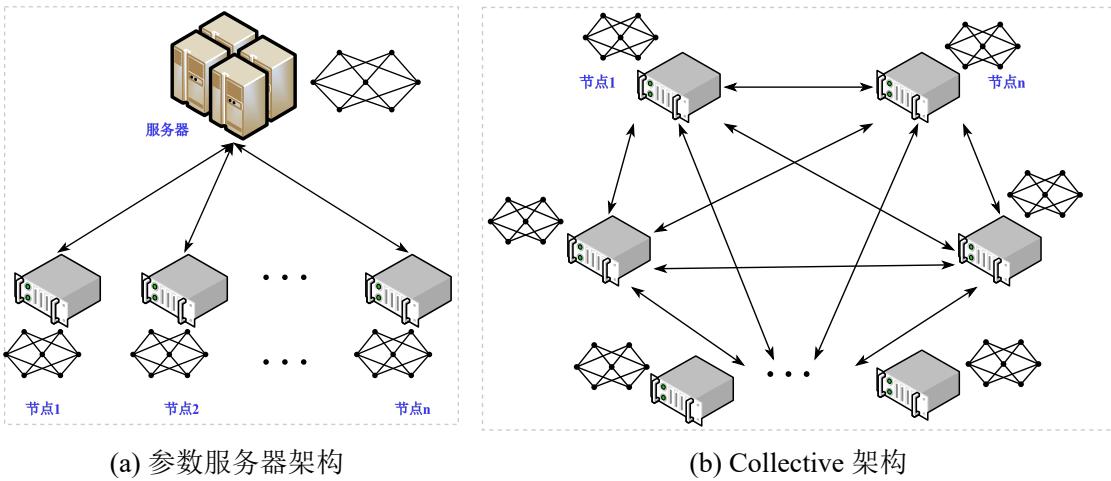


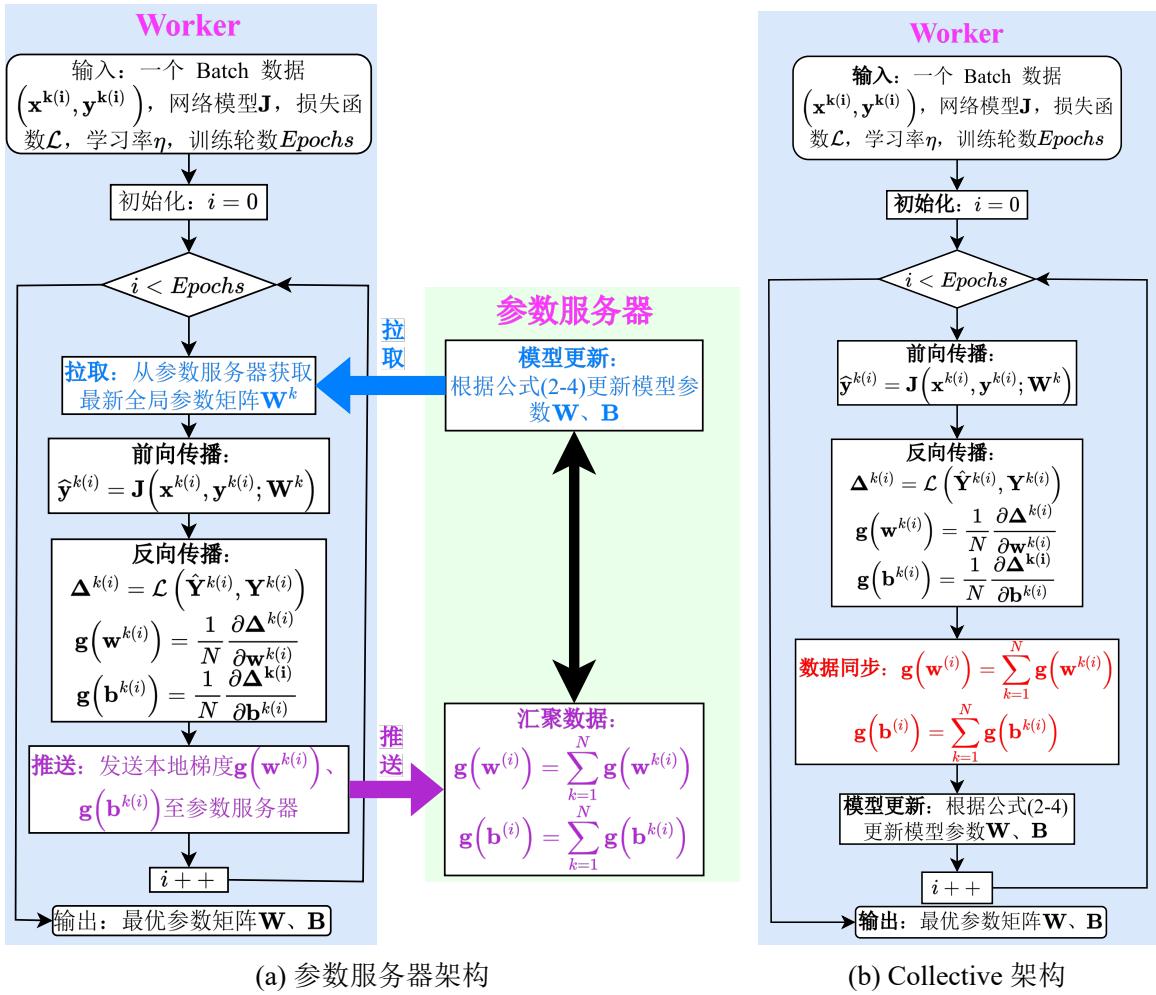
图 2.5 数据并行策略下分布式训练场景中不同的通信拓扑架构<sup>[67]</sup>

1) **参数服务器架构:** 参数服务器架构<sup>[68]</sup>是一种中心化 (Centralized) 的分布式训练通信拓扑, 它通常由一个服务器 (Server) 和若干计算节点 (Worker) 构成, 此时便构成了星型拓扑结构。另外, 当采用这种星型拓扑架构时, 各个 Worker 之间相互隔离, 不会发生数据交换操作, 只会与 Server 之间通过拉取 (Pull) 和推送 (Push) 通信原语的方式进行数据交换。

在参数服务器网络拓扑中, Server 持有整个神经网络模型的全部参数, 负责参数的分片存储和更新; Worker 持有模型网络结构, 根据分配到的子训练数据集进行模型前向传播和反向传播的并行计算, 得到网络模型对应的参数和梯度, 分布式训练的大致过程如图 2.6 (a) 所示。Worker 每经过一次迭代训练, 都可以向 Server 发送数据交换请求, 获取网络模型的最新参数, 然后基于该参数计算当前训练子数据集的梯度, 择机推送给服务器; Server 等待所有 Worker 推送完毕后进行梯度聚合, 然后采用 SGD 等优化算法进行全局参数更新。

参数服务器网络通信架构的优势有: 1) 实现简单、原理清晰易于理解; 2) 支持在不重启全部设备的情况下, 动态更新 Worker 的数量; 3) 同时支持分布式网络模型的同步训练和异步训练, 能够缓解因某个 Worker 计算过慢而导致的同步时间过长问题。关于分布式训练过程中的网络模型的同步/异步更新将在本章的下一小节进行详细介绍。

2) **Collective 架构:** 参数服务器网络通信架构致命缺点是, 随着 Worker 数量的增加, Server 与各个 Worker 之间的网络通信带宽受限问题暴露的愈加明显, 严重阻碍了分布式训练的速度, 该问题被称为通信瓶颈。为了解决该问题, 仅仅通过增加 Server 的数量并不一定能够可行, 因为目前主流的启发式的参数划分方案仍可能导致



(a) 参数服务器架构

(b) Collective 架构

图 2.6 不同拓扑架构下同步更新的网络模型训练流程

各个 Server 的任务负载不均衡<sup>[69]</sup>, 从而导致多对一的通信瓶颈问题。

目前, 解决了参数服务器网络拓扑架构中通信瓶颈问题的方案是去中心化 (Decentralized) 的 Collective 架构。在该架构中, 没有模型参数管理 Server, 每一个 Worker 都是平等的, 都持有网络模型的最新全局信息, 执行相同的工作, 但各个 Worker 之间需要通过多次通信操作, 以实现梯度同步, 并且网络通信时需要使用 Allreduce 原语。

聚合通信, 来源于 HPC (High Performance Computing) 领域的消息传递接口 (Message Passing Interface, MPI) 技术, 它能够在多个进程之间进行高效通信, 被广泛用于大规模数值计算领域, 包括流体力学模拟、天气预报等。Allreduce 则是众多聚合通信原语 (Reduce、Scatter、Allreduce、Allgather 等) 中的一种, 它支持所有 Reduce 规则运算, 包括求和、求平均值以及求最值等。在数据并行策略的分布式训练过程中, 使用 Allreduce 原语对网络模型的梯度、参数等信息进行聚合操作, 可以大幅改善其训练速度。

图 2.6 (b) 展示了基于 Allreduce 通信原语的数据并行策略下的分布式训练过程：首先，Worker 加载网络模型并读取子训练数据集，并进行前向、反向传播，获取局部梯度；然后，各 Worker 通过 Allreduce 原语实现梯度聚合、交换，得到最新全局梯度；最后，各 Worker 基于最新全局梯度进行模型参数更新，进入下一次迭代训练过程。

Allreduce 原语在 MPI 技术中仅仅被规定用来全局性地规约数据，并没有给出具体的实现。因此，不同的实现方式在通信次数、通信数据量等方面存在差异，表 2.2 列出了目前常见 Allreduce 原语的实现方式，包括 Reduce + Broadcast、Recursive Halving and Doubling、Butterfly 和 Ring Allreduce 等，其中， $N$  表示网络通信拓扑架构中的 Worker 数量， $\alpha$  表示单次通信固定的启动花销， $\beta$  表示发送单位数据量的时延 ( $\beta = 1/\text{带宽}$ )。

表 2.2 Allreduce 原语常见实现的通信时间<sup>[29]</sup>

实现逻辑	通信时间
Reduce + Broadcast	$2(\alpha + \beta)N$
Recursive Halving and Doubling	$2(\alpha + \beta) \log_2 N$
Butterfly	$(\alpha + \beta) \log_2 N$
Ring Allreduce	$2(\alpha + \frac{\beta}{N})(N - 1)$

虽然 Allreduce 原语的实现各种各样，但 Patarasuk 等人在 2009 年提出的 Ring All Reduce 算法<sup>[70]</sup> 在分布式训练中最受欢迎。Ring All Reduce 算法采用较少的点对点通信轮数，就可以完成各个 Worker 之间的全局模型信息的同步，同时它还被证明是带宽最优的。Baidu 在 2017 年首次将 Ring All Reduce 算法应用到大规模神经网络模型的训练场景中。值得注意的是，Uber 开源的 Horovod 框架<sup>[34]</sup>改善了现有的分布式训练通信模块，协同 MXNet<sup>[71]</sup>、TensorFlow<sup>[56]</sup>、PyTorch<sup>[72]</sup>等深度学习框架，使用 Ring Allreduce 算法加速了分布式训练过程。

由于网络通信拓扑中的所有 Worker 都使用了相同的 Allreduce 接口调用，因此，Allreduce 算法的引入导致在分布式训练过程中，全局梯度更新操作只能采用同步的方式，而无法采用异步更新。这意味着，基于 Allreduce 算法的分布式训练不支持异步训练以及动态更新网络拓扑，健壮性和容错性也相对较差。

### 2.3.3 分布式训练的同步方式

在了解了分布式训练的并行方式、网络拓扑架构之后，本小节将介绍分布式训练的同步方式。分布式训练过程中，各个 Worker 不仅需要根据训练子数据集进行并

行计算，得到各自的局部梯度，还需要进行梯度聚合以更新网络模型，而同步方式恰恰控制了各个 Worker 的训练进度。目前现有的分布式训练同步方式可以分为同步（Synchronize）更新与异步（Asynchronize）更新两种。

**同步更新方式：**这种同步方式需要在每一轮迭代结束后，各个 Worker 相互等待对方计算结束，然后根据具体拓扑架构采取合适的通信原语进行数据交换，最后完成梯度聚合、参数更新并进入下一轮的迭代训练过程。

该同步方式可以控制每一个 Worker 的训练进度保持一致，能保证分布式训练与单机训练时的算法等价性，有利于神经网络模型快速收敛。然而，同步更新的缺点就是网络拓扑中的各个 Worker 需要相互等待，可能发生掉队者（Straggler）问题，即由于计算能力的不同，某些 Worker 长期处于等待状态，造成计算资源浪费。

**异步更新方式：**当分布式训练采用异步更新的同步方式时，对网络拓扑中各 Worker 的模型、计算能力、存储能力等方面没有一致性的要求，它们在每次独立的并行迭代计算完成后，不需要相互等待。这种同步方式能够使计算集群的资源利用率最大化，但可能会导致各个 Worker 上的模型信息不一致，以致于网络模型收敛速度慢甚至发散的问题。

在实际分布式训练应用场景中，大规模网络模型训练任务往往部署在 Worker 同质化程度较高的集群上，即网络拓扑中各个 Worker 的计算能力、存储能力、网络通信带宽基本一致。因此，本论文提出的梯度压缩算法是都基于数据并行策略和同步更新方式的 Collective 拓扑架构之上。

### 2.3.4 现有分布式平台介绍

在深度学习技术领域，无论分布式拓扑架构采用 PS 亦或是 Collective，它们都会在现有通信库的 API 基础上实现底层通信操作接口。

在 PS 网络拓扑架构中，由于点对点（Point to Point, P2P）通信请求频繁，一般采用 Socket 实现进程间通信。如 Google 开源的远程过程调用（Remote Procedure Call, RPC）库，也就是 gRPC，作为 PyTorch 和 TensorFlow 的底层通信库。gRPC 采用 Protobuf 数据序列化格式进行网络模型的梯度、参数交换，大大提高了通信传输速率。MXNet 则采用了消息队列库 ZeroMQ 进行底层数据通信。ZeroMQ 是一个开源的分布式消息传递库，采用了零拷贝技术，旨在提供高性能、高可靠和高可扩展性的消息传输，支持多种消息传递模式，包括一对一、一对多、多对一和多对多等多种拓扑架构。

Collective 网络架构则有更多的通信库可供选择，例如，MVAPICH 实现了 MPI 标准协议的同时进行了一些列的优化工作<sup>[73][74][75][76]</sup>。NVIDIA 开发的通信库 NCCL (NVIDIA Collective Communications Library) 对单机多节点和多机多节点的 GPU 集

群应用场景进行了优化，NCCL 通过指定 CUDA 核函数使用固定数量的多线程进行处理，实现了数据计算与高效快速搬移的并行处理。GLOO 通信框架由 Facebook 提出并开源，基于 MPI、NCLL 等代码库之上，实现并优化了特定场景下分布式深度学习领域中的常见通信算法。

主流的深度学习框架主要有 PyTorch<sup>[72]</sup>、TensorFlow<sup>[56]</sup>、MXNet<sup>[71]</sup>等，并随着分布式训练的流行，都引入了分布式的相关算法。PyTorch 提供了丰富的分布式训练 API，包括 DataParallel、DistributedDataParallel 等，可以快速上手，易用性较高，支持单机多 GPU、多机多节点训练。TensorFlow 同样提供了易用的分布式 API，在大规模训练任务中性能表现更好，可扩展性更强。MXNet 分布式训练性能也很出色，几乎可以与 TensorFlow 相媲美，支持大规模训练任务。Uber 开源的 Horovod 分布式训练库，可以 PyTorch、TensorFlow、MXNet 等框架配合使用，性能非常出色，分布式训练的 API 相对易用。截至目前，Horovod 在 GitHub 上的 Star 数量接近 14K，受到广大用户的喜爱。因此，本论文中所涉及的验证实验都将部署在 Horovod + PyTorch 的分布式平台上。

## 2.4 任务场景介绍

### 2.4.1 高光谱目标检测流程

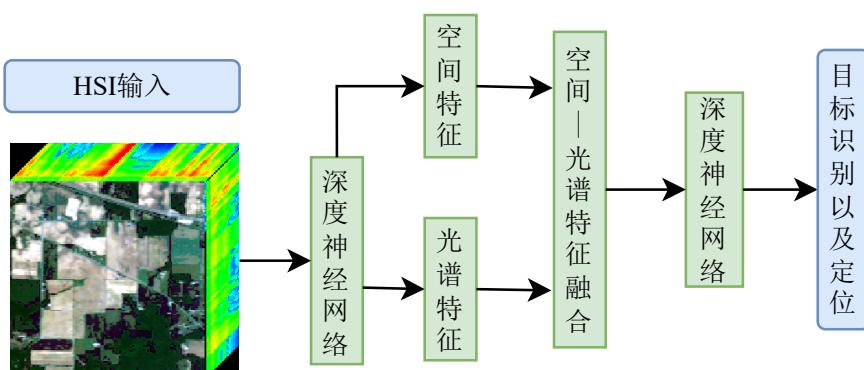


图 2.7 高光谱目标检测大致流程示意图

相较于传统遥感图像而言，高光谱遥感图像中目标场景更加多样、地物的空间和光谱两大特性更加精细，因此，通过 HSI 进行目标检测任务（例如物质勘测、海洋探测等）具有重要意义。HTD 已被广泛应用于监测和实施农业、地质、环境资源评估、城市规划、军事/国防、灾害管理等众多领域<sup>[77][78][24]</sup>。

从本质上讲，目标检测可以看作是分类和定位的问题<sup>[79][80]</sup>。目标检测算法首先通过将目标场景中感兴趣目标的光谱信息和空间信息进行融合，然后基于融合信息将 HSI 划分为背景和目标两大类别，最后通过各种现有的检测算法进行识别与定位。图 2.7 展示了高光谱目标检测任务场景的大致流程，其核心步骤主要包括以下三步：

- 1) 利用深度学习技术提取高光谱图像中高级别层次的光谱、空间特征信息；
- 2) 将提取到的空间特征信息和光谱特征信息进行融合；
- 3) 采用 Fast R-CNN、YOLO 等现有目标检测算法进行识别与定位。

### 2.4.2 相关网络模型介绍

本小节将介绍本文实验中涉及到的网络模型，主要包括对抗自编码网络（Adversarial Autoencoder, AAE）、VGGNet、ResNet 等。

**AAE:** 2015 年 Makhzani 等人<sup>[81]</sup>将先验分布与 AE 隐空间相结合，并在 AE 和判别器的基础之上提出了 AAE 网络，具体网络结构如图 2.8 所示。在该图中，上面一行为 AE 网络模型， $q(\mathbf{z}|\mathbf{x})$  表示编码器网络模型，负责对输入数据进行学习、重构， $p(\mathbf{x}|\mathbf{z})$  表示解码器网络模型， $\mathbf{x}$  和  $\hat{\mathbf{x}}$  分别表示编码器的输入数据和解码器的重构结果， $\mathbf{z}$  表示 AE 网络模型隐空间的输出结果。该图中下面一行的网络模型被称为判别器，负责区分输入样本是预设的概率分布还是 AE 网络的隐空间输出， $p(\mathbf{z})$  表示概率分布。AE 网络模型隐空间的后验分布表示如下：

$$q(\mathbf{z}) = \int_x q(\mathbf{z}|\mathbf{x}) p_d(\mathbf{x}) dx \quad (2-10)$$

生成器的目标是生成分布相似高质量数据，以欺骗判别器，而判别器的目标则是准确地区分生成的数据和真实的数据。与传统 GAN 网络模型相比，AAE 网络不仅表现更加稳定可靠，而且可以生成与训练数据分布相似的高质量重构数据，常用于图像生成、数据增强、目标检测等各种任务。

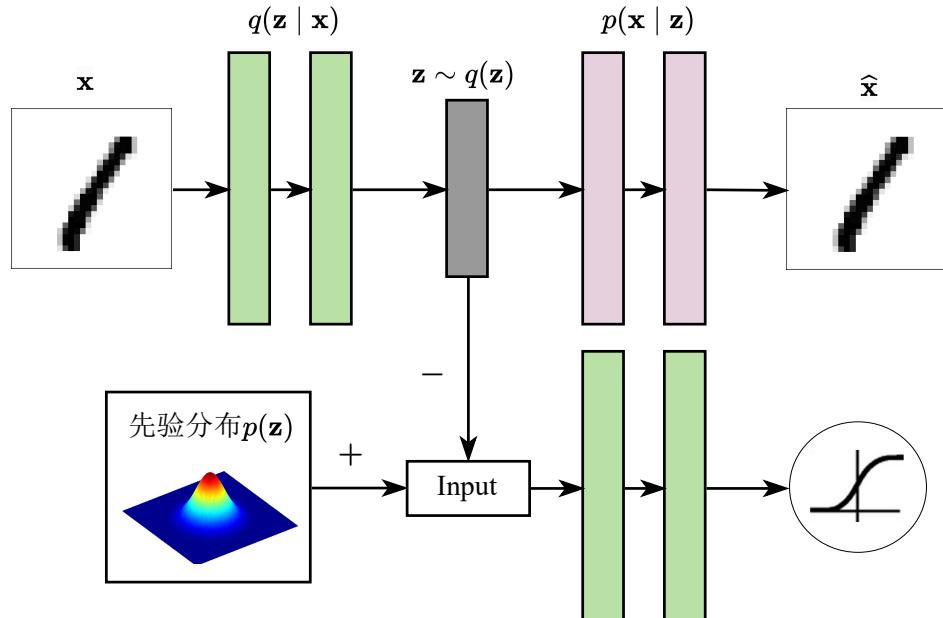


图 2.8 对抗自编码网络结构示意图<sup>[81]</sup>

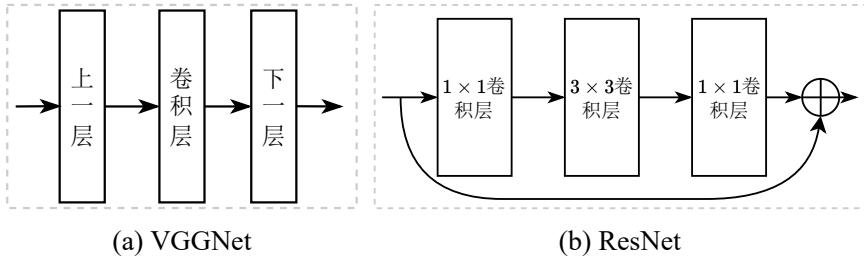


图 2.9 VGGNet 和 ResNet 网络模型核心结构示意图

**VGGNet:** VGGNet 是一种卷积神经网络模型，设计理念简单，将大量小卷积核 ( $3 \times 3$ ) 和池化层 ( $2 \times 2$ ) 堆叠成一个很深的网络结构，例如 VGG-16, VGG-19 等，如图 2.9 (a) 展示了该类型网络的核心结构。其中，VGG-16 包含 16 个卷积层和 3 个全连接层，是最受欢迎的一个网络模型变体。本文在自然图像分类场景下，会将提出的梯度压缩算法应用于该网络进行算法验证。

**ResNet:** ResNet 是一种残差块堆叠而成的深度神经网络，它的设计思想在于引入残差块，如图 2.9 (b) 所示。一个残差块包含多个卷积层以及一个恒等映射，将输入数据与卷积层的输出数据进行相加，从而帮助网络模型学习残差信息，解决了深度神经网络训练时梯度消失或爆炸的问题。相较于 VGGNet，该网络模型通常更深，具有更好的泛化能力，计算效率高的优点，但也具有计算量大、模型参数多等劣势。本文将在 ResNet-56 以及 ResNet-110 两个不同大小的网络模型上验证所提出的梯度压缩算法。

## 2.5 实验数据集与评价指标

### 2.5.1 实验数据集

**AeroRIT<sup>[82]</sup>:** AeroRIT 数据集是由罗彻斯特理工学院大学校园上空的塞斯纳飞机上的两种摄像系统采集得到，第一个采集系统是工作在 RGB 域的 80 magepixel (MP) 的框架型硅传感器，第二个采集系统使用的是可见红外 (VNIR) 高光谱 Headwall Photonics Micro Hyperspec E 系列 CMOS 传感器。该数据集中的图像数据包含 372 个波段，波长范围从 397 nm 到 1003 nm，空间尺寸分辨率为  $1973 \times 3975$ ，包含了将近 700 万个注释数据，却只被划分为 5 个类别。

由于该数据集的空间分布不规则，我们进行了适当的切割处理，即选取位于原始图像的某块矩形图像作为训练数据集，该矩形区域的对于应原始图像左上角和右下角的坐标分别为 (401, 101) 和 (1600, 3900)。因此，在评估实验中，对所有波段的图像数据来说，其裁切后的实际有效空间分辨率为  $1200 \times 3800$ ，高光谱目标和背景分别占用 48.67Mb 和 3196.13Mb 内存空间，如表 2.3 描述了验证高光谱目标检测实验中

的目标和背景的像素、占比等总体分布情况。如图 3.4 (a) 以及图 3.4 (b) 两幅子图所示，分别展示了该数据集的伪彩图和目标真实标签的分布状况。

表 2.3 Xiongan 和 AeroRIT HSI 中目标和背景的样本数量以及空间占比

数据集	类别	样本数量	占比 (%)
Xiongan New Area (Matiwan Village)	水	165647	2.80
	背景	5759353	97.20
AeroRIT	水	68576	1.50
	背景	4491424	98.50

**Xiongan New Area (Matiwan Village)**<sup>[83]</sup>: 2017 年 10 月，该航空高光谱遥感图像采用中国科学院技术物理研究所设计的可见光及近红外成像光谱仪拍摄。该数据集有 256 个波段，波长范围为 400 至 1000nm，空间分辨率为  $3750 \times 1580$ ，包含 19 种土地覆盖类别。在本文的验证实验中，“水”将被视为感兴趣的高光谱目标，而其他类别被视为背景样本，分别占用近 0.896GB 和 31.104GB 的内存空间，具体的像素数量及其空间占比比如表 2.3 所示。如图 3.5 (a) 以及图 3.5 (b) 两幅子图所示，分别展示了该数据集的伪彩图和目标真实标签的分布状况。为了简化后续图表标题，“Xiongan New Area (Matiwan Village)” 将被简化为 “Xiongan”。

**CIFAR10**: 该数据集包含 10 种具有标签的图片集合，它们相互独立、互不重叠。每一种类型的图片包括 5000 张训练样本以及 1000 张测试样本，每张图片大小为  $32 \times 32$ ，如图 2.10 展示了类别标签和一些图片样张。在本论文的自然图像分类实验中，我们将首先使用图片增强技术对该数据集进行简单预处理，然后再将预处理后的图像作为训练数据。

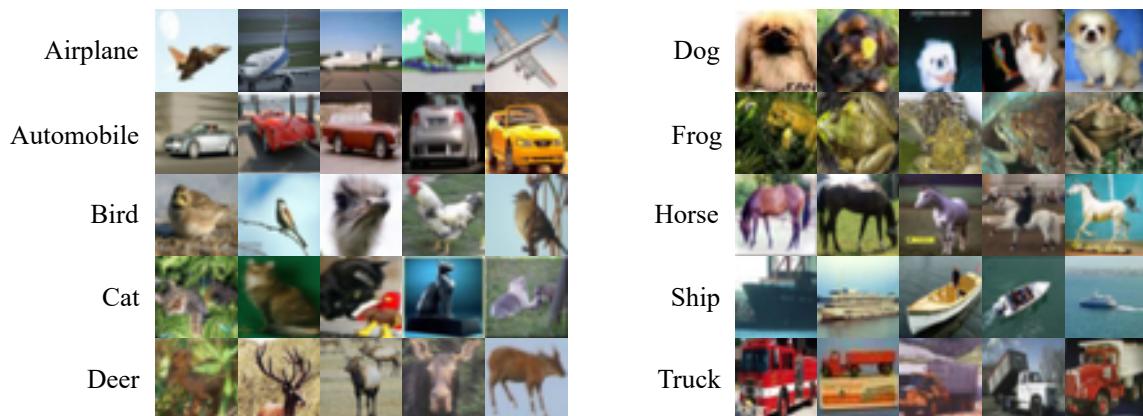


图 2.10 CIFAR10 数据集标签及其部分图片样张

## 2.5.2 评价指标

在目标检测领域，从评价个体的角度可以将性能指标分以下两种：主观评价、客观评价。主观评价，通常指的是人们基于实验结果，采用视觉与直观感受进行评价的一种评价指标，但评价结果会受到较强的主观因素的影响。客观评价，通常指的是采用某种量化数据对实验结果进行客观分析，可以很好的避免主观因素介入，应用更为广泛。本文将根据不同的下游任务进行不同性能评价指标的选择。例如，高光谱目标检测任务采用主观评价和客观评价，而自然图像分类任务将仅采用客观评价。接下来，本小节将对本文中涉及到的评价指标进行阐述。

ROC (Receiver Operating Characteristic) 曲线<sup>[84]</sup>：它是一种检测器性能的二维评价指标，是一条关于检测器的检测率和误报率的曲线。但为了通过单个标量来表示检测器的性能，通常采取的措施是计算 ROC 曲线下的面积，被称为 AUC<sup>[85]</sup>，如公式 (2-12) 所示。

$$\begin{aligned} F_0(\tau) &= \frac{TP(\tau)}{TP(\tau) + FN(\tau)} \\ F_1(\tau) &= \frac{TP(\tau)}{TN(\tau) + FP(\tau)} \end{aligned} \quad (2-11)$$

其中， $\tau$  表示给定的分割阈值，并定义分数大于等于分割阈值的像素为正样本， $F_0(\tau)$  表示检测率， $F_1(\tau)$  表示虚警率，给定阈值  $\tau$  时， $TP(\tau)$  表示被预测为正样本的目标像素数量， $FN(\tau)$  表示被预测为负样本的目标像素数量， $FP(\tau)$  表示被预测为正样本的背景像素数量， $TN(\tau)$  表示被预测为负样本的背景像素数量。虚警率 (False Positive Rate, FPR) 以及 AUC 计算公式 (2-12) 如下：

$$\begin{aligned} FPR &= \int_0^1 F_1(s) ds \\ AUC &= \int_0^1 F_0(s) dF_1(s) \end{aligned} \quad (2-12)$$

在本文高光谱目标检测任务场景中，将采用 AUC 和 FPR 作为其客观评价指标， $AUC(F_0(\tau), F_1(\tau))$  的值越大表示检测精度越高， $AUC(F_1(\tau), F_0(\tau))$  的值越小表示虚警率越低。

本文自然图像分类任务场景中，将采用 Top-k 分类精度 ( $acc_k$ ) 作为客观评价指标， $k$  通常设为 {1, 5}，具体来说，其计算公式 (2-13) 如下：

$$acc_k = \frac{N_{C^k}}{N_{C^T}} \quad (2-13)$$

其中， $N_{C^k}$  表示所有测试结果中前  $k$  类标签中正确的预测数量， $N_{C^T}$  表示所有测试图像的标签数量。

除了以上常用的评价指标之外，梯度压缩算法的复杂度、鲁棒性等性能指标也可作为目标检测任务、分类任务的参考指标。

## 2.6 本章小结

本章主要针对深度学习技术、分布式训练、高光谱目标检测任务、图像分类任务等涉及到的知识进行介绍。首先介绍了深度学习的相关基础理论知识，紧接着介绍了分布式训练相关的理论基础，并指出了目前大数据大模型场景下引入分布式训练的必要性，然后阐述了分布式训练中的并行方式、拓扑架构、同步方式等必要组件，同时对现有分布式平台进行简单介绍，并选择使用 Horovod 作为本论文实验的部署平台。接下来，介绍了高光谱目标检测以及图像分类的相关知识，包括检测流程以及相关网络模型。最后，对本文下游任务的相关实验数据集与评价指标进行阐述。



## 第三章 基于近似质心的梯度压缩算法

### 3.1 引言

本章首先根据第二章相关理论基础，发现并指出了目前分布式训练的技术难点，即高频的、全尺寸全精度数据格式的网络通信行为可能导致分布式训练速度慢，甚至无法在短时间内顺利完成训练。为了解决该技术难题，本章提出了基于近似质心的梯度压缩算法（Gradient Compression via Approximate Centroid, GCAC），该算法根据网络模型每一层的梯度分布状况选择接近质心的梯度进行优选，降低了不同 Worker 之间网络通信的数据量，并采用 Horovod + PyTorch 网络框架搭建分布式平台进行实现和验证。

本章第 3.2 节详细介绍提出的基于近似质心的梯度压缩算法；第 3.3 节对提出的基于近似质心的梯度压缩算法进行实验验证，主要包括高光谱目标检测和图像分类两大任务场景，并对实验结果进行分析；第 3.4 节对本章的研究工作进行总结。

### 3.2 基于近似质心的梯度压缩算法

#### 3.2.1 网络模型中局部梯度分析

在分布式训练过程中，虽然 Allreduce 算法优化加速了各 Worker 之间的梯度同步，但是待传输梯度数据量过大仍然可能导致通信瓶颈问题。为了减少分布式训练的通信开销并最大化利用分布式集群中各个 Worker 的利用率，我们将梯度压缩算法应用于分布式训练技术。同时，本节对梯度特征进行了实验性分析，以更好地实施梯度压缩算法。

目前，大多数梯度压缩算法以梯度值或范数大小作为重要性度量，通常需要满足以下两个隐含条件才能取得比较不错的效果：

- 1) 梯度值或范数越小，对模型优化贡献越小；
- 2) 梯度值或范数的标准差足够大。

然而，在实际训练过程中，上述提及的两个隐含条件不总是成立。因此，我们抓取并分析了局部梯度之间的冗余相关性，以便更好地裁剪冗余梯度。如图 3.1 (a) 展示了压缩前的梯度分布状况，可以看到，绝大多数梯度值较小且接近于 0。当采用基于梯度值的梯度压缩算法且压缩比设置为 2 时，压缩后的梯度分布状况如图 3.1 (b) 所示，其中位于设定阈值线（红色点状线）之间的梯度将被重置为 0，并且不会参与网络各 Worker 之间的网络通信。需要注意的是，当采用这种重要性度量时，梯度值或范数越大，通常该梯度对网络模型参数优化的贡献越大；但度量相对较小的梯度无

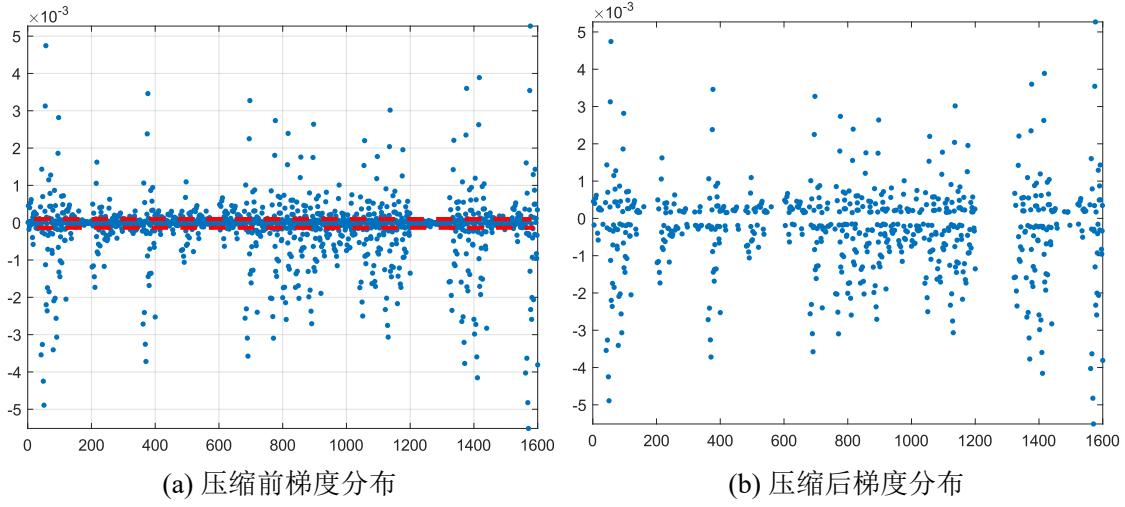


图 3.1 网络模型采用传统方法压缩前后的梯度分布

法被证明其重要性小，因为度量较小不一定是因为其梯度值或范数小，也可能是因为在分布式训练的一次迭代中所有梯度的值或范数都比较大。在这种情况下，裁剪掉梯度值或范数小的梯度无疑将严重影响网络模型的性能表现。另外，当局部梯度值或范数的标准差很小时，以压缩比为目标去选择一个合适的分割阈值通常是困难的。一旦选择的阈值不合适，大量梯度关键信息将会被裁剪，同样会导致网络模型性能下降严重。

基于以上分析可知，仅仅依赖梯度值或范数大小就决定某个梯度是否重要存在不可靠的情况。因此，我们在受到 Fletcher 等人<sup>[86]</sup>的启发，基于 Euclidean 空间中的质心揭示共同属性的假设，即质心等效于局部梯度张量的中心或公共属性，将梯度映射到 Euclidean 空间并计算梯度张量的质心，并通过分析 Euclidean 空间中的点进一步揭示各个梯度之间的冗余关系，提出一种基于质心的重要性度量方法，克服了传统方法不总满足的依赖条件。

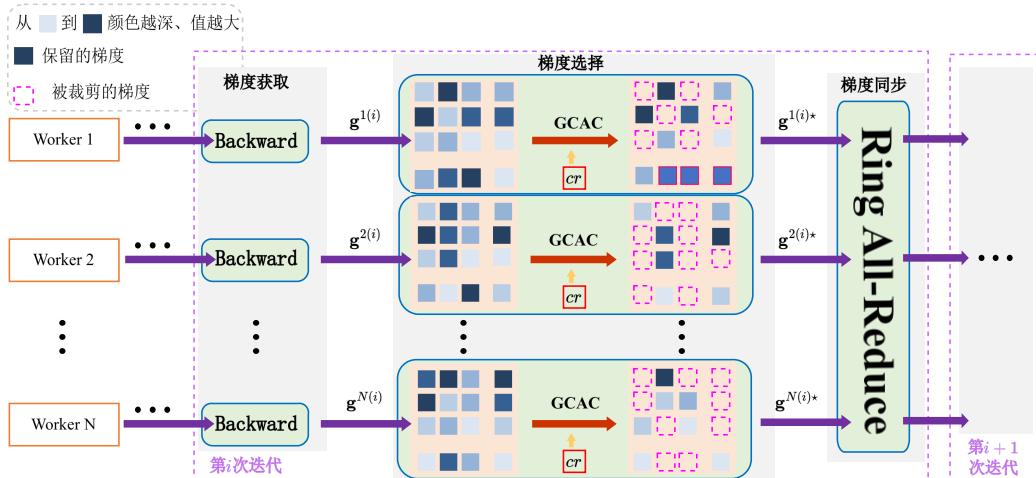


图 3.2 基于近似质心的梯度压缩算法应用框图

### 3.2.2 基于近似质心的梯度压缩算法

为了避免 3.2.1 小节所讨论的梯度值或范数依赖条件问题，本小节将介绍我们提出基于近似质心的梯度压缩算法。GCAC 算法的核心思想是从信息冗余的角度出发，替换掉那些可被代替的梯度。如图 3.2 为 GCAC 算法的应用框图，展示了 GCAC 算法如何在分布式训练过程中发挥作用。

第  $m$  个 Worker 上网络模型的第  $k$  层梯度  $\mathbf{g}^{m[k]} \in \mathbb{R}^{(n^{m[k]}+1) \times Q^{m[k]}}$  构成的梯度矩阵如公式 (3-1) 所示：

$$\mathbf{g}^{m[k]} = \begin{bmatrix} g_{0,0}^{m[k]} & \cdots & g_{0,Q^{m[k]}}^{m[k]} \\ \vdots & \ddots & \vdots \\ g_{n^{m[k]},0}^{m[k]} & \cdots & g_{n^{m[k]},Q^{m[k]}}^{m[k]} \end{bmatrix} \quad (3-1)$$

其中， $Q^{m[k]}$  表示  $m$  节点中第  $k$  层中神经元梯度的尺寸。局部梯度矩阵的质心  $\mathcal{C}^{m[k]}$  可通过计算公式 (3-2) 求得：

$$\begin{aligned} \psi(\mathcal{C}) &\triangleq \sum_{i=0}^{n^{m[k]}} \sum_{j=0}^{Q^{m[k]}} \left\| \mathcal{C} - g_{i,j}^{m[k]} \right\|_2, \\ \mathcal{C}^{m[k]} &= \arg \min_{\mathcal{C} \in \mathbb{R}^d} \psi(\mathcal{C}) \end{aligned} \quad (3-2)$$

基于已获得的质心，可通过公式 (3-3) 获取最接近质心的梯度  $\mathbf{g}_{i^*, j^*}^{m[k]}$ ：

$$\begin{aligned} \mathbf{g}_{i^*, j^*}^{m[k]} &= \arg \min_{g_{i', j'} \in \mathbb{R}^d} \left\| \mathbf{g}_{i', j'}^{m[k]} - \mathcal{C}^{m[k]} \right\|_2, \\ s.t. i' &\in \{0, 1, \dots, Q^{m[k]}\}, j' \in \{0, 1, \dots, Q^{m[k]}\} \end{aligned} \quad (3-3)$$

根据质心相关特性可知， $\mathbf{g}_{i^*, j^*}^{m[k]}$  可被本层网络模型中其他梯度线性表示，裁剪掉这些接近质心的冗余梯度并不会对网络模型参数优化产生太大的影响。值得注意的是，通过深入分析公式 (3-2) 以及公式 (3-3)，可得出以下两个结论：

1) 通过公式 (3-2) 求解局部梯度质心是极其耗时的：由于梯度矩阵中各梯度不满足均匀分布的特性，则需要进行遍历求解，时间复杂度为  $O(n^{m[k]} \times Q^{m[k]})$ ；

2) 接近质心的待裁剪梯度候选者  $\mathbf{g}_{i^*, j^*}^{m[k]}$  来源于本层梯度矩阵：基于公式 (3-2) 得到的质心  $\mathcal{C}^{m[k]}$ ，采用公式 (3-3) 进行梯度选择得到梯度候选者，且满足  $\mathbf{g}_{i^*, j^*}^{m[k]} \in \mathbf{g}^{m[k]}$ 。

因此，为了解决结论 1) 的梯度计算耗时问题，我们在充分利用结论 2) 的基础上，提出了一个求解该问题的加速计算技术。该加速计算算法是一个计算更方便、耗时更短的近似公式，其主要核心思想是最小化梯度候选者与其他梯度之间的距离之和，如公式 (3-4) 所示，其中， $\mathbf{g}_{i^*, j^*}^{m[k]}$  将作为本层梯度质心的近似替代者，称为“近似质心”，并在之后的计算中代替局部梯度质心的作用。为了方便后续描述，将  $\mathbf{g}_{i^*, j^*}^{m[k]}$  简

记为  $\hat{\mathcal{C}}^{m[k]}$ 。

$$\begin{aligned} \mathbf{g}_{i^*, j^*}^{m[k]} &= \arg \min_{\mu} \sum_{i'=0}^{n^{m[k]}} \sum_{j'=0}^{Q^{m[k]}} \left\| g_{i', j'}^{m[k]} - \mu \right\|_2, \\ \text{s.t. } & \left| g_{i', j'}^{m[k]} - \mu \right| > 0, \mu \in \left\{ g_{0,0}^{n[k]}, \dots, g_{Q^{m[k]}, Q^{m[k]}}^{m[k]} \right\} \end{aligned} \quad (3-4)$$

质心在 Euclidean 空间中是经典的数据中心估计器<sup>[86]</sup>，被选中的近似质心  $\hat{\mathcal{C}}^{m[k]}$  和剩下的梯度共享某些共同属性。换句话说，质心等效于局部梯度矩阵的中心或公共属性，然后将其余梯度视为一组基向量，这组基向量可以将接近质心的梯度  $\hat{\mathcal{C}}^{m[k]}$  完备地线性表示，也可以引导模型朝着最优的梯度下降方向进行优化。

另外，为了满足梯度压缩多样性的要求，必须对局部梯度进行排序操作。在根据公式 (3-4) 求得最接近质心的梯度候选者  $\hat{\mathcal{C}}^{m[k]}$  后，然后通过公式 (3-5) 进行计算“近似质心”与其他梯度之间的距离  $s$  (本文采用欧几里得范数作为距离度量)：

$$\begin{aligned} s_{j'}^{m[k]} &\triangleq \phi \left( \hat{\mathcal{C}}^{m[k]}, g_{i,j}^{m[k]} \right) = \left\| \hat{\mathcal{C}}^{m[k]} - g_{i,j}^{m[k]} \right\|_2, \\ \text{s.t. } \Lambda^{m[k]} &\triangleq n^{m[k]} \times Q^{m[k]}, \quad i \in [0, n^{m[k]}], \quad j \in [0, Q^{m[k]}], \quad j' \in [1, \Lambda^{m[k]}] \end{aligned} \quad (3-5)$$

并定义梯度距离和列表  $\mathcal{S} \in \mathbb{R}^{\Lambda^{m[k]}}$ ，如公式 (3-6) 所示：

$$\mathcal{S}^{m[k]} \triangleq \left\{ s_1^{n[k]}, \dots, s_{\Lambda^{m[k]}}^{n[k]} \right\} \quad (3-6)$$

然后将局部的  $\mathcal{S}$  进行排序 (默认采用升序)，得到  $\mathcal{S}^*$ ，如公式 (3-7) 所示：

$$\mathcal{S}^{m[k]*} = Sort (\mathcal{S}^{m[k]}, increase = True) \quad (3-7)$$

最后，根据排序后结果  $\mathcal{S}^*$  有选择性地裁剪某些梯度，并在适当时机采用 Allreduce 通信原语进行网络模型梯度更新：

1) 根据给定的梯度压缩条件 (目标梯度压缩比  $cr$ )，在  $\mathcal{S}^*$  中搜索对应的索引  $\kappa$ ，使其满足压缩条件，得到保留目标梯度列表  $\mathbf{g}^{n[k]}$ ，如公式 (3-8) 所示：

$$\begin{aligned} \kappa &= Search \left( \mathcal{S}^{m[k]*}, cr \right), \\ \hat{\mathcal{S}}^{m[k]} &= \mathcal{S}^{m[k]*} [\kappa, \Lambda^{m[k]}] = \left\{ s_{\kappa}^{m[k]}, s_{\kappa+1}^{m[k]}, \dots, s_{\Lambda^{m[k]}}^{m[k]} \right\}, \\ \text{s.t. } & \min_{\hat{cr} \geq cr} |\hat{cr} - cr| \end{aligned} \quad (3-8)$$

其中， $\hat{cr} \in [1, +\infty)$  表示实际梯度压缩比，计算公式 (3-9) 所示， $|\hat{cr} - cr|$  表示实际梯度压缩比与目标梯度压缩比两个标量之间绝对值计算操作，然后通过得到的  $\hat{\mathcal{S}}^{m[k]}$  反向推导出目标梯度处于原始梯度中的位置信息  $\mathbf{g}^{m[k]}$ 。

$$\begin{aligned}\hat{cr} &= \frac{\Lambda^{m[k]}}{\sum_{t=0}^{\Lambda^{m[k]}} \delta(s_t^{m[k]})}, \\ \delta(s^{m[k]}) &= \begin{cases} 1 & \text{if } s^{m[k]} \in \hat{\mathcal{S}}^{m[k]}, \\ 0 & \text{others.} \end{cases}\end{aligned}\quad (3-9)$$

2) 网络拓扑中各 Worker 之间更新神经网络模型梯度、参数, 如公式 (3-10) 所示:

$$\mathbf{g}^{n[k]*} = Update(\mathbf{g}^{n[k]}, \mathbf{g}^{n[k]}) \quad (3-10)$$

其中,  $\mathbf{g}^{n[k]*}$  表示更新后的最新网络模型梯度, 处于  $\mathbf{g}^{n[k]}$  位置的梯度值保持不变的同时其他梯度值被重置为 0。在分布式训练过程中, 由于非零梯度保留了重要信息, 仍可以用来引导网络模型继续朝着正确方向进行优化, GCAC 算法应用于分布式训练的大致流程如算法 3.1 所示。

---

#### 算法 3.1 单节点上 GCAC 算法整体框架及流程

---

**输入:** 训练数据  $X^{n[k]}$ , 目标梯度压缩比  $cr$ , 最大迭代轮数  $Epoch_{max}$ , 网络模型权重  $\mathbf{W}$

**输出:** 训练完成的网络模型权重  $\hat{\mathbf{W}}$

**算法:**

- 1: **for**  $e = 1; e \leqslant Epoch_{max}; e++$  **do**
  - 2:     获取局部梯度  $\mathbf{g}^{n[k]}$
  - 3:     **for**  $k = 1; k \leqslant n^{[k]}; k++$  **do**
  - 4:         初始化:  $\mathcal{S}_{i,j}^{n[k]*} \leftarrow \emptyset$
  - 5:         根据公式 (3-4) 获取最接近质心的梯度  $\mathbf{g}_{i^*,j^*}^{n[k]}$
  - 6:         根据公式 (3-5)、(3-6)、(3-7) 更新已排序梯度列表  $\mathcal{S}_{i,j}^{n[k]*}$
  - 7:         根据公式 (3-8)、(3-9) 搜索满足  $cr$  的列表索引, 并获取  $\mathbf{g}^{n[k]}$
  - 8:         根据公式 (3-10), 将获取到的  $\mathbf{g}^{n[k]}$  置零并更新  $\mathbf{g}^{n[k]*}$
  - 9:     根据设定的 Allreduce 算法, 在合适的时机与其他 Worker 进行梯度同步
  - 10:     $\hat{\mathbf{W}} \leftarrow \mathbf{W}^{(t+1)} \leftarrow SGD(\mathbf{W}^{(t)}, \mathbf{g}^{n[k]*})$
  - 11: **return**  $\hat{\mathbf{W}}$
- 

为了更好地展示本章提出的基于近似质心的梯度压缩算法效果, 如图 3.3 所示, 其中,  $t$  是梯度列表中对应索引。我们针对获取到的梯度进行了简单变换操作, 即将笛卡尔坐标系中的梯度分布情况同步到极坐标中。极角通过公式 (3-11) 转换得到, 极径为当前梯度  $g$  与近似质心  $\hat{\mathcal{C}}^{m[k]}$  之间的欧式距离  $s_t^{m[k]}$ :

$$\begin{aligned}\gamma &= \frac{2\pi}{t}, \\ s_t^{m[k]} &= \phi\left(\hat{\mathcal{C}}^{m[k]}, g_t^{m[k]}\right)\end{aligned}\quad (3-11)$$

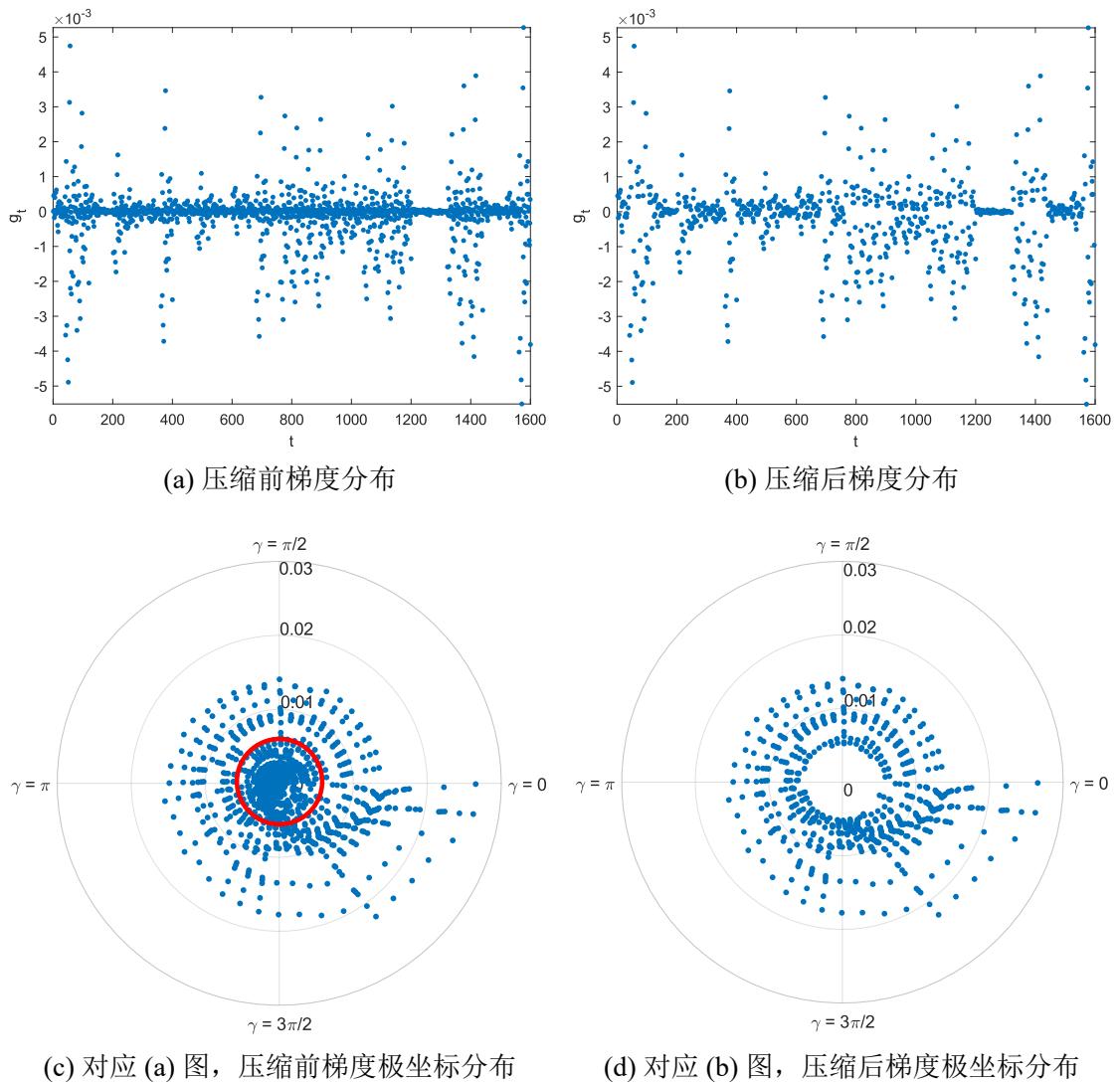


图 3.3 基于 GCAC 算法压缩前后梯度分布

在图 3.3 的 (a) 和 (c) 子图中, 不难发现, 绝大多数的梯度都分布在较小的范围内, 并且十分接近该局部梯度矩阵质心, 所以这些梯度可以被那些远离质心的梯度进行表示, 也就是说, 当梯度压缩比较大时, 仍然不会对神经网络模型性能优化产生较大影响。图 3.3 (c) 中红色圆圈对应于图 3.1 (a) 中红色点状线, 它们梯度压缩比都是 2。显然, 圈内的梯度更加接近局部梯度矩阵的质心, 它们包含高度相似的信息, 并且可以通过圆外的梯度(即远离质心的梯度)进行线性组合获得, 因此, 他们对模型参数更新的贡献可被替代, 对它们实施裁剪操作并不会对网络模型性能产生实质性的影响。

如图 3.3 (b) 和图 3.3 (d) 所示, 与基于梯度值或范数的梯度压缩方法(见图 3.1 (b))相比, 当采用 GCAC 算法时, 其结果保留了相对较大和相对较小的梯度, 因此即使不满足上述两个隐含条件, 也保留了大部分重要的梯度信息。

### 3.3 实验与结果分析

#### 3.3.1 实验环境以及相关参数配置

本章所有的验证实验都将部署在由 4 个 Worker 节点构成的 Linux 集群环境下。具体来说，单个 Worker 节点的主要硬件环境包括：CPU 处理器为主频 4.50 GHz 的 Intel® Core™ i9-10900X、内存大小为 64 GB、GPU 采用的是显存大小为 11 GB 的 GeForce GTX 2080Ti，网络接口设备带宽为 1000 Mb/s。另外，为了便于部署与管理分布式集群中各个节点实验环境，采用 Docker 20.10 进行镜像封装，且镜像内部主要装有 OpenMPI 4.0、CUDA 11.7、Python 3.8、PyTorch 1.7 以及 Horovod 0.21 等软件。另外，针对高光谱目标检测任务，运行在 Linux 操作系统上的 MATLAB 2021b 被用来进行输入数据的预处理以及输出结果的后处理操作。

当处理不同的下游任务时，我们采取了不同的神经网络模型和数据集，关于下游任务、网络模型以及相关数据集具体设置如表 3.1 所示。另外，分布式训练部署过程涉及众多超参数设置，主要包括 Batch 大小  $\varphi$ 、最大迭代轮数  $Epochs$ 、学习率  $\eta$ 、动量 Momentum、权重衰减因子、优化器等，实验实施过程中需要针对不同数据集进行不同超参数设置，具体设置如表 3.2 所示。另外，需要注意的是，当采用 AAE 网络模型时，其中 AE 子网络结构采用对称结构，编码器中隐藏层数量设置为 2，每一个隐藏层神经元数量设置为 1000。

表 3.1 下游任务、网络模型以及训练数据集相关设置

下游任务	网络模型	数据集
高光谱目标检测	AAE	AeroRIT Xiongan New Area (Matiwan Village)
图像分类	VGGNet-16	CIFAR10
	ResNet-56	
	ResNet-110	

#### 3.3.2 高光谱目标检测实验结果分析

为了验证本章提出的梯度压缩算法的可行性，首先选择高光谱目标检测作为下游任务进行验证。如图 3.4 所示，针对 AeroRIT 高光谱数据集，我们分别考察了不同压缩比和梯度压缩算法对高光谱目标检测结果的影响。其中，图 3.4 (a) 展示了该场景的伪装图，而图 3.4 (b) 则描述了该场景中感兴趣目标的分布情况，图 3.4 (c) 给出了不进行梯度压缩的情况下，对 AeroRIT 数据集进行分布式训练的效果图，图 3.4 (d)、

表 3.2 GCAC 梯度压缩算法超参数设置

数据集	$Bs$	$Epochs$	学习率 $\eta$	动量	权重衰减因子	优化器
AeroRIT	310000	100				
Xiongan New Aera (Matiwan Village)	200000	200	0.0001	-	-	Adam
CIFAR10	128	200	0.01	0.9	0.0005	SGD

(e) 分别展示了在压缩比  $cr$  分别设置为 10、100 的情况下，采用本章提出的梯度压缩算法（GCAC）时的检测效果图，图 3.4 (f) 展示了在压缩比  $cr$  设置为 10，且以 L1-范数作为梯度重要性度量的情况下，进行目标检测的效果图，图 3.4 (g)、图 3.4 (h) 分别展示了在压缩比  $cr$  设置为 10 的情况下，采用 DGC、Top-K 算法进行分布式训练的检测效果图。

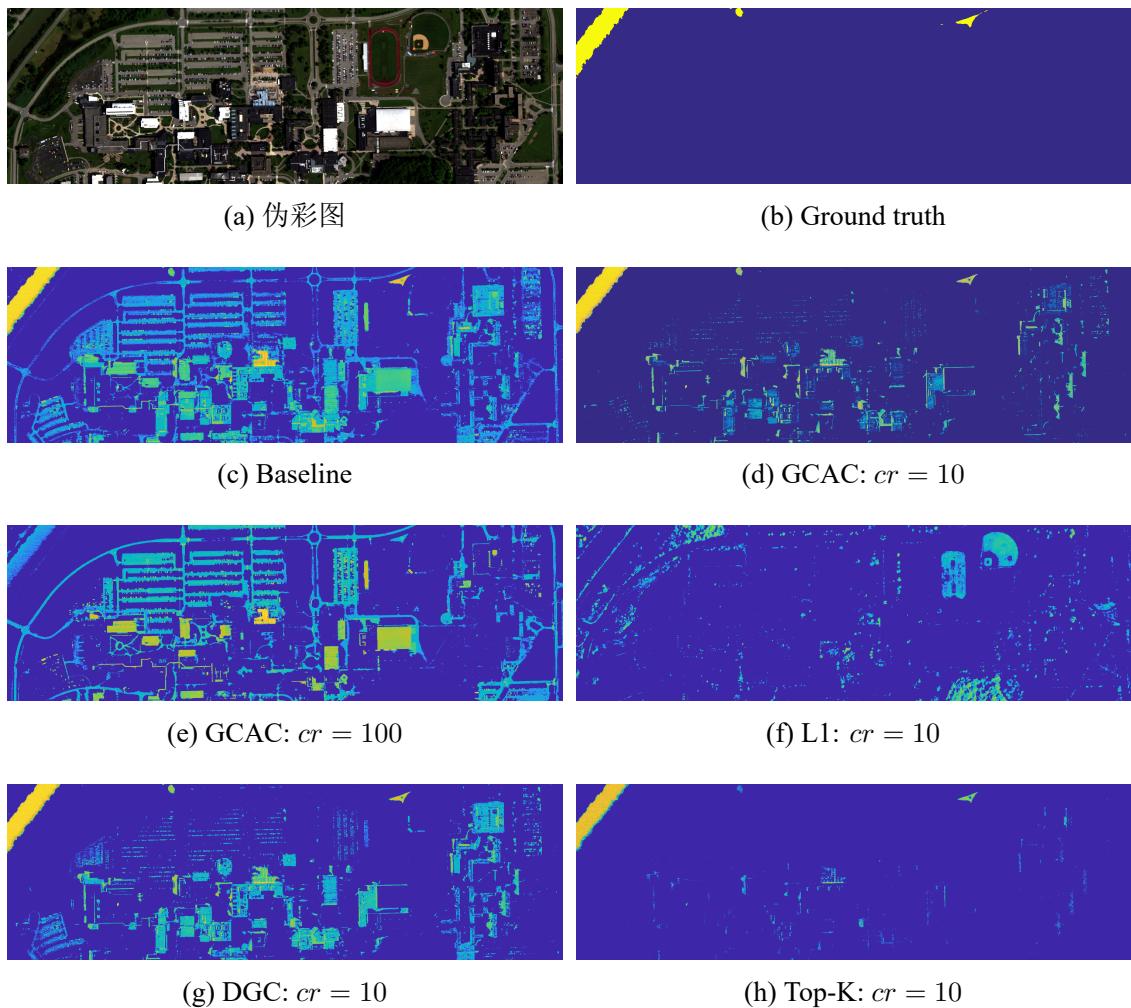


图 3.4 AeroRIT 数据集上，使用不同梯度压缩算法进行分布式训练的高光谱目标检测可视化结果

在高光谱目标检测任务中，我们不仅仅在 AeroRIT 数据集上进行实验，同时也在规模更大的遥感数据集 Xiongan New Area (Matiwan Village) 上进行了验证。图 3.5 展示了分布式训练的高光谱目标检测效果对比。其中，(a)~(h) 子图分别展示了高光谱数据集 Xiongan New Area (Matiwan Village) 的伪彩图、感兴趣目标的真实标签分区情况 (Ground truth)、不进行梯度压缩的高光谱目标检测效果图 (Baseline)、采用 GCAC 算法并设置压缩比  $cr$  为 10 和 100 的情况下，进行高光谱目标检测效果图，以及压缩比  $cr$  为 10 的情况下，采用 L1-范数作为梯度重要性度量、DGC、Top-K 等梯度压缩算法的高光谱目标检测效果图。

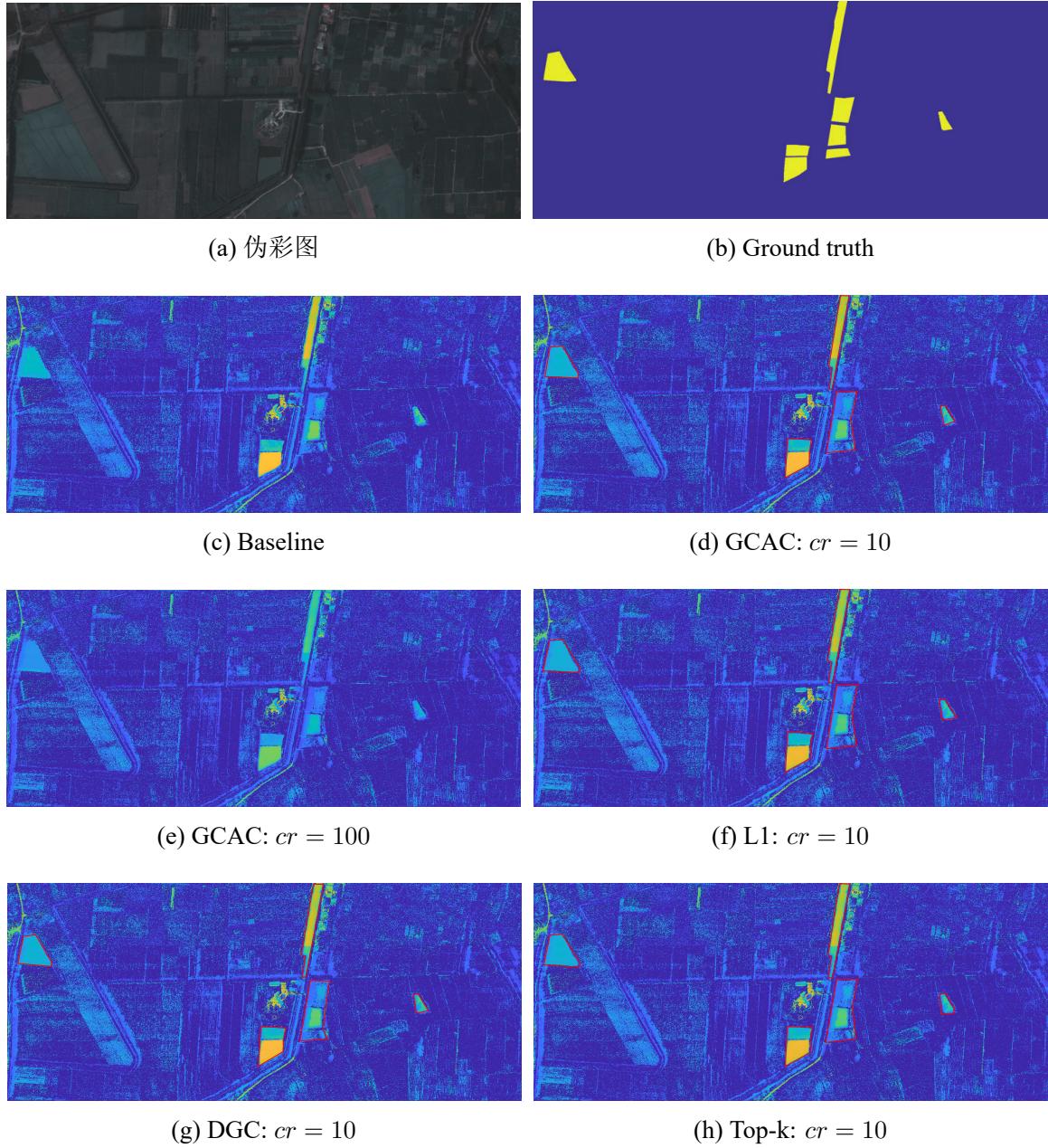


图 3.5 Xiongan 数据集上，使用不同梯度压缩算法进行分布式训练的高光谱目标检测可视化结果

通过图 3.4 以及图 3.5 的效果进行对比，我们可以清楚地看到，在相同的压缩比条件下，GCAC 算法获取到的目标检测结果明显优于 L1、DGC、Top-K 等算法。而当与 Baseline 进行对比时，可以发现在较小压缩比条件下，两者之间的差距非常小，目标都可以被很好地检测出来。然而，通过图 3.4 (d) 与图 3.4 (e) 以及图 3.5 (d) 与图 3.5 (e) 的对比，我们可以发现，当压缩比从 10 增大 100 时，GCAC 算法并不能很好地适应这种变化，从而导致检测效果不佳。这种情况出现的可能原因是：在实践中，由于我们为了加快计算梯度张量的质心位置，采用了“近似质心”进行代替，导致在采用保留下来的梯度进行模型训练时，梯度更新的方向与原优化方向有所偏差。

如图 3.4、3.5 所示，采用了目标检测效果图来评估算法性能。然而，这种评价方式由于主观性较强且不够全面，因此我们采用了更为客观和量化的评价指标，包括 AUC、FPR 等，以对算法的性能进行更深入的评估。如表 3.3 所示，总结了在不同数据集和不同压缩比的情况下，不同梯度压缩算法对网络模型检测性能的影响，并对本数据集中数值最大的 AUC 值以及最小的 FPR 值进行了加粗显示。

表 3.3 不同训练数据集上，使用不同梯度压缩算法进行分布式训练的 HTD 实验结果

数据集	梯度压缩算法	压缩比 $cr$	AUC(%)	FPR(%)
AeroRIT Xiongan New Area (Matiwan Village)	Baseline	-	<b>95.99</b>	<b>1.16</b>
			<b>94.10</b>	7.93
AeroRIT	GCAC	10×	<b>95.99</b>	1.18
		30×	95.08	1.35
		50×	93.28	2.15
	L1	100×	87.28	5.04
		10×	83.94	6.67
		10×	95.23	1.58
Xiongan New Area (Matiwan Village)	GCAC	10×	93.38	2.36
		10×	<b>94.10</b>	7.94
		30×	93.35	<b>7.50</b>
	L1	50×	92.09	7.67
		100×	89.41	11.31
		10×	89.63	10.67
	DGC <sup>[31]</sup>	10×	93.63	7.78
		10×	93.38	8.36
		10×	93.38	8.36

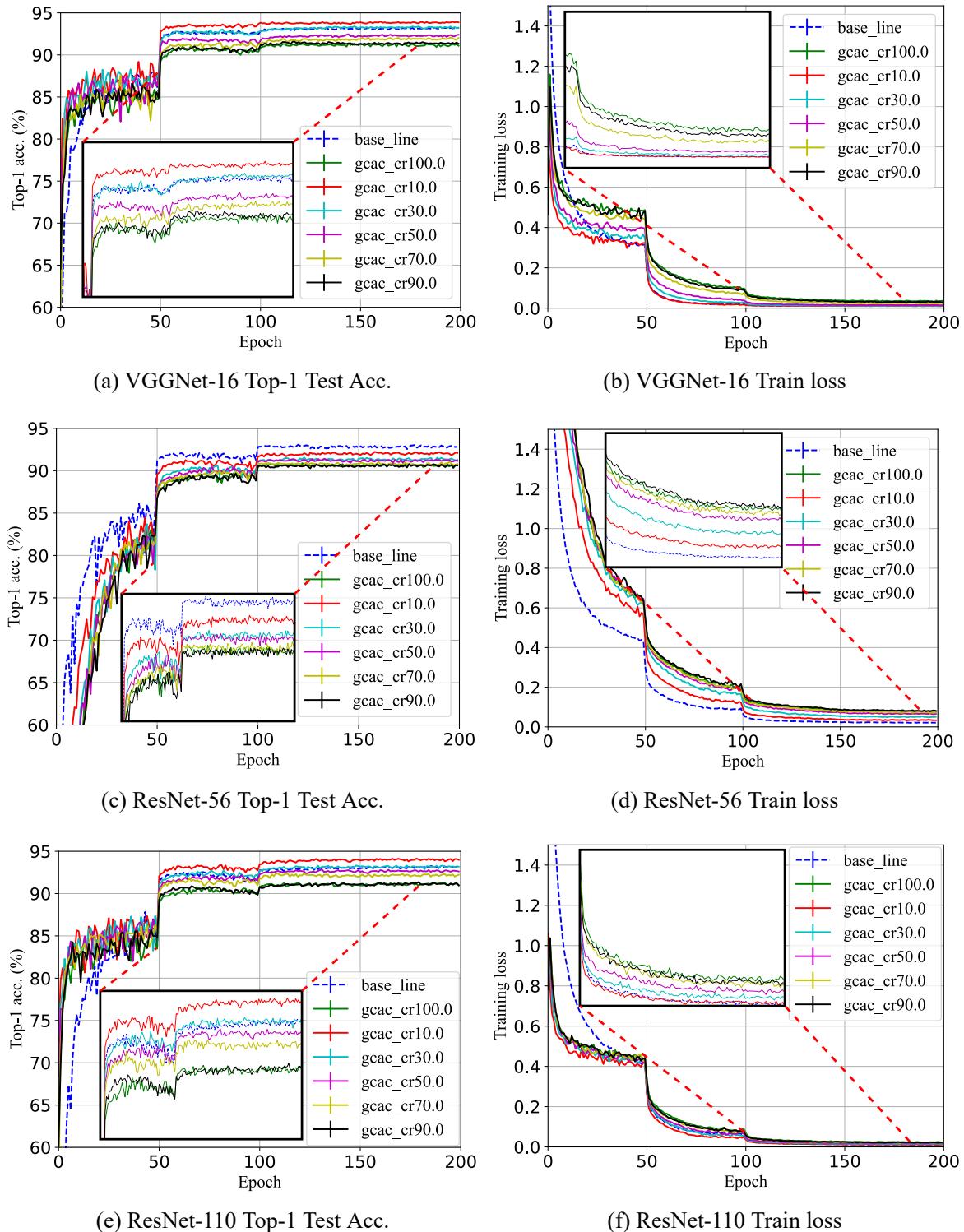
从表 3.3 中的 AUC 值可以看出，本章所提出的 GCAC 算法在大规模遥感数据集上的表现非常具有竞争力，甚至已然超过 DGC 等先进的梯度压缩算法。无论是在 AeroRIT 还是 Xiongan New Area (Matiwan Village) 遥感数据集上，当压缩比为 10 时，GCAC 算法都实现了无损压缩的效果，即 AUC 值与 Baseline 表现相同，同时明显优于其他梯度压缩算法。另外，通过表 3.3 中的 FPR 值可以看出，使用 GCAC 算法时，FPR 值与 Baseline 十分接近，这说明在分布式训练场景中应用 GCAC 算法并没有对网络模型的虚警率产生较大影响。然而，需要注意的是，由于 GCAC 算法在实践过程中采用了“近似质心”的计算加速技术，因此在面对大压缩比时，GCAC 算法的表现仍有较大的提升空间。上述的高光谱目标检测任务的实验表明，本章提出的基于近似质心的梯度压缩算法在小压缩比的情况下表现优异，但在大压缩比的情况下，网络模型的性能有较大的下降，成为了该算法的一个不足之处。

### 3.3.3 图像分类实验结果分析

为了充分验证所提出的基于近似质心的梯度压缩算法的泛化能力，我们将其应用到自然图像分类这一下游任务中。第 3.3.1 小节详细描述了本任务所涉及到的模型部署环境、网络模型选择以及模型超参数设置等相关内容。

图 3.6 展示了在 CIFAR10 数据集上，使用 GCAC 算法进行分布式训练不同规模的网络模型（VGGNet-16、ResNet-56 以及 ResNet-110）的测试精度和训练损失曲线。其中，第一行中图 3.6 (a)、图 3.6 (b)，第二行中图 3.6 (c)、图 3.6 (d)，第三行中图 3.6 (e)、图 3.6 (f) 分别展示了不压缩（Baseline）以及不同压缩比的情况下，VGGNet-16、ResNet-56 以及 ResNet-110 三个网络模型的测试精度和训练损失曲线。可以看出，无论是 VGGNet 还是 ResNet 网络模型，在分布式训练过程中采用 GCAC 算法时，其收敛速度都比于 Baseline 更快。此外，当压缩比较小时，网络模型的性能表现甚至超过了 Baseline，这表明在梯度张量中存在一部分对模型优化起到反作用的梯度，因此，在整个分布式训练过程中，针对不同的目标选择合适的算法进行梯度稀疏化是相当重要的。需要注意的是，在压缩比较大（例如 100）的情况下，采用 GCAC 算法进行分布式训练的网络模型性能表现有所下降，这中现象归因于“近似质心”的加速计算技术。

如图 3.7 所示，在 CIFAR10 数据集上，对不同网络模型（VGGNet-16、ResNet-56 以及 ResNet-110）采用了不同的梯度压缩算法进行分布式训练，并展示了其测试精度以及训练损失曲线。需要注意的是，该图仅选取了两个具有代表性的压缩比（10 和 100）进行比较。与采用 L1-范数为梯度度量梯度压缩算法相比，无论是从收敛速度还是网络模型性能表现来看，GCAC 算法的表现更好，并且当网络模型收敛后，分布式训练的 loss 值更小。因此，可以得出结论，GCAC 算法可以在保证网络模型精度的

图 3.6 CIFAR10 数据集上，不同网络模型采用 GCAC 算法进行分布式训练的  $acc_1$  与 loss 曲线

同时，还可以大幅提高分布式的训练效率。

如表 3.4 展示了在 CIFAR10 数据集上，不同网络模型采用不同梯度压缩算法进行分布式训练的  $acc_1$  与损失 loss 收敛值。正如图 3.6 与图 3.7 所展示的，当分布式训

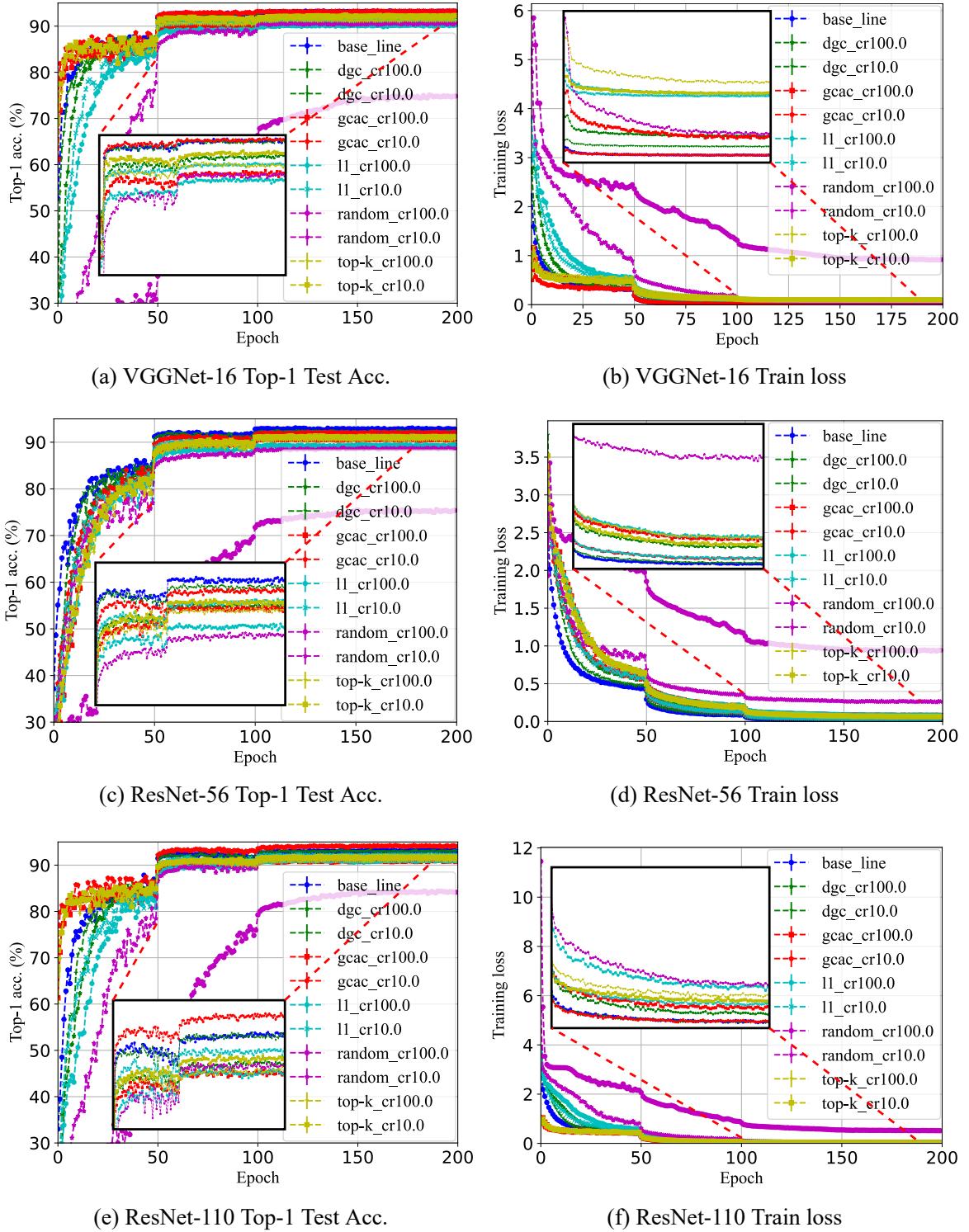


图 3.7 CIFAR10 数据集上，不同网络模型采用不同梯度压缩算法进行分布式训练的  $acc_1$  与 loss 曲线

练习过程中的梯度压缩比  $cr$  较小时，GCAC 算法的表现非常出色。尤其面向较大规模的网络模型时，该算法训练出来模型 Top-1 精度甚至超过了 DGC 算法。与其他梯度压缩算法对比，采用 GCAC 算法的网络模型损失 loss 收敛值更接近于零，这表明该

网络模型与训练数据集的贴合度更高。然而，该表格也反映出，本章提出的 GCAC 算法在面对大压缩比的挑战时，表现不尽如人意，导致网络模型的性能下降严重，该现象在理论上归因于“近似质心”的加速计算技术。

表 3.4 CIFAR10 数据集上，GCAC 与其他算法的分布式训练  $acc_1$  与 loss 收敛值

网络模型	梯度压缩算法	梯度压缩比 $cr$	Top-1 Acc. (%)	loss 收敛值
VGGNet-16	Baseline	-	93.17	0.0105
	GCAC	10× / 100×	<b>93.28</b> / 90.84	0.0101 / 0.0359
	L1	10× / 100×	91.46 / 90.32	0.0118 / 0.0140
	Top-K	10× / 100×	92.45 / 91.48	0.0141 / 0.0291
	DGC	10× / 100×	93.24 / 92.04	<b>0.0100</b> / 0.0123
	Random	10× / 100×	90.69 / 75.06	0.0378 / 0.9185
ResNet-56	Baseline	-	<b>93.07</b>	<b>0.0211</b>
	GCAC	10× / 100×	92.23 / 90.89	0.0334 / 0.0745
	L1	10× / 100×	91.30 / 89.51	0.0354 / 0.0816
	Top-K	10× / 100×	91.41 / 90.77	0.0657 / 0.0811
	DGC	10× / 100×	92.52 / 91.08	0.0243 / 0.0589
	Random	10× / 100×	88.84 / 75.53	0.2631 / 0.9435
ResNet-110	Baseline	-	93.24	0.0116
	GCAC	10× / 100×	<b>94.07</b> / 91.14	<b>0.0110</b> / 0.0224
	L1	10× / 100×	92.38 / 91.08	0.0252 / 0.0372
	Top-K	10× / 100×	91.87 / 91.03	0.0265 / 0.338
	DGC	10× / 100×	93.20 / 91.65	0.0113 / 0.0174
	Random	10× / 100×	91.41 / 84.61	0.0421 / 0.5033

### 3.3.4 耗时分析

分布式训练过程中采用梯度压缩技术的目的有两个：1) 减小网络拓扑中各 Worker 之间通信的数据量；2) 缩短节点间的通信时间。显然，梯度压缩算法可以轻松实现第一个目标，而缩短通信时间不仅与节点数量有关，也与网络通信内容数据量息息相关，即网络通信中节点数量越多、通信内容数据量越大，通信时间越长。GCAC 算法旨在减少通信内容数据量，在控制网络节点数量不变的情况下，表 3.5 展示了不

进行梯度压缩与采用梯度压缩网络模型训练相关时长统计，包括每次迭代平均用时、梯度压缩平均用时及其占比。

表 3.5 CIFAR10 数据集上，不同网络模型采用不同梯度压缩算法的耗时比较

网络模型	梯度压缩算法	迭代平均用时 (毫秒)	梯度压缩平均 用时 (毫秒)	梯度压缩用时 占比 (%)
VGGNet-16	Baseline	640.99	-	-
	GCAC	828.70	348.78	42.09
	L1	828.17	<b>347.18</b>	<b>41.92</b>
	Top-K	862.27	418.46	48.53
	DGC	1375.42	906.56	65.91
ResNet-56	Baseline	157.33	-	-
	GCAC	333.28	<b>230.15</b>	69.06
	L1	335.80	230.38	<b>68.61</b>
	Top-K	395.59	277.66	70.19
	DGC	557.86	460.00	82.46
ResNet-110	Baseline	289.08	-	-
	GCAC	629.85	<b>382.52</b>	<b>60.73</b>
	L1	612.90	397.79	64.90
	Top-K	653.27	494.00	75.62
	DGC	1067.71	866.94	81.20

根据该表可见，本文提出的 GCAC 算法与 L1 算法的梯度压缩平均用时接近，这主要得益于采用了“近似质心”的加速计算技术以及 GPU 并行计算。与 Top-K 梯度压缩算法对比，当网络模型参数量较大时（如 VGGNet-16、ResNet-110），GCAC 算法进行梯度压缩平均耗时占比优势更加明显，仅占用总迭代时长的 6% 左右。相较于 DGC 算法，GCAC 算法梯度压缩用时占比更小，主要是因为 DGC 采用的延迟通信操作被计入了梯度压缩用时。更重要的是，无论是小规模（ResNet-56）还是大规模网络模型（VGGNet-16、ResNet-110），GCAC 算法只需要花费少量时间对梯度进行压缩操作，就可以大幅减少各节点之间的通信数据量，进而大幅缩减通信时间。例如，在 CIFAR10 数据集上，大规模网络模型 VGGNet-16 进行分布式训练过程中，采用 GCAC 算法进行梯度压缩操作，只需花费原迭代平均用时的 0.29 倍。

### 3.3.5 “近似质心”加速计算组件分析

在 3.3.2 与 3.3.3 两小节，我们发现 GCAC 算法在应对大压缩比挑战时，最终网络模型的性能相对于 Baseline 有所下降，并在 3.2.2 小节通过理论定性分析将其归因于“近似质心”加速计算组件。为了充分论证“近似质心”加速计算技术对 GCAC 算法所带来的影响，本小节将通过消融实验对该组件进行定量分析。

表 3.6 展示了“近似质心”加速计算组件对 VGGNet-16 以及两种不同规模的 ResNet 网络模型的性能影响，其中，“-”和“√”分别表示在分布式训练过程中不采用和采用“近似质心”加速计算组件。

表 3.6 CIFAR10 数据集上，“近似质心”加速计算组件对不同网络模型的影响

网络模型	VGGNet-16		ResNet-56		ResNet-110	
“近似质心”加速计算组件	-	√	-	√	-	√
单次迭代平均用时（毫秒）	8342.78	<b>347.18</b>	2025.32	<b>230.15</b>	3412.08	<b>382.52</b>
Top-1 Acc. (%)	<b>92.08</b>	90.84	<b>91.32</b>	90.89	<b>91.94</b>	91.14

从该表可以发现，不采用“近似质心”加速计算组件的情况下，网络模型可以取得更好的分类精度，但随着网络模型单层平均参数量的增加，网络训练的单次迭代平均用时剧增，这将无疑导致分布式训练速度急剧下降，无法实现加速训练的目标。因此，为了尽量保证网络模型性能表现的情况下，实现分布式训练的优化加速，GCAC 算法引入了“近似质心”加速计算组件，这也导致了 GCAC 算法无法很好地适应大压缩比训练场景，是该算法的一个不足之处。

## 3.4 本章小结

为了降低分布式训练过程中各 Worker 之间的通信数据量，本章提出一种新的梯度压缩算法，即基于近似质心的梯度压缩算法（Gradient Compression via Approximate Centroid, GCAC）。该算法利用网络模型局部梯度张量的质心特性，选择部分梯度代替全局梯度进行通信，旨在保证网络模型性能的情况下提高分布式训练速度。算法的核心思路是：网络拓扑中每个 Worker 首先获取本地的局部网络模型梯度张量，然后根据“近似质心”的加速计算技术计算每一个梯度与“近似质心”之间的距离，并基于该距离进行梯度排序，随后根据预设压缩比选择相应数量的梯度索引，据此得到待传输梯度，最后进行梯度更新并进入下一次迭代。我们在多个下游任务上进行了 GCAC 算法的验证工作，并与现有梯度压缩算法进行了对比。实验结果表明，基于近

似质心的梯度压缩算法具有很强的通用性，可以适应高光谱目标检测、自然图像分类等不同类型的任务。同时，GCAC 算法只需在各 Worker 通信之前花费少量时间进行梯度压缩操作，从而缓解了整个分布式训练的瓶颈难题，大大提高了网络模型训练速度。另外，实验还发现 GCAC 算法在压缩比较小时可以完美应用到大数据大模型场景下，但在大压缩比场景下会导致网络模型性能下降较多，是该算法的一个不足之处。



## 第四章 基于 $k$ -互异近邻的梯度压缩算法

### 4.1 引言

在大规模数据集、高复杂度网络模型场景下，与现有一些梯度压缩算法（如 Top-K、DGC 等）相比，第三章提出的 GCAC 算法在最终网络模型的性能表现、梯度压缩平均耗时等方面取得了一定的优势。但在大压缩比的情况下，采用 GCAC 算法的网络模型最终性能表现下降较多，是该算法的一个不足之处。因此，发现在大压缩比场景下 GCAC 算法表现不足的难题后，本章进一步思考、分析梯度张量的相关特性，提出基于  $k$ -互异近邻的梯度压缩算法（Gradient Compression based on  $k$ -Reciprocal Nearest neighbors,  $k$ -RNGC）。

$k$ -RNGC 算法实现了独立计算感知梯度重要性的测量方法，根据梯度测量结果对梯度进行排序和预选择，同时还设计了  $k$ -互异近邻梯度选择方案，从推荐的梯度中选择出待保留的梯度。另外，为了提高大压缩比场景下的算法表现，引入动量修正和预热训练两大核心组件，克服了通信频率降低导致的梯度陈旧问题，保证稀疏化网络模型的性能表现。

本章首先在第 4.2 小节介绍  $k$ -RNGC 算法的核心内容，包括计算感知算法、 $k$ -互异近邻梯度选择算法、动量修正、预热训练以及整体框架等。接着，第 4.3 小节介绍高光谱目标检测和自然图像分类两个下游任务上的验证实验以及结果分析，主要包括组件分析、耗时分析以及对比实验。最后，第 4.4 小节总结本章主要的研究工作。

### 4.2 基于 $k$ -互异近邻的梯度压缩算法

#### 4.2.1 计算感知获取局部梯度压缩比

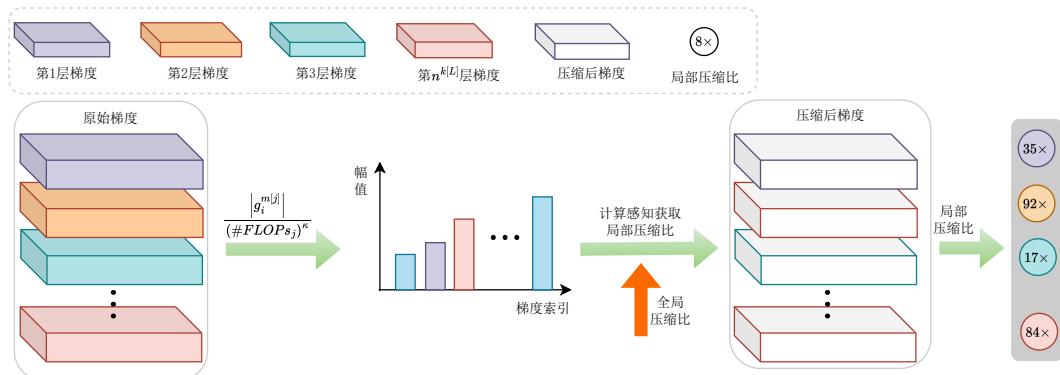


图 4.1 通过计算感知算法以获取网络模型局部压缩比原理图

假设给定全局梯度压缩比为  $cr$ ，全局排序的目标则是通过假定裁剪掉某些排名

靠后的梯度，使得裁剪后的梯度张量满足压缩比条件，然后根据保留梯度反向推导出其在网络模型中所处的位置，最后再统计每一层的保留梯度得到对应的局部压缩比，该过程如图 4.1 所示。需要注意的是，该过程中排序操作只用于梯度预裁剪，旨在获取局部压缩比，也就说用于网络通信的梯度并不是由全局排序的结果确定，而是通过  $k$ -互异近邻梯度选择算法得到。

在全局排序过程中，梯度的重要性度量通过计算感知获取，具体如公式 (4-1) 所示：

$$\omega(g_i^{m[j]}) = \frac{|g_i^{m[j]}|}{(\#FLOPs_j)^\kappa} \quad (4-1)$$

其中， $\#FLOPs_j$  表示网络模型当前层（第  $j$  层）的 FLOPs 规模大小， $\kappa \geq 0$  是应用于整个网络模型的一个超参数。公式 (4-1) 是绝对值度量的推广，当设置  $\kappa$  为 0 时，它将退化为基于绝对值的重要性度量，仅依赖于梯度幅值。当固定  $\kappa$  不变时，梯度的绝对值越小、网络中该层 FLOPs 规模越高，梯度的重要性就越低，则往往不会被选中作为保留梯度。另外，当网络模型以及  $\kappa$  确定后， $(\#FLOPs_j)^\kappa$  表现为一个常量，公式 (4-1) 与基于幅值的度量方法相比，不仅没有增加计算量，而且很好地考虑了网络模型中梯度全局性和各层计算量，可以更好地感知梯度的重要程度。

为了后续描述方便，定义分布式训练过程中第  $i$  次迭代时第  $m$  个 Worker 上的局部压缩比  $\mathcal{R}^{m(i)}$  如下：

$$\begin{aligned} \mathcal{R}^{m(i)} &= \{r^{m(i)[1]}, r^{m(i)[2]}, \dots, r^{m(i)[L]}\}, \\ r^{m(i)[j]} &\in [1, +\infty), \quad \forall j \in [1, L] \end{aligned} \quad (4-2)$$

其中， $r^{m(i)[j]}$  定义了网络模型中第  $j$  层的稀疏度，具体数值由公式 (4-3) 计算得到：

$$r^{m(i)[j]} = \frac{n^{m[j]} \cdot Q^{m[j]}}{\sum_s^{n^{m[j]}} \sum_t^{Q^{m[j]}} \delta(g_{s,t}^{m(i)[j]})} \quad (4-3)$$

其中， $\delta(g_{s,t}^{m(i)[j]})$  的输出结果由全局梯度压缩比与当前梯度在全局排序结果的位置共同决定，即当  $g_{s,t}^{m(i)[j]}$  被裁剪掉时， $\delta(g_{s,t}^{m(i)[j]})$  输出 0，否则输出 1。

根据已获得的局部压缩比，可计算出网络模型中各层梯度保留数量  $\mathfrak{N}^{m(i)}$ ，如第  $j$  层待传输梯度数量  $\mathfrak{n}^{m(i)[j]}$  的计算公式如 (4-4) 所示：

$$\begin{aligned} \mathfrak{N}^{m(i)} &= \{\mathfrak{n}^{m(i)[1]}, \mathfrak{n}^{m(i)[2]}, \dots, \mathfrak{n}^{m(i)[L]}\} \\ \mathfrak{n}^{m(i)[j]} &= \lceil n^{m(i)[j]} \times r^{m(i)[j]} \rceil \end{aligned} \quad (4-4)$$

其中， $\lceil \cdot \rceil$  表示四舍五入后最接近输入的整数。

算法 4.1 展示了通过计算感知算法对网络模型进行全局梯度排序，然后根据排序

的结果得到各层保留梯度数量，从而进一步获取局部梯度压缩比的具体流程。

---

**算法 4.1 通过计算感知算法以获取网络模型局部压缩比流程**

---

**输入：**训练数据  $X$ ，全局目标梯度压缩比  $cr$

**输出：**网络模型局部梯度压缩比  $\mathcal{R}^m$  以及局部梯度保留数量  $\mathfrak{N}^m$

**算法：**

- 1: 初始化:  $\mathcal{H}^m \leftarrow \emptyset$
  - 2: **for**  $j = 1; j \leq L; j++$  **do**
  - 3:     获取局部梯度  $\mathbf{g}^{m[j]}$
  - 4:     **for**  $i = 1; i \leq n^{[j]} \times Q^{m[j]}; i++$  **do**
  - 5:         根据公式 (4-1) 计算梯度  $g_i^{m[j]}$  的重要性度量值  $\omega$ ，并将其追加到  $\mathcal{H}^m$  中，即  

$$\mathcal{H}^m \leftarrow Append(\mathcal{H}^m, \omega)$$
  - 6:     将  $\mathcal{H}^m$  进行排序（升序），形成网络模型梯度全局排序，即  $\mathcal{H}^m \leftarrow Sort(\mathcal{H}^m)$
  - 7:     根据全局目标梯度压缩比  $cr$ ，剔除掉排序靠后的梯度，即  $\hat{\mathcal{H}}^m \leftarrow Select(\mathcal{H}^m, cr)$
  - 8:     **for**  $j, g := range \hat{\mathcal{H}}^m$  **do**
  - 9:         根据公式 (4-3) 以及 (4-4)，计算获得  $r^{m[j]}, \mathfrak{n}^{m[j]}$
  - 10:         $\mathcal{R}^m \leftarrow Append(\mathcal{R}^m, r^{m[j]}), \mathfrak{N}^m \leftarrow Append(\mathfrak{N}^m, \mathfrak{n}^{m[j]}),$
  - 11:    **return**  $\mathcal{R}^m, \mathfrak{N}^m$
- 

### 4.2.2 $k$ -互异近邻梯度选择算法

根据公式 (4-4) 计算得到网络模型中每一层保留的梯度数量之后，接下来就是在局部梯度张量中找到相应数量最重要的梯度。也就是说，在该梯度选择算法中将通过测量  $\mathfrak{n}^{m(i)[j]}$  个梯度构成梯度子集的重要性，而不是简单地考虑单个梯度的重要性。思路如下：每层梯度对网络模型优化做出的贡献是该层梯度共同作用的结果，因此应该考虑梯度集体的重要性而不是某一个梯度的重要性。在该思想的启发下，提出了一种基于推荐的梯度选择算法，每一个梯度推荐子集都包含了  $\mathfrak{n}^{m(i)[j]}$  个梯度，而这些梯度推荐子集都有可能用于网络通信和梯度同步，最后根据  $k$ -互异近邻原则选出最终的梯度子集，我们称这种梯度选择算法为“ $k$ -互异近邻梯度选择算法”。

为了通过  $k$ -互异近邻算法实现梯度选择，需要先对网络模型中第  $j$  层的梯度进行归一化紧密度建模，构建出相似矩阵  $S^{m[j]}$ ，其元素定义如下：

$$s_{i,h}^{m[j]} = \frac{\exp\left(-\mathcal{D}^2\left(g_i^{m[j]}, g_h^{m[j]}\right)\right)}{\sum_{t=1}^{n^{m[j]}} \exp\left(-\mathcal{D}^2\left(g_i^{m[j]}, g_t^{m[j]}\right)\right)} \quad (4-5)$$

$$1 \leq j \leq L, \quad 1 \leq i, h \leq n^{m[j]}$$

其中， $\mathcal{D}(\cdot, \cdot)$  表示某种距离函数。通过实践发现，欧几里得范数是最简单实用的一种，可以很好地反映出梯度与梯度之间的接近程度。在定义好接近程度的度量基础上，则

可以进一步定义接近度度量函数，例如，给定梯度  $g_i^{m[j]}$  时，梯度  $g_h^{m[j]}$  的接近度度量可采用公式 (4-6) 描述：

$$\phi\left(g_h^{m[j]} \mid g_i^{m[j]}\right) = 1 + \sum_{t=1}^{n^{m[j]}} \delta\left(s_{i,g}^{m[j]} > s_{i,h}^{m[j]}\right) \quad (4-6)$$

其中， $\delta(\cdot)$  表示单位函数，如果输入为真，则输出 1，否则输出 0。

对于节点  $m$  上的第  $j$  层梯度，将会推荐一组最与  $g_i^{m[j]}$  最接近的梯度，因此，可以使用来自于  $g_i^{m[j]}$  的  $k$  个梯度构建一个梯度推荐子集，其定义如下：

$$\mathcal{G}_{g_i^{m[j]}}^k = \left\{ g_h^{m[j]} \mid \phi\left(g_h^{m[j]} \mid g_i^{m[j]}\right) \leq k, h = 1, 2, \dots, n^{m[j]}\right\} \quad (4-7)$$

该公式表明， $\mathcal{G}_{g_i^{m[j]}}^k$  捕获了  $k$  个最接近  $g_i^{m[j]}$  的梯度 ( $k$ -nearest neighbors,  $k$ -NN)，构成了一个推荐子集。虽然对于第  $j$  层的梯度  $g_i^{m[j]}$  而言， $k$ -NN 中的每个梯度都是良好的候选者，但不同的梯度有很大可能会存在不同的推荐子集。因此，简单地选择某一个推荐子集是不合适的，因为该推荐梯度子集可能与其他梯度相距甚远。为了解决该问题，我们提出了如公式 (4-8) 所示的  $k$ -互异近邻梯度推荐子集：

$$\hat{\mathcal{G}}_{g_i^{m[j]}}^k = \mathcal{G}_{g_1^{m[j]}}^k \cap \mathcal{G}_{g_2^{m[j]}}^k \cap \dots \mathcal{G}_{g_n^{m[j]}}^k \quad (4-8)$$

不难发现， $k$ -互异近邻梯度推荐子集被定义为  $g^{m[j]}$  中所有梯度的  $k$ -NN 交集。这对最终选择的梯度推荐子集中的梯度提出更严格的要求，即每一个被选中的梯度应该属于每个梯度的  $k$ -NN，而不是某一个梯度  $k$ -NN。因此，通过该方法可以排除一些低值邻居，也就是靠近特定的梯度但远离其他梯度的邻居。

还有一点需要注意， $\hat{\mathcal{G}}_{g_i^{m[j]}}^k$  中梯度的数量可能小于计算感知算法得到的保留梯度目标数量  $n^{m(i)[j]}$ ，即  $|\hat{\mathcal{G}}_{g_i^{m[j]}}^k| < n^{m(i)[j]}$ 。为解决这一选择异常问题，我们初始化  $k$  为  $n^{m[j]}$ ，并协同全局二分算法<sup>[87]</sup> 与局部遍历算法，快速缩小  $k$  的搜索范围，直至找到最优值且满足保留梯度目标数量的条件。

算法 4.2 展示了基于计算感知算法获取的局部梯度压缩比之上，通过  $k$ -互异近邻梯度选择算法获取最终  $k$ -互异近邻梯度推荐子集的具体流程。

### 4.2.3 动量修正

根据第三章以及现有研究<sup>[88]</sup>，不难发现当网络模型中梯度稀疏度非常高时，若仍在通信延迟技术的基础上，采用稀疏化梯度进行更新，网络模型的收敛性将会受到极大的影响，然而，动量修正组件可以缓解这个问题。因此，在现有分布式训练流程的基础上，将动量修正组件引入了  $k$ -RNGC 算法。

---

**算法 4.2 基于局部梯度压缩比的  $k$ -互异近邻梯度选择算法流程**


---

**输入:** 训练数据  $X$ , 网络模型局部梯度压缩比  $\mathbf{R}^m$  以及局部梯度保留数量  $\mathfrak{N}^m$

**输出:**  $k$ -互异近邻梯度子集  $\hat{\mathcal{G}}^k$

**算法:**

```

1: for  $j = 1; j \leq L; j++$  do
2:   初始化:  $k \leftarrow \mathfrak{N}^m[j], \mathcal{G}^k \leftarrow \emptyset$ 
3:   while  $|\mathcal{G}^k| \neq \mathfrak{N}^m[j]$  do
4:     根据公式 (4-8) 计算  $\mathcal{G}^k$ 
5:      $k \leftarrow binaryTraversSearch^{[87]}(k)$ 
6:      $\hat{\mathcal{G}}^k \leftarrow \hat{\mathcal{G}}^k \cap \mathcal{G}^k$ 
7:      $k \leftarrow k + 1$ 
8: return  $\hat{\mathcal{G}}^k$ 

```

---

在网络模型的分布式训练过程中, 动量 SGD 已被广泛使用, 并代替了普通的 SGD 优化算法。例如, 在 Collective 网络架构上使用普通动量 SGD 进行分布式训练, 网络模型参数优化过程可表述如下<sup>[89]</sup>:

$$\begin{aligned} \mathbf{u}^{(t)} &= m\mathbf{u}^{(t-1)} + \sum_{k=1}^N \mathbf{g}_i^{k(t)}, \\ \mathbf{W}^{(t+1)} &= \mathbf{W}^{(t)} - \eta \mathbf{u}^{(t)} \end{aligned} \quad (4-9)$$

其中,  $m$  表示动量,  $N$  是分布式训练网络的 Worker 数量。将网络模型权重  $\mathbf{W}$  中的第  $i$  个位置的权重展平并记为  $\varsigma_i$ , 则经过  $T$  次迭代后的  $\varsigma_i$  如公式 (4-10) 所示:

$$\varsigma_i^{(t+T)} = \varsigma_i^t - \eta \left[ \cdots + \left( \sum_{\tau=0}^{T-2} m^{(\tau)} \right) \mathbf{g}_i^{k(t+1)} + \left( \sum_{\tau=0}^{T-1} m^{(\tau)} \right) \mathbf{g}_i^{k(t)} \right] \quad (4-10)$$

此时, 如果将动量 SGD 直接应用于稀疏化场景, 则更新规则如公式 (4-11) 所示, 不再等价于公式 (4-9)。

$$\begin{aligned} \mathbf{v}^{k(t)} &= \mathbf{v}^{k(t-1)} + \mathbf{g}^{k(t)} \\ \mathbf{u}^{(t)} &= m\mathbf{u}^{(t-1)} + \sum_{k=1}^N sparse(\mathbf{v}^{k(t)}) \\ \mathbf{W}^{(t+1)} &= \mathbf{W}^{(t)} - \eta \mathbf{u}^{(t)} \end{aligned} \quad (4-11)$$

其中, 第一项  $\mathbf{v}^{k(t)}$  表示训练节点  $k$  上的局部梯度累计。一旦累计值大于某个阈值或到达通信延迟时间节点,  $\mathbf{v}^{k(t)}$  将通过稀疏化函数  $sparse(\cdot)$  进行硬阈值处理并通过第二项进行编码, 然后发送到网络中的其他节点进行梯度同步操作。相似地, 经过稀疏

化更新间隔  $T$  后,  $\varsigma_i$  可由公式 (4-12) 表示。

$$\varsigma_i^{(t+T)} = \varsigma_i^t - \eta \left( \dots + \mathbf{g}_i^{k(t+1)} + \mathbf{g}_i^{k(t)} \right) \quad (4-12)$$

对比公式 (4-10) 与公式 (4-12) 可以发现, 积累系数  $\sum_{\tau=0}^{T-1} m^{(\tau)}$  消失, 而这必将导致网络模型收敛性下降。当梯度稀疏度变大, 更新间隔  $T$  剧增时, 积累系数消失的副作用将严重影响模型性能表现。为了避免这一副作用带来的影响, 需要在公式 (4-11) 基础上进行动量修正, 以确保梯度稀疏更新与公式 (4-9) 表示的梯度密集更新二者等价。

将公式 (4-11) 中的速度  $\mathbf{v}^{(t)}$  视为“梯度”, 那么公式 (4-11) 中的第三项则可以视为“梯度  $\mathbf{v}$ ”的普通 SGD。又因局部梯度累计已被证明对于普通 SGD 是有效的<sup>[31]</sup>, 因此可以将局部累计的对象由“真实梯度  $\mathbf{g}$ ”变更为“速度  $\mathbf{v}$ ”, 并将公式 (4-11) 推广应用到公式 (4-9), 得到公式 (4-13):

$$\begin{aligned} \mathbf{u}^{k(t)} &= m \mathbf{u}^{k(t-1)} + \mathbf{g}^{k(t)} \\ \mathbf{v}^{k(t)} &= \mathbf{v}^{k(t-1)} + \mathbf{u}^{k(t)} \\ \mathbf{W}^{(t+1)} &= \mathbf{W}^{(t)} - \eta \sum_{k=1}^N \text{sparse}(\mathbf{v}^{k(t)}) \end{aligned} \quad (4-13)$$

其中, 前两项是动量修正后的局部梯度累加操作, 且累加结果  $v^{k(t)}$  将用于后续的梯度稀疏化以及网络通信等操作。

需要注意的是, 对局部梯度积累的简单修正, 就可以基于公式 (4-13) 推导出公式 (4-10) 中的积累系数  $\sum_{\tau=0}^{T-1} m^{(\tau)}$ , 很好地避免了积累系数消失的副作用。这种修正方法被称为  $k$ -RNGC 算法中的“动量修正”组件, 它只需要对更新方程进行的简单调整, 而且不会产生任何新的超参数。

#### 4.2.4 预热训练

在网络模型训练的早期阶段, 模型参数变化较快, 梯度更加多方向化和激进, 而梯度稀疏化限制了网络模型的变化范围, 无疑会影响了网络的收敛速度。同时, 训练早期剩余的激进梯度在被选中用于下一次更新之前被积累, 因此它们对模型优化方向的影响可能会超过最新梯度, 从而需要更多轮迭代训练进行优化方向的修正。在大规模小批量训练中, 引入预热训练组件已被证实很有帮助<sup>[90]</sup>。具体来说, 就是在预热训练期间, 使用不太激进的学习率来减缓训练开始时网络优化方向的变化速度, 并采用不太激进的梯度稀疏度, 以减少被延迟的极端梯度数量。通过本章第 4.3.2 小节  $k$ -RNGC 算法预热组件分析实验, 可得出如下结论:

- 1) 预热训练有助于提升网络模型的最终性能表现, 且模型表现提升值随着预热

迭代轮数的增加，逐渐趋近与一个稳定值。

2) 在预热训练阶段不使用线性的形式增加梯度稀疏度，而是将稀疏度从相对较小的值以指数级的形式增加到最终预设值，可以很大程度上帮助网络模型适应较大的稀疏度。

如算法 4.3 所示，展示了本章提出的基于  $k$ -互异近邻的梯度压缩算法的整体框架，详细描述了该算法在分布式训练过程的使用方式，其核心组件主要包括预热训练、局部梯度累计、动量修正、计算感知、 $k$ -互异近邻梯度选择、梯度稀疏化、梯度延迟等。

---

#### 算法 4.3 基于 $k$ -互异近邻的梯度压缩算法整体框架及流程

**输入：**训练数据  $\mathbf{X}^m$ ，网络模型权重  $\mathbf{W}$ ，目标压缩比  $cr$ ，最大迭代轮数  $Epoch_{max}$ ，预热训练迭代轮数  $Epoch_{wm}$

**输出：**训练完成的网络模型权重  $\hat{\mathbf{W}}$

**算法：**

```

1: 初始化:  $\mathbf{u}^m \leftarrow 0, \mathbf{v}^m \leftarrow 0$ 
2: for  $e = 1; e \leq Epoch_{max}; e++$  do
3:   // 局部梯度累计
4:   for  $i = 1; i \leq \varphi; i++$  do
5:      $\mathbf{g}^{m(t)} \leftarrow \mathbf{g}^{m(t)} + \frac{1}{\varphi N} \nabla \mathcal{L}(\mathbf{X}; \mathbf{W});$ 
6:   if  $i \leq Epoch_{wm}$  then
7:     // 局部梯度更新
8:     根据公式 (4-13) 进行局部梯度更新;
9:   else
10:    根据算法 4.1 获取网络模型局部梯度压缩比  $\mathbf{R}^m$  以及局部梯度保留数量  $\mathfrak{N}^m$ ;
11:    根据算法 4.2 获取  $k$ -互异近邻梯度子集  $\hat{\mathcal{G}}^k$ ;
12:    // 梯度稀疏化 + 局部梯度更新
13:    根据公式 (4-13) 以及  $\hat{\mathcal{G}}^k$  依次进行梯度稀疏化、局部梯度更新;
14:    根据预先设定的 Allreduce 算法，在合适的时机与其他 Worker 进行梯度同步操作
15:     $\hat{\mathbf{W}} \leftarrow \mathbf{W}^{(t+1)} \leftarrow SGD(\mathbf{W}^{(t)}, \mathbf{g}^{n[k]^\star})$ 
16: return  $\hat{\mathbf{W}}$ 

```

---

## 4.3 实验与结果分析

### 4.3.1 实验环境及其相关参数配置

本章所涉及的分布式训练实验环境与第三章保持一致，且面对不同的下游任务时，本章验证实验所涉及到的网络模型、数据集之间的对应关系也与第三保持一致，如表 3.1 所示。需要注意的是，由于 GCAC 算法与本章提出的  $k$ -RNGC 算法超参数区

别较大，因此分布式训练实施过程中的参数需要重新设置，主要包括 Batch 的大小  $\varphi$ 、最大迭代轮数  $Epoch_{max}$ 、预热训练迭代轮数  $Epoch_{wm}$ 、学习率  $\eta$ 、动量 Momentum、权重衰减因子、优化器、梯度同步延迟阈值等，具体参数值如表 4.1 所示。

表 4.1  $k$ -RNGC 算法相关超参数设置

数据集	AeroRIT	Xiongan New Area (Matiwan Village)	CIFAR10
Batch 大小 $\varphi$	310000	200000	128
梯度同步延迟阈值		3	
$Epoch_{max}$		200	
$Epoch_{wm}$		5	
学习率 $\eta$		0.0001	0.01
动量		-	0.9
权重衰减因子		-	0.0005
优化器		Adam	SGD
压缩比		1 / 10 / 50 / 100	

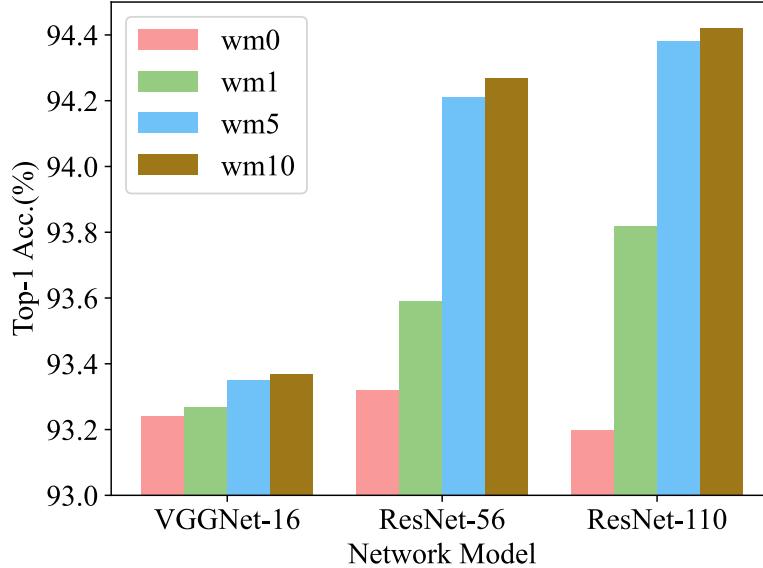
### 4.3.2 预热训练组件分析

为了深入分析、讨论预热训练组件，我们采用控制变量实验方法，将预热组件作为实验变量。表 4.2 展示了在 CIFAR10 数据集上，VGGNet-16、ResNet-56 以及 ResNet-110 三个不同规模的网络模型采用  $k$ -RNGC 梯度压缩算法时的 Top-1 分类精度。在该表中，“-”表示分布式训练过程中不使用预热训练组件，也可书写成“wm0”，表示从第一次迭代开始就进行梯度压缩操作，这无疑会导致网络模型的前几步优化方向有所偏差，将需要更多次迭代操作来弥补，具体表现就是模型收敛速度慢。“wm5”表示采用预热训练组件且  $Epoch_{wm}$  设置为 5，也就是说预热期的终止迭代轮数为 5，并且在预热期间分布式训练的学习率逐渐增大，梯度压缩比采用指数增加的方式逐步增加到预设值  $cr$ 。“wm5o”表示采用预热训练组件、 $Epoch_{wm}$  设置为 5 并且预热训练期间中  $cr$  保持为 1。另外，该实验的目的是探究  $k$ -RNGC 算法中预热训练组件对网络模型最终性能表现的影响，为了控制变量突出该组件的作用，我们将设置全局梯度压缩比  $cr$  为 10。可以看出，无论是 VGGNet 神经网络模型，还是 ResNet 神经网络模型，当预热训练组件设置为“wm5”时性能表现最优。

表 4.2 CIFAR10 数据集上，预热训练组件对不同网络模型精度的影响

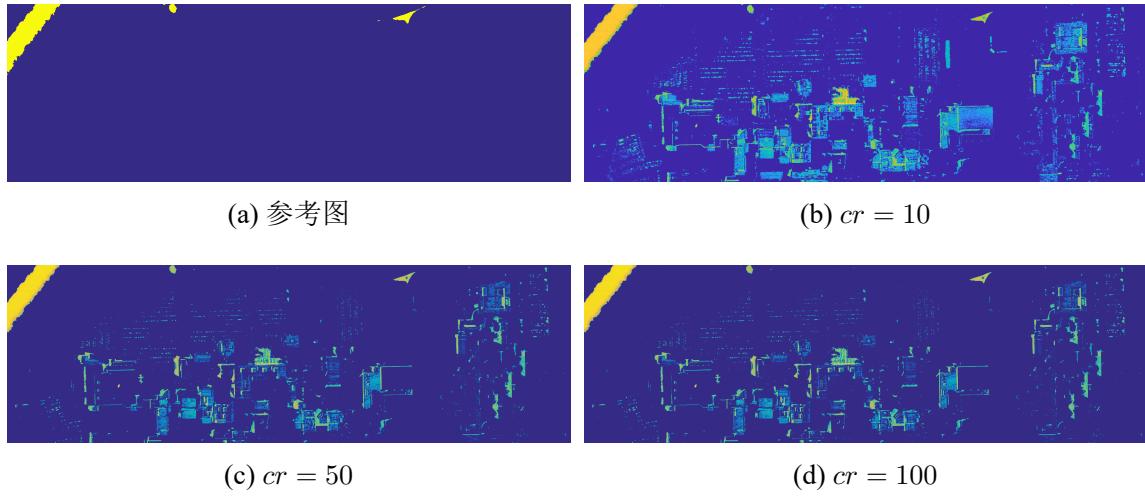
网络模型	VGGNet-16			ResNet-56			ResNet-110		
预热训练	-	wm5	wm5o	-	wm5	wm5o	-	wm5	wm5o
Top-1 Acc. (%)	93.24	<b>93.35</b>	93.26	93.32	<b>94.12</b>	94.07	93.20	<b>94.38</b>	94.22

另外，在得知具有预热训练组件的模型性能表现后，我们又探究  $Epoch_{wm}$  对模型性能的影响做了一个实验。在该实验中，采用网络模型为 VGGNet-16、ResNet-16 以及 ResNet-110，实验变量为  $Epoch_{wm} \in \{1, 5, 10\}$ ，具体实验结果如图 4.2 所示。从该图可以看到，随着  $Epoch_{wm}$  的增加网络模型性能越好，但是在预热期间网络模型的全局梯度压缩比不能满足预设条件，也就是说， $Epoch_{wm}$  的增加会导致网络拓扑中通信数据量的相对增加，进一步导致分布式训练时长增加，并且该效应随  $Epoch_{wm}$  的增大愈加明显，因此不能无限制地增大。另外，从该图中的 ResNet 网络模型结果可以发现， $Epoch_{wm}$  从 1 增大到 5 带来的性能提升远远高于从 5 增大到 10，该现象表明随着  $Epoch_{wm}$  增大所带来的性能提升在逐渐降低。基于该实验结果综合考虑后，决定后续的验证实验将设置  $Epoch_{wm}$  为 5，即预热组件设置为“wm5”。

图 4.2  $Epoch_{wm}$  对不同网络模型性能的影响

### 4.3.3 高光谱目标检测实验结果分析

本小节，我们通过将  $k$ -RNGC 算法应用于高光谱目标检测场景，来说明该算法的有效性以及在分布式训练中的加速效果。与第三章类似，我们将在 AeroRIT 以及 Xiongan New Aera (Matiwan Village) 两个大规模高光谱遥感数据集上运行 AAE 网络

图 4.3 AeroRIT 数据集上，不同压缩比对  $k$ -RNGC 算法的 HTD 可视化结果

模型。

图 4.3 展示了 AAE 网络模型在 AeroRIT 高光谱遥感图像数据集上采用  $k$ -RNGC 算法时的可视化目标检测结果图，其中图 4.3(a) 展示了 AeroRIT 的真实目标参考图，图 4.3(b)、图 4.3(c) 以及图 4.3(d) 分别展示了压缩比为 10、50、100 时的检测结果图。通过如图 4.3 所示的实验结果可以看出，相较于 GCAC 算法，尽管  $k$ -RNGC 算法步骤有所增加且更为复杂，但该算法在网络模型性能表现上更加优秀，感兴趣的目标都可以被很好地检测出来。

表 4.3 AAE 网络模型采用  $k$ -RNGC 算法在不同压缩比下的量化结果

数据集	压缩比 $cr$	ACU (%)	FPR (%)
AeroRIT	10	<b>96.03</b>	<b>1.29</b>
	50	95.21	1.76
	100	94.83	2.13
Xiongan New Area (Matiwan Village)	10	<b>94.30</b>	<b>7.23</b>
	50	93.59	8.62
	100	93.82	10.18

表 4.3 展示了将 AAE 网络模型应用于 AeroRIT 高光谱遥感图像数据集，并在分布式训练时采用  $k$ -RNGC 算法进行梯度压缩的相关量化结果，其中梯度压缩比  $cr$  值与图 4.3 中  $cr$  值一一对应。结合该表与表 3.3 可以发现，本章所提出的  $k$ -RNGC 算法在 AeroRIT 数据集上的检测精度表现更优。进一步结合表 3.5 与表 4.5，对比其他梯度压缩算法可以发现，虽然基于  $k$ -互异近邻梯度压缩算法由于步骤复杂导致在分

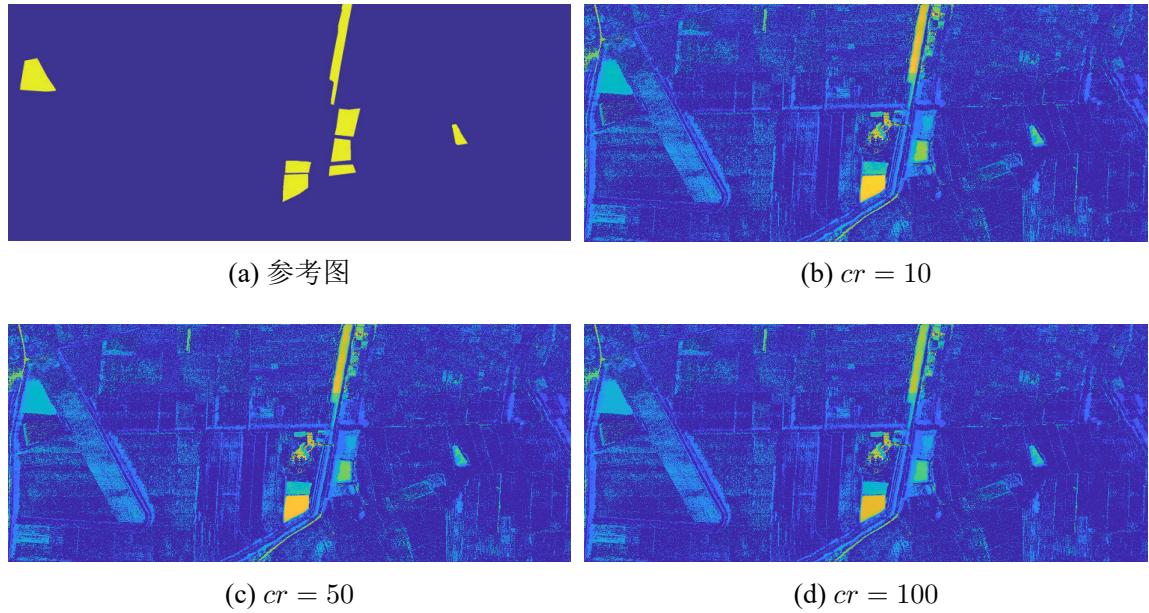
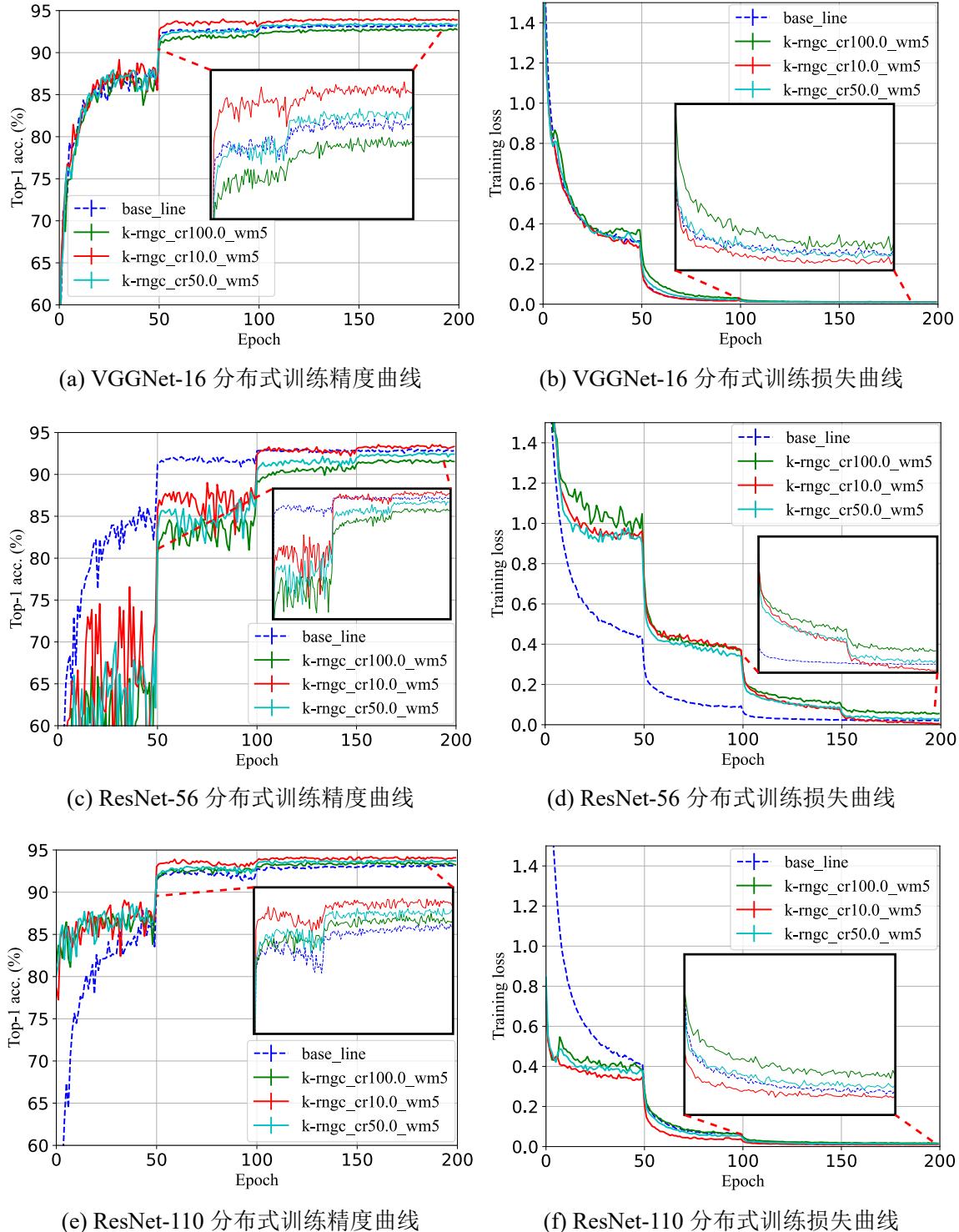


图 4.4 Xiongan 数据集上，不同压缩比对  $k$ -RNGC 算法的 HTD 可视化结果

布式训练中迭代耗时有所增加，但它取得了更好的网络模型性能表现，尤其是解决了 GCAC 所面临的大压缩比难题。例如当压缩比  $cr$  设置为 100 时， $k$ -RNGC 算法的目标检测精度仅仅下降了 1.16%，而 GCAC 算法则下降了 5.71%。

为了更好地展示  $k$ -RNGC 的优势，我们还将其应用到了更大规模的高光谱遥感图像数据集 Xiongan New Aera (Matiwan Village)。如图 4.4 展示了 AAE 神经网络模型采用  $k$ -RNGC 算法进行高光谱目标检测任务的相关检测结果图。其中，图 4.4(a) 展示了该数据集真实目标的参考图，图 4.4(b)、图 4.4(c)、图 4.4(d) 分别展示了压缩比  $cr$  设置为 10、50、100 时的检测结果图。对比结果图可以发现， $k$ -RNGC 算法下的 AAE 网络模型能够在不同压缩比的情况下，很好地检测到数据集中的真实目标，尤其是大压缩比的情况下，模型的性能表现依然优异。与图 3.5 中的其他算法的检测结果对比可知，本章提出的梯度压缩算法不仅可以准确地检测并定位到真实目标的位置，而且检测结果图中背景干扰信息相对较少，即在大压缩比、大规模数据集的场景下也具有很好的表现。

为了进一步量化高光谱目标检测的效果，表 4.3 展示了 AAE 网络模型在大规模高光谱遥感图像数据集 Xiongan New Aera (Matiwan Village) 上，采用  $k$ -RNGC 算法进行高光谱目标检测任务的量化结果。与表 3.3 进行对比可以得出，当采用  $k$ -RNGC 算法时，AAE 网络模型在该超大规模的数据集上性能表现更好。当压缩比  $cr$  为 10 时，目标检测精度不仅超过其他梯度压缩算法，甚至要优于 Baseline (94.30% vs. 94.10%)。在大压缩比 ( $cr$  设置为 100) 场景下，本章所提出的梯度压缩算法相较于 Baseline 也仅仅下降了 0.28 个百分点，而 GCAC 算法则下降了 3.69 个百分点。

图 4.5 CIFAR10 数据集上，不同压缩比对不同网络模型采用  $k$ -RNGC 算法的分布式训练曲线

#### 4.3.4 图像分类实验结果分析

上一小节将  $k$ -RNGC 算法应用到高光谱目标检测的任务场景中，实验结果表明该算法取得了不错的实验效果。为了验证本章提出的  $k$ -互异近邻梯度压缩算法的泛

化性，本小节将  $k$ -RNGC 算法应用于自然图像分类场景，来进一步说明该算法的有效性以及在分布式训练中的加速效果。与第三章类似，我们将在 CIFAR10 自然图像数据集上运行 VGGNet-16、ResNet-56 以及 ResNet-110 三个不同规模的网络模型，并且实验最终能够结果仍然采用与上一章相同的评价指标。

图 4.5 展示了在不同压缩比的情况下，不同网络模型在 CIFAR10 自然图像数据集上，采用基于  $k$ -互异近邻的梯度压缩算法进行分布式训练时的 Top-1 精度曲线以及损失曲线。当  $cr$  设置为 1 时，网络模型将不进行梯度压缩，也不使用动量修正、预热训练等组件，即 Baseline 训练。通过图 4.5 中的各个实验曲线可以发现，无论是 VGGNet 网络模型还是 ResNet 网络模型都取得了不错的性能表现，如 Top-1 精度以及损失曲线的最终收敛值都在 Baseline 曲线附近。当采用 ResNet-56 网络模型时，分布式训练损失曲线收敛速度不如 Baseline，可能的原因是该网络模型复杂度较低、参数数量较小，导致其高阶抽象能力欠缺，但是最终收敛后的网络模型性能表现却相当具有竞争力。当压缩比  $cr$  设置为 100 时，VGGNet-16 和 ResNet-100 两个高复杂度网络模型的损失曲线收敛速度都不逊色于 Baseline，而且最终 Top-1 精度并没有损失太多，说明  $k$ -RNGC 算法克服了 GCAC 算法面临的大压缩比问题。

表 4.4 CIFAR10 数据集上，不同网络模型采用  $k$ -RNGC 算法进行分布式训练的量化结果

网络模型	压缩算法	梯度压缩比 $cr$	Top-1 Acc. (%)	损失函数收敛值
VGGNet-16	-	Baseline	93.17	0.0105
	$k$ -RNGC	10× / 50× / 100×	<b>93.35</b> / 93.32 / 92.78	<b>0.0103</b> / 0.0107 / 0.0122
ResNet-56	-	Baseline	93.07	<b>0.0211</b>
	$k$ -RNGC	10× / 50× / 100×	<b>93.19</b> / 92.80 / 92.10	0.0257 / 0.0551 / 0.0756
ResNet-110	-	Baseline	93.24	0.0116
	$k$ -RNGC	10× / 50× / 100×	<b>94.22</b> / 93.84 / 93.64	<b>0.0100</b> / 0.0132 / 0.0160

表 4.4 展示了在 CIFAR10 图像数据集上使用不同网络模型进行分布式训练的最终量化结果。与表 3.4 中的其他梯度压缩算法相比，可以看出本章所提出的  $k$ -RNGC 算法表现更稳定且性能更好，并且很好地克服了大压缩比所带来的网络模型性能严重下降的问题。 $k$ -RNGC 算法在 VGGNet 以及 ResNet 网络模型上都取得了不错的性能表现，说明该算法对不同类型网络模型的兼容性很强。

### 4.3.5 耗时分析

对于分布式网络中的任一 Worker 来说，一次完整迭代训练过程耗时如下：

$$T_{iter} = T_{calculate} + T_{compress} + T_{synchronize} \quad (4-14)$$

其中， $T_{calculate}$  表示单次迭代过程中，网络模型读取数据、前向传播以及反向传播的计算耗时， $T_{compress}$  表示梯度同步前的压缩（稀疏化和/或量化）以及编码操作耗时， $T_{synchronize}$  表示网络各个 Worker 之间进行梯度同步操作的耗时。各个阶段耗时与网络模型体积、节点的计算能力以及网络带宽存在较强的相关性，描述如下：

$$T_{calculate} \propto \frac{M}{F} \quad (4-15)$$

$$T_{compress} \propto \frac{G}{F} \quad (4-16)$$

$$T_{synchronize} \propto \frac{G/cr}{B} = \frac{\hat{G}}{B} \quad (4-17)$$

其中， $G$  表示整个网络模型中梯度规模大小， $M$  表示网络模型完整计算一次数据流（从前向传播开始到反向传播结束）所需计算量， $F$  表示分布式网络中各 Worker 的计算能力， $B$  表示有效网络带宽， $cr$  表示梯度压缩比， $\hat{G}$  为网络模型训练时压缩后的梯度规模大小。显然， $G$  只与网络模型本身有关， $B$  与  $F$  则随着网络设备状态以及训练的物理环境变化而变化， $M$  则与网络模型复杂度、输入数据规模以及单次迭代时 Batch 的大小有关。

表 4.5 展示了 CIFAR10 数据集上，Batch 大小  $\varphi$  为 128 时，不同网络模型采用  $k$ -RNGC 算法进行分布式训练时的单次迭代用时。梯度同步阈值决定了两次梯度同步间隔内进行多少个 Batch 训练数据的输入，其值越大，输入到网络模型中的数据量越大，梯度同步时间间隔越长。通过分析不难知道，当梯度同步阈值为 3 时，梯度压缩用时占比约为 50%；当阈值为 1 时，整个算法不会进行局部梯度累计操作，提高了网络通信频次，没有进行梯度修正操作，对最终网络模型的性能表现造成较大影响；当阈值为 5 时，梯度压缩平均用时占比超过 50%，会严重影响分布式训练进度的推进。因此，本文将设置梯度同步阈值为 3。

与表 3.5 对比可以发现，相较于 DGC 算法，本章提出的  $k$ -RNGC 算法单次迭代耗时大幅降低（VGGNet-16: 906.56ms vs. 639.73ms，ResNet-56: 460.00ms vs. 236.65ms，ResNet-110: 866.94ms vs. 413.84ms），取得了不错的加速效果。与第三章提出的 GCAC 算法相比，本章提出的  $k$ -RNGC 算法步骤增多，实际场景中分布式部署难度有所上升，并因全局梯度计算感知算法的引入，导致计算复杂度以及  $T_{compress}$  变大，最终

表 4.5 CIFAR10 数据集上，不同模型采用  $k$ -RNGC 算法进行分布式训练的单次迭代耗时

网络模型	梯度同步	迭代平均用时	梯度压缩平均用时	梯度压缩用时
	延迟阈值	$T_{iter}$ (毫秒)	$T_{calculate} + T_{compress}$ (毫秒)	占比 (%)
VGGNet-16	1	916.11	275.12	30.03
	3	1188.43	<b>639.73</b>	53.83
	5	2183.81	1559.02	71.39
ResNet-56	1	227.93	66.49	29.17
	3	430.04	<b>236.65</b>	55.03
	5	714.49	536.08	75.03
ResNet-110	1	301.40	86.17	28.59
	3	797.08	<b>413.84</b>	51.92
	5	1204.74	886.57	73.59

表现在迭代平均用时上的现象就是： $k$ -RNGC 算法多消耗了 25% ~ 40% 左右的时间，但相较于 Baseline 来说， $k$ -RNGC 算法仍然取得了不错的分布式训练加速效果。

## 4.4 本章小结

本章首先分析了第三章提出的 GCAC 梯度压缩算法的优势与不足，对网络模型中的梯度张量及其特性进行重新思考、分析，提出了一种基于  $k$ -互异近邻的梯度压缩算法（ $k$ -RNGC）。该算法的核心思路是，在梯度压缩和通信延迟的协同思想指导下，通过计算感知梯度的重要性，然后采用  $k$ -互异近邻梯度选择算法确定待传输梯度，并在分布式训练过程中引入了动量修正、预热训练等关键组件。接着，详细介绍了  $k$ -RNGC 算法的相关理论，主要包括该算法的相关核心步骤、相关公式及其作用、整体框架以及分布式训练流程。然后，通过组件分析、耗时分析实验，确定了分布式训练中的相关超参数设置，验证了  $k$ -RNGC 算法中每一个组件的必要性。最后，为验证本章提出的梯度压缩算法的可行性以及泛化性，在高光谱遥感数据集以及自然图像数据集上进行了高光谱目标检测和图像分类两大任务，实验结果证明了  $k$ -RNGC 算法的有效性以及通用性。



## 第五章 总结与展望

### 5.1 工作总结

近年来，诸如高光谱遥感技术以及深度学习技术的高速发展，用于各种任务场景的网络模型复杂度以及数据集规模呈现爆炸式增长趋势，导致面临模型部署困难、训练时间长等众多挑战。分布式训练突破了单机训练的算力瓶颈，协同多个 Worker 上的计算资源，加速了网络模型训练，极大程度上缩短了开发、部署周期。然而，随着大规模数据集以及网络模型的广泛使用，分布式训练过程中各 Worker 之间由于具有通信频次高、通信流量大、通信数据全尺寸全精度的特点，产生了大量的通信开销，导致无法完全发挥并行计算的优势，严重拖慢了训练速度。本文的主要研究内容聚焦于深度学习场景中的分布式数据并行训练，针对通信开销大的问题，从通信流量大的角度出发，协同压缩梯度和缩减通信频次两大优化技术，以实现减少网络通信数据量、加速分布式训练的目的。本文的主要贡献包括以下几个方面：

第一，提出了基于近似质心的梯度压缩算法 GCAC。该算法首先提取出网络模型中局部梯度张量的近似质心，然后在满足预设的梯度压缩比的同时，依据梯度张量中各个梯度与近似质心之间的距离度量筛选出合适的梯度用以网络通信，大幅减少了网络通信流量，提高了分布式训练速度。GCAC 算法在分布式训练过程中保留了模型参数优化所需的关键梯度信息，有助于网络模型更快收敛。本文还通过实验的方式，验证了 GCAC 算法在不同任务场景、不同网络模型中的有效性，能够保证网络模型快速收敛及其性能表现。尤其是在小压缩比的情况下，无论是大模型还是小模型，GCAC 算法都能够达到不错模型的最终精度，甚至还能达到提高精度的效果。然而，我们发现在大压缩比小模型的场景下，GCAC 算法由于实践过程采用了近似质心的加速计算技术，导致网络模型收敛性较差，性能下降较多。

第二，协同梯度压缩与缩减通信频次等技术，本文提出了基于  $k$ -互异近邻的梯度压缩算法  $k$ -RNGC。GCAC 算法无法适应大压缩比小模型的分布式训练场景，其根本原因是采用近似质心而导致的梯度选择偏差。因此，依据网络模型训练流程与梯度张量特性，提出了另外一种梯度压缩算法  $k$ -RNGC，该算法结合了梯度压缩与通信频次缩减技术，即通过梯度计算感知算法与  $k$ -互异近邻梯度选择算法的结合，优选满足预设全局压缩比阈值的梯度，并引入动量修正和预热训练等组件，显著降低了分布式训练过程中的网络通信流量，大幅度加快了训练速度。实验证明，相较于 GCAC 算法， $k$ -RNGC 算法虽然在单次迭代过程中的平均耗时较多，但是，它不仅能够适应大数据大模型的分布式训练场景，而且网络模型的性能表现更为优异。另外，值得注意

的是，面临大压缩比小模型的场景时， $k$ -RNGC 算法不存在网络模型收敛性速度慢、性能下降严重的问题，并且具有极其优秀的通用性。

## 5.2 未来研究展望

本论文面向大规模数据高复杂度模型的分布式训练场景，针对训练过程中的网络通信开销大的瓶颈问题，基于梯度压缩、通信频次缩减等技术，对其进行进一步优化，提出了基于近似质心的梯度压缩算法（GCAC）以及基于  $k$ -互异近邻梯度压缩算法（ $k$ -RNGC）。本文提出的两种算法对分布式训练的过程进行优化，能够在尽量保证模型性能表现的情况下，加速完成训练过程，极大程度缩短了模型训练时间。然而，由于本人知识、时间以及精力有限，目前的工作仍存在诸多不足之处，为了进一步优化，可基于本文工作从以下几点进行改进。

第一，GCAC 算法是根据梯度与近似质心之间的距离进行排序，而  $k$ -RNGC 算法则将模型中所有梯度进行全局排序确定局部梯度压缩比，本文提出的两个梯度压缩算法的梯度分析、排序都是在空域中进行的。后续工作可以考虑引入频域分析，并将空频域分析结果融合，对模型中的梯度特征进行进一步挖掘以确定更优的排序规则。

第二，本文工作所提出的梯度压缩算法面向的是数据并行结合 Collective 架构下的分布式训练场景，然而，参数服务器架构在某些情况下也是相当主流的架构。因此，后续研究可以对 GCAC 算法以及  $k$ -RNGC 算法进行完善优化，并做出相应的验证实验，使其兼容不同网络架构，进一步提高其通用性。

第三，本文工作所提出的梯度压缩算法在 VGGNet 以及 ResNet 网络模型进行了实验验证，均取得不错的模型训练加速效果以及性能表现。然而，仍有一些现有的网络模型并没有进行实验验证，例如 GoogleNet、RNN、MobileNet、Transformer 等，后续研究工作中可以考虑将 GCAC 算法以及  $k$ -RNGC 算法应用于更多类型的网络模型训练中，并根据网络模型表现及时做出相应的算法评估与修正。

第四，GCAC 算法在分布式训练实践中仅仅涉及梯度压缩技术， $k$ -RNGC 则是结合了梯度压缩技术与通信频次压缩技术。因此，后续研究工作可以提出或实现更多的梯度压缩算法（例如通信频次缩减算法、梯度量化算法等），然后结合这两种算法应用于各种场景下的分布式训练场景，进一步验证 GCAC 算法和  $k$ -RNGC 算法对其他梯度压缩算法的兼容性。

## 参考文献

- [1] LIU W, ANGUELOV D, ERHAN D, et al. Ssd: Single shot multibox detector[C]//Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. 2016: 21-37.
- [2] LIN T, STICH S U, PATEL K K, et al. Don’t use large mini-batches, use local SGD[J]. arXiv preprint arXiv:1808.07217, 2018.
- [3] LIN T Y, DOLLÁR P, GIRSHICK R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.
- [4] SONG L, PAN P, ZHAO K, et al. Large-scale training system for 100-million classification at alibaba [C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020: 2909-2930.
- [5] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
- [6] XIE Q, LUONG M T, HOVY E, et al. Self-training with noisy student improves imagenet classification[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 10687-10698.
- [7] ZHANG C L, ZHANG H, WEI X S, et al. Deep bimodal regression for apparent personality analysis [C]//European conference on computer vision. 2016: 311-324.
- [8] LEE H, PHAM P, LARGMAN Y, et al. Unsupervised feature learning for audio classification using convolutional deep belief networks[J]. Advances in neural information processing systems, 2009, 22.
- [9] BRATTOLI B, TIGHE J, ZHDANOV F, et al. Rethinking zero-shot video classification: End-to-end training for realistic applications[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 4613-4623.
- [10] XIE S, SUN C, HUANG J, et al. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 305-321.
- [11] YUAN L, WANG T, ZHANG X, et al. Central similarity quantization for efficient image and video retrieval[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 3083-3092.
- [12] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25.

- [13] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
- [14] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
- [15] VASWANI A, SHAZER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.
- [16] ZARE A, JIAO C, GLENN T. Discriminative multiple instance hyperspectral target characterization [J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 40(10): 2342-2354.
- [17] JIANG K, XIE W, LEI J, et al. LREN: Low-rank embedded network for sample-free hyperspectral anomaly detection[C]//Proceedings of the AAAI Conference on Artificial Intelligence: vol. 35: 5. 2021: 4139-4146.
- [18] XIE W, ZHANG X, LI Y, et al. Weakly supervised low-rank representation for hyperspectral anomaly detection[J]. IEEE Transactions on Cybernetics, 2021, 51(8): 3889-3900.
- [19] GREEN R O, EASTWOOD M L, SARTURE C M, et al. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)[J]. Remote sensing of environment, 1998, 65(3): 227-248.
- [20] CHANG C I. Hyperspectral imaging: techniques for spectral detection and classification: vol. 1[M]. Springer Science & Business Media, 2003.
- [21] LI J, BENEDIKTSSON JA, ZHANG B, et al. Spatial technology and social media in remote sensing: A survey[J]. Proceedings of the IEEE, 2017, 105(10): 1855-1864.
- [22] ZHANG B, CHEN Z, PENG D, et al. Remotely sensed big data: Evolution in model development for information extraction [point of view][J]. Proceedings of the IEEE, 2019, 107(12): 2294-2301.
- [23] 马纪涛, 谢卫莹, 雷杰, 等. 端到端分布式联合优化的空谱自编码密度估计模型[J]. 电子学报, 2023, 51(4): 1006.
- [24] PLAZA A, PLAZA J, PAZ A, et al. Parallel hyperspectral image and signal processing [applications corner][J]. IEEE Signal Processing Magazine, 2011, 28(3): 119-126.
- [25] ZINKEVICH M, WEIMER M, LI L, et al. Parallelized stochastic gradient descent[J]. Advances in neural information processing systems, 2010, 23.
- [26] CHILIMBI T, SUZUE Y, APACIBLE J, et al. Project adam: Building an efficient and scalable deep learning training system[C]//11th USENIX symposium on operating systems design and implementation (OSDI 14). 2014: 571-582.
- [27] XING E P, HO Q, DAI W, et al. Petuum: A new platform for distributed machine learning on big data[C]//Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2015: 1335-1344.

- [28] MORITZ P, NISHIHARA R, STOICA I, et al. Sparknet: Training deep networks in spark[J]. arXiv preprint arXiv:1511.06051, 2015.
- [29] 刘铁岩, 陈薇, 王太锋. 分布式机器学习: 算法、理论与实践[M]. 北京: 机械工业出版社, 2018.
- [30] ALISTARH D, GRUBIC D, LI J, et al. QSGD: Communication-efficient SGD via gradient quantization and encoding[J]. Advances in neural information processing systems, 2017, 30.
- [31] LIN Y, HAN S, MAO H, et al. Deep gradient compression: Reducing the communication bandwidth for distributed training[J]. arXiv preprint arXiv:1712.01887, 2017.
- [32] SEIDE F, FU H, DROPOPO J, et al. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns[C]//Fifteenth annual conference of the international speech communication association. 2014.
- [33] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.
- [34] SERGEEV A, DEL BALSO M. Horovod: fast and easy distributed deep learning in TensorFlow[J]. arXiv preprint arXiv:1802.05799, 2018.
- [35] WANGNI J, WANG J, LIU J, et al. Gradient sparsification for communication-efficient distributed optimization[J]. Advances in Neural Information Processing Systems, 2018, 31.
- [36] WANG H, SIEVERT S, LIU S, et al. Atomo: Communication-efficient learning via atomic sparsification[J]. Advances in neural information processing systems, 2018, 31.
- [37] STRÖM N. Scalable distributed DNN training using commodity GPU cloud computing[J]. 2015.
- [38] STICH S U, CORDONNIER J B, JAGGI M. Sparsified SGD with memory[J]. Advances in Neural Information Processing Systems, 2018, 31.
- [39] AJI A F, HEAFIELD K. Sparse communication for distributed gradient descent[J]. arXiv preprint arXiv:1704.05021, 2017.
- [40] CUI L, SU X, ZHOU Y, et al. ClusterGrad: Adaptive gradient compression by clustering in federated learning[C]//GLOBECOM 2020-2020 IEEE Global Communications Conference. 2020: 1-7.
- [41] LUO P, YU F R, CHEN J, et al. A novel adaptive gradient compression scheme: Reducing the communication overhead for distributed deep learning in the Internet of Things[J]. IEEE Internet of Things Journal, 2021, 8(14): 11476-11486.
- [42] KARIMIREDDY S P, REBJOCK Q, STICH S, et al. Error feedback fixes signsgd and other gradient compression schemes[C]//International Conference on Machine Learning. 2019: 3252-3261.
- [43] WEN W, XU C, YAN F, et al. Terngrad: Ternary gradients to reduce communication in distributed deep learning[J]. Advances in neural information processing systems, 2017, 30.
- [44] BERNSTEIN J, WANG Y X, AZIZZADENESHELI K, et al. signSGD: Compressed optimisation for non-convex problems[C]//International Conference on Machine Learning. 2018: 560-569.

- [45] CHEN T, GIANNAKIS G, SUN T, et al. LAG: Lazily aggregated gradient for communication-efficient distributed learning[J]. Advances in neural information processing systems, 2018, 31.
- [46] MCDONALD R, HALL K, MANN G. Distributed training strategies for the structured perceptron [C]//Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics. 2010: 456-464.
- [47] POVEY D, ZHANG X, KHUDANPUR S. Parallel training of deep neural networks with natural gradient and parameter averaging[J]. arXiv preprint arXiv:1410.7455, 2014: 124.
- [48] ZHANG S, CHOROMANSKA A E, LECUN Y. Deep learning with elastic averaging SGD[J]. Advances in neural information processing systems, 2015, 28.
- [49] MCMAHAN B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. 2017: 1273-1282.
- [50] SONG L, ZHAO K, PAN P, et al. Communication efficient SGD via gradient sampling with Bayes prior[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 12065-12074.
- [51] GHOSH A, MAITY R K, KADHE S, et al. Communication-efficient and byzantine-robust distributed learning with error feedback[J]. IEEE Journal on Selected Areas in Information Theory, 2021, 2(3): 942-953.
- [52] ZHANG X, LIU J, ZHU Z, et al. Communication-efficient network-distributed optimization with differential-coded compressors[C]//IEEE INFOCOM 2020-IEEE Conference on Computer Communications. 2020: 317-326.
- [53] KHIRIRAT S, MAGNUSSON S, JOHANSSON M. Compressed gradient methods with hessian-aided error compensation[J]. IEEE Transactions on Signal Processing, 2020, 69: 998-1011.
- [54] 陈开周. 最优化计算方法[M]. 西北电讯工程学院出版社, 1985.
- [55] 朱泓睿, 元国军, 姚成吉, 等. 分布式深度学习训练网络综述[J]. 计算机研究与发展, 2021, 58(98-115).
- [56] ABADI M, BARHAM P, CHEN J, et al. {TensorFlow}: a system for {Large-Scale} machine learning[C]//12th USENIX symposium on operating systems design and implementation (OSDI 16). 2016: 265-283.
- [57] KETKAR N, MOOLAYIL J, KETKAR N, et al. Introduction to pytorch[J]. Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch, 2021: 27-91.
- [58] JIA Y, SHELHAMER E, DONAHUE J, et al. Caffe: Convolutional architecture for fast feature embedding[C]//Proceedings of the 22nd ACM international conference on Multimedia. 2014: 675-678.
- [59] GULLI A, PAL S. Deep learning with Keras[M]. Packt Publishing Ltd, 2017.

- [60] SEIDE F, AGARWAL A. CNTK: Microsoft's open-source deep-learning toolkit[C]//Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016: 2135-2135.
- [61] LI S, ZHAO Y, VARMA R, et al. Pytorch distributed: Experiences on accelerating data parallel training[J]. arXiv preprint arXiv:2006.15704, 2020.
- [62] SHOEYBI M, PATWARY M, PURI R, et al. Megatron-lm: Training multi-billion parameter language models using model parallelism[J]. arXiv preprint arXiv:1909.08053, 2019.
- [63] HUANG Y, CHENG Y, BAPNA A, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism[J]. Advances in neural information processing systems, 2019, 32.
- [64] GHOLAMI A, YAO Z, KIM S, et al. Ai and memory wall[J]. RiseLab Medium Post, 2021, 1: 6.
- [65] HARLAP A, NARAYANAN D, PHANISHAYEE A, et al. Pipedream: Fast and efficient pipeline parallel dnn training[J]. arXiv preprint arXiv:1806.03377, 2018.
- [66] FAN S, RONG Y, MENG C, et al. DAPPLE: A pipelined data parallel approach for training large models[C]//Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2021: 431-445.
- [67] TANG Z, SHI S, CHU X, et al. Communication-efficient distributed deep learning: A comprehensive survey[J]. arXiv preprint arXiv:2003.06307, 2020.
- [68] LI M, ANDERSEN D G, PARK J W, et al. Scaling distributed machine learning with the parameter server[C]//11th USENIX Symposium on operating systems design and implementation (OSDI 14). 2014: 583-598.
- [69] CHEN Y, PENG Y, BAO Y, et al. Elastic parameter server load distribution in deep learning clusters [C]//Proceedings of the 11th ACM Symposium on Cloud Computing. 2020: 507-521.
- [70] PATARASUK P, YUAN X. Bandwidth optimal all-reduce algorithms for clusters of workstations [J]. Journal of Parallel and Distributed Computing, 2009, 69(2): 117-124.
- [71] CHEN T, LI M, LI Y, et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems[J]. arXiv preprint arXiv:1512.01274, 2015.
- [72] PASZKE A, GROSS S, MASSA F, et al. Pytorch: An imperative style, high-performance deep learning library[J]. Advances in neural information processing systems, 2019, 32.
- [73] ANTHONY Q, AWAN A A, JAIN A, et al. Efficient training of semantic image segmentation on summit using horovod and mvapich2-gdr[C]//2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). 2020: 1015-1023.
- [74] JAIN A, AWAN A A, SUBRAMONI H, et al. Scaling tensorflow, pytorch, and mxnet using mvapich2 for high-performance deep learning on frontera[C]//2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS). 2019: 76-83.

- [75] AWAN A A, MANIAN K V, CHU C H, et al. Optimized large-message broadcast for deep learning workloads: MPI, MPI+ NCCL, or NCCL2?.[J]. Parallel Computing, 2019, 85: 141-152.
- [76] AWAN A A, BÉDORF J, CHU C H, et al. Scalable distributed dnn training using tensorflow and cuda-aware mpi: Characterization, designs, and performance evaluation[C]//2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). 2019: 498-507.
- [77] GOETZ A F, VANE G, SOLOMON J E, et al. Imaging spectrometry for earth remote sensing[J]. science, 1985, 228(4704): 1147-1153.
- [78] TEKE M, DEVECI H S, HALILOĞLU O, et al. A short survey of hyperspectral remote sensing applications in agriculture[C]//2013 6th International Conference on Recent Advances in Space Technologies (RAST). 2013: 171-176.
- [79] NASRABADI N M. Hyperspectral target detection: An overview of current and future challenges [J]. IEEE Signal Processing Magazine, 2013, 31(1): 34-44.
- [80] ZHU D, DU B, ZHANG L. Binary-class collaborative representation for target detection in hyperspectral images[J]. IEEE Geoscience and Remote Sensing Letters, 2019, 16(7): 1100-1104.
- [81] MAKHZANI A, SHLENS J, JAITLEY N, et al. Adversarial autoencoders[J]. arXiv preprint arXiv:1511.05644, 2015.
- [82] RANGNEKAR A, MOKASHI N, IENTILUCCI E J, et al. Aerorit: A new scene for hyperspectral image analysis[J]. IEEE Transactions on Geoscience and Remote Sensing, 2020, 58(11): 8116-8124.
- [83] YI C, ZHANG L, ZHANG X, et al. Aerial hyperspectral remote sensing classification dataset of Xiongan New Area (Matiwan Village)[J]. National Remote Sensing Bulletin, 2020, 24(11): 1299-1306.
- [84] SWETS J A, DAWES R M, MONAHAN J. Better decisions through science[J]. Scientific American, 2000, 283(4): 82-87.
- [85] BRADLEY A P. The use of the area under the ROC curve in the evaluation of machine learning algorithms[J]. Pattern Recognition, 1997, 30(7): 1145-1159.
- [86] FLETCHER P T, VENKATASUBRAMANIAN S, JOSHI S. Robust statistics on Riemannian manifolds via the geometric median[C]//2008 IEEE Conference on Computer Vision and Pattern Recognition. 2008: 1-8.
- [87] CORMEN T H, LEISERSON C E, RIVEST R L, et al. Introduction to algorithms[M]. MIT press, 2022.
- [88] CHEN C Y, CHOI J, BRAND D, et al. Adacomp: Adaptive residual gradient compression for data-parallel distributed training[C]//Proceedings of the AAAI conference on artificial intelligence: vol. 32: 1. 2018.
- [89] QIAN N. On the momentum term in gradient descent learning algorithms[J]. Neural networks, 1999,

## 参考文献

---

- 12(1): 145-151.
- [90] GOYAL P, DOLLÁR P, GIRSHICK R, et al. Accurate, large minibatch sgd: Training imagenet in 1 hour[J]. arXiv preprint arXiv:1706.02677, 2017.



## 致 谢

阳光洒在西电校园的每一寸土地上，微风拂过树梢，似乎在为我的硕士生涯奏响一曲悠扬的赞歌。在这个充满回忆和成长的时光里，我终于站在了终点，怀着无尽的感恩和深深的敬意，写下这篇致谢。

感谢我的导师，谢卫莹教授。您的博学和严谨，点亮了我前行的道路；每一次的指导和批评，都如清风拂面，让我在学术的海洋中不断追求卓越。您不仅教会我如何进行科研，更教会了我如何保持对真理的执着和热爱。您的教诲，如烛火般长明，将照亮我未来的岁月。感谢雷杰教授，是您的推荐让我有机会加入图像传输与处理研究所（图像所）这个大家庭，让我能够将所学知识应用于项目实践中。感谢李云松教授，您如同图像所的一座灯塔，为我们科研的道路指明方向。

感谢我的师兄师姐：范舒然、郭文劲、谷怡洁、贾超群、蒋恺、李艳林、刘攀、刘易丹、秦皓楠、王嘉轩、杨梗、张佳青、张鑫，同窗好友：陈晓钰、樊潇怡、郭怡、梁皓天、马纪涛、马松林、王世轩、徐杰涛、吴林安、徐思敏、徐小鸿、叶航宇、张昊辉、张俊荣、张钰以及我的师弟师妹：李岱勋、李嘉琪、黎雄杰、刘卓然、田佳渊等人。我们共同走过的时光，是我人生中最美好的记忆。每一次的讨论，每一场的辩论，都使我获益匪浅；每一次的欢笑，每一段的共鸣，都将成我生命中不可或缺的一部分。因为有你们，我的成长旅程不再孤单。

感谢为评阅本文付出宝贵时间的各位老师和专家们。你们严谨的学术态度和独到的见解，为我的论文增色不少。你们的建议和批评，使我得以完善论文的内容和结构，提升学术水平。

感谢我的父母。从我呱呱坠地的那一刻起，你们便无私地给予我无限的爱和支持。你们是我坚实的后盾，是我在困难时刻坚持下去的力量源泉。你们的默默付出和不懈支持，使我能够心无旁骛地追求梦想。这份恩情，唯有在心底默默铭记，终生难忘。

感谢西电和图像所。这里的每一处景色，每一个角落，都记录了我求学的点滴。我会永远铭记在这里度过的美好时光，这里的每一个瞬间，都是我人生的重要篇章。

在这段硕士研究生的旅程中，我收获了知识，收获了友谊，也收获了人生的智慧。在未来的日子里，我将继续怀着一颗感恩的心，继续追求自己的梦想。愿这份感恩之情，能够化作我前行的动力，让我在未来的道路上，不忘初心，砥砺前行。



## 作者简介

### 1. 基本情况

卢天恩，男，河南商丘人，1995年10月出生，西安电子科技大学通信工程学院信息与通信工程专业2021级硕士研究生

### 2. 教育背景

2017.09 ~ 2021.07, 长春理工大学, 本科, 专业: 通信工程

2021.09 ~ , 西安电子科技大学, 硕士研究生, 专业: 信息与通信工程

### 3. 攻读硕士学位期间的研究成果

#### 3.1 申请（授权）专利

[1] 谢卫莹, 卢天恩, 张鑫等. 基于滤波器特征图的全局秩感知神经网络模型压缩方法: 中国, 申请号 [202111373645.6]. 2021.11.19.

[2] 谢卫莹, 马纪涛, 蒋恺等. 基于空谱深度协同的高光谱异常检测方法: 中国, 专利号 [ZL 2021 1 1539339.5]. 2023.06.20.

#### 3.2 参与科研项目及获奖

[1] XXX 探测器有效载荷中视场彩色相机地检系统, 2021.10-2022.05, 已交付, 软件系统负责人

[2] XXX 探测器有效载荷地形地貌相机地检系统, 2022.09-2023.03, 已交付, 软件系统负责人

[3] 2021 - 2022 学年硕士研究生一等学业奖学金

[4] 2023 - 2024 学年硕士研究生三等学业奖学金