并行机器学习算法的性能与能耗

Xidong Wu、Preston Brazzle 和 Stephen Cahoon

arXiv:2305.00798v1 [cs.DC] 1 May 2023

I. 摘要

机器学习模型已在数据科学、计算机视觉和自然语言处理等各种现实应用中取得了显著成果。然而，机器学习中的模型训练需要大规模数据集，并且需要经过多次迭代才能正常运行。训练算法的并行化是加快训练过程的常见策略。然而，许多关于模型训练和推理的研究只关注性能方面。功耗也是任何类型计算的重要指标，尤其是对于高性能应用。研究人员已研究可用于传感器和移动设备等低功耗平台的机器学习算法，但针对为高性能计算设计的算法，功耗优化仍较少。

本文介绍了逻辑回归和遗传算法的 C++ 实现，以及用于分类任务、采用随机梯度下降（随机梯度下降）算法的神经网络 Python 实现。我们将展示模型复杂度和训练数据规模对算法并行效率的影响，同时考察功耗和性能。我们还测试了这些实现采用共享内存并行、分布式内存并行和 GPU加速来加快机器学习模型训练的效果。

II. 引言

机器学习是一类数据驱动算法和模型，模型会随着经验的积累而逐步改进。它有许多应用，从图像分类到机器人控制 [1]。通过提供一组训练数据，模型可以自行训练，从而准确处理训练集之外的新数据。然而，随着训练数据和模型规模不断扩大，为了处理更大规模的问题，有必要开发这些算法的并行版本。监督学习算法是一类高度适合并行化的机器学习算法。

监督式机器学习算法会针对预先设定的一组训练数据，为模型寻找最优参数。这个优化问题可以通过许多不同的方法解决，但有些算法比其他算法更受欢迎。随机梯度下降（SGD）是训练机器学习模型时最常用的优化器之一 [2]。作为梯度下降的一种变体，随机梯度下降允许模型在每次迭代时使用训练数据中的随机抽样子集进行训练，而不是每次都处理整个数据集；这种方法速度相对较快、计算成本较低，也适合并行化。此外，我们还将考虑使用遗传算法训练模型，因为它具有很强的并行化能力。遗传算法会跟踪由多个模型组成的种群，根据每个模型的效果进行评分，并由排名最高的个体产生新一代。由于遗传算法会同时为多个模型评分，因此很容易将其拆分为多个批次进行多线程处理。

即使采用了这些技术，训练具有大量参数的机器学习模型仍可能需要巨大的计算能力和训练时间。并行机器学习算法可以通过加速计算 [3]来克服这些限制。共享内存并行和分布式并行都可以利用并行计算来加速机器学习训练过程 [4]。此外，分布式并行可以将训练工作负载分配到多台机器上，每台机器都拥有自己的内存和处理能力，从而能够在海量数据集上训练大规模机器学习模型。

然而，随着数据集和模型规模的增大，能耗也随之增加。机器学习算法的能耗分析是一个新兴的研究领域，但早在 1996 [5]时就已有人对此进行研究。当前，机器学习领域的大部分研究都聚焦于所获得的性能提升，却很少关注不同技术和方法所消耗的能耗。通过比较不同实现的加速比和能耗，这类研究对设计人员和用户都很有帮助。例如，与串行对应算法相比，并行训练算法可以实现显著的加速，但随着工作被分配给多个处理器，并行性的开销成本也会逐渐增加。通过确定功耗与性能效率开始出现分歧的临界点，研究人员和用户能够更好地理解性能提升所伴随的权衡。

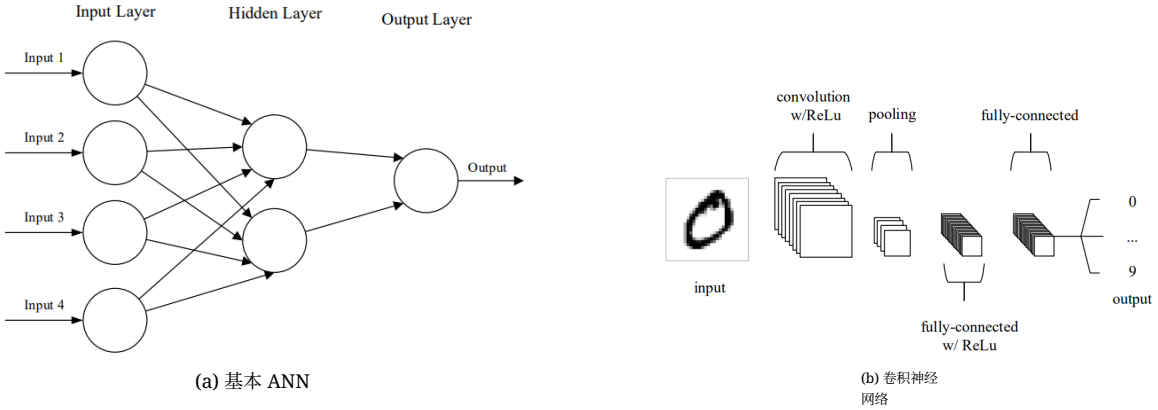
本文将采用多种方法对不同的机器学习算法进行并行化和训练。我们将把每种方法的性能和功耗与串行基线进行比较，并据此得出相关结论。在解释结果之前，必须先讨论本文主题的背景以及所采用的方法。

III. 背景

A. 机器学习模型

1）人工神经网络：在人工神经网络（ANN）中，数据会输入一系列节点（或神经元），这些节点计算数据的线性组合，并将结果传递到下一层。该组合的系数称为权重。由于同一层中的节点彼此独立，因此可以轻松地并行执行其计算。最后一层表示模型的结果，通常是一系列概率。例如，在图像分类模型中，输出层中的每个节点都可以表示图像属于某个类别的概率。

训练过程包括为模型找到理想的权重组合，使其能够最有效地解读数据。事实证明，经过适当训练的人工神经网络具备通用函数逼近能力 [6]。提供 ANN 模型和一组训练数据（包含输入及预期结果）后，机器学习算法便能优化模型，使其拟合训练数据。然而，当输入超出训练集范围时，人工神经网络通常难以进行外推。在训练数据上表现良好、但在其他数据上表现不佳的人工神经网络被视为过拟合。



(a) 基本 ANN (b) 卷积神经网络

图 1. [7]中的人工神经网络（a）和卷积神经网络（b）的可视化示意图。ReLu 是线性整流单元的简称。

2）卷积神经网络：人工神经网络虽然非常有用，但对于图像等较大的输入，其计算复杂度会呈指数级增长。随着人工神经网络模型规模的扩大，模型也更容易对训练数据过拟合。卷积神经网络（CNN）被提出，用于解决这些问题 [8]。卷积神经网络旨在有效处理图像，因为卷积核能够概括相邻信息，而不同层则可视为多尺度特征提取器。

卷积神经网络包含三种类型的层：卷积层、池化层和全连接层。卷积层对输入图像执行多次卷积运算，以识别特定特征。经过训练后，处理后的图像称为特征图。池化层只是对前一层的结果进行下采样，以降低后续层的计算负荷。例如，最大池化层可以将每个 2x2 像素网格替换为一个像素，其值为该网格中的最大值。这会显著缩小数据规模，从而降低后续层的计算需求。全连接层的工作方式与人工神经网络类似，只需计算所有输入的线性组合并应用激活函数。常见的激活函数是 ReLu，它会将所有负值替换为 0。全连接层生成模型的整体输出 [7]。 [9] 关于深度残差网络（ResNet）的论文是所有研究领域中被引用次数最多的论文之一（引用次数为 162,170）

3）逻辑回归：逻辑回归是一种用于二元分类任务的经典机器学习模型。逻辑回归是一种简单而强大的算法，广泛应用于信用风险建模、营销分析和医学研究等各种领域。在新算法的研究与开发中，它也常被用作基线模型，以便与更复杂的模型进行比较。它使用逻辑函数对二元响应变量的概率进行建模。逻辑回归模型通过寻找能够使观测数据的似然最大化的参数来进行训练。该函数的输出表示在给定预测变量的条件下，二元响应变量发生的概率。逻辑函数如下所示：

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{1}$$

由此得到的概率表示输入数据与最终标签之间是否存在某种关系。例如，如果使用逻辑回归模型对具有特定婚姻状况的人进行训练，将其标记为“快乐”或“不快乐”，那么模型就会根据其婚姻状况 [10]，预测该人“快乐”的概率。因此，模型会将数据分类到似然最高的类别中。

逻辑回归可以扩展为预测具有多个类别的结果。这通常称为多项逻辑回归 [11]。在多项逻辑回归中，会为每个可能的结果类别计算似然，模型则将输入数据分配给结果最高的类别。

算法 1 随机梯度下降（SGD）算法

输入： 训练周期数 T、学习率 η；以及小批量大小 b；
初始化： 模型参数 x_0；
对于 $t = 1, 2, \ldots, T$ **do**
　抽取小批量样本 $\mathcal{B}_t = \{\xi^j\}_{j=1}^{b}$ 使用 $|\mathcal{B}_t| = b$，从训练数据集中抽取 $\mathcal{D}$
　$x_t = x_{t-1} - \eta \nabla_x f(x_{t-1}; \mathcal{B}_t^i)$
结束循环
输出： x_T。

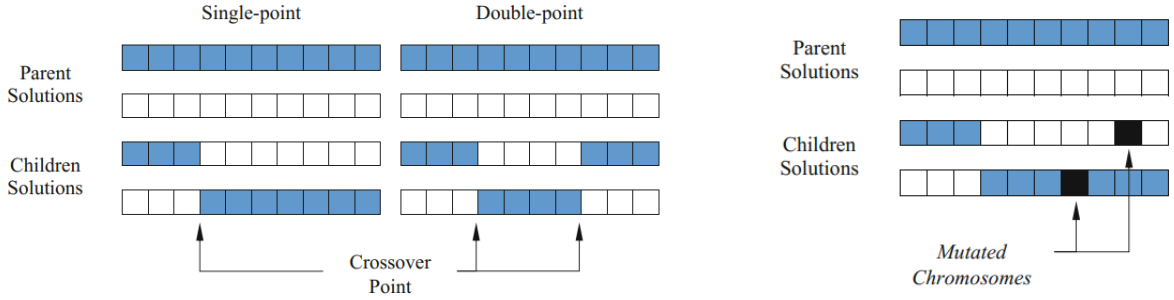
B. 模型训练方法

1)（随机）梯度下降：梯度下降可用于寻找预定义损失函数的最优参数。在监督式机器学习中，损失函数通常是模型输出与训练数据集中预期输出之间的差值。通过求出损失函数的梯度，我们可以沿着该方向更新参数，沿最陡下降路径前进。反复计算梯度并沿该方向移动一小段距离的过程称为梯度下降。

梯度下降的主要缺点之一是需要使用整个训练数据集来求取梯度，这在计算上成本高昂且速度缓慢。随机梯度下降并不使用完整数据集，而是随机选择训练数据的一个子集，并在每一步中使用该子集。尽管这会降低总体准确率，并在梯度方向中引入噪声，但随机梯度下降能够以高得多的速度推进 [2]。算法 1 总结了 SGD 算法，其中 $f(\cdot)$ 是损失函数，$\mathcal{D}$ 是训练数据集。

随着模型和数据集规模的增大，深度学习中的分布式训练日益普及 [12]。近年来，同步训练和异步训练已成功应用于许多优化算法，例如随机梯度下降（SGD） [13], [14] 和 SVRG [15], [16]。 [13] 对 SGD 进行并行化，并证明并行计算能够有效加速机器学习训练。 [14] 研究了 SGD 在非凸优化（深度学习模型）中的异步并行实现，并提供了理论支持。 [15] 利用方差缩减技术加速了分布式训练的收敛。分布式优化已在多个深度学习领域取得成功，例如计算机视觉 [17], 、生成式建模 [18] 和强化学习 [19]。一些通信高效的分布式方法也得到了研究 [20], [21]。然而，其中大多数方法都侧重于同步训练，并未比较不同的并行方式（同步与异步、共享内存与分布式）对加速模型训练的效果。

随机梯度下降算法可以采用同步或异步方式进行并行化。在同步并行随机梯度下降中，我们在每次迭代中使用并行计算。在共享内存并行计算中，我们使用多进程方法来加速矩阵乘法。我们在每次迭代中使用“#pragma omp parallel for”并行更新模型。在分布式并行计算中，每个处理节点在本地抽取数据点并计算自己的随机梯度。随后，服务器汇总来自各客户端的梯度，并使用总体梯度更新模型。最后，服务器将更新后的模型发送给每个处理节点。另一方面，异步训练不需要同步步骤，因此可以避免模型更新瓶颈。在共享内存并行计算中，我们让每次迭代并行执行。在分布式并行计算中，服务器收到随机梯度后会立即更新模型，并仅将更新后的模型发回计算该梯度的处理节点。



(a) 选定的解如何组合以产生后代　　(b) 子代解中的变异。

图2。来自 [22] 的图展示了(a)选定的解如何组合以产生后代；(b)子代解中的变异。“染色体”指单个模型参数。

2）遗传机器学习：遗传算法受到达尔文进化论的启发。给定一组具有随机参数的模型（称为种群），对每个模型进行测试，并赋予其一个适应度分数，该分数决定了它将参数传递给下一代的似然。被选中的模型会将自身参数与其他模型的参数组合，生成“子代”。此外，子代的参数还可能发生“变异”，表现为小幅随机变化 [22]。

总体而言，该算法由若干中间步骤组成：适应度评分、父代选择、父代组合和变异。针对每个子步骤，都有许多已提出的策略。例如，一种流行的父代选择算法会将个体模型的适应度分数与分数总和进行比较，以计算该模型被选中的概率。除这些步骤外，人们还提出了其他算法改进方法。例如，每一代都可以保留种群中得分最高的成员，使其延续到下一代。这个过程称为精英保留策略。

适应度函数通常被认为是遗传算法中最重要的部分。没有设计良好的适应度函数，算法就无法收敛到可用的解。使用带有预期结果的训练数据时，一种常见的适应度函数就是模型结果与期望结果之间的距离。另一种可能的适应度函数是准确率百分比。例如，在图像分类任务中为模型评分时，其适应度分数可以是正确分类的图像所占的百分比。虽然这种适应度函数会奖励正确分类，但它并不考虑模型结果的确定性。如果适应度函数取决于实际结果与期望结果之间的总距离，那么它就会奖励最接近某个猜测的模型。

选定适应度函数后，算法便会使用这些分数为下一代选择父代。通常，一个模型被选中的概率与其适应度分数或在种群中的总体排名成正比。一种简单的选择算法可以每次都只选择得分最高的模型，但这样会大幅降低种群的多样性。多样性使遗传算法区别于随机搜索或梯度下降，因为它可以同时探索多个局部极小值。因此，即使是表现最差的模型，也必须有机会被选中。

给定两个或更多选定的父代模型，遗传算法的组合步骤将为下一代创建一个或多个子代模型。组合算法可以预先设定交叉点的数量，用于细分父代模型参数。每个细分部分仅由单个父代的参数组成，并交替由不同父代提供参数。交叉点的确切位置通常是随机的。图 2 展示了单点和双点交叉组合算法的可视化示意。组合算法也可以针对每个参数随机选择其中一个父代。

变异使遗传算法能够在原始种群之外搜索理想解。然而，变异率会对算法的效能产生巨大影响。如果变异率非常高，该算法就类似于随机搜索。

IV. 方法

所有实验均在匹兹堡大学研究计算中心（CRC）进行，使用了共享内存处理（SMP）、消息传递接口（MPI）和GPU集群。遗憾的是，由于用户权限限制，无法收集具体的功耗结果。不过，CRC 提供了每个集群中所有硬件的规格，因此可以获得热设计功耗（TDP）数值。热设计功耗是处理器在理论最大负载下消耗的功率 [23]。这一假设虽不理想，但可以让我们大致了解功耗如何随处理器数量增加而变化。

A. 使用随机梯度下降的逻辑回归

为比较不同的训练方法和配置，我们针对逻辑回归进行了多项测试。两种配置 MP 和 MPI 分别表示在 SMP 集群上使用 OpenMP，以及在 MPI 集群上使用 MPI。两个集群均采用异步（A）和同步训练。每种情况分别使用 1、2、4、8 和 16 个线程运行，并为每个集群设置串行基线。MPI 集群使用英特尔至强金牌 6342 处理器。

使用 C++ 实现了采用随机梯度下降训练的逻辑回归模型，并分别使用 OpenMP 和 OpenMPI 处理共享内存架构和分布式内存架构。我们采用四种不同的并行方法（同步共享内存、异步共享内存、同步分布式、异步分布式）来加速模型训练。训练数据采用来自 LIBSVM 的 gisette 数据集。训练数据按特征缩放至 [-1,1]。数据量为 7,000，特征数为 5,000。批量大小为 256。

共享内存并行方法使用 OpenMP 库。在同步共享内存方法（MP）中，我们使用 ”#pragma omp parallel for” 和 ”#pragma omp parallel for reduction(+: cp)” 在每次迭代中加速向量乘法。我们不使用批大小并行计算，以避免模型更新中的竞态条件。在异步共享内存并行方法（MPA）中，我们使用 ” #pragma omp parallel for schedule(static, 1) ” 对每次更新迭代进行并行化。在模型更新步骤中，我们使用 ”#pragma omp critical” 来防止竞态条件，并强制多个线程逐个更新模型。

在分布式并行方法中，我们使用 OpenMPI 库。在模型训练之前，我们首先需要将训练数据集（虚拟地）分发到各个节点。在同步分布式方法（MPI）中，每个客户端在当前迭代中采样部分训练数据，并计算梯度。随后，服务器汇总来自各个客户端的这些梯度，并使用总体梯度更新模型。最后，服务器将更新后的模型发送给各个处理节点。在异步分布式方法（MPIA）中，我们不要求进行同步通信。当服务器收到随机梯度时，会立即更新模型，并且只将当前模型发送回计算出该梯度的处理节点。

B. 使用随机梯度下降的卷积神经网络

CNN 模型使用 PyTorch 实现，并采用随机梯度下降作为优化器；此外，我们使用 CUDA 库添加了 GPU加速，以加快训练速度。CNN 模型通常包含大量参数，需要大量计算能力和内存才能进行训练。随着模型和数据集规模不断增大，训练过程的计算开销可能变得难以承受，导致仅使用中央处理器训练这些模型十分困难，甚至不可能。由于图形处理器具备并行处理能力，且执行矩阵运算的速度很快，因此是理想的选择。本文比较了在 CIFAR-10 上使用中央处理器或图形处理器训练 CNN 模型 10 个训练轮次的性能。CIFAR-10 数据集包含 50,000 张训练图像和 10,000 张测试图像。60,000 张（32×32）彩色图像被分为 10 个类别（飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船、卡车）。学习率设为 3e-2。批量大小设为 32。模型架构如下所示。我们选择交叉熵作为损失。所用中央处理器是 CRC SMP 集群中的 Intel Xeon Gold 6126 CPU，图形处理器则是 CRC GPU 集群中的 NVIDIA A100。

表 I CIFAR-10 [24] 的模型架构

层类型	形状	填充
卷积 + ReLU	$3 \times 3 \times 16$	1
最大池化	2×2	
卷积 + ReLU	$3 \times 3 \times 32$	1
最大池化	2×2	
卷积 + ReLU	$3 \times 3 \times 64$	1
最大池化	2×2	
全连接层 + ReLU	512	
全连接层 + ReLU	64	
全连接层 + ReLU	10	

C. 遗传算法库

所有遗传算法实验均使用 pca 遗传算法库完成。该库使用 C++ 编写，旨在服务于本项目；我们使用 Catch2 进行单元测试，并使用 libjpeg-turbo 读取训练数据集中的 JPEG 图像。该库已使用 OpenMP 进行并行化。

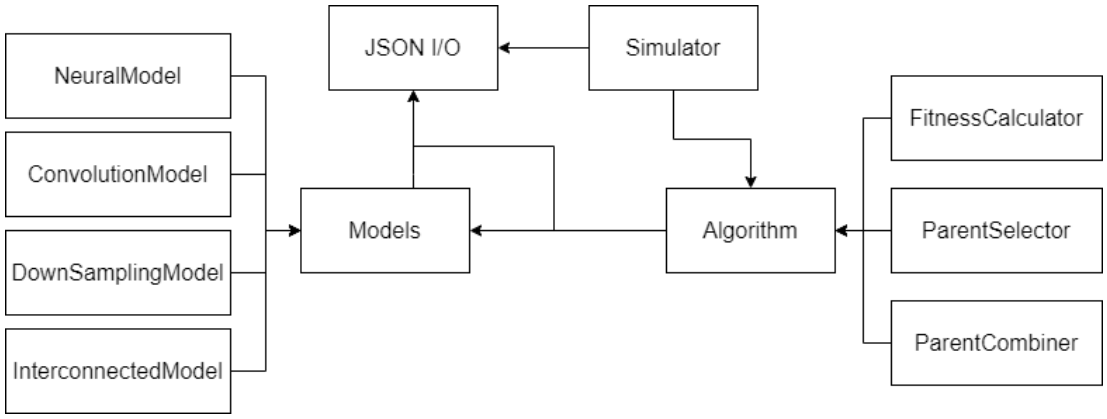


图 3. pca 遗传算法的高级架构。图中包含实验中使用的具体模型类

模型包包含实验中使用的所有机器学习模型，以及用于解析不同类型输入数据的代码。构建人工神经网络时使用了 NeuralModel 类。构建卷积神经网络时，使用 InterconnectedModel 类将 NeuralModel、Convolution Model 和 DownSamplingModel 组合起来，构成完整网络。InterconnectedModel 可以处理任意数量按层连接在一起的模型，因此可以轻松重新配置，以包含多个卷积和池化步骤。所有模型都可以将其参数表示为数字向量，并进行自我变异，因此遗传算法代码可以适用于任何模型。

遗传算法代码包含该算法子步骤的接口。适应度计算、父代选择和父代组合都被分配给独立的类。这使得算法代码可以轻松配置，并且能够在运行时为每个子步骤替换不同的算法。

遗传算法还将处理一些用于规定种群的参数。变异率决定新子模型发生变异的概率，而变异幅度决定变异可以达到的程度（以百分比表示）。种群规模决定种群中存在多少个模型，精英保留策略则根据适应度排名决定种群中有多大比例能够延续到后代。最后，偏移量大小参数用于创建

第一代种群，这一代中的每个模型都会先从模型模板复制而来，然后以 100% 的变异率和偏移量大小的变异幅度进行变异。

Simulation 类包含运行实验所需的所有相关信息：带有参数和子步骤的算法类、模板模型类、训练数据，以及要运行的代数。该类会按指定的代数运行遗传算法，并将每一代的平均适应度和最佳适应度保存到 .csv 文件中。

此外，还为该库编写了 JSON 解释器，以便将实验保存为文本。每个模型、算法子步骤类和数据类都能将自身表示为 JSON 字符串。这样，Simulation 类便可在执行前从 JSON 文件中读取所有相关信息。此外，还编写了一个 Python 脚本，用于生成这些 JSON 文件并自动执行实验，将结果保存到 .csv 文件中。这样，测试算法参数多个取值的实验便可自动运行。最后，还编写了另一个 Python 脚本，用于将生成的 .csv 文件转换为图表并汇编成 PDF。

未来可以为 pca genetic 添加若干额外功能。目前，pca genetic 仅采用共享内存并行，但很容易将其扩展到分布式网络。由于它能够解析 JSON，处理节点可以将大部分消息作为 JSON 文本发送。例如，可以向每个处理节点发送一份训练数据副本和一部分种群，然后由其向中央服务器节点报告各个模型的适应度。服务器节点随后可以根据适应度数据更新种群。这个思路还可以扩展为类似随机梯度下降训练的同步算法和异步算法。此外，一些关于遗传算法的研究建议增加额外的子步骤 [25]。

我们使用这个库运行了几项不同的实验。首先，我们创建了三种不同的模型，用于处理 MINST 数据集中的手写数字图像：双层人工神经网络、三层人工神经网络和卷积神经网络。每张图像为 28x28 像素，因此每个模型都设计为接收相应数量的输入数值。每个模型会生成 10 个数值，表示图像对应于各个数字的置信度。数值最高的数字被视为模型的猜测结果。为了测定并行效率，我们使用每个模型进行了简短的仿真，并计算每一代所需时间的平均值。我们使用不同的处理器数量，并将平均时间与单线程基线进行比较。我们还通过改变输入图像的数量再次进行了这些效率仿真。

V. 结果

本节将详细介绍各项测试所得的结果。

表 II 使用中央处理器与图形处理器在 CIFAT-10 上训练卷积神经网络的准确率、时间和能耗

设备	准确率（%）	时间（秒）	能耗 (J)
CPU	47	27978.98796	3497373.495
GPU	48	327.3535366	130941.4146

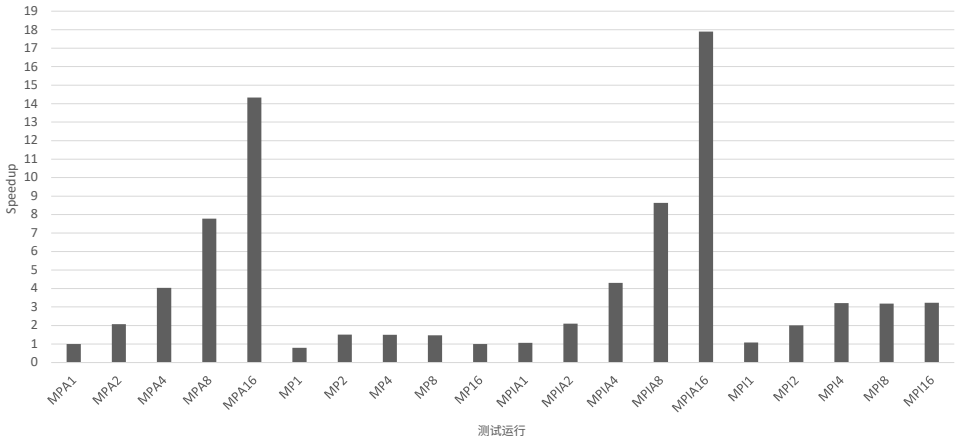


图 4. 使用 MPI 和 OpenMP，以及在 gisette 数据集上采用异步和同步训练所获得的加速比。

表 III 同步训练 250个训练轮次后的最终损失结果

Test	MP1	MP2	MP4	MP8	MP16	MPI1	MPI2	MPI4	MPI8	MPI16
Loss	0.0183	0.0182	0.0183	0.0183	0.0182	0.0182	0.0185	0.0182	0.0177	0.0188

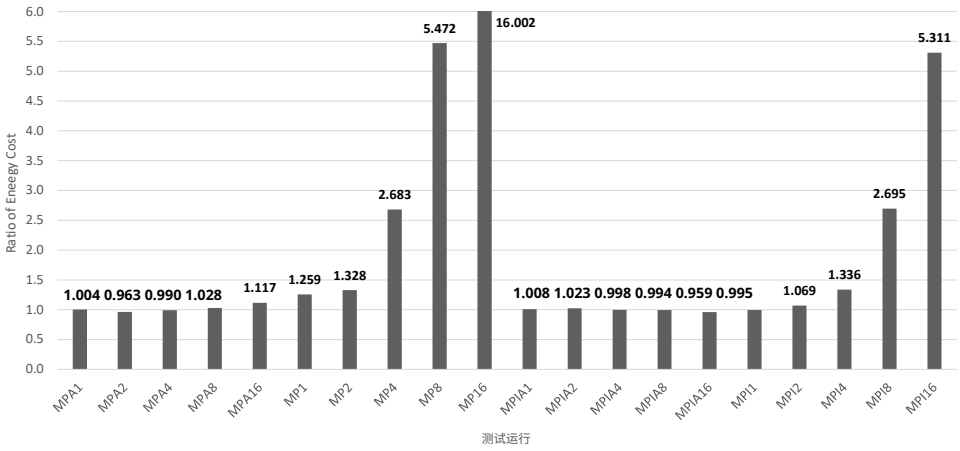


图 5. 在 Gisette 数据集上，采用异步和同步训练时，MPI 和 OpenMP 的能耗与串行基线之比

表 IV 异步训练经过 250个训练轮次后的最终损失结果

Test	MPA1	MPA2	MPA4	MPA8	MPA16	MPIA1	MPIA2	MPIA4	MPIA8	MPIA16
Loss	0.0182	0.0194	0.0212	0.0219	0.0242	0.0182	0.0194	0.0212	0.0219	0.0242

A. 使用随机梯度下降的逻辑回归

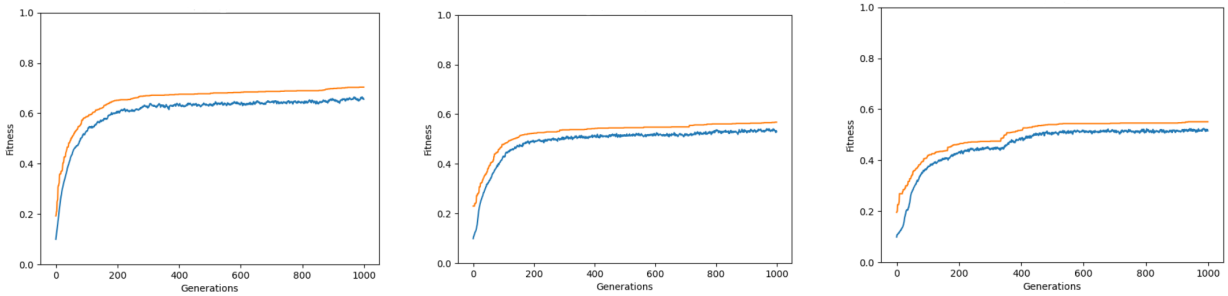
图 V 所示的加速比结果表明，在性能方面，无论是 MP 还是 MPI 情况，异步训练都优于同步训练。不过，需要注意的是，同步训练情况下的最终损失大致相同（0.018），而异步训练情况下的最终损失则会随着线程数增加而上升（16 个线程时为 0.024）。表 III 和表 IV 展示了这些结果。在某些情况下，例如 MPIA4-MPIA16，加速比超过了线程数。这种超线性加速很可能归因于分布式环境中数据的拆分和缓存。由于在分布式训练中，每个处理器处理的数据较少，数据更有可能存放在缓存中，因此访问速度更快。

观察能耗图时，可以发现结果相似，但并不完全相同。请注意，在图 V 中，尽管 MPIA4 的加速比更高，其功率效率却低于对应的 MPA4。事实上，MPIA 只有在使用 8 个和 16 个处理器时才胜过 MPA，这是因为它在这些情况下获得了更高的超线性加速。该图还显示了与图 V 相符的结果：由于加速比不尽如人意，同步情况下的能耗始终高于串行情况。

总体而言，由于当前模型在每次迭代中都会通过同步训练生成总体梯度，因此其准确率更高。然而，异步训练无需经过同步这一开销步骤，因此加速性能更好。因此，这些不同的方法体现了速度和准确率之间的权衡。与共享内存并行计算相比，分布式并行计算的加速性能更好。

B. 使用随机梯度下降的卷积神经网络

在功率方面，图形处理器的功耗超过 3 倍，但如表 II 所示，其能耗最终降低了 26 倍以上。这些能耗节省源于使用图形处理器硬件后时间的大幅缩短（85 倍）；这类硬件专为处理此类并行问题而设计。该表还显示准确率略有提高，这凸显了图形处理器在机器学习训练中的重要作用。



（a）2层 ANN（28*28，10） （b）3层 ANN（28*28，10，10） （c）CNN（1个特征图，3x3最大池化，2层 ANN）

图6. 3种不同模型在1000代中的适应度值。2层 ANN 的准确率达到约0.75，而其他模型的准确率约为0.6。

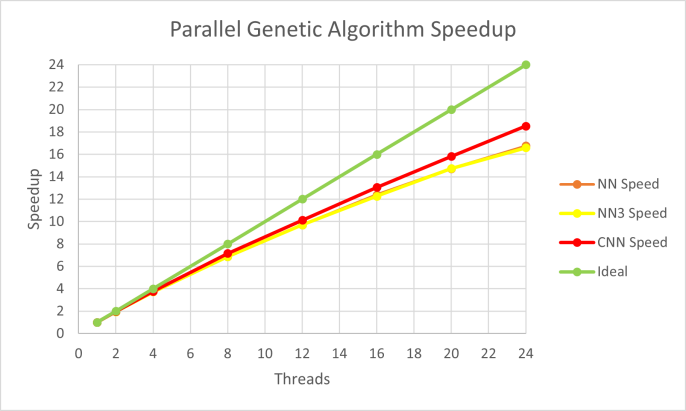


图 7。3 种不同模型的并行效率。每次仿真均运行 100 代，使用 10,000 张图像，记录的每代时间是所有代的平均值。

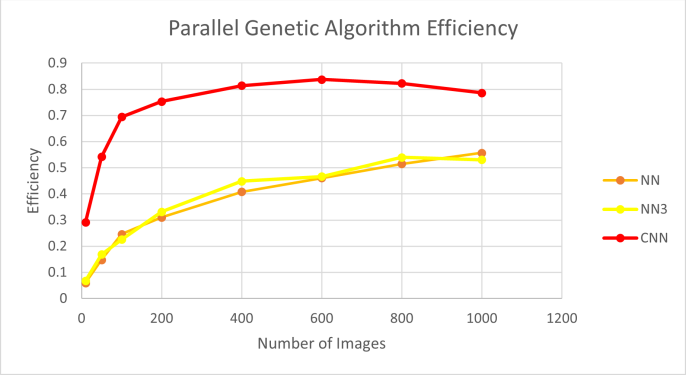


图 8。 3 种不同模型的并行效率。随着图像数量增加，并行效率也会提高 s.

C. 遗传算法

在运行长期仿真时，复杂度较低的模型往往能达到更高的准确率。这很可能是因为其总体参数较少，从而缩小了搜索空间。两层 ANN 的准确率约为 75%，而三层 ANN 和卷积神经网络的准确率仅约为 60%。虽然随着时间推移，准确率有可能进一步提高，但这需要非常多代。每个模型的性能都明显提升，直到约 200 代；此后提升速度放缓，但并未停滞。有趣的是，卷积神经网络的准确率在约第 350 代时突然大幅提高，但很快又趋于平稳。这很可能是由于种群多样性下降，导致后续的任何改进都取决于变异。

随着线程数增加，每个模型都表现出良好的加速比。正如预期，复杂度更高的卷积神经网络效率略高。额外的复杂度增大了每个线程的粒度，从而降低了每一代同步的开销。训练数据集的大小也对效率有很大影响。在使用.....进行训练时。

少于 200 张图像时，3 个模型的效率都非常低。然而，超过这一点后，效率会迅速提升。卷积神经网络从训练数据规模的增加中获益最多，最高效率约为 83%。

VI. 结论

本文介绍了在分类任务上，使用遗传算法和随机梯度下降（SGD）算法对机器学习模型（逻辑回归模型和卷积神经网络模型）进行并行实现。我们采用共享内存并行计算、分布式并行计算和图形处理器来加速模型训练。首先，我们考察了以随机梯度下降为优化器的逻辑回归模型。实验结果表明，同步训练的准确性更高，而异步训练的加速性能和能耗表现更好。此外，与共享内存并行计算相比，分布式并行计算的加速效果和能耗表现更好。随后，我们考察了以随机梯度下降为优化器的卷积神经网络模型。我们分别在中央处理器和图形处理器上训练模型，图形处理器在功率和能耗方面均优于中央处理器。结果表明，图形处理器最多可将训练时间缩短 85 倍，同时将能耗降低 26 倍。最后，我们探讨了遗传算法及其性能。遗传算法也能从并行化中获益良多。尽管本文讨论的方法效率很高，但仍可进一步向分布式并行发展。还可以开发遗传算法的异步版本，以更高效地利用并行资源。总的来说，在无法使用随机梯度下降或其效果不佳的情况下，该算法展现出良好的应用前景。

参考文献

[1] M. I. Jordan 和 T. M. Mitchell，《机器学习：趋势、观点与前景》，《科学》，第 349 卷，第 6245 期，第 255–260 页，2015年。

[2] 2017， 113 132 [ ] 10 1007 978 1 4842 2766 4 8 N. Ketkar，《随

[3] 机梯度下降》。加利福尼亚州伯克利：Apress，第 – 页。在线。获取地址：https://doi.org/。/---- S. R. Upadhyaya，《机器学习的并行方法——全面综述》，《并行与分布式计算杂志》，第 73 卷，第 3 期，第 284–292 页，2013年，《高性能分布式数据挖掘模型与算法》，[在线]。获取地址：

https://www.sciencedirect.com/science/article/pii/S[4] 07437315 12002705 J. Keuper 和 F.-J. Pfreundt，《异步并

行随机梯度下降：可扩展分布式机器学习算法的数值核心》，载于《高性能计算环境中的机器学习研讨会论文集》，丛书：MLHPC '15。美国纽约州纽约：计算机协会， 2015 [ ] 10 1145 2834892 2834893 。在线。获取地址：https://doi.org/。/。E. Garcıa-Martın、

[5] ´ ´ C. F. Rodrigues、G. Riley 和 H. Grahn，《机器学习能耗估算》，《并行与分布式计算杂志》，第 134 卷，第 75–88 页，2019年。[6] O. I. Abiodun、A. Jantan、A. E. Omolara、K. V. Dada、N. A. Mohamed 和 H. Arshad，《人工神经网络应用的最新进展：综述》，《Heliyon》，第

4， 11， 00938, 2018 [ ] 2405844018332067 卷，第期，e。在线。获取

[7] 2015 [ ] 10 48550 1511 地址：https://www.sciencedirect.com/science/article/pii/S K. O'Shea 和 R. Nash，《卷积神经网络简介》。在线。获取地址：https://doi.org/。/arXiv.。 08458

[8] 521， 7553， 436 444, 2015 Y. LeCun、Y.

Bengio 和 G. Hinton，《深度学习》，《自然》，第卷，第期，第–页。[9] K. He、X. Zhang、S. Ren 和 J. Sun，《用于图像识别的深度残差学习》，载于《电气与电子工程师协会计算机视觉与模式识别会议论文集》，2016年，第 770–778 页。[10] A. DeMaris，《逻辑回归教程》，《婚姻与家庭杂志》，第 57 卷，第 4 期，第 956–968 页，1995年。[11] C. Kwak 和 A. Clayton-Matthews，《多项逻辑回归》，《护理研究》，第 51 卷，第 6 期，第 404–410 页，2002年。[12] J. Dean、G. Corrado、R. Monga、K. Chen、M. Devin、M. Mao、M. Ranzato、A. Senior、P. Tucker、K. Yang 等，《大规模分布式深度网络》，《神经信息处理系统进展》，第 25 卷，2012年。[13] B. Recht、C. Re、S. Wright 和 F. Niu，《Hogwild!：一种并行化随机梯度下降的无锁方法》，《神经信息处理系统进展》，第 24 卷，2011年。[14] X. Lian、Y. Huang、Y. Li 和 J. Liu，《用于非凸优化的异步并行随机梯度》，《神经信息处理系统进展》，第 28 卷，2015年。[15] Z. Huo 和 H. Huang，《用于非凸优化的方差缩减异步小批量梯度下降》，载于《AAAI人工智能会议论文集》，第 31 卷，第 1 期，2017年。[16] R. Bao、X. Wu、W. Xian 和 H. Huang，《用于随机复合优化的双重稀疏异步学习》，载于第三十一届国际人工智能联合会议论文集，IJCAI，2022年，第 1916–1922 页。[17] P. Goyal、P. Doll´ar、R. Girshick、P. Noordhuis、L. Wesolowski、A. Kyrola、A. Tulloch、Y. Jia 和 K. He，《准确的大批量随机梯度下降：1 小时内训练 ImageNet》，arXiv 预印本 arXiv:1706.02677，2017年。[18] A. Brock、J. Donahue 和 K. Simonyan，《用于高保真自然图像合成的大规模生成对抗网络训练》，arXiv 预印本 arXiv:1809.11096，2018 年。[19] X. Zhang 、 Z. Liu 、 J. Liu 、 Z. Zhu 和 S. Lu，《协作式多智能体强化学习中去中心化策略评估的通信与样本复杂度控制》，《神经信息处理系统进展》，第 34 卷，第 18 825–18 838 页，2021年。[20] X. Wu、F. Huang、Z. Hu 和 H. Huang，《更快的自适应联邦学习》，arXiv 预印本 arXiv:2212.00974，2022年。[21] X. Wu、Z. Hu 和 H. Huang，《用于非凸极小极大问题的去中心化黎曼算法》，arXiv 预印本 arXiv:2302.03825，2023年。

[22] 2019 [ ] 10 1007 978 3 319 93025 1 4 S. Mirjalili，《遗传

[23] [ ] 000055611 算法》。Cham：Springer国际出版，。在线。获取地址：https://doi.org/。/----《英特尔处理器中的热设计功耗（TDP）》。在线。获取地址：

https://www.intel.com/content/www/us/en/support/articles/ /processors.html [24] F. Huang、X. Wu 和 H. Huang，《用于非光滑极小极大问题的高效镜像下降上升方法》，《神经信息处理系统进展》，第 34 卷，第 10 431–10 443 页，2021年。[25] S. Yang，《用于动态优化问题的精英保留移民遗传算法》，载于《进化计算应用》，M. Giacobini 编。柏林、海德堡：Springer柏林海德堡，2007年，第 627–636 页。