文章编号:1671-4598(2025)07-0227-07 DOI:10.16526/j.cnki.11-4762/tp.2025.07.028 中图分类号:TP393.4 文献标识码:A

基于储备池计算的深度学习模型数据并行训练优化方法

黎祥远[1], 吕峻闽[2], 徐胜超[2]
(1. 广州华商学院 实验教学与网络技术管理中心, 广州 511300;
2. 广州华商学院 人工智能学院, 广州 511300)

摘要: 为了提升深度学习模型训练效率低, 提升收敛性, 对基于储备池计算的深度学习模型数据并行训练优化方法进行了研究; 对训练数据进行平衡和切分, 进行了储备池构建, 采用了飞蛾算法求取连接矩阵中神经元之间的连接密度、权重参数的最优值; 以平衡和切分的训练数据为输入, 采用了构建的储备池计算优化深度学习模型并进行并行训练; 经实验测试, 该模型损失更小且模型曲线更早地趋于平稳, 表明模型更加稳定, 能够更快地收敛到了较好的解; 另外, 该方法训练过程中 CPU 使用率持续较高且波动较小, F_1 分数明显更高, 表明其能够更有效地利用 CPU 资源, 具有较好的泛化能力, 证明了该方法的训练效果。

关键词: 储备池计算; 深度学习模型; 训练数据; 飞蛾算法; 并行训练优化

Data Parallel Training Optimization Method for Deep Learning Model Based on Reservoir Computing

LI Xiangyuan[1], LÜ Junmin[2], XU Shengchao[2]
(1. Center of Experimental Education and Network Technology Management, Guangzhou Huashang College, Guangzhou 511300, China;
2. School of Artificial Intelligence, Guangzhou Huashang College, Guangzhou 511300, China)

Abstract: In order to improve the low training efficiency and convergence of deep learning models, this paper presents a data parallel training optimization method for deep learning model based on reservoir computing, balances and splits the training data, constructs a reserve pool, and uses the moth algorithm to obtain the optimal density and weight parameters between neurons in the connection matrix. Using the balanced and segmented training data as the input, and adopting the constructed reservoir computing to optimize the deep learning model and conduct parallel training. Through experimental testing, the model has smaller losses, and its curve tends to stabilize earlier, indicating that the model is more stable and can converge to a better solution faster. In addition, during training process, this method has the features of high CPU usage and minimal fluctuations, with a significantly higher F_1 score, the results show that this method can more effectively utilize CPU resources and has good generalization ability, proving the training effectiveness of this method.

Keywords: reservoir computing; deep learning models; training data; moth algorithm; parallel training optimization

0 引言

随着模型规模的不断扩大和数据量的急剧增长, 如何高效地训练深度学习模型成为了一个关键挑战。传统的串行训练方式已难以满足现代计算需求, 而数据并行训练作为一种有效的并行化策略, 逐渐成为提升深度学

收稿日期:2025-02-05; **修回日期**:2025-03-13。
基金项目:国家自然科学基金面上项目(61972444);广州华商学院校内科研导师制项目资助(2024HSDS20)。
作者简介:黎祥远(1981-),男,硕士,高级工程师。
吕峻闽(1970-),男,博士,教授。
徐胜超(1980-),男,硕士,副教授。
引用格式:黎祥远, 吕峻闽, 徐胜超.基于储备池计算的深度学习模型数据并行训练优化方法[J].计算机测量与控制,2025,33(7):227-233,242.

习训练效率的重要手段[1]。数据并行训练通过将大规模数据集分割成多个小块，并分配给不同的计算节点进行独立处理，每个节点负责更新模型的一部分参数。这种方式显著减少了单个节点的计算负担，并通过并行化处理加速了整体训练过程[2]。然而，数据并行训练也面临着诸多挑战，即计算资源瓶颈和训练效率低下的问题，因此探索深度学习模型的数据并行训练优化方法对于提升训练效率、缩短训练周期具有重要意义。

例如，文献［3］模拟生物进化过程用来搜索和优化深度学习模型的参数或结构配置。深度学习模型的参数空间通常具有高维度特性，这使得遗传算法在优化过程中难以针对高维度编码设计合适的交叉、变异操作。高维度优化问题不仅增加了算法的计算复杂度，还可能导致算法性能下降，因为遗传算法在高维空间中搜索有效解的能力相对较弱。文献［4］利用鲸鱼优化算法不断调整长短期记忆神经网络模型的参数，以最小化预报误差为目标。鲸鱼优化算法需要对整个种群进行多次迭代计算，且每次迭代都需要评估所有个体的适应度值。对于深度学习模型而言，这通常意味着需要进行大量的模型训练和验证工作，从而导致计算量显著增加，收敛速度变慢。当面对大规模数据集和复杂模型时，这一问题尤为明显。文献［5］将 AVOA 算法与 DELM 模型相结合，利用 AVOA 算法优化 DELM 的关键参数，构建出 WPD-AVOA-DELM 组合预测模型。AVOA 算法在进化过程中容易陷入早熟收敛，即算法在达到全局最优解之前，就提前收敛到某个局部最优解，导致算法性能受限，特别是在处理复杂、高维度的深度学习模型时，这一问题尤为突出。文献［6］将每个粒子代表一组长短期记忆神经网络模型的超参数，然后通过多次迭代，根据粒子的适应度来更新粒子的位置和速度，最终找到使长短期记忆神经网络模型性能最优的超参数组合。粒子群算法属于随机类算法，其搜索过程受到随机数生成器的影响较大，因此在不同的运行条件下，粒子群算法可能得到不同的最优解，这种不确定性降低了算法结果的可靠性和稳定性，使得深度学习模型的优化效果难以预测和控制。

深度学习模型在处理复杂任务时展现出强大的能力，但其训练过程往往耗时且资源密集。储备池计算作为一种简化的循环神经网络框架，以其独特的储备池结构和简单的读出层训练机制，为优化深度学习训练提供了新的思路。储备池机制可以适配多种深度学习模型，包括但不限于 CNN、RNN、LSTM 和 Transformer 等。适配的关键在于：储备池可以根据不同模型的特性和需求调整数据分配策略。例如，RNN 和 LSTM 可能需要考虑时间序列数据的连续性，而 CNN 可能需要考虑图像数据的批量处理。

储备池计算的核心优势之一在于其结构简单且计算高效。传统深度学习模型在并行训练时，各层之间存在复杂的反馈连接，参数更新涉及大量的矩阵运算，计算量庞大。而储备池计算采用固定的随机连接储备池，仅输出层参数需要学习和更新。这大大减少了训练过程中的计算量，尤其在大规模数据并行训练场景下，能有效降低计算资源的消耗，提升训练速度。因此，本文探索基于储备池计算的深度学习模型的数据并行训练优化方法。该优化方法不仅充分利用了储备池计算框架中储备池结构的高效信息处理能力和读出层训练的简便性，还巧妙地结合了数据并行化的策略，有效缓解了深度学习模型训练过程中的耗时与资源密集问题。通过这一方法，本文为深度学习模型的训练效率和性能提升开辟了新的途径。

1 深度学习模型数据并行训练优化

随着深度学习应用的广泛普及，模型训练的效率与可扩展性成为制约其进一步发展的关键因素[7]。为此，本文提出一种基于储备池计算的深度学习模型数据并行训练优化方法，旨在通过结合储备池计算的独特优势与数据并行训练的高效性，提升大规模深度学习模型的训练效率，构建局部储备池以提取数据特征，结合全局同步策略，实现了计算资源的有效分配与利用，同时减少了通信开销，加速了训练过程。本研究分为 3 部分，即训练集平衡和切分、储备池构建以及储备池优化并行训练。

1.1 深度学习模型训练数据平衡和切分

1.1.1 训练数据平衡

训练数据平衡是指调整数据集中各类别样本的数量，使其分布更加均匀，以减少模型因数据不平衡而产生的偏差。数据不平衡通常发生在某些类别的样本数量远多于其他类别时，这可能导致模型在多数类上表现良好，而在少数类上表现不佳[8]。为此，通过 SMOTE 技术进行训练数据平衡，计算欧氏距离 d_i，即

$$d_i = \| X_i - X_k \| = \sqrt{\sum_{j=1}^{n} (x_{ij} - x_{kj})^2} \tag{1}$$

式中，X_i 代表“种子”样本，取值为 90%；X_k 代表少数类样本集中所有其他样本；x_{ij}、x_{kj} 代表 X_i、X_k 样本的第 j 个特征属性。找到该“种子”样本的 M 个最近邻，本文 M 选择 7，选取奇数以避免投票平局。根据 d_i，在两者之间的连线上随机选择一个点作为新的合成样本。重复此过程，直到达到所需的类别平衡[9-10]。

在数据并行的环境中，储备池可以动态地管理和分配数据，使得数据分配更灵活。在这种方法中，可以实时监控每个类别的样本数量，并根据需要从储备池中补

充样本，以维持平衡。

1.1.2　训练数据切分

对训练集切分，即在每个训练周期开始前，将整个训练数据集根据并行进程数（即计算单元数）划分，每个进程只读取自身切分的数据[11]。对于训练数据切分策略，采用分层随机划分方法，确保每个并行进程中包含各类别样本的比例与整体数据一致。这样既能保证数据分布一致性，又利于后续并行计算的均衡负载。切分后的训练数据表示如下：

$$P_s = \{X_1, \cdots, X_m \mid Y_s\};\quad i = 1,2,\cdots,m; s = 1,2,\cdots,S \tag{2}$$

式中，P_s 代表切分后的第 s 个训练集；m 代表 P_s 中训练样本数量，也就是储备池输入节点数；S 代表训练集数量；Y_s 代表对应的输出[12]。经过处理后，数据样本更符合后期训练要求。数据切分策略通常包括以下步骤：

首先，按照任务类型决定切分维度，对于时间序列数据可按时间步切分，对于图像数据可按通道或区域切分；其次，采用分块均匀分配原则，确保每个并行单元接收到的数据量大致相等。以图像分类任务为例，若共有 1 000 张图片且有 4 个并行进程，则可将数据分为每组 250 张的 4 块，并尽量保持每组内各类别样本比例一致。

在分布式训练中，数据切分可能需要考虑计算节点的负载均衡。储备池可以帮助动态分配数据块，确保每个节点在训练过程中都能获得相对均衡的数据量和类别分布。

1.2　储备池构建与参数优化

图 1 为构建的储备池结构。储备池由大量相互连接的神经元组成，这些神经元具有短时程非线性特性，能够捕捉输入数据的动态变化，这大大降低了训练的复杂性和计算成本。储备池构建的关键是创建储备池的内部连接矩阵以及神经元之间的连接密度、权重。

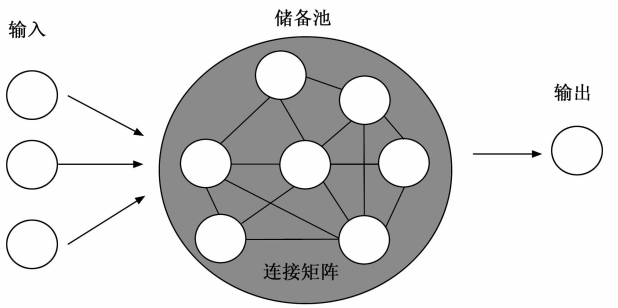

图 1　储备池结构图

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{m} \end{pmatrix} \tag{3}$$

式中，$\boldsymbol{A}$ 代表 $n \times n$ 的连接矩阵，其中 n 是储备池神经元的数量，对应切分后的训练集的数量。

储备池计算（Reservoir Computing）在实时情感分析领域展现出强大潜力。在优化储备池的连接密度和权重时，适应度值函数的设计至关重要。若目标是提高情感分类准确率，则适应度值可定义为分类正确样本数的比例，即 $F = \frac{C}{T}$，其中，C 表示分类正确的样本数量，T 为总样本数。借助飞蛾算法搜索最优参数组合，使得适应度值最大化，进而显著提升模型性能。为此，针对储备池连接矩阵，利用飞蛾算法求取矩阵中神经元之间的连接密度、权重，具体过程如图 2 所示。

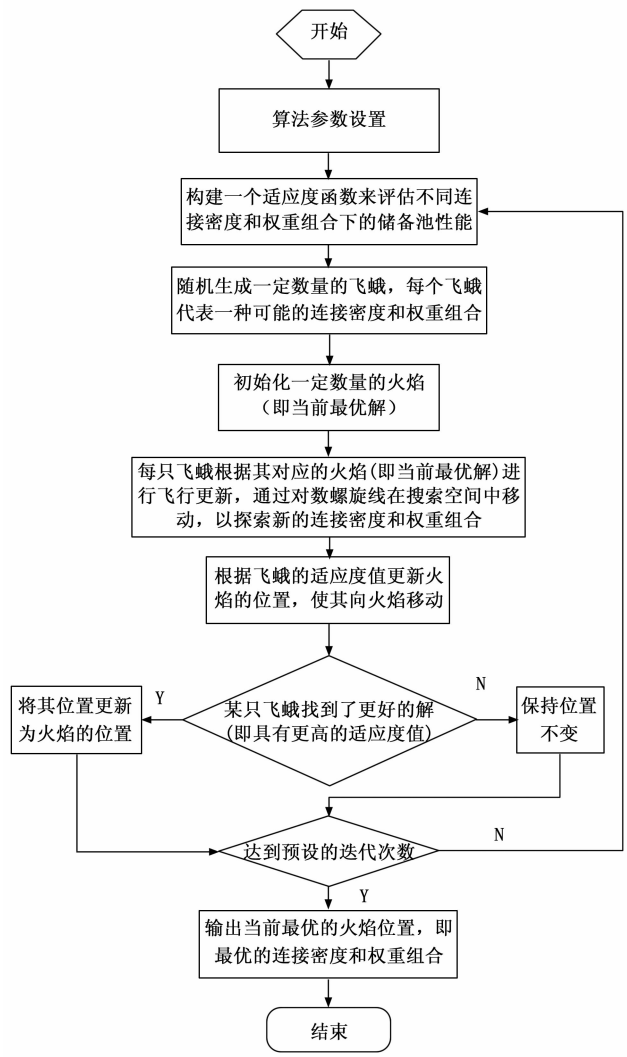

图 2　飞蛾算法求取储备池连接矩阵连接密度和权重

飞蛾算法（MFO，moth-flame optimization）被用于优化深度学习模型的超参数和网络架构。具体实现细节如下。

1）种群大小：飞蛾算法的种群大小直接影响算法的搜索能力和计算复杂度。种群大小设置为 50，这样

可以在保证算法性能的同时控制计算开销。

2）. 迭代次数：迭代次数决定了算法的收敛速度和最终优化效果。一般来说，迭代次数可以设置为 100 次，以确保算法有足够的时间搜索解空间，找到最优解。

3）位置更新机制：飞蛾算法通过模拟飞蛾围绕光源飞行的行为来更新位置。每只飞蛾的位置根据其与火焰的距离进行更新，具体公式为：

$$\mathrm{Moth}_i^{t+1} = \mathrm{Distance}_i \cdot e^{b\cdot l} \cdot \cos(2\pi l) + \mathrm{Flame}_j^t \quad (4)$$

其中：$\mathrm{Distance}^i$ 是飞蛾与火焰之间的距离，b 是常数，l 是一个随机数。

4）火焰数量更新：在每次迭代中，火焰的数量逐渐减少，以增加算法的局部搜索能力。

在上述过程中，设置的适应度函数公式如下：

$$F(w,\rho) = \frac{1}{B} - \lambda(\rho - \tilde{\rho})^2 \quad (5)$$

式中，$F(w,\rho)$ 代表适应度函数；B 代表给定权重下储备池网络在验证集上的均方误差；w 代表储备池连接矩阵的权重；ρ 代表连接密度；λ 代表一个正则化系数，用于控制连接密度与预测误差之间的权衡；$\tilde{\rho}$ 代表期望达到的目标连接密度。$\lambda(\rho-\tilde{\rho})^2$ 用于惩罚与目标连接密度偏差过大的解。如果 $\lambda=0$，则不考虑连接密度；如果 $\lambda>0$，则会在优化过程中考虑连接密度。

经过上述过程，求出的最佳连接密度、权重赋值给各个连接神经元连接矩阵 $\boldsymbol{U}$ 表示如下：

$$u_{ij} = w_{ij}a_{ij} + \rho_{ij} \quad (6)$$

$$\boldsymbol{U} = \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{nm} \end{pmatrix} \quad (7)$$

式中，u_{ij} 代表连接密度、权重赋值后的连接矩阵中元素值；a_{ij} 代表连接矩阵中第 i 行第 j 列元素[13]。构建后的储备池，用于下一章节的训练优化当中。

1.3 储备池计算优化深度学习模型并行训练

储备池计算（RC，reservoir computing）与长短期记忆网络（LSTM）均属于递归神经网络（RNN）的范畴，但二者存在本质区别。LSTM 通过门控机制显式地控制信息流动，具有较强的建模能力，适合处理长依赖问题，但其训练过程复杂且计算开销大。相比之下，RC 中的核心部分——回声状态网络（ESN），仅需在固定连接权重的稀疏储备池上执行线性回归，大幅降低了计算成本，同时保持良好的非线性拟合能力。LSTM 侧重于精确建模复杂的动态行为，而 RC 更注重高效性和简洁性。因此，RC 可以被视为一种“轻量版”的 LSTM，在特定场景下提供了一种权衡性能与效率的新选择。

在模型并行训练中，储备池计算作为一种组件被嵌入到更大的深度学习模型中，通过长短期记忆网络（LSTM），输入大小是 100，隐藏层大小是 128，使用的 TensorFlow 2.3 框架，简化了训练过程，避免了传统循环神经网络（RNN，recurrent neural networks）训练中的梯度消失或梯度爆炸问题，同时保持了良好的计算性能和泛化能力[14-15]。

应用如长短期记忆网络（LSTM）设计网络架构通常包括若干卷积层、池化层和全连接层。具体层数和每层的神经元数量需要根据任务调整。通过飞蛾算法优化的超参数包括学习率、修正线性单元函数（ReLU，rectified linear unit）。通过飞蛾算法的全局搜索能力，可以在较大的参数空间中找到较优的超参数组合。采用数据并行的方法，将训练数据划分成多个子集，并行地在多个计算节点上进行训练。每个节点计算梯度后，通过参数服务器或分布式通信协议（如 MPI）进行梯度聚合和参数更新。通过最小化损失函数来提高模型的泛化能力。通过结合飞蛾算法和深度学习的并行训练方法，有效提高模型的训练效率和性能，在处理大规模数据集时，能够显著缩短训练时间，同时提升模型的准确性和稳健性。

下面针对储备池计算优化深度学习模型并行训练过程进行具体分析。

步骤 1：输入平衡并切分好的训练集 $P_s = \{X_1, \cdots, X_m\}; i = 1,2,\ldots,m; s = 1,2,\ldots,S$。

步骤 2：将输入数据映射到一个计算节点上，在每个计算节点上，使用不同的数据子集对深度学习模型进行训练[16]。

步骤 3：选取一个子集，提取训练子集的特征向量，如式（8）所示[17]：

$$v_1 = f(X_i \mid \mu) \in \Im^{[C,H,L]} \quad (8)$$

式中，v_1 代表特征向量；C 特征图的通道数，H,L 分别为特征图的高和宽；f 代表传递函数。

步骤 4：更新储备池内部神经元状态。

$$D_{t+1} = \tanh(u_1 w_{in} + D_t S), t = 0,1,\ldots,C-1 \quad (9)$$

式中，D_t、D_{t+1} 代表更新前、后的储备池内部神经元的状态；w_{in} 代表的输入层权重；tanh 表示双曲正切函数。

步骤 5：计算储备池输出 q_t，即

$$q_t = w_{\mathrm{out}} D_{t+1} \quad (10)$$

式中，w_{out} 代表输出权重矩阵。

步骤 6：以储备池输出 q_t 为输入，训练深度学习模型[18-19]。前向传播公式如下：

$$Z_s = \mathrm{ReLU}\Big(\sum_{j=1}^{M} g_1 \mathrm{ReLU}\Big(\sum_{t=1}^{C} g_2 q_t + l_1\Big) + l_2\Big) \quad (11)$$

式中，Z_s 代表第 s 个输出层输出；ReLU 代表非线性激活函数；g_1、g_2 代表连接权值；l_1、l_2 代表偏置。

将产生模型的最终输出 Z_k 与真实目标值进行比较，并计算损失 φ：

$$\varphi=\sqrt{\sum_{s=1}^{S}(Z_s-Y_s)^2} \tag{12}$$

式中，Z_s 代表神经元的真实目标值。

根据前向传播产生的损失，通过梯度下降来更新深度学习模型的参数（如权重和偏置）。前向传播和反向传播的过程会重复多次，直到模型在验证集上的性能停止提高或达到预定的训练轮次[20]。在每次迭代中，都会使用训练集划分不同子集来更新模型参数。最后，就可以用训练好的深度学习模型对新的输入数据进行预测或分类。

2　实验测试与性能分析

2.1　实验环境

基于储备池计算的深度学习模型数据并行训练优化方法的实验环境设置如下所示。以文献［4］、文献［5］、文献［6］的 3 个方法作为性能分析比较对象。

文献［4］的方法为利用鲸鱼优化算法不断调整 LSTM 模型的参数；文献［5］的方法为基于非洲秃鹫优化算法的方法；文献［6］的方法为基于粒子群优化的方法。

本文的测试深度模型是经过飞蛾算法优化后的回声状态网络（ESN）。该模型首先通过训练集数据调整其储备池的连接密度和权重，然后利用优化后的参数对测试集进行预测。

1）硬件环境：

显卡种类：NVIDIA Tesla V100 或 A100

显卡数量：8 张

显存大小：16 GB

CPU：Intel Xeon Gold 6226R

内存：256 GB RAM

存储：1TB SSD，用于快速数据读取和写入。

2）软件环境：

操作系统：Ubuntu 20.04 LTS

深度学习框架：PyTorch 2.0

Python：Python 3.8

CUDA：CUDA 11.7（用于 GPU 加速）

cuDNN：cuDNN 8.x

通过合理的实验环境的配置，可以确保其他研究人员或开发者能够按照相同的条件复现实验结果，这对于验证实验结果的有效性、推动科学进步以及软件的可维护性至关重要。

2.2　数据准备与划分

选择适合深度学习训练的大型数据集——CIFAR-10、MNIST 。CIFAR-10 分为 10 个类别[21]，每个类别有 6 000 张图像。这些图像大部分是自然场景中的物体，如飞机、汽车、鸟类等。根据并行训练的进程数（GPU 数）从数据集中选取一定数量的数据并划分为多个子集，每个子集包含部分数据用于独立训练，具体划分情况如表 1 所示。

表 1　数据准备与划分

类别	图像数量	子集数量
飞机（airplane）	300	462
汽车（automobile）	300	213
鸟类（bird）	300	484
猫（cat）	300	623
鹿（deer）	300	695
狗（dog）	300	558
蛙（frog）	300	741
马（horse）	300	625
船（ship）	300	412
卡车（truck）	300	525

MNIST 数据集由手写数字的图像组成，包含 60 000个训练样本和 10 000 个测试样本，主要用于简单图像分类任务的基准测试。

将训练集随机打乱，以确保类别在数据中的分布是随机的。然后，根据 GPU 的数量 N，将打乱后的数据集平均分配到 N 个子集中。

2.3　实验参数设置

储备池以及深度学习模型在并行训练中设置的参数如表 2 所示。

表 2　储备池实验参数设置表

类别	参数值
储备池类型	Echo State Network（ESN）
储备池节点数	1 000
储备池激活函数	Sigmoid
储备池稀疏度	0.1（10%）
储备池谱半径	0.9
数据并行策略	数据分片与参数同步
并行设备数	8 GPUs
梯度同步方法	Allreduce
优化算法	Adam
学习率	0.001
批量大小	64
训练迭代数	250
批归一化策略	SyncBatchNorm
损失要求	10^{-5}

梯度同步采用 AllReduce 算法实现全局梯度聚合，将每个并行单元独立计算本地梯度后，通过通信库将所有本地梯度累加并广播回各单元，最后更新模型参数。引入层级通信结构或混合精度训练技术。根据硬件环境调整通信频率，每隔若干轮迭代同步一次，以平衡计算

与通信开销。

表 2 中这些参数的合理设置对于提升模型的整体性能和训练效率具有至关重要的作用。

2.4 并行训练结果

借助 Matlab，对储备池计算优化后的深度学习模型进行并行训练并与储备池计算优化前进行对比，结果如图 3 所示。

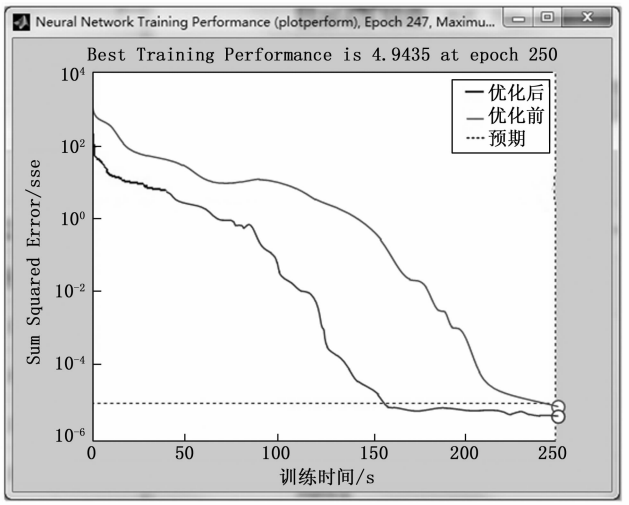

图 3 并行训练结果

从图 3 中可以看出，优化后的模型曲线在优化前模型的曲线下方，表明优化后的模型损失小，在相同的训练条件下达到了更好的性能。另外，利用数据切片策略进行独立训练，通过增加数据的多样性和独立性，为模型提供了在每个数据切片上进行独立训练的机会，从而促使模型能够学习到多样化的局部最优解。最终通过集成方式获得更为鲁棒的全局最优解，使模型曲线更早地趋于平稳。

2.5 模型精度对比

对比不同方法下模型精度，评估模型在验证集或测试集上的准确率。

CIFAR-10 数据集下模型精度见表 3。

表 3 模型精度对比

方法	训练时间/h	模型精度/%
文献[4]方法	10	82.5
文献[5]方法	6	85.0
文献[6]方法	8	84.3
本文方法	3.2	88.9

分析表 3 可知，文献［4］方法的训练时长为 10 h，文献［5］方法的训练时长为 6 h，文献［6］方法的训练时长为 8 h，本文方法的训练时长为 3.2 h；本文方法的训练时间远远低于其他方法，而文献［4］方法的模型精度为 82.5%，文献［5］方法的模型精度为 85.0%，文献［6］方法的模型精度为 82.5%，本文方法的模型精度为 88.9%，上述结果表明，本文方法能够有效提升模型精度，并且缩短训练时长。

MNIST 数据集下模型精度见表 4。

表 4 模型精度对比

方法	训练时间/h	模型精度/%
文献[4]方法	12	72.9
文献[5]方法	7.5	79.6
文献[6]方法	9.6	82.5
本文方法	2.9	96.5

分析表 4 可知，文献［4］方法的训练时长为 12 h，文献［5］方法的训练时长为 7.5 h，文献［6］方法的训练时长为 9.6 h，本文方法的训练时长为 2.9 h；本文方法的训练时间远远低于其他方法，而文献［4］方法的模型精度为 72.9%，文献［5］方法的模型精度为 79.6%，文献［6］方法的模型精度为 82.5%，本文方法的模型精度为 96.5%，上述结果表明，本文方法能够有效提升模型精度，并且缩短训练时长。

2.6 泛化能力测试

在深度学习领域，模型的训练只是第一步，真正的挑战在于将训练好的模型应用于实际场景中，以解决实际问题。为了全面评估深度学习模型的训练效果，将其与 3 种传统训练后的方法进行比较，通过实际测试数据来验证模型的训练成果和泛化能力。方法为五折交叉验证法，验证指标为 F_1 分数。F_1 分数对比如图 4 所示。

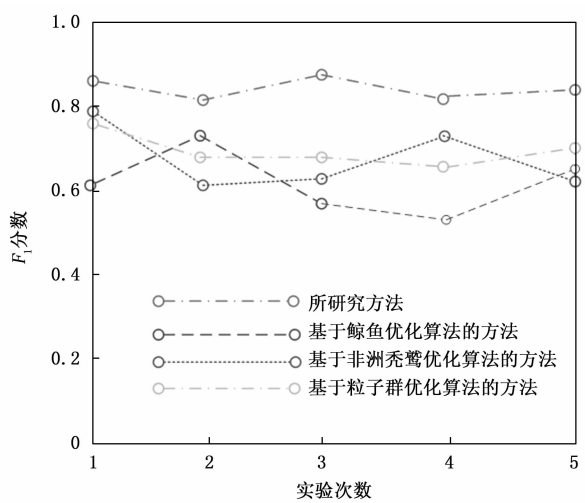

图 4 F_1 分数对比结果

从图 4 中可以看出，所研究方法训练后的模型 F_1 分数明显高于其他 3 种模型，通过在不同数据切片上实施独立训练的策略，训练后的模型通过减轻对整体数据集特定特征的过度依赖，从而显著降低了过拟合的风险。这种策略有效地增强了模型的泛化能力，使该模型在综

合评估下表现最佳，进一步证明了的方法的训练效果。

2.7　梯度消失或爆炸解决效果对比

为验证储备池机制对梯度消失或爆炸问题的解决效果，在相同的模型架构和数据集上，比较使用和不使用储备池机制的模型训练表现，如表 5。

表 5　梯度消失或爆炸解决效果对比

指标	使用储备池机制	不使用储备池机制
训练时间/h	3.2	6.8
模型准确性/%	88.9	85.2

通过对比表 5 结果可知，在相同的模型架构和数据集上，本文方法在使用储备池机制后的训练时间与模型准确性都明显上升，表明本文方法能够有效梯度消失或爆炸解决效果。

为进一步验证本文方法性能，在 ImageNet 数据集上对不同方法训练过程中准确率进行了验证得的结果见表 6 所示。

表 6　ImageNet 数据集下准确率

数据量	方法	准确率/%
500 GB	文献[4]方法	67.8
	文献[5]方法	69.2
	文献[6]方法	75.6
	本文方法	98.6

由表 6 可知，数据量为 500 GB 时，文献［4］方法的训练过程中准确率为 67.8%，文献［5］方法的训练过程中准确率为 75.6%，文献［6］方法的训练过程中准确率为 98.6%，本文方法的训练过程中准确率为 98.6%，上述结果表明，本文方法能够有效提升训练过程中准确率。

3　结束语

本文通过采用储备池计算原理的深度学习模型，在融入数据并行训练策略后，展现了一种针对大规模模型训练的高效优化途径。该方法将储备池计算内在的高效信息处理特性与数据并行化的强大并行计算能力相结合。最终，通过一系列实验验证，表明该方法相比于其他传统方法。期待在以下几个方面看到更多的进展：一是进一步优化储备池的计算机制，提高其在处理高维、非线性数据时的效率；二是探索更加高效、灵活的数据并行化技术，以适应不同规模和特性的数据集；三是将该方法应用于更多实际场景中，如自然语言处理、图像处理、时间序列预测等，推动人工智能大模型的进一步发展[22]。

参考文献：

[1] LIU C，SUI H，ZENG S. Efficient building damage assessment from post-disaster aerial video using lightweight deep learning models［J］. International journal of remote sensing，2023，44（22）：6954－6980.

[2] WEN Y，QIU Z，ZHANG D，et al. Accelerating massively distributed deep learning through efficient pseudo-synchronous update method［J］. International Journal of Parallel Programming，2024，52（3）：125－146.

[3] ELIZABETH PALACIOS-MOROCHO，SAUL INCA，JOSE F，et al. Multipath planning acceleration method with double deep R-Learning based on a genetic algorithm［J］. IEEE Transactions on Vehicular Technology，2023，72（10）：12681－12696.

[4] 丁艺鼎，蒋名亮，徐力刚，等. 基于鲸鱼优化算法的长短期记忆模型水库洪水预报［J］. 湖泊科学，2024，36（1）：320－332.

[5] 王忠义，崔东文. 基于小波包分解－非洲秃鹫优化算法－深度极限学习机的水文预报模型及其应用［J］. 水电能源科学，2022，40（8）：26－31.

[6] 吴　飞，农皓业，马晨浩. 基于粒子群优化算法－长短时记忆模型的刀具磨损预测方法［J］. 吉林大学学报（工学版），2023，53（4）：989－997.

[7] NOURANE ANDERRAHIM，AMIR ECHTIOUI，RAFIK KHEMAKHEM，et al. Epileptic seizures detection using iEEG signals and deep learning models［J］. Circuits，systems，and signal processing：CSSP，2024，43（3）：1597－1626.

[8] MIRA A，HELLWICH O. Deep learning models beyond temporal frame-wise features for hand gesture video recognition［J］. The Journal of Supercomputing，2024，80（9）：12430－12462.

[9] HEIDARI A，NAVIMIPOUR N J，DAG H，et al. A novel blockchain-based deepfake detection method using federated and deep learning models［J］. Cognitive Computation，2024，16（3）：1073－1091.

[10] SARMUN R，CHOWDHURY M E H，MURUGAPPAN M，et al. Diabetic foot ulcer detection：combining deep learning models for improved localization［J］. Cognitive Computation，2024，16（3）：1413－1431.

[11] NADEEM M，SOHAIL S S，JAVED L，et al. Vision-Enabled large language and deep learning models for image-based emotion recognition［J］. Cognitive Computation，2024，16（5）：2566－2579.

[12] ENIREDDY V，KARTHIKEYAN C，BABU D V. One-HotEncoding and LSTM-based deep learning models for protein secondary structure prediction［J］. Soft Computing，2022，26（8）：3825－3836.

（下转第 242 页）