基于 OpenCL 的卷积神经网络并行化设计

计算机科学与技术专业　王乃博
指 导 教 师　张剑贤

[摘　要]本文完成了基于 OpenCL 的卷积神经网络并行化设计，并在异构并行平台上验证了所设计算法的可行性及正确性。在分析手写数字识别的卷积神经网络基本结构的基础上，本文归纳总结了基于 OpenCL 的卷积神经网络训练的优化方法，提出了单卷积过程并行、多卷积任务并行、多卷积数据并行和批处理等多个基于 OpenCL 的并行优化方案，并完成优化方案的编程实现。在 Intel CPU、AMD GPU 和 NVIDIA GPU 平台上实现了 CNN 算法的训练、测试过程以及仿真验证。测试结果表明在相同训练正确率的情况下，本文所提出的并行优化方案比串行执行方法在速度上提升了约 375 倍。
[关键词] OpenCL 卷积神经网络 异构并行计算

[Abstract] This paper completes the parallel design of convolution neural network based on OpenCL, and verifies the feasibility and correctness of the designed algorithm on heterogeneous parallel platform. Based on the analysis of the basic structure of convolution neural network of handwritten numeral recognition, this paper summarizes the optimization method of convolution neural network training based on OpenCL, and proposes a single convolution process parallel, multi-convolution task parallel, multi-convolution data Parallel and batch processing and other openCL-based parallel optimization program, and complete the optimization program programming. The training, testing process and simulation of CNN algorithm are realized on Intel CPU, AMD GPU and NVIDIA GPU platform. The test results show that the parallel optimization scheme proposed in this paper is about 375 times faster than the serial execution method in the case of the same training accuracy rate.
[Key Words] OpenCL　Convolution neural network　heterogeneous parallel computing

一、引言

随着深度学习技术[1]的快速发展，卷积神经网络（Convolutional Neural Network,CNN）[1]已成为当前语音分析和图像识别领域的研究热点。由于多层神经网络结构计算的复杂度比较高，使得单一结构的计算平台以串行化进行神经网络训练的方法已难以满足系统实时性需求。异构并行计算逐渐成为了提升算法性能的主要方法。OpenCL[2]是一种跨平台的异构并行编程架构，可以在不同平台上进行移植，而且支持数据并行和任务并行。本文以卷积神经网络为优化目标，在异构并行计算平台上采用 OpenCL 并行语言

设计实现可以识别手写数字的卷积神经网络，并完成基于 OpenCL 的卷积神经网络算法的测试验证。实验结果表明所提出算法能够有效提高卷积神经网络的训练速度，降低训练时间。。

二、基于 OpenCL 的卷积神经网络算法优化方法

本文首先分析 OpenCL 的基本架构[2]，包括平台模型、执行模型、存储器模型、编程模型等四个组成部分以及基于 OpenCL 的编程开发方法。然后分析本文所用到的用于手写数字识别的卷积神经网络结构[1]如图 1 所示。它由一个输入层（INPUT），两个卷积层（C1 和 C3），两个下采样层（S2 和 S4）以及一个输出层（OUTPUT）组成。每层的规模如图中数字所示。

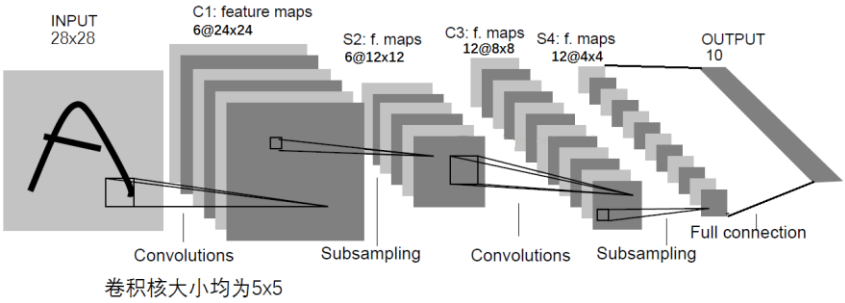

图 1 用于手写数字识别的卷积神经网络结构

根据手写数字识别的卷积神经网络基本结构，本文归纳总结了基于 OpenCL 的卷积神经网络并行化设计方法[4]如图 2 所示。OpenCL 主要对前向传到、后向传播及更新权值模块进行加速。

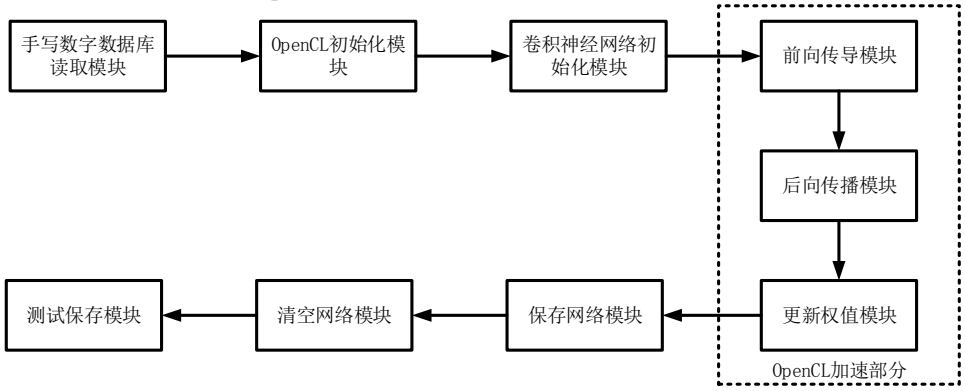

图 2 基于 OpenCL 的卷积神经网络优化方法

在并行优化过程中，论文对在线处理网络和批处理网络[5]进行了多次不同层次和不同方法的并行优化。首先进行 OpenCL 环境的搭建及平台（CPU、GPU）的选择，然后根据具体模块的优化思路编写不同的 OpenCL 内核函数和主机端代码，最后对训练结果进行输出保存。

1. 单卷积过程并行优化

影响网络训练速度的主要因素是网络训练过程中的卷积操作速度较慢，把卷积操作写成并行化的卷积内核函数[6]，然后向它传入不同的输入图像和卷积图像（即不同的输入数据）就可以完成对卷积操作的并行化运算。定义内核函数 convolutionNaive 来实现并行卷积操作。由于 OpenCL 只支持一维数组传入，因此先在主机端把卷积核、输入图像和保存输出图像的数组由二维转化成一维，然后再传入 OpenCL 内核函数中进行操作。

将运算过程中保持不变的数据放到常量存储区域可以有效提高运算速度。因此，本文将卷积过程中不变的卷积核数据存放到常量存储区

同样，如果把计算的原始图像放到工作组的局部存储器[7]中，将会大大加速工作项读取信息的速度，因此，这里定义了局部存储器优化版本的内核函数，通过在内核函数体中加入局部数组来进行全局访问到局部访问的转换，从而实现对卷积过程中读取速度的优化。

论文根据卷积操作本身具有的并行性实现了对单个卷积过程的三层优化。而 CNN 本身也具有并行化，因此，后续的论文工作将针对 CNN 本身的并行化特性进行并行优化设计。

2. 多卷积任务并行优化

CNN 在训练过程中存在 4 种不同的卷积操作，而进行每一种卷积操作时，各卷积操作之间没有数据依赖关系。同理，对卷积过程中其他互不相关的卷积操作，也可以采用并行化方法实现。

OpenCL 支持任务并行模式，因此，论文采用任务并行模式对上述卷积操作进行优化。首先应在主机端建立好足够的内存来存储多种卷积并行计算的结果，然后在主机端不停地发出卷积命令，最后用 OpenCL 同步机制等待卷积操作执行完毕，具体优化过程如图 3 所示。

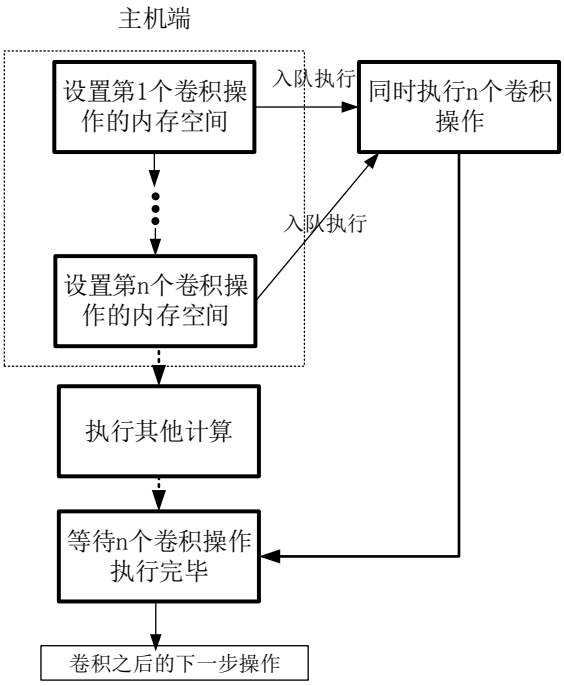

图 3 CNN 卷积过程中的任务并行卷积流程

3. 多卷积数据并行优化

由于 OpenCL2.0 规范[8]增加了对细粒度系统 SVM 的支持，使得在内核函数中操作主机端的二维及以上的数组成为了可能。因此，可以对卷积操作进行进一步的优化。

首先，由于 OpenCL2.0 支持直接访问二维数组，因此在主机端可以去掉二维转一维和输出之后的一维转二维的操作，这将节省大量的运算时间。

最后，可以使用数据并行的模型，对多卷积操作进行数据并行的优化，流程如图 4 所示。

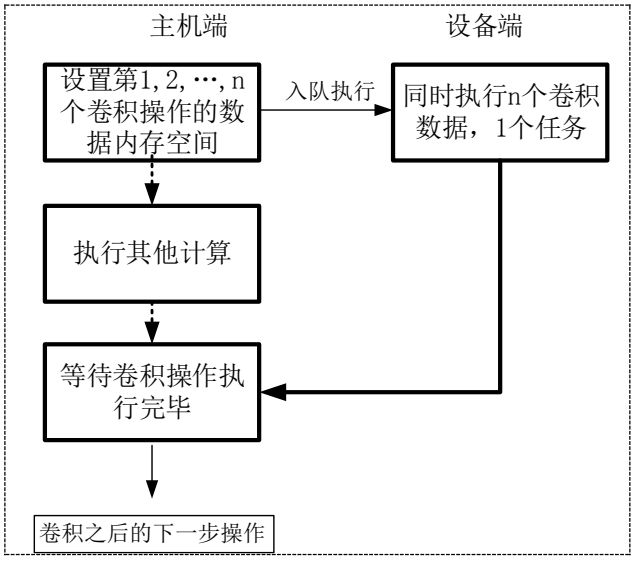

图 4 CNN 卷积过程中的数据并行卷积流程

以上三种 CNN 训练过程的并行优化方案属于在线处理方式。若要进一步提升执行速度，则需要修改训练算法，即可通过批处理方式进行并行优化。

4. 批处理优化

前面论文提出了一次训练过程中使用 OpenCL 对卷积过程进行优化的设计方案。本节则通过批处理的方式对 CNN 进行训练以期达到更快的训练速度。批处理优化的特点是向 CNN 输入多

幅图像后才统一更新一次权值，具体流程如图 5 所示。

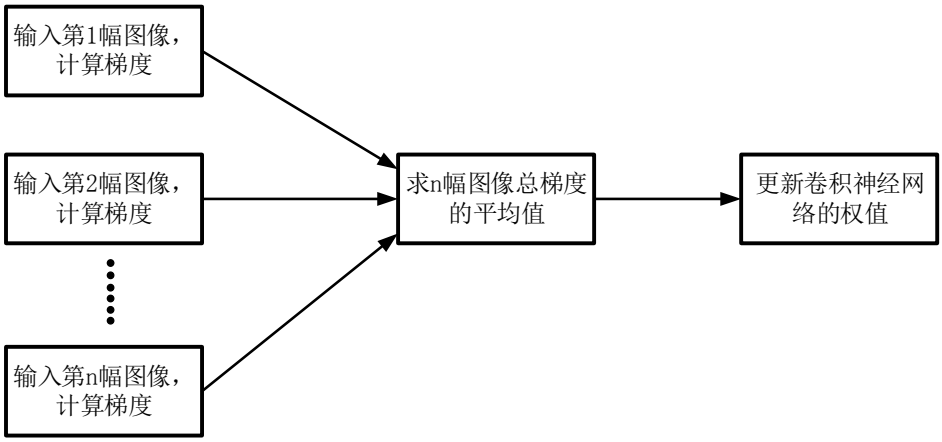

图 5 批处理训练 CNN 流程

由图5可得，训练n幅图像的过程本身没有任何先后顺序和数据关联。因此，可以利用OpenCL同时对 n 幅图像进行训练。例如同时对 50 幅图像进行前向传导、后向传播、计算梯度的训练过程，然后主机端等待执行完成后对 50 幅图像计算出的梯度进行求平均操作，最后更新 CNN 各层的权值。

同样，需要分任务并行和数据并行两种方式对批处理方式的卷积神经网络进行训练，以数据并行为例，流程如图 6 所示。

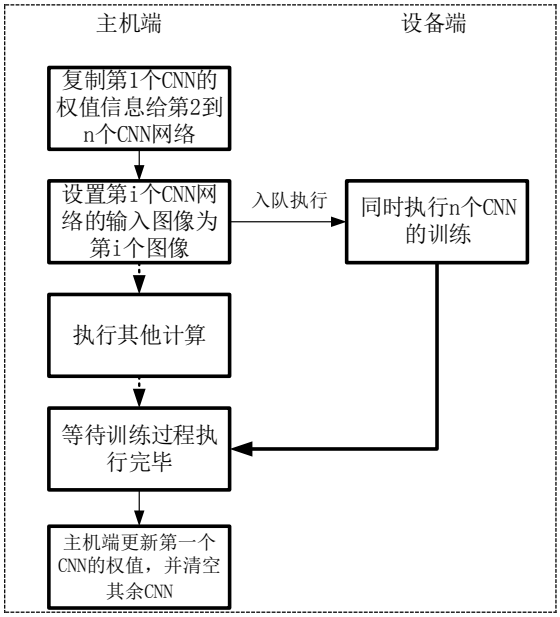

图 6 CNN 批处理数据并行执行流程

三、并行算法实现及结果分析

1. 实验数据选取以及错误率检测

本文选取的实验数据是纽约大学的公开手写数字识别库 mnist，它包含四个文件，文件名称和作用如表 1 所示，其中训练样本和测试样本规模分别是 60000 和 10000。

表 1 mnist 手写数字识别数据库说明

文件名称	文件说明	样本大小
train-images-idx3-ubyte.gz	训练样本集图像	60000
train-labels-idx1-ubyte.gz	训练样本集标签	60000
t10k-images-idx3-ubyte.gz	测试样本集图像	10000
t10k-labels-idx1-ubyte.gz	测试样本集标签	10000

算法并行优化的目标是在不降低网络正确率前提下有效提升 OpenCL 并行算法的速度。论文首先对串行化版本的 CNN 训练过程的训练错误率/正确率进行测试。每 1000 组数据训练完计算一次 CNN 网络的错误率，并将每个网络的学习率 α 都设置为 1.0。测试结果如图 7 所示，经过 60 万次训练后的测试错误率最终达到了 1.5%左右，已经低于人眼识别错误率，表明了所设计的 CNN 网络的可用性。

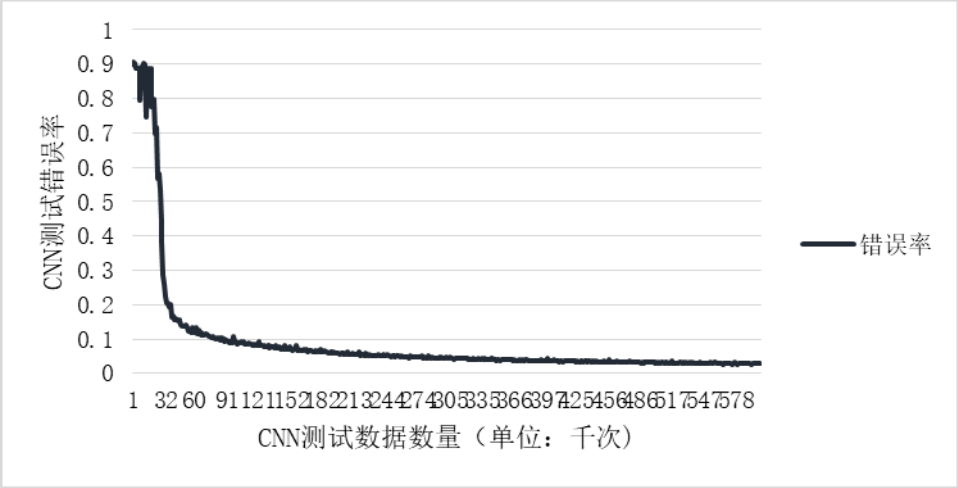

图 7 CNN 测试错误率折线图

2. 并行计算方法设计及测试验证

论文采用 OpenCL 语言编程实现了所提出的四种并行方案，并在不同计算平台上进行了测试验证，并对测试和训练的时间，加速比以及 CNN 的训练正确率进行对比分析。

1） 单卷积核优化及结果分析

对于单卷积核优化设计方案，本文分别采用了 Intel 的 E5CPU、NVIDIA 的 GPU、AMD 的 GPU 以及 CPU 四种设备对 3 种不同的内核函数进行了测试和训练优化，优化结果分别如图 8、图 9 所示。

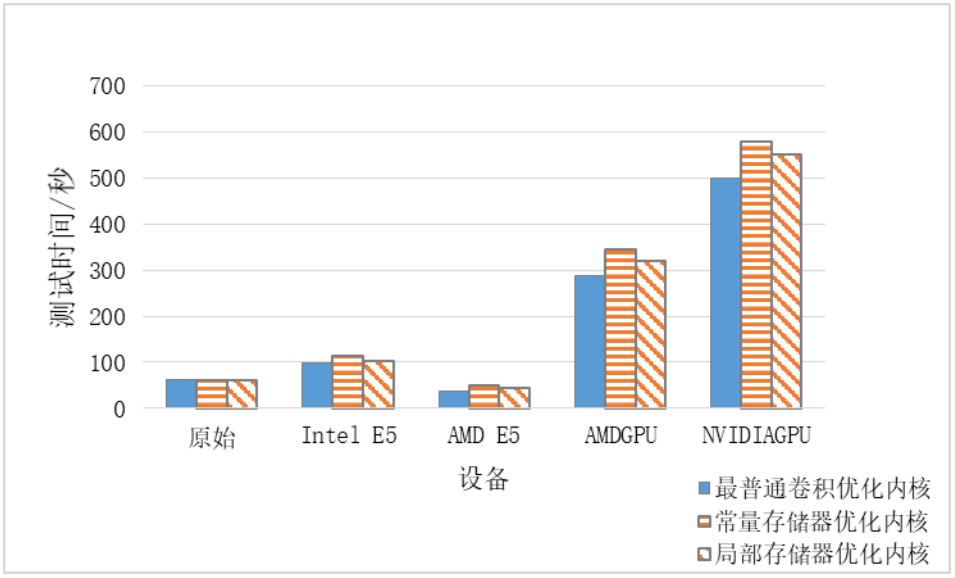

图 8 单卷积核优化测试过程在不同平台下的加速效果图

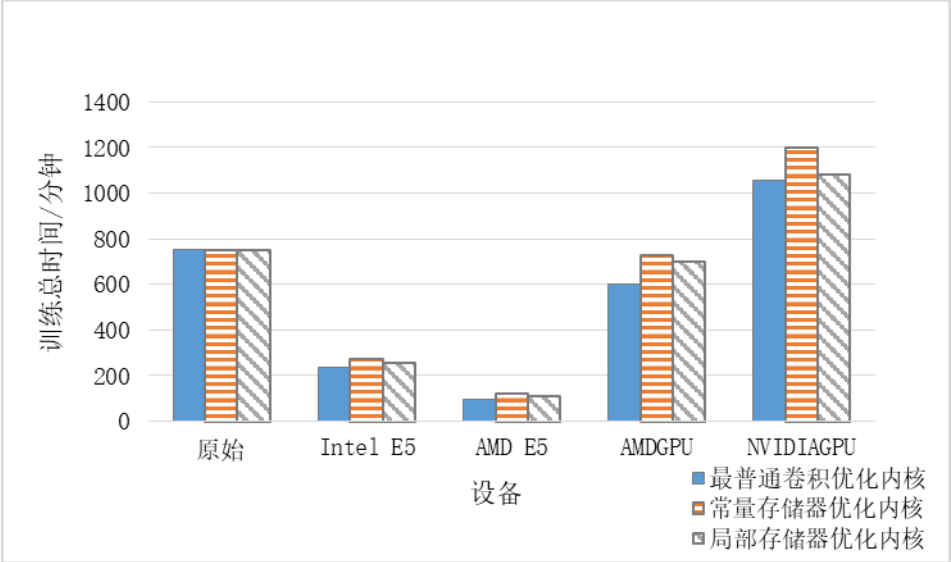

图 9 单卷积核优化训练过程在不同平台下的加速效果图

根据优化结果可得，相对于原始串行化版本的程序，只有使用 AMD 平台下的 CPU 才能对 CNN 测试过程实现加速，而其他平台反而会导致速度下降；而对于训练过程，可以看出除了 NVIDIA 的 GPU 用 OpenCL 后速度比原来还慢，其他的平台都成功的实现了对 CNN 的训练加速过程。根据分析得，前向传导时间和后向传播时间都远远小于更新权值所用的时间，因此，单卷积核优化的最终效果是对更新权值过程进行了卷积优化，并主要优化了 C1 层计算卷积梯度的卷积过程，而对前向传播的卷积过程，并没有实际的速度提升，甚至出现了速度下降。因此，只有当卷积运算量较大的时候，才能体现出并行化加速的优势。

此外，使用最普通的卷积内核函数的速度>使用局部存储器优化的卷积内核函数>仅使用常量存储器优化的卷积内核函数。经过分析可知，在卷积图像和卷积核都比较小的情况下，存储器访问优化并不是影响计算速度的主要原因，但因为使用局部存储器、常量存储器优化后有数据的迁移过程，因此在少量计算的卷积过程中，运算速度反而会降低。

2） 多卷积核并行优化及结果分析

根据提出的多卷积任务并行优化方式，论文对四种不同的设备进行了测试和训练过程的优化，并分别对单命令队列和多命令队列两种情况下的性能进行了比较，结果如表 2 和表 3 所示。

表 2 使用命令队列对 CNN 测试时间的不同平台下的优化结果

平台	单命令队列优化时间/秒	多命令队列优化时间/秒
INTEL E5	68.2	69.1
INTEL AMD 平台	19.7	28.5
NVIDIA	140	150
AMD	149	148

表 3 使用命令队列对 CNN 训练时间的不同平台下的优化结果

平台	单命令队列优化时间/分钟	多命令队列优化时间/分钟
INTEL E5	170	174
INTEL AMD 平台	59.8	80
NVIDIA	317	330
AMD	311	325

由表可知在任务并行优化的环境下，程序的测试速度不变，而对于训练速度则有了大幅度的提升，所有 OpenCL 平台下的设备都相比原始训练速度有了显著提升。并且实验证明了向一个命

令队列异步发送多条可并行执行的命令，其效果和多个命令队列相同。但由于主机端需要等待多个命令队列，所以前者反而更快。

相比原来的训练过程，优化后的训练最快加速了 12.5 倍。测试过程也加速了 3 倍。由于数据并行方案只适用于支持细粒度系统 SVM 的平台，只有 Intel 平台下的 E5CPU 支持，因此，论文只对 E5CPU 进行数据并行优化，同时对去掉二维转一维、一维转二维操作之后的训练和测试时间进行了比较，如表 4 所示。

表 4 数据并行卷积对 CNN 训练和测试过程的优化比较

优化方式	测试时间/秒	训练时间/分钟
INTEL E5 一维卷积	24	55
INTEL E5 二维卷积	11	44

由表 4 可知使用数据并行对卷积过程进行优化之后，训练和测试速度都对原始版本有了明显提升；同时，去掉二维和一维的互相转换的过程后，速度照原始版本速度提升了大约 20%，从而证明了该方法的有效性。

论文采用 OpenCL 设计的 CNN 并行优化方案将原来运行 750min 的训练过程减少到了 44min，把原来 61s 的测试过程减少到了 11s，速度分别提升了 17 倍和 5.5 倍。同时对所有加速之后的网络进行正确率测试，实验结果网络错误识别率都在 1.5%左右，从而说明了使用 OpenCL 对 CNN 进行训练和测试加速的正确性和可用性。

3） 批处理并行优化及结果分析

首先编写串行化版本的批处理 CNN 网络训练程序，并进行了训练，最终网络的正确率同图 7，测试时间：61.5s，训练时间：748min，和在线处理方式无明显差别。

按照批处理优化方案，论文把整个网络的训练和测试过程全部编写成 OpenCL 内核，通过设备端来进行训练和测试。论文只在支持细粒度系统 SVM 的 Intel E5 平台下进行了相关优化，设置一批数据规模为 50，并使用了任务并行和数据并行两种方式，最终的优化效果如表 5 所示。

表 5 批处理并行方案对 CNN 训练和测试过程的优化比较

优化方式	测试时间/秒	训练时间/分钟
INTEL E5 任务并行	4	20
INTEL E5 数据并行	0.05	2

由表 5 可得，当使用任务并行方式对批处理 CNN 进行优化时，测试速度提升了 15 倍，训练速度提升了 37.5 倍；而使用数据并行方式对批处理 CNN 进行优化时，测试速度提升了 1200 倍，训练速度提升了 375 倍，极大地提升了 CNN 网络的训练和测试的速度。同时证明了数据并行比任务并行速度更快的事实。

四、总结

本文在异构计算平台上使用 OpenCL 语言实现了用于手写数字识别的卷积神经网络的训练工程，通过一系列不同的优化方案，最终成功的在保持正确率不变的情况下，把训练速度提升到了原来的 375 倍，测试速度提升到原来的 1200 倍，证明了本文提出的并行计算方案的有效性。论文采用 OpenCL 语言设计实现了卷积神经网络的并行化算法，对单卷积过程并行、单卷积任务并行、单卷积数据并行以及批处理并行四个部分进行和优化方案设计与实现，并在四种不同的平台上进行了测试验证。实验结果表明了所设计的并行算法的正确性和有效性。

参考文献

[1] 吴岸城. 神经网络与深度学习[M]. 电子工业出版社. 2016.

[2] Aaftab Munshi,Benedict R.Gaster,Timothy G.mattson 等. OpenCL 编程指南[M]. 机械工业出版社. 2013.
[3] 周志华. 机器学习[M]. 清华大学出版社. 2016.
[4] 乐毅，王斌. 深度学习——Caffe 之经典模型详解与实战[M]. 电子工业出版社. 2016.
[5] 赵永科. 深度学习——21 天实战 Caffe[M]. 电子工业出版社. 2016.
[6] 刘文志，陈轶，吴长江. OpenCL 异构并行计算[M]. 机械工业出版社. 2016.
[7] 詹云,赵新灿,谭同德. 基于 OpenCL 的异构系统并行编程[J]. 计算机工程与设计. 2012,33(11): 4192-4193.
[8] Khronos OpenCLWorking Group. The OpenCL Specification[M].. Version1.1. Khronos Group. 2011.