doi:10.3772/j.issn.1002-0470.2022.04.001

基于软硬件协同加速的关系网络推理优化方法①

张志超②[*,**,***,****] 王 剑[*,**,***] 章隆兵[*,**,***] 肖俊华[*,**,***]

([*]计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)

([**]中国科学院计算技术研究所 北京 100190)

([***]中国科学院大学 北京 100049)

([****]中国电子科技集团公司第十五研究所 北京 100083)

摘 要 针对数据中心基于图形处理器(GPU)平台的关系网络推理计算中存在的低效能问题,本文提出了一种基于软硬件协同加速的关系网络优化方法。该方法采用基于GPU 提取的支持集特征池与现场可编程门阵列(FPGA)推理异构协同的方式处理关系网络的推理计算,在高效能计算的同时保持关系网络的推理计算与 GPU 平台一致的准确率。利用基于高级综合(HLS)优化浮点卷积神经网络的计算方式,提高关系网络的处理能效。利用多运算单元异构多核处理的方式,满足 FPGA 时序收敛的同时,提升 FPGA 片上吞吐能力。本文在 FPGA 平台上实现了关系网络推理运算单元,在 Omniglot 数据集上构建的加速器功耗为 15.867 W,相对于 GPU 加速比为 1.4 ~ 17.2;在 miniImageNet 数据集上构建的加速器功耗为 12.359 W,相对于 GPU 加速比为 1.5 ~ 3.4。本文方法与同类 FPGA 加速浮点卷积神经网络相比,达到了最优的计算效能。实验数据表明,该方法有效利用了软硬件协同计算以及 FPGA 可重构计算的优势,降低了软硬件协同开发的耦合度,在保持关系网络推理计算准确率的同时,提升了关系网络推理的计算效能。

关键词 关系网络;软硬件协同加速;卷积神经网络;异构多核

0 引 言

基于深度学习的卷积神经网络技术广泛应用于图像处理任务中,在大规模标注的 ImageNet 数据集上提出的 AlexNet[1]、VGG16[2]、ResNet[3]等大规模深度模型,展现了较高的识别准确率。然而,在少样本学习应用中,尤其是对新的未知类别分类任务中,需要新的学习模式和方法,包括 Matching Nets[4]、Meta Nets[5]、MAML[6]、Prototypical Nets[7]、Relation Net[8]等少样本学习方法,通过构建多批次的不同类别任务对模型进行训练,引入支持集作为先验知识,用以处理未知类别任务的分类,关系网络[8](Relation Net)相对于其他模型[4-7],在 Ominiglot[9]数据集和 miniImageNet[4]数据集上取得了较高的识别准确率。

关系网络采用浅层卷积块的设计方式构建了特征提取模块和关系计算模块,通常的推理计算方式采用中央处理器(central processing unit, CPU)或者图形处理器(graphics processing unit, GPU)进行处理[8],使用 CPU 处理速度较慢,使用 GPU 处理能效不高。关系网络以及少样本学习技术[4-8]受限于少量样本在大模型网络的过拟合问题,通常采用浅层卷积块的方式构建特征提取模块和关系计算模块,计算复杂度和模型参数存储量相对较少,适合基于

① 国家自然科学基金(61432016)和国家重点研发计划(2018YFC0832306,2018YFC0831203,2018YFC0831206)资助项目。

② 男,1990 年生,博士生;研究方向:计算机系统结构;联系人,E-mail:zhangzhichao@ict.ac.cn。

(收稿日期:2021-03-30)

现场可编程门阵列(field programmable gate array, FPGA)处理的加速方式。

FPGA 由于其协助通用服务器进行高性能计算在亚马逊云计算数据中心[10]以及百度云计算数据中心[11]已经大规模部署,展现了高效能可重构计算的应用潜力。现有的基于 FPGA 的加速处理卷积神经网络技术大多采用参数量化[12-15]等方式实现卷积神经网络计算加速,量化模型加速的方式通常对深度卷积大模型进行训练微调,而且会降低模型的精度,在浅层网络的量化加速问题上目前也缺少相关研究。基于 FPGA 的浮点模型加速[16-23]采用循环优化、专用计算单元设计等方法,在一些卷积网络加速上取得了较好的计算效能。

针对关系网络推理计算的高效能处理需求,在不降低关系网络推理计算准确率的要求下,采用软硬件协同加速的方式解决处理效能问题。对于关系网络 FPGA 片上计算单元设计采用高级综合(high-level synthesis, HLS)循环优化、异构多核的方式在提升处理能效的同时提升处理吞吐量。

本文的主要贡献如下。

(1)提出了一种软硬件协同加速关系网络推理计算的方法,能够保持关系网络在实际应用中推理精度的同时,相对于 GPU 平台处理达到较高的处理速度与处理效能。

(2)提出一种基于 HLS 高级综合的方式优化卷积神经网络的卷积、池化以及全连接计算,通过设置不同的循环展开粒度,提升计算模块的并行度。

(3)提出了一种基于多核互联的加速器设计方法,避免由于单层卷积核过大造成的 FPGA 大规模资源综合的时序收敛问题,从多核的角度提升了加速器的计算吞吐量。

1　关系网络推理优化方法

1.1　关系网络推理计算流程描述

典型的关系网络推理输入为支持集、测试图片,输出为关系分数,由多个支持集的关系分数最终选出测试图片的类别,同一张测试图片在不同的支持集相关的关系计算下会输出不同的结果。根据支持集的类别数(C-Way)、每类支持集的数量(K-Shot),可以构成不同的 C-Way K-Shot 推理任务。关系网络推理计算包括两个模块的卷积计算部分,分别是特征提取模块和关系计算模块,主要由卷积计算、最大池化计算以及全连接计算组成。特征提取模块用于支持集和测试图片的特征提取,对于 C-Way K-Shot 模式的支持集特征提取,K 为 1 时,输出为单张图片的卷积输出特征图;K 大于 1 时,则是多张图片的卷积输出特征对应元素累加值。经过特征提取模块提取了支持集特征和测试图片特征后,测试图片特征与支持集特征拼接起来构成输入特征图送入关系计算模块,分别计算特征的相似度为输出值,对于 C-Way 的特征值,选取分数最大的作为该测试图片的类别输出。具体的关系网络推理计算流程如图 1 所示。

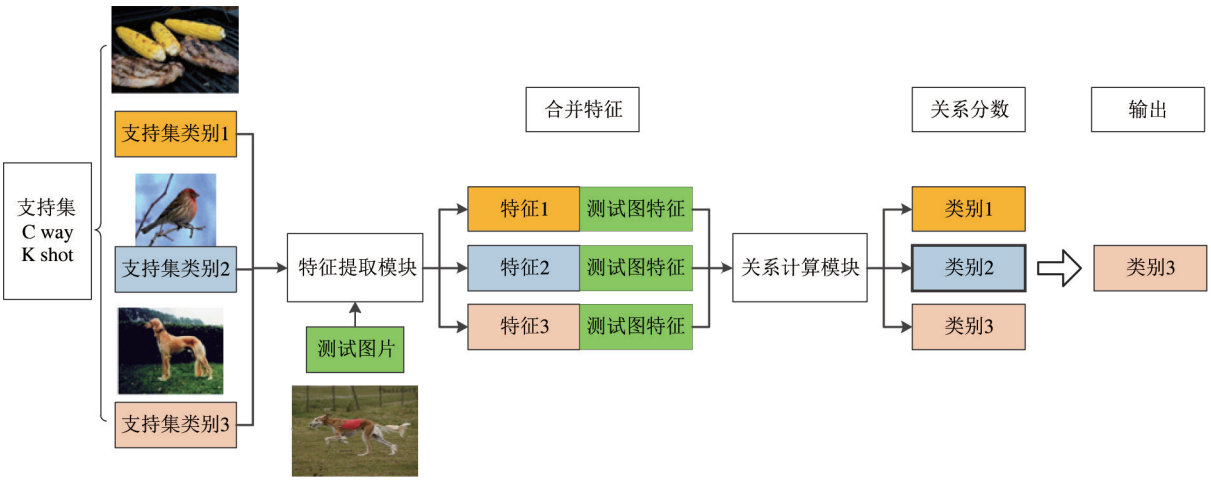

图 1　关系网络推理计算流程

关系网络的推理计算中存在测试图片与多批次支持集之间的关系计算需求,关系分数的准确率以及置信度依赖于支持集数量的大小。多批次的支持集关系计算引入了非常大的计算、存储开销,基于通用的 CPU、GPU 处理的关系网络推理计算速度慢、能耗较高,需要设计一个高效能的推理计算加速器来提升关系网络推理计算的效能。

1.2 基于软硬件协同加速处理的关系网络计算流程

针对关系网络推理计算速度慢、能耗高的问题,本文采用软硬件协同加速计算的方式提升处理速度与处理效能。通过 CPU 或者 GPU 提取可复用的支持集特征供推理计算使用,降低计算开销;采用 HLS 循环优化设计的特征提取模块以及关系计算模块,提升关系网络推理计算的速度与效能;采用异构多核的设计方式综合利用多核的处理能力进一步提升关系网络推理计算速度。

基于关系网络的 C-Way K-Shot 分类推理任务采用 X86/GPU 平台与 FPGA 平台协同计算的方式进行加速计算。X86/GPU 平台用于支持集的特征提取,FPGA 平台用于测试图片的特征提取以及支持集特征下的关系计算。具体的基于软硬件协同加速处理的关系网络计算流程如图 2 所示。

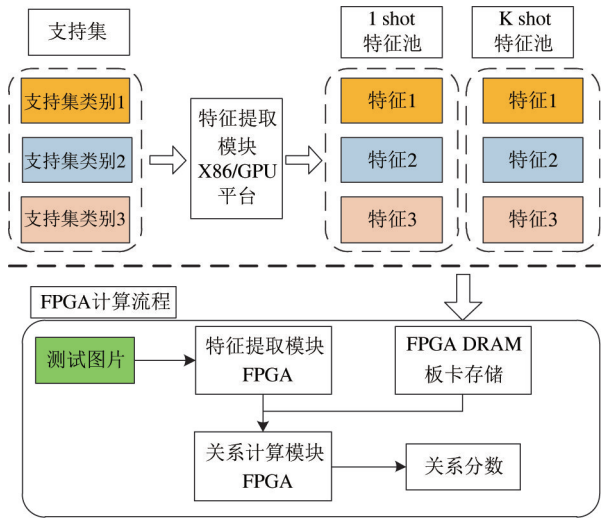

图 2 基于软硬件协同加速处理的关系网络计算流程

通常一个关系网络针对一个任务数据集训练后,可以用于图像分类,甚至对未学习类别的图像分类。图像分类的关键是依赖于支持集的特征作为关系计算的判据输入,不同的支持集会对测试图片的分类结果造成影响,越多的支持集越有助于待测图片的分类准确率提升。

支持集一般是由专家标注的标准数据集,其分类标签是固定的,支持集特征可以利用 X86 或者 GPU 平台提前计算出,支持集特征构成支持集特征池,便于后续计算使用,同时节省计算时间与能耗。对于不同的 C-Way K-Shot 任务,可以构建不同的 K-Shot 特征池,在 Omniglot 数据集下 K 为 1、5 和 20,在 miniImageNet 数据集下,K 为 1 和 5。

FPGA 计算流程采用离线支持集特征直接缓存到动态随机存取存储器(dynamic random access memory, DRAM)的方式节约支持集特征的计算时间与能耗,在片上配置特征提取模块和关系计算模块,根据计算速度的要求和片上资源的约束,构建不同大小的计算模块,多个计算模块配置成异构多核的方式协同加速,进一步提升推理计算能力。

1.3 基于 HLS 循环优化的计算单元设计

关系网络采用多个卷积块、最大池化层以及全连接层构成特征提取模块和关系计算模块。本部分采用基于 HLS 的优化手段对卷积、池化、全连接的计算循环进行展开优化,增加计算模块的并行处理能力,从而提升计算模块的吞吐量。对于特征提取模块和关系计算模块的多层计算采用数据流的优化方式构成一个计算单元,简化计算模块的核外调度处理,使得输入为图像或者特征图,输出为特征图或者关系计算分数。以下阐述卷积、池化以及全连接层的关键设计部分。

(1)卷积乘累加计算

卷积乘累加计算完成输入特征图与权重数据的卷积乘累加计算。卷积乘累加模块的计算输入为输入特征图,维度为 $IH \times IW \times IC$,其中 IH 为输入特征图高度,IW 为输入特征图宽度,IC 为输入特征图通道数;以及权重数据,维度为 $OC \times K^2 \times IC$,其中 OC 为输出特征图通道数,K 为卷积核大小。输出为输出特征图,维度为 $OH \times OW \times OC$,其中 OH 为输出特征图高度,OW 为输出特征图宽度,OC 为输出特征图通道数。

卷积乘累加计算模块包含多个处理单元(process element, PE),每个神经元内部按照单指令

多数据(single instruction multiple data, SIMD)进行并行计算设计。输入特征图调度为 $P \times S$ 宽度的数据,其中 P 为 PE 的个数、S 为 SIMD 的个数。每个 PE 计算的输出为一个神经元的计算输出,PE 内部采取 S 个计算单元并行计算进一步提升并行数,S 个相关的数据完成神经元内部的乘累加计算。卷积的权重调度为 $P \times S$ 宽度的数据,对应相应的特征图输入维度。具体的卷积乘累加计算如算法 1 所示,具有 5 重循环,其中针对 P 和 S 并行展开两重循环,对应的并行数为 $P \times S$, 每个卷积层可以根据计算量的大小以及前后层的计算输入输出速度进行单独的 PE 和 SIMD 配置,在提升处理速度的同时尽可能节约片上资源数量。

算法 1　卷积乘累加计算

输入:输入特征图 In,权重数据 W

输出:输出特征图 Out

```
for h = 0, h < OH, h + + do
for w = 0, w < OW, w + + do
for c = 0, c < OC, c + + do
for pe = 0, pe < P, pe + + do
        //循环展开
for simd = 0, simd < S, simd + + do
        //循环展开
Out[w][h][c/P + pe] + = In[pe][simd] * W[pe][simd]
end for
end for
end for
end for
end for
```

(2) 全连接乘累加计算

全连接乘累加计算完成输入特征图以及权重数据的全连接计算,输出为输出特征图。全连接计算模块的输入特征图、输出特征图跟卷积计算模块保持一致,均为 $H \times W \times C$ 的数据维度,其中 H 为特征图的高度、W 为特征图的宽度、C 为特征图的通道数。全连接计算可以抽象为 $K = 1$ 的卷积计算,具体的全连接乘累加计算算法如算法 2 所示,包含 3 重循环。该算法对 PE 循环和 SIMD 循环进行展开,设计为包含 $P \times S$ 并行度处理模块,每个 PE 计算一个全连接神经元的输出,神经元内部具有 S 个乘累加单元并行计算。每个全连接层实例可以根据前后层的输入输出速度进行特定的 PE、SIMD 数量配置,在满足处理的同时,节约片上资源消耗量。

算法 2　全连接乘累加计算

输入:输入特征图 In,权重数据 W

输出:输出特征图 Out

```
for c = 0, c < OC, c + + do
for pe = 0, pe < P, pe + + do
        //循环展开
for simd = 0, simd < S, simd + + do
        //循环展开
Out[c/P + pe] + = In[pe][simd] * W[pe][simd]
end for
end for
end for
```

(3)最大池化计算

最大池化计算完成输入特征图的池化计算功能,输入输出特征图与卷积层、全连接层保持一致,偏于多层网络组合为数据流调度的计算单元。最大池化计算算法如算法 3 所示,包含 6 重循环,只对最内层循环展开,展开数为 P,即并行度为 P,每个池化层根据计算需要单独配置。其中 $Pool\_Size$ 为池化大小,max 为选取最大值。输入特征图 In 的数据经过输入调度满足伪代码 6 重循环的计算要求。

算法 3　最大池化计算

输入:输入特征图 In

输出:输出特征图 Out

```
for h = 0, h < OH, h + + do
for ph = 0, ph < Pool_size, ph + + do
for w = 0, w < OW, w + + do
for pw = 0, pw < Pool_size, pw + + do
for c = 0, c < C, c + + do
for pe = 0, pe < P, pe + + do
        //循环展开
old = Out[w][h][c/P + pe]
        Out[w][h][c/P + pe] = max(In[pe], old)
    end for
end for
end for
end for
end for
end for
```

1.4 基于异构多核的片上处理系统设计

关系网络的构建由基本的卷积块组成,卷积块由 3×3 的卷积、BatchNorm 和 Relu 计算模块组成,卷积块的输出为 64 个神经元。特征提取模块由 4 个卷积块和 2 个最大池化计算组成,关系计算由 2 个卷积块、2 个最大池化计算以及 2 个全连接计算组成。全连接 1 的维度为 $H\times 8$,全连接 2 的维度为 8×1,最终输出为关系分数。具体的关系网络结构如图 3 所示。对于 Omniglot 和 miniImageNet 2 个不同大小的数据集,其输入图片维度分别为 $28\times 28\times 1$ 和 $84\times 84\times 3$,全连接中的 H 分别为 64 和 576。由于 2 个数据集输入大小以及内部配置的差异,导致具体的各个模块计算量也不同,因此针对 Omniglot 数据集和 miniImageNet 数据集分别设计不同的加速器进行推理加速。

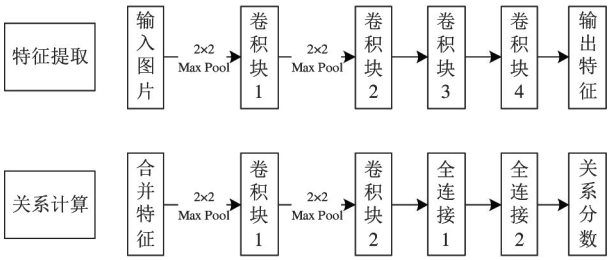

图 3 关系网络结构

Omniglot 28 片上多核设计如图 4 所示,包含控制接口、AXI(advanced extensible interface)互联、DRAM 以及计算模块。计算模块包括特征提取模块以及 4 个关系计算模块。控制接口完成主机的控制指令传输、数据传输功能,对具体的关系计算和特征提取模块进行控制。多个计算模块共享使用一个 DRAM 的 4 GB 内存空间,多个计算模块可以同时并行运行,利用多核能力提供更高的运算吞吐量。

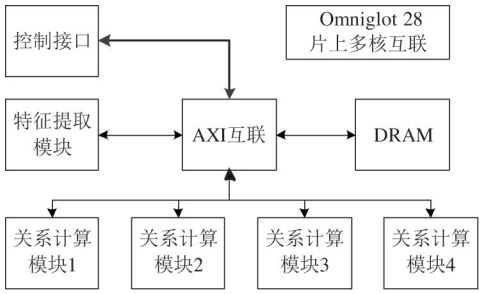

图 4 Omniglot 28 片上多核互联设计

Omniglot 28 加速器配置如表 1 所示,为了描述简便,将特征提取模块和关系计算模块的 P、S 配置放在一个表格进行描述。特征提取模块包括 cnv1 至 cnv4 部分,关系计算模块包括 cnv5 至 fc2 部分。通过设置 P 和 S 的大小可以控制每个模块的并行粒度,进而控制其吞吐量,但是受限于浮点数据的 32 位宽度和 P、S 带来的大线宽、大规模计算矩阵导致其在 Vivado 综合时序难以收敛,故采用多核的方式进一步利用资源提升吞吐量。Omniglot 28加速器的特征提取模块设计速度为2198.39 fps,关系计算模块设计速度为 7430.56 fps,相关吞吐量数据是在 214 MHz 主频下估算得到。

表 1 Omniglot 28 加速器配置

名称	P	S	卷积循环次数	吞吐量/fps
cnv1	4	1	97 344	2198.39
mp1	16			
cnv2	4	16	69 696	3070.48
mp2	8			
cnv3	2	8	57 600	3715.28
cnv4	2	8	57 600	3715.28
cnv5	4	16	28 800	7430.56
mp5	8			
cnv6	1	8	18 432	11610.24
mp6	1			
fc1	1	1	512	417 968.75
fc2	1	1	8	26 750 000.00
特征提取				2198.39
关系计算				7430.56

miniImageNet 84 片上多核互联设计如图 5 所示,包含控制接口、AXI 互联、DRAM 以及计算模块。计算模块包括 1 个特征提取模块和 2 个关系计算模块。

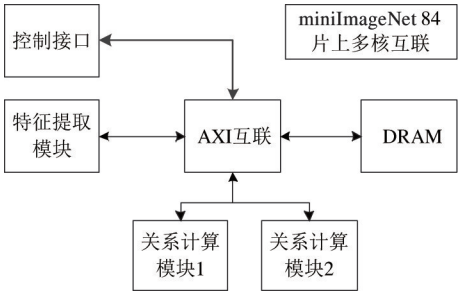

图 5 miniImageNet 84 片上多核互联设计

miniImageNet 84 加速器配置如表 2 所示，为了描述简便，将特征提取模块和关系计算模块的 P、S 配置放在一个表格进行描述。特征提取模块包括 cnv1 至 cnv4 部分，关系计算模块包括 cnv5 至 fc2 部分。通过设置 P 和 S 的大小可以控制每个模块的并行粒度，进而控制其吞吐量。miniImageNet 84 加速器的特征提取模块设计速度为 221.02 fps，关系计算模块设计速度为 642.78 fps，相关吞吐量数据是在 214 MHz 主频下估算得到。

表 2　miniImageNet 84 加速器配置

名称	P	S	卷积循环次数	吞吐量/fps
cnv1	4	3	968 256	221.02
mp1	16			
cnv2	4	16	876 096	244.27
mp2	8			
cnv3	2	8	831 744	257.29
cnv4	2	8	831 744	257.29
cnv5	4	16	332 928	642.78
mp5	4			
cnv6	1	4	331 776	645.01
mp6	9			
fc1	1	9	512	417 968.75
fc2	1	1	8	26 750 000.00
特征提取				221.02
关系计算				642.78

2 实验结果和分析

2.1　基于 HLS 的计算模块资源占用与性能分析

关系网络加速器的 PE、SIMD 配置后，可以进入 HLS 进行仿真分析，从而获得每个模块的具体资源占用情况。实验所用 HLS 为 Vivado _ HLS 2019.1 版本，通过 C 语言综合获得具体的加速器资源使用数据。使用的 FPGA 开发板卡为 VCU 1525 开发板，板载芯片为 VU9P。

(1)基于 HLS 的 Omniglot 28 关系网络推理设计分析

通过 HLS 对 Omniglot 28 加速器进行分析，特征提取模块资源占用如表 3 所示，其中 BRAM 为块随机存储器，DSP 为数字信号处理器，FF 为触发器，LUT 为查找表，URAM 为超级随机存储器。由表 3 可知，片上资源占用各部分不超过 10%，超级逻辑区(super logic region，SLR)资源占用各部分不超过 31%，其中使用最多的资源为数字信号处理(digital signal processing，DSP)资源，由于使用的是浮点乘累加设计的加速器，DSP 资源使用相对多些。关系计算模块资源占用如表 4 所示。由表 4 可知，片上资源占比不超过 7%，SLR 资源占比不超过 22%。数据表明，设计的 Omniglot 28 加速器资源占用量较少，能够顺利地被 Vivado 工具综合实现为 FPGA 比特流。

表 3　Omniglot 28 特征提取模块资源占用表

名称	BRAM	DSP	FF	LUT	URAM
用量	407	708	213 331	118 455	48
片上资源总量	4320	6840	2 364 480	1 182 240	960
SLR 资源总量	1440	2280	788 160	394 080	320
片上资源用量占比	9%	10%	9%	10%	5%
SLR 资源用量占比	28%	31%	27%	30%	15%

表 4　Omniglot 28 关系计算模块资源占用表

名称	BRAM	DSP	FF	LUT	URAM
用量	328	520	177 383	88 751	41
片上资源总量	4320	6840	2 364 480	1 182 240	960
SLR 资源总量	1440	2280	788 160	394 080	320
片上资源用量占比	7%	7%	7%	7%	4%
SLR 资源用量占比	22%	22%	22%	22%	12%

(2)基于 HLS 的 miniImageNet 84 关系网络推理设计分析

通过 HLS 对 miniImageNet 84 加速器进行分析，特征提取模块资源占用如表 5 所示。由表 5 可知，片上资源占用各部分不超过 10%，SLR 资源占用各部分不超过 32%。关系计算模块资源占用如表 6

所示,片上资源占比不超过 8%,SLR 资源占比不超过 24%。数据表明,设计的 miniImageNet 84 加速器资源占用量较少,能够顺利地被 Vivado 工具综合实现为 FPGA 比特流。

表 5　miniImageNet 84 特征提取模块资源占用表

名称	BRAM	DSP	FF	LUT	URAM
用量	438	742	220 683	123 021	48
片上资源总量	4320	6840	2 364 480	1 182 240	960
SLR 资源总量	1440	2280	788 160	394 080	320
片上资源用量占比	10%	10%	9%	10%	5%
SLR 资源用量占比	30%	32%	27%	31%	15%

表 6　miniImageNet 84 关系计算模块资源占用表

名称	BRAM	DSP	FF	LUT	URAM
用量	317	546	196 576	90 738	42
片上资源总量	4320	6840	2 364 480	1 182 240	960
SLR 资源总量	1440	2280	788 160	394 080	320
片上资源用量占比	7%	7%	8%	7%	4%
SLR 资源用量占比	22%	23%	24%	23%	13%

2.2　基于 Vivado 综合后的推理设计分析

实验使用的 Vivado 工具为 2019.1 版本,对加速器的综合实现后获得资源占用及功耗数据。

(1)基于 Vivado 综合的 Omniglot 28 加速器资源占用与功耗分析

基于 Vivado 的 Omniglot 28 加速器资源占用如表7 所示,其中,LUTRAM 为基于查找表的随机存储器,IO 为输入输出,GT 为千兆位收发器,BUFG 为一般时钟缓存,MMCM 为混合模式时钟管理器,PLL 为锁相环。由表 7 可知,各部分资源占用约 40% 以下,能够在 VU9P 芯片上综合加速器处理逻辑。具体的 Omniglot 28 加速器功耗信息如表 8 所示,片上功耗为15.867 W。

(2)基于 Vivado 综合的 miniImageNet 84 加速器资源占用与功耗分析

基于 Vivado 的 miniImageNet 84 加速器资源占用如表 9 所示,其中各部分资源占用约 30% 以下,能够在 VU9P 芯片上综合加速器处理逻辑。具体的 miniImageNet 84 加速器功耗信息如表 10 所示,片上功耗为 15.359W。

表 7　基于 Vivado 的 Omniglot 28 加速器资源占用表

名称	用量	资源总量	占比
LUT	472 635	1 182 240	39.98%
LUTRAM	34 870	591 840	5.89%
FF	917 280	2 364 480	38.79%
BRAM	552.50	2160	25.58%
URAM	212	960	22.08%
DSP	2699	6840	39.46%
IO	144	676	21.30%
GT	1	76	1.32%
BUFG	66	1800	3.67%
MMCM	3	30	10.00%
PLL	3	60	5.00%

表 8　基于 Vivado 的 Omniglot 28 加速器功耗分析

名称	片上功耗	结温
Omniglot 28	15.867 W	33.2 ℃

表 9　基于 Vivado 的 miniImageNet 84 加速器资源占用表

名称	用量	资源总量	占比
LUT	322 426	1 182 240	27.27%
LUTRAM	23 381	591840	3.95%
FF	626 300	2 364 480	26.49%
BRAM	353.50	2160	16.37%
URAM	132	960	13.75%
DSP	1763	6840	25.77%
IO	144	676	21.30%
GT	1	76	1.32%
BUFG	53	1800	2.94%
MMCM	3	30	10.00%
PLL	3	60	5.00%

表 10　基于 Vivado 的 miniImageNet 84 加速器功耗分析

名称	片上功耗	结温
miniImageNet 84	12.359 W	31.4 ℃

2.3　与 FPGA 平台的推理效能对比

本文将关系网络推理加速器的性能与之前的 FPGA 加速浮点卷积神经网络相关工作对比，具体结果如表 11 所示。其中，文献[18，20]的工作采用了基于硬核浮点 DSP 的 Intel Arria 10 器件，吞吐量较高。除了采用浮点硬核加速的相关工作，本文提出算法的运算吞吐量达到了最高值，分别为 Omniglot 28 加速器为 147.79 GFlops，miniImageNet 84 加速器为99.89 GFlops；运算效能也达到了最高值，分别为 9.31 和 8.08 GFlops/W。实验结果表明，本文提出的基于 HLS 循环展开优化的方法取得了较高的加速计算效能。

表 11　FPGA 浮点加速卷积网络相关工作对比

名称	位宽	描述语言	设备	主频/MHz	吞吐量/GFlops	功耗/W	LUT/K	DSP	吞吐量/功耗(GFlops/W)
文献[16]	Float 32	HLS	Virtex7 VX485T	100	61.62	18.61	186	2240	3.31
文献[17]	Float 32	HLS	Virtex7 VX485T	100	75.16	33.93	28	2695	2.22
文献[18]	Float 32	OpenCL	Arria10 GX1150	370	866	41.7	437	1320	20.77
文献[19]	Float 32	OpenCL	Stratix-V GXA7	181	33.9	27.3	-	162	1.24
文献[20]	Float 32	OpenCL	Arria10 GX 1150	270.8	360.4	-	350	1290	-
文献[21]	Float 32	-	ZC706	150	22.74	-	145	596	-
文献[22]	Float 32	Verilog	XCZU9EG	100	42.13	24	124	784	1.75
文献[23]	Float 32	HLS	ZYNQ 7100	100	17.11	4.083	142	1926	4.19
Omniglot 28（本文算法）	Float 32	HLS	VU + VU9P	214	147.79	15.867	472	2699	9.31
miniImageNet 84（本文算法）	Float 32	HLS	VU + VU9P	214	99.89	12.359	322	1763	8.08

2.4　与 GPU 平台的推理效能对比

在 GPU 平台上对关系网络推理进行了性能测试，实验硬件配置为 E5-2650-v4 CPU 以及 Nvidia Titan Xp GPU，软件配置为 Ubuntu14.04，Cuda 版本为 9.0.176，Python 版本为 2.7，Pytorch 版本为 0.3。基于 GPU 平台和 FPGA 平台对关系网络的推理效能进行对比分析，测试数据集包括 Omniglot 数据集和 miniImageNet 数据集，分别对相应的加速器进行吞吐测试。原始的关系网络测试代码用的是多批处理进行测试，批次大小为 2500，每批次包含多个测试图片，用以复用支持集特征减少计算时间，具体的多批次对比测试结果如图 6 所示。在 Omniglot 数据集上实现了 5-Way K-Shot 任务 2198 fps 的吞吐量，20-Way K-Shot 任务 1473 fps 的吞吐量，K 为 1 和 5。在 miniImageNet 数据集上实现了 5-Way K-Shot 任务 220 fps的吞吐量，其中 K 为 1 和 5。数据表明，多批次的吞吐量测试，FPGA 平台优于 GPU 平台。

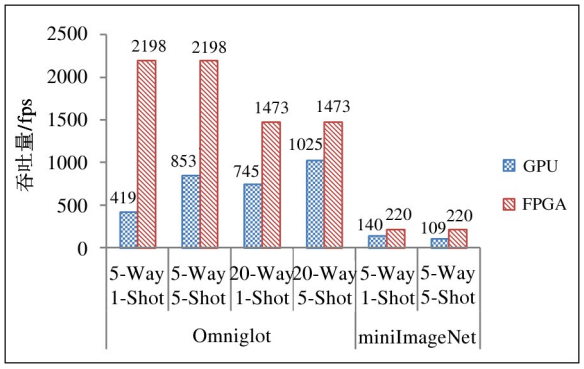

图 6　多批次测试结果对比

考虑推理任务一般为小批次测试,本文进行了单批次吞吐量实验,具体结果如图 7 所示。在 Omniglot 数据集上实现了 5-Way 1-Shot 任务 1779 fps 的吞吐量,5-Way 5-Shot 任务 2101 fps 的吞吐量,20-Way 1-Shot 任务 2075 fps 的吞吐量,20-Way 5-Shot 任务 1482 fps 的吞吐量,K 为 1 和 5。在 miniImageNet 数据集上实现了 5-Way K-Shot 任务220 fps的吞吐量,其中 K 为 1 和 5。单批次的实验 GPU 和 FPGA 吞吐量性能都略有下降,较小的批次数据导致处理器的流水线调度的开销占比偏大,导致了处理效率的低下,但是 FPGA 针对流处理进行了优化,其单批次吞吐量性能优于 GPU 的单批次性能。

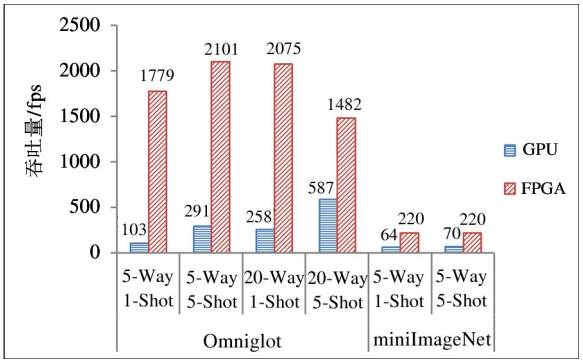

图 7　单批次测试结果对比

根据实验数据,本文对关系网络实现的两个加速器 Omniglot 28 和 miniImageNet 84 与 GPU 平台进行了功耗以及加速比分析,具体结果如表 12 所示。GPU 平台功耗较高,约为 150 W,FPGA 平台的 Omniglot 28 加速器功耗为 15.867 W,多批次加速比为 GPU 的 1.4 ~ 5.24 倍,单批次加速比为 GPU 的 2.5 ~ 17.2 倍;FPGA 平台的 miniImageNet 84 加速器功耗为 12.359 W,多批次加速比为 GPU 的 1.5 ~ 2.0 倍,单批次加速比为 GPU 的 3.1 ~ 3.4 倍。结果表明,相对 GPU 平台的关系网络推理过程,FPGA 加速器达到了低功耗高能效的预期效果,对于小批量的数据集 FPGA 加速器比 GPU 加速器更有性能功耗优势。

表 12　加速器性能功耗对比分析

平台	功耗	多批加速比	单批加速比
GPU	~150 W	1	1
FPGA(本文算法) Omniglot 28	15.867 W	1.4 ~ 5.24	2.5 ~ 17.2
FPGA(本文算法) miniImageNet 84	12.359 W	1.5 ~ 2.0	3.1 ~ 3.4

3　结　论

本文提出的基于软硬件协同加速的关系网络推理优化方法,充分利用了关系网络浅层卷积设计模式带来的计算与存储优势,通过对卷积计算的 HLS 循环优化、异构多核协同处理以及 GPU 预处理支持集特征等方法,降低了软硬件协同开发的耦合度,达到了高效能的关系网络推理计算的目的,同时保持了原有模型的准确率。随着深度学习技术的发展,越来越需要揭开深度学习的黑盒面纱,提升模型的可解释性,降低应用的难度,在少样本学习、元学习以及可解释性学习等方面开展更深入的研究。在研究高精度、少样本适用、可解释的识别算法的同时,体系结构研究者可以跟进研究相关算法的计算效能问题,进一步针对新的少样本学习、元学习等技术展开相关的效能计算研究。

参考文献

[1] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. *Advances in Neural Information Processing Systems*, 2012, 25: 1097-1105
[2] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[EB/OL]. https://arxiv.org/abs/1409.1556:arXiv, (2015-04-10), [2021-03-24]
[3] 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, Las Vegas, USA, 2016: 770-778
[4] VINYALS O, BLUNDELL C, LILLICRAP T, et al. Matching networks for one shot learning[C]//Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 2016: 3637-3645
[5] MUNKHDALAI T, YU H. Meta networks[C]//International Conference on Machine Learning, Sydney, Australia, 2017: 2554-2563
[6] FINN C, ABBEEL P, LEVINE S. Model-agnostic meta-learning for fast adaptation of deep networks[C]//International Conference on Machine Learning, Sydney, Australia, 2017: 1126-1135
[7] SNELL J, SWERSKY K, ZEMEL R. Prototypical networks for few-shot learning[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems, New York, USA, 2017: 4080-4090
[8] SUNG F, YANG Y, ZHANG L, et al. Learning to compare: relation network for few-shot learning[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 1199-

1208
[9] LAKE B, SALAKHUTDINOV R, GROSS J, et al. One shot learning of simple visual concepts[C]// Proceedings of the Annual Meeting of the Cognitive Science Society, Boston, USA, 2011:2568-2573
[10] Amazon. AWS cloud services[EB/OL]. https://aws.amazon.com/: Amazon,[2021-03-24]
[11] Baidu. Baidu FPGA cloud services[EB/OL]. https://cloud.baidu.com/product/fpga.html: Baidu,[2021-03-24]
[12] RASTEGARI M, ORDONEZ V, REDMON J, et al. XNOR-Net: imagenet classification using binary convolutional neural networks[C]// European Conference on Computer Vision, Amsterdam, Netherlands, 2016: 525-542
[13] COURBARIAUX M, HUBARAL I, SOUDRY D, et al. Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1[EB/OL]. https: arxiv.org/abs/1602.02830: arXiv, (2016-03-17),[2021-03-24]
[14] HUBARA I, COURBARIAUX M, SOUDRY D, et al. Quantized neural networks: training neural networks with low precision weights and activations[J]. *The Journal of Machine Learning Research*, 2017, 18(1): 6869-6898
[15] CAI Z, HE X, SUN J, et al. Deep learning with low precision by half-wave Gaussian quantization[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, 2017: 5918-5926
[16] ZHANG C, LI P, SUN G, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks[C]// Proceedings of the 2015 ACM/SIGDA International Symposium on Field-programmable Gate Arrays, Monterey, USA, 2015: 161-170
[17] RAHMAN A, LEE J, CHOI K. Efficient FPGA acceleration of convolutional neural networks using logical-3D compute array[C]// 2016 Design, Automation and Test in Europe Conference and Exhibition, Dresden, Germany, 2016: 1393-1398
[18] ZHANG J, LI J. Improving the performance of OpenCL-based FPGA accelerator for convolutional neural network[C]// Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2017: 25-34
[19] WANG D, AN J, XU K. PipeCNN: an OpenCL-based FPGA accelerator for large-scale convolution neuron networks[EB/OL]. http://arxiv.org/abs/1611.02450: arXiv, (2016-11-08),[2021-03-24]
[20] WEI X, YU C H, ZHANG P, et al. Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs[C]// Proceedings of the 54th Annual Design Automation Conference 2017, Austin, USA, 2017: 1-6
[21] 蹇强, 张培勇, 王雪洁. 一种可配置的 CNN 协加速器的 FPGA 实现方法[J]. 电子学报, 2019, 47(7): 1525-1531
[22] 赵烁, 范军, 何虎. 基于 FPGA 的 CNN 加速 SoC 系统设计[J]. 计算机工程与设计, 2020, 41(4): 939-944
[23] LIU B, ZOU D, FENG L, et al. An FPGA-based CNN accelerator integrating depthwise separable convolution[J]. *Electronics*, 2019, 8(3): 1-18

Relation network inference optimization method based on software and hardware co-acceleration

ZHANG Zhichao[*,**,***,****], WANG Jian[*,**,***], ZHANG Longbing[*,**,***], XIAO Junhua[*,**,***]
([*] State Key Laboratory of Computer Architecture, Institute of Computer Technology, Chinese Academy of Sciences, Beijing 100190)
([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
([***] University of Chinese Academy of Sciences, Beijing 100049)
([****] The 15th Research Institute of China Electronics Technology Group Corporation, Beijing 100083)

Abstract

Aiming at the problem of low efficiency in relation network inference computing based on graphics processing unit (GPU) platform, this paper proposes a relation network optimization method based on software and hardware co-acceleration. In this method, the inference calculation of the relation network is processed by means of heterogeneous collaboration between the feature pool of support set extracted by GPU and the inference of field programmable gate array (FPGA). The inference calculation of the relation network and the GPU platform are maintained with the same accuracy while the calculation is efficient. The processing energy efficiency of the relation network is improved by using the high-level synthesis (HLS) optimized floating point convolutional neural network. The heterogeneous multi-core processing method of multiple computing units is used to satisfy the convergence of FPGA timing sequence and improve the throughput capacity of FPGA chip. In this paper, a relation network inference operation unit is implemented on FPGA platform. The power consumption of the accelerator built on Omniglot dataset is 15.867 W, and the acceleration ratio relative to GPU is 1.4 ~ 17.2. The power consumption of the accelerator built on the miniImagenet dataset is 12.359 W, and the acceleration ratio relative to the GPU is 1.5 ~ 3.4. Compared with similar FPGA accelerated floating-point convolutional neural networks, the proposed method achieves the optimal computational performance. The experimental data show that this method effectively utilizes the advantages of software and hardware collaborative computing and FPGA reconfigurable computation, reduces the coupling degree of software and hardware collaborative development, and improves the computational efficiency of relation network inference while maintaining the accuracy of relation network inference calculation.

Key words: relation network, software and hardware co-acceleration, convolutional neural network, heterogeneous multi-core