

XX论文

|  |  |
| --- | --- |
| 论文题目： | 基于嵌入式FPGA平台的高能效深度学习框架与应用 |
| 学 院： | 信息科学技术学院 |
| 年 级： | 2012级 |
| 专 业： | 计算机科学与技术 |
| 姓 名： | 赵睿哲 |
| 学 号： | 1200012778 |
| 指导教师： | 梁云 |

年 月 日

# 

# 摘要

2012年，基于深度学习的图像识别与分类算法取得了突破性的进展：在当年的ImageNet[[1]](#footnote-1)比赛中遥遥领先。自此，如何利用深度学习和神经网络算法，便成为计算机视觉领域的热门研究方向：基于图像的识别，分类，检索，以及基于动态视频流的SLAM算法等都获得性能与效果上的长足进步。但是这种方法有明显的局限性：传统深度学习算法往往基于大规模集群与GPU架构，而许多计算机视觉算法需要搭载于汽车、机器人乃至无人机等计算资源有限、能耗限制高且开发难度大的嵌入式设备上。因此，如何高效地实现与优化基于嵌入式平台的深度学习算法，对计算机视觉领域的进一步发展至关重要。

本研究实现了SoCaffe —— 一个全新的基于深度学习框架Caffe与嵌入式SoC平台的深度学习框架，并提出基于该框架开发深度学习应用的方法论。硬件平台选用赛灵思（Xilinx）公司推出的“全可编程”嵌入式SoC ZYNQ进行开发：双核ARM处理器与FPGA硬件在效率与能耗上均有优秀表现。开发平台使用Xilinx最新推出的SDSoC套件，极大地提高了开发效率。本研究将广泛应用于学术界和工业界的Caffe框架通过交叉编译部署于ZYNQ，并使用FPGA加速热点函数，相对于原始版本具有平均x倍的加速比。本研究最后选取基于卷积神经网络的立体匹配算法作为案例，以展现平台的计算效率与开发效率。

关键词

深度学习；ZYNQ；Caffe；SDSoC

# **Abstract**

Since when deep learning based image recognition algorithm

**Key words**

Key words;key words; key words(英文关键词内容必须与中文关键词完全对应。英文关键词采用Times New Roman小四号字书写，毕业论文、毕业设计行与行之间、段落和层次标题以及各段落之间均为1.5倍行距。关键词与关键词之间用“;”隔开)

目录

[摘要 I](#_Toc451122189)

[**Abstract** II](#_Toc451122190)

[引言 1](#_Toc451122191)

[第一章 基本原理 1](#_Toc451122192)

[1.1 深度学习 1](#_Toc451122193)

[1.1.1 深度学习算法 1](#_Toc451122194)

[1.1.2 卷积神经网络 1](#_Toc451122195)

[1.1.3 深度学习系统架构 1](#_Toc451122196)

[1.2 可编程逻辑门阵列（FPGA） 2](#_Toc451122197)

[1.2.1 架构概述 2](#_Toc451122198)

[1.2.2 高层次综合（HLS） 2](#_Toc451122199)

[1.3 片上系统（SoC） 2](#_Toc451122200)

[1.4 论文的主要工作 2](#_Toc451122201)

[第二章 总体设计 3](#_Toc451122202)

[2.1 深度学习框架Caffe 3](#_Toc451122203)

[2.1.1 软件架构 3](#_Toc451122204)

[2.1.2 GPU加速设计 3](#_Toc451122205)

[2.1.3 第三方库依赖 3](#_Toc451122206)

[2.1.4 使用方法 3](#_Toc451122207)

[2.2 ZYNQ平台 3](#_Toc451122208)

[2.2.1 系统架构 3](#_Toc451122209)

[2.2.2 ARM处理器 4](#_Toc451122210)

[2.2.3 FPGA硬件 4](#_Toc451122211)

[2.2.4 Zedboard开发板卡 4](#_Toc451122212)

[2.3 SDSoC开发工具 4](#_Toc451122213)

[2.3.1 基本原理与组成 4](#_Toc451122214)

[2.3.2 工作流程 4](#_Toc451122215)

[2.4 综合设计方案 4](#_Toc451122216)

[第三章 主体工作 5](#_Toc451122217)

[3.1 系统架构设计 5](#_Toc451122218)

[3.2 FPGA加速器设计 5](#_Toc451122219)

[3.2.1 GEMM定义 6](#_Toc451122220)

[3.2.2 GEMM的FPGA实现与优化 7](#_Toc451122221)

[3.2.3 GEMM的矩阵分块计算实现与优化 7](#_Toc451122222)

[3.2.4 总结 8](#_Toc451122223)

[3.3 ARM处理系统与数据通路设计 8](#_Toc451122224)

[3.3.1 ARM处理系统设计 8](#_Toc451122225)

[3.3.2 数据通路设计 9](#_Toc451122226)

[3.4 应用：基于卷积神经网络的立体匹配算法实现 9](#_Toc451122227)

[3.4.1 立体匹配算法 9](#_Toc451122228)

[3.4.2 神经网络设计 9](#_Toc451122229)

[3.4.3 前处理与后处理 9](#_Toc451122230)

[第四章 测试结果 10](#_Toc451122231)

[4.1 FPGA加速器测试 10](#_Toc451122232)

[4.1.1 GEMM性能测试 10](#_Toc451122233)

[4.1.2 GEMV性能测试 10](#_Toc451122234)

[4.1.3 综合性能测试 10](#_Toc451122235)

[4.1.4 资源利用 10](#_Toc451122236)

[4.2 Caffe整体性能测试 11](#_Toc451122237)

[4.2.1 单元测试 11](#_Toc451122238)

[4.2.2 样例应用测试 11](#_Toc451122239)

[4.3 立体匹配算法效率测试 11](#_Toc451122240)

[第五章 总结与展望 12](#_Toc451122241)

[5.1 主要工作总结 12](#_Toc451122242)

[5.2 未来优化方向 12](#_Toc451122243)

[结论 13](#_Toc451122244)

[参考文献 14](#_Toc451122245)

[附录一 15](#_Toc451122246)

[致谢 16](#_Toc451122247)

（备注：目录按2～3级标题编写“目录”二字使用黑体小二号字居中书写，隔行书写目录内容，“摘要、Abstract、正文的一级标题、结论、参考文献、附录目录、致谢”采用黑体小四号字书写，正文其他层次标题均采用宋体小四号字书写；“摘要、Abstract”与“正文”之间隔一行；正文中的二级标题、三级标题相对于上一级标题均缩进二个空格书写；目录行与行之间均为1.5倍行距；目录内容多者，正文中的二、三级标题可使用宋体五号字书写，行与行之间可采用单倍行距。目录修改时单击目录点右键选择更新域，选择更新整个目录。之后，在“Abstract”的页码后回车加一空行。）

# 引言

本研究的目的是优化深度学习应用在嵌入式平台上的运行效率与开发效率。传统的深度学习算法的研究方向往往重结果而

近年来，有关深度学习在嵌入式平台上的实现的文献综述

本研究的主要内容是在嵌入式平台ZYNQ上实现优化基于Caffe的深度学习框架， 包含FPGA加速器的设计，Caffe的编译和部署，以及基于深度学习的立体匹配算法的实现案例。

# 第一章 基本原理

正文采用宋体（英语用Times New Roman）小四号字，毕业论文、毕业设计行与行之间、段落和层次标题以及各段落之间均为1.5倍行距。

## 1.1 深度学习

正文采用宋体（英语用Times New Roman）小四号字，毕业论文、毕业设计行与行之间、段落和层次标题以及各段落之间均为1.5倍行距。

具体内容 具体内容 具体内容 具体内容 具体内容 具体内容 具体内容[[[2]](#endnote-1)]。

### 1.1.1 深度学习算法

正文采用宋体（英语用Times New Roman）小四号字，毕业论文、毕业设计行与行之间、段落和层次标题以及各段落之间均为1.5倍行距。

### 1.1.2 卷积神经网络

正文

### 1.1.3 深度学习系统架构

正文

## 1.2 可编程逻辑门阵列（FPGA）

正文

### 1.2.1 架构概述

正文

### 1.2.2 高层次综合（HLS）

正文

## 1.3 片上系统（SoC）

正文

## 1.4 论文的主要工作

正文

# 第二章 总体设计

本章阐述本研究的整体设计，包含对深度学习框架Caffe、Zynq SoC架构、以及SDSoC开发工具的基本介绍。

## 2.1 深度学习框架Caffe

正文

### 2.1.1 软件架构

正文

### 2.1.2 GPU加速设计

正文

### 2.1.3 第三方库依赖

正文

### 2.1.4 使用方法

正文

## 2.2 ZYNQ平台

正文

### 2.2.1 系统架构

正文

### 2.2.2 ARM处理器

正文

### 2.2.3 FPGA硬件

正文

### 2.2.4 Zedboard开发板卡

正文

## 2.3 SDSoC开发工具

正文

### 2.3.1 基本原理与组成

正文

### 2.3.2 工作流程

正文

## 2.4 综合设计方案

正文

# 第三章 主体工作

本研究的主要工作是实现了SoCaffe——基于深度学习框架Caffe并且可以运行在任一Zynq架构的嵌入式SoC设备上的神经网络框架。SoCaffe的实现主要分为三个步骤，首先是整体系统架构设计：即如何充分利用Zynq提供的可编程逻辑（Programmable Logic）和处理系统（Processing System）。确定好哪些部分需要硬件加速之后，第二步是充分优化硬件加速器设计，这部分工作主要利用了高层次综合工具Vivado HLS。最后是对基于ARM CPU运行代码的编译、与硬件加速器的数据通路设计，以及整个系统的生成，通过SDSoC工具可以将这部分工作完美整合起来。

接下来对上述三步实现进行细致描述，在本章最后的给出基于SoCaffe的实现案例：基于卷积神经网络的立体匹配算法实现。该实现体现了SoCaffe的易用性和高效性。

## 3.1 系统架构设计

Zynq平台分为可编程逻辑与处理系统两部分，二者之间通过AXI总线连接。可编程逻辑是FPGA，处理系统则主要是ARM CPU。Caffe的软件架构可以粗略分为三层：最上层是应用层，深度学习应用通过prototxt与caffemodel文件配置网络与数据；中间层是网络层的实现，比如Conv层、tanh层等等；最下层则是各种数学函数和第三方依赖的库文件。基于模块化和分层设计的考虑，本研究主要从底层数学函数出发，将计算耗时长、适合FPGA运行的计算任务放到硬件加速器中实现。

（为什么选择GEMM进行优化？）

综上所述，SoCaffe基于如下的系统架构（如图），Caffe大部分代码与全部第三方库都运行在处理系统上，该部分通过Xilinx提供的GNU工具链进行编译与链接。GEMM函数运行在FPGA上，使用Vivado HLS进行硬件逻辑的生成。最后使用SDSoC工具生成整个系统的镜像文件，复制到SD卡上即可直接运行。

## 3.2 FPGA加速器设计

FPGA加速器中主要包含对GEMM的加速。GEMM加速器设计分为对计算核心的设计和数据通路两部分，其中计算核心主要使用Vivado HLS工具进行实现，重点是优化矩阵乘法计算的效率；数据通路则使用SDSoC工具生成，重点是选择合适的通路类型以及合理安排对计算核心的调用。数据通路的内容在3.3节具体讲述，本节主要概述GEMM的实现。

### 3.2.1 GEMM定义

GEMM是非常常见的线性代数计算函数，BLAS（Basic Linear Algebra Subprograms, 基础线性代数集）[[3]](#footnote-2)标准中有明确的定义。GEMM（GEneral Matrix Multiplication，通用矩阵乘法）的优点在于通用性：矩阵的乘法和加法操作都可以用GEMM定义。GEMM的公式形式为：

其中分别为三个矩阵，根据输入参数决定对是否对进行转置，与分别为两个系数。在数学计算库中（比如CBLAS[[4]](#footnote-3)，MKL[[5]](#footnote-4)等等）会给出GEMM的函数形式，一般表示为：

图 1：GEMM函数接口

[s|d]gemm(TransA, TransB, M, N, K, alpha, A, lda, B, ldb, BETA, C, ldc)

s与d分别表示单精度（single）与双精度（double），TransA与TransB分别指示矩阵A与B是否需要进行转置之后再进行计算。M，N，K分别为矩阵的三个大小参数。lda，ldb，ldc分别为矩阵A，B，C的首维度（leading dimension）大小：用来通过二维坐标在一维矩阵数组中寻址，比如A[i\*lda+j]就是A中的第(i, j)个矩阵元素。有些库对GEMM的定义中还包含对数组中数据排列方式的参数，比如行优先（Row major）和列优先（Column major），这里暂时不考虑。

GEMM的时间复杂度为，由于M，N，K的数值不是固定的，显然无法直接把现有的GEMM实现移植到FPGA上：FPGA上的资源有限，而且硬件设计必须给出矩阵大小的范围。因此本研究提出如下硬件加速方案：在FPGA上放置固定大小的矩阵乘法与加法计算单元，然后基于矩阵块乘法和块转置的原理在CPU上进行矩阵分块和数据传输以完成任意大小矩阵的GEMM计算。接下来分别从FPGA计算单元设计和矩阵分块计算两个层面介绍GEMM加速器的实现。

### 3.2.2 GEMM的FPGA实现与优化

GEMM在FPGA上执行的操作主要是固定大小矩阵的乘法与加法运算，算法简单，但取得高效率却不容易。本节首先定义基于固定大小的矩阵块的GEMM接口，并给出不加优化的实现。接下来在该实现的基础上增加Vivado HLS综合优化指令，并给出原因。相关的对比实验在第四章给出。

基于固定大小的矩阵块的GEMM接口（以下称为gemm\_accel，只给出单精度浮点数的接口）的定义如下：

图 2：gemm\_accel接口定义

**void** gemm\_accel (**float** A[DIM\*DIM], **float** B[DIM\*DIM], **float** C[DIM\*DIM],

**float** ALPHA, **float** BETA);

相对于标准的GEMM接口定义，gemm\_accel接口有如下两个不同点：一是所有的数组定义都包含固定的大小DIM\*DIM，其中DIM为矩阵块的尺寸（行列相同）；二是A，B在传输到gemm\_accel之前已经进行转置处理，因此这里不需要转置条件TransA与TransB。

基于该接口的GEMM实现的最原始版本只是非常简单的三重循环，如下图所示。假设DIM的值为32，通过简单的计算可以知道该函数的延迟为32\*32\*32=32768左右，相应的HLS报告也指出综合后的具体延迟为\_\_。

该原始版本的问题主要在于：

1. 没有使用流水线：因此硬件资源没有被充分利用。如果第二层循环可以被充分流水线化的话，延迟可以理论上降低到1024个时钟周期。
2. 没有使用BRAM缓存：在计算过程中每次对矩阵的读取都需要从输入流中读取，而且同一个元素会读取多次，造成整体计算性能下滑。因此在进行计算之前，使用额外的BRAM来缓存矩阵A和B，可以进一步缩短延迟。

使用流水线与板上BRAM缓存之后，HLS代码与综合结果如下图。可以发现延迟缩短到2352个时钟周期，板上资源利用率也得到很大提升。

相应的性能测试在第四章。

### 3.2.3 GEMM的矩阵分块计算实现与优化

目前已经有针对每个固定大小矩阵块的GEMM计算接口，对于任意大小的矩阵可以在该接口的基础上进行实现。矩阵分块计算的原理依赖如下两个公式：

1. 分块矩阵乘法：

其中，分别为对应三个边上的块的个数。分块矩阵乘法相当于把矩阵块当作矩阵元素进行计算。

1. 分块矩阵转置：  
   首先对每个块进行转置，之后把每个块当作矩阵元素进行整个矩阵的转置。

综上所述，对于任意大小的矩阵可以按照如下图的顺序进行计算：首先按照一般矩阵算法的三重循环对块进行遍历，对遍历到的每个块，根据转置与否把对应的块复制到缓冲区中，最后把缓冲区中的数据传给FPGA进行矩阵块的GEMM计算。

整个计算流程可以划分为如下几步，由此可以看出，在FPGA进行计算的过程中CPU处于空闲的状态。如果把对FPGA运算单元的同步调用变为异步，调用运算单元之后随即进行CPU端的数据复制和处理，整个系统的性能可以进一步提升。相关的异步调用和流水线化操作可以用SDSoC工具提供的编译指令实现，具体实现方式在3.3节详述。使用与不使用流水线的GEMM计算的性能对比测试在第四章给出。

### 3.2.4 总结

## 3.3 ARM处理系统与数据通路设计

正文

### 3.3.1 ARM处理系统设计

正文

### 3.3.2 数据通路设计

正文

## 3.4 应用：基于卷积神经网络的立体匹配算法实现

本研究选用基于卷积神经网络实现的立体匹配算法作为实现案例，以展示SoCaffe的易用性、高效率和实用价值。

立体匹配算法是双目立体视觉中重要的计算步骤：双目立体视觉是通过模拟人类双目感知深度的机制，使用平行放置的两台摄像机获取拍摄场景的深度信息的方法。立体匹配算法需要确定左右两幅图片中像素点的对应关系，方法是像素点之间匹配代价（Matching cost）计算。匹配代价计算以往是用启发式的、计算缓慢的算法实现的。2015年，Jure Žbontar与Yann LeCun使用基于卷积神经网络的深度学习算法[[6]](#endnote-2)优化了匹配代价计算过程（以下简称MC-CNN方法），在效率上和准确率上都有进展，因此将该算法移植到密集使用双目视觉算法的平台至关重要。

将MC-CNN移植到SoCaffe上主要有两方面的工作：一是基于Caffe的标准进行神经网络结构与输入输出的移植；二是对图像前处理和后处理的移植。接下来从算法原理出发，逐步解释向SoCaffe的移植过程。

### 3.4.1 立体匹配算法

正文

### 3.4.2 神经网络设计

正文

### 3.4.3 前处理与后处理

正文

# 第四章 测试结果

正文

## 4.1 FPGA加速器测试

正文

### 4.1.1 GEMM性能测试

正文

### 4.1.2 GEMV性能测试

正文

### 4.1.3 综合性能测试

正文

### 4.1.4 资源利用

正文

## 4.2 Caffe整体性能测试

正文

### 4.2.1 单元测试

正文

### 4.2.2 样例应用测试

正文

## 4.3 立体匹配算法效率测试

正文

# 第五章 总结与展望

正文

## 5.1 主要工作总结

正文

## 5.2 未来优化方向

正文

# 结论

毕业论文的结论是对整个论文主要成果的归纳，应突出论文的创新点，以简练的文字对论文的主要工作进行评价。若不可能得出应有的结论，则需进行必要的讨论。可以在结论或讨论中提出建议、研究设想及尚待解决的问题等。

毕业设计的结论是概括说明设计的情况和价值，分析其优点和特色，有何创新，性能达到何水平，并应指出其中存在的问题和今后改进的方向。

结论要单独成页，“结论”二字采用黑体小二号字居中书写，隔行采用宋体（英语用Times New Roman）小四号字书写具体内容，行与行之间、各段落之间均为1.5倍行距。

# 参考文献

# 附录一

附录序号采用“附录一”、“附录二”、“附录1”、“附录2” 、“附录A”、“附录B”等，用四号黑体字左起顶格排写，其后不加标点符号，空一行书写附录内容。附录内容采用宋体（英语用Times New Roman）小四号字书写。行与行之间、各段落之间均为1.5倍行距。

附录可有可无，当无附录内容时，此页可删除；当附录唯一时，只写“附录”即可。

# 致谢

致谢要单独成页，“致谢”二字采用黑体小二号字居中书写，隔行采用宋体（英语用Times New Roman）小四号字书写具体内容。行与行之间、各段落之间均为1.5倍行距。

1. ImageNet 2012 [↑](#footnote-ref-1)
2. [] 主要责任者.文献题名[J].刊名.出版年份,卷号(期号):起止页码． [↑](#endnote-ref-1)
3. [↑](#footnote-ref-2)
4. [↑](#footnote-ref-3)
5. [↑](#footnote-ref-4)
6. [↑](#endnote-ref-2)