中国科学院大学计算机组成原理实验课

实 验 报 告

学号： 2017K8009929044 姓名： 李昊宸 专业： 计算机科学与技术

实验序号：5 实验名称： 深度学习算法及硬件加速

注 1：请在实验项目个人本地仓库中创建顶层目录 doc。撰写此 Word 格式实验报告后以 PDF 格式保存在 doc 目录下。文件命名规则：学号-prjN.pdf，其中学号中的字母“K”为大写，“-”为英文连字符，“prj”和后缀名“pdf”为小写，“N”为 1 至 5 的阿拉伯数字。例如：2015K8009929000-prj1.pdf。PDF 文件大小应控制在 5MB 以内。

注 2：使用 git add 命令将 doc 目录下的实验报告 PDF 文件添加到本地仓库，然后 git push 推送提交。

注 3：实验报告模板下列条目仅供参考，可包含但不限定如下内容。实验报告中无需重复描述讲义中的实验流程。

一、 逻辑电路结构与仿真波形的截图及说明（比如关键 RTL 代码段{包含注释}及其对应的逻辑电路结构、相应信号的仿真波形和信号变化的说明等）

2D 卷积代码：

```
void convolution() {
    short* in = (short*)addr.rd_addr;
    short* weight = (short*)addr.weight_addr;
    short* out = (short*)addr.wr_addr;

    unsigned output_offset = 0;                                     //d0:
    unsigned input_offset = 0;                                      //d1:
    unsigned weight_offset = 0;                                     //d2: hidth
                                                                    //d3: width
    unsigned input_fm_w = rd_size.d3;
    unsigned input_fm_h = rd_size.d2;

    unsigned pad = KERN_ATTR_CONV_PAD;
    unsigned pad_len = pad << 1;    //Extra length brought by two pads

    unsigned conv_out_w = rd_size.d3 - weight_size.d3 + pad_len;    //weight/rd_size.d3: the width  of weight/input
    unsigned conv_out_h = rd_size.d2 - weight_size.d2 + pad_len;    //weight/rd_size.d2: the height of weight/input

    unsigned stride = KERN_ATTR_CONV_STRIDE;                        //one move

    conv_out_w = div(conv_out_w, stride);
    conv_out_h = div(conv_out_h, stride);

    conv_out_w++;   //Size of Output Pictures
    conv_out_h++;   //using oh = (ih -kh + pad * 2) / stride + 1

    conv_size.d0 = wr_size.d0;  //Number of Input  Pictures
    conv_size.d1 = wr_size.d1;  //Number of Output Pictures
    conv_size.d2 = conv_out_h;  //Height of Output Pictures
    conv_size.d3 = conv_out_w;  //Width  of Output Pictures

    //TODO: Please add your own algorithm implementaion here

    int no, ni, y, x, padding_y, padding_x, ky, kx;
    int temp;
    int  temp_in, temp_w;

    for(no = 0; no < conv_size.d1; ++no)                                                    //number of output pictures
    {
        input_offset = 0;   //reset input_offset

        for(ni = 0; ni < conv_size.d0; ++ni)                                                //number of input pictures
        {
```

```
        for(y = 0; y < conv_size.d2; ++y)   //point(x, y) of output picture
        {
            for(x = 0; x < conv_size.d3; ++x)                                                                   //point (x,y) of output pictures
            {
                padding_y = y * stride;                                                     //padding_y is the y-coordinate in the input pictures(which include padding zone)
                padding_x = x * stride;                                                     //padding_x is the x-coordinate in the input pictures(which include padding zone)
                temp = 0;                                                           //reset temp
                if(ni == 0)                                             //reset output picture(add bias, which is already of short type)
                    out[output_offset + y * conv_size.d3 + x] = weight[weight_offset];          //add bias to the output picture

                for(ky = 0; ky < weight_size.d2; ++ky)
                {
                    for(kx = 0; kx < weight_size.d3; ++kx)                                                      //point(kx,ky) of weight map
                    {
                        if(padding_x + kx >= pad && padding_x + kx < input_fm_w + pad && padding_y + ky >= pad && padding_y + ky < input_fm_h + pad)//not in the range of padding zone
                            temp_in = (short int)in[input_offset + (padding_y + ky - pad) * input_fm_w + (padding_x + kx - pad)];    //temp_in is the data of  point(padding_x+kx-pad, padding_y+ky-pad)
of input pictures
                        else
                            temp_in = 0;    //point(padding_x+kx, padding_y+ky) is in padding zone

                        temp_w = (short int)(weight[weight_offset + ky * weight_size.d3 + kx + 1]);  //Filter[oc][0][0] is bias, thus plus 1 to skip it
                        temp += (int)(temp_in * temp_w);
                    }
                }
                out[output_offset + y * conv_size.d3 + x] += (short)(temp >> FRAC_BIT);  //MUL make the last 20 bits are decimal, only get first 10 of them,shift left 10 bits

                //calculate one point
            }
                                                        //complete one row
         }                                          //complete one input picture
        input_offset += input_fm_h * input_fm_w;                                                                //offset to the next input && weight
        weight_offset += weight_size.d2 * weight_size.d3 + 1;                                                   //the size of the third field of Filter is 1+K*K

    }                                                                                                           //handle ont output picture
    output_offset += conv_size.d2 * conv_size.d3;                                                               //offset to next output picture
}


}
```

池化代码：

```
    void pooling() {
        short* out = (short*)addr.wr_addr;

        unsigned output_offset = 0;
        unsigned input_offset = 0;

        unsigned input_fm_w = conv_size.d3;
        unsigned input_fm_h = conv_size.d2;

        unsigned pad = KERN_ATTR_POOL_PAD;
        unsigned pad_len = pad << 1;

        unsigned pad_w_test = conv_size.d3 - KERN_ATTR_POOL_KERN_SIZE;
        unsigned pad_h_test = conv_size.d2 - KERN_ATTR_POOL_KERN_SIZE;

        unsigned pool_out_w = pad_w_test + pad_len;
        unsigned pool_out_h = pad_h_test + pad_len;

        unsigned stride = KERN_ATTR_POOL_STRIDE;

        unsigned pad_w_test_remain = pad_w_test - mul(div(pad_w_test, stride), stride);
        unsigned pad_h_test_remain = pad_h_test - mul(div(pad_h_test, stride), stride);

        pool_out_w = div(pool_out_w, stride);
        pool_out_h = div(pool_out_h, stride);
        pool_out_w++;
        pool_out_h++;

        if ( (!pad) && (pad_w_test_remain || pad_h_test_remain) )
        {
            pool_out_w++;
            pool_out_h++;
        }

        //TODO: Please add your own algorithm implementaion here
        int no, y, x, oy, ox, i, j;
        int maxium; //oy, ox: work on output from convolution
        int temp;

        for(no = 0; no < conv_size.d1; ++no)                                    //number of output pictures
        {


            for(y = 0; y < pool_out_h; ++y)                                     //point(x, y) of pooling output
            {
                for(x = 0; x < pool_out_w; ++x)
```

```
        {
            oy = y * stride;                                            //pooling stride
            ox = x * stride;
            maxium = -2147483648;   //INT_MIN

            for(j = 0; j < stride; ++j)                                 //point(ox+i,oy+j) of pooling area
            {
                for(i = 0; i < stride; ++i)
                {
                    if(ox + i >= pad && ox + i < input_fm_w + pad && oy + j >= pad && oy + j < input_fm_h + pad)  //if point(ox+i,oy+j) is not in the padding
                        temp = (short)out[input_offset + (oy + j - pad) * input_fm_w + (ox + i - pad)];
                    else
                        temp = 0;                                                                                //point(ix+kx, iy+ky) is in padding zone

                    if(temp > maxium)
                        maxium = temp;
                }
            }
            out[output_offset + y * pool_out_w + x] = maxium;

        }

    }
    input_offset += input_fm_h * input_fm_w;  //Change into another input, which is the output of convolution

    output_offset += pool_out_h * pool_out_w;  //Change into another output
}


}
```

硬件加速器驱动：

```
#include "printf.h"
#include "trap.h"
#include "perf_cnt.h"

#define HW_ACC_START    0x0000
#define HW_ACC_DONE             0x0008


int main()
{
        //TODO: Please add your own software to control hardware accelerator
        unsigned long *base = (void *)0x40040000;
        unsigned long val;
        //unsigned long val1;
        volatile unsigned long *val1 = (void *)0x40040008;
        unsigned long val2;

        Result res;

        bench_prepare(&res);

        printf("starting convolution\n");
        val = *base&0xfffffffe;
        *base = val+1;

        while(1)
        {
           // val1 = *(base + HW_ACC_DONE/4);
           // val1 = val1&0x00000001;
            val2 = *val1&0x00000001;

            //printf("%d\n",val1);
           // if(val1 > 0)
            if(val2>0)
            break;

        }

        bench_done(&res);

        printf("Cycles of sw: %u\n", res.msec);
        printf("Memory visit times is: %u\n",res.mem_cycle);

        return 0;
}
```

二、 实验过程中遇到的问题、对问题的思考过程及解决方法（比如 RTL 代码中出现的逻辑 bug，仿真、本地上板及云平台调试过程中的难点等）

数据处理：由于自己写好的 mips 处理器不支持浮点运算，故需要用定点数表示浮点数。

用 short int 16 位定点数表示浮点数，其中低 10 位表示小数部分。中间变量用 32 位，这样在做完乘法之后，低 20 位为小数部分，高 16 位为整数部分。其中最高位为符号位。

最后处理掉溢出部分：先将中间变量右移 10 位，还原小数部分；然后取其低 16 位作为结果。

volatile 关键词：

在书写 hw_conv 时，使用的变量 val1 在循环内部读取 0x40040008 地址的最低位，判断硬件加速器是否做完卷积计算。发现如下问题：

1） 不使用 volatile 关键词定义指针型变量时，编译器会直接将对 val1 定义语句优化掉，导致 val1 的值始终不会更新，从而循环不会停止。

2） volatile 关键词只能对指针型变量使用，对普通变量无效。

三、 对于此次实验的心得、感受和建议（比如实验是否过于简单或复杂，是否缺少了某些你认为重要的信息或参考资料，对实验项目的建议，对提供帮助的同学的感谢，以及其他想与任课老师交流的内容等）

总体来说实验在充分理解之后不难书写，但是充分理解这个过程缺少有效的查阅资料，导致理解阶段较为困难。感谢徐逸斌同学的帮助。