My LaTeX Template

pastglory*

目录

*sunyata000@hotmail.com

摘 要

这是一段摘要, 这个仓库主要保存我的 LaTeX 模版，用于各种文档的书写，目前实现的功能较少，有待日后在使用中不断优化。

深度神经网络已经发展成为解决传统机器学习任务最流行的技术，然而因为存在计算量大，参数量多的问题，将深度神经网络部署到硬件资源有限的嵌入式设备上非常困难。由于深度神经网络的规模随着应用场景的复杂化而增大，为了使其能部署到硬件资源有限的嵌入式设备上，网络压缩是一种有效的方法。在应用一个网络时需要先进行训练，得到合适的参数后将其部署到硬件设备上进行推理。训练是从大量数据中学习正确的参数的一个过程，而推理是使用训练好的参数对新的输入进行运算得到预测结果的过程。通常训练时需要大量的参数，但训练完成后便不需要如此多的参数，因此可以在训练完成后，部署到硬件设备前对网络进行压缩。网络压缩主要分为剪枝和数据量化，剪枝是指剪去网络中不重要的参数，数据量化是指减少表示权重的数据位宽。网络压缩后，不仅减小了进行推断所需的计算量，同时也减小了存储量和读写次数，如果压缩效果比较好，那么原本需要存储在 DRAM 中的数据就可以存储在 SRAM 中，进一步提高了访存效率。

第一章 简介

你好，LaTeX! 这个仓库主要保存我的 LaTeX 模版，用于各种文档的书写。为了实现自由扩展的需求，一切格式上的改动都放在 cls 文件中，并且所有实质性内容都放在 src 文件夹下，main.tex 只用于整理，作为顶层。

为了测试参考文献格式是否正确，使用一篇稀疏运算加速的论文[1] 以及一篇老化预测的论文[2] 作为参考文献样例。

深度神经网络已经发展成为解决传统机器学习任务最流行的技术，然而因为存在计算量大，参数量多的问题，将深度神经网络部署到硬件资源有限的嵌入式设备上非常困难。由于深度神经网络的规模随着应用场景的复杂化而增大，为了使其能部署到硬件资源有限的嵌入式设备上，网络压缩是一种有效的方法。在应用一个网络时需要先进行训练，得到合适的参数后将其部署到硬件设备上进行推理。训练是从大量数据中学习正确的参数的一个过程，而推理是使用训练好的参数对新的输入进行运算得到预测结果的过程。通常训练时需要大量的参数，但训练完成后便不需要如此多的参数，因此可以在训练完成后，部署到硬件设备前对网络进行压缩。网络压缩主要分为剪枝和数据量化，剪枝是指剪去网络中不重要的参数，数据量化是指减少表示权重的数据位宽。网络压缩后，不仅减小了进行推断所需的计算量，同时也减小了存储量和读写次数，如果压缩效果比较好，那么原本需要存储在 DRAM 中的数据就可以存储在 SRAM 中，进一步提高了访存效率。

第二章 测试

2.1 结构

图1为 FPGA 基本单元结构图。

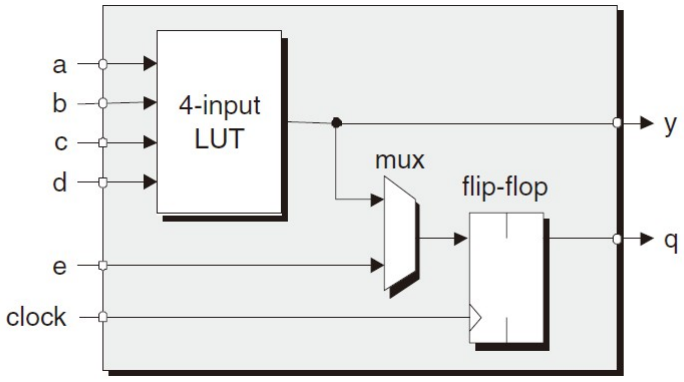

图 1: Slice of FPGA

接下来是代码风格的描述。

2.2 数学

这里有一个数学公式

$$\int_1^2 x dx = \frac{3}{2} \tag{2-1}$$

2.2.1 数学公式说明

这是一个积分

2.3 代码

推荐的 verilog 代码风格如下所示。推荐的 verilog 代码风格如下所示。推荐的 verilog 代码风格如下所示。

```
module keyboard(
    input            clk,
    input            rst_n,
    input  [3 : 0]   col,  // column input, 1 enable
    output [3 : 0]   key_pulse // if key pushed, output 1
);

reg  [11 : 0] treg0;  // tmp reg
reg  [11 : 0] treg1;  // tmp reg
reg  [11 : 0] treg2;  // tmp reg
reg  [11 : 0] treg3;  // tmp reg
wire [11 : 0] treg0_nxt = treg0 + 1'b1;  // tmp reg next val
wire [11 : 0] treg1_nxt = treg1 + 1'b1;  // tmp reg next val
wire [11 : 0] treg2_nxt = treg2 + 1'b1;  // tmp reg next val
wire [11 : 0] treg3_nxt = treg3 + 1'b1;  // tmp reg next val

// when tregx = 12'hffe and tregx_nxt = 12'hfff
// key_pulse will be 1
// then, tregx will be 12'hfff
// when tregx == 12'hfff, it will keep until keyboard not pushed
always @ (posedge clk or negedge rst_n) begin
    if (~rst_n) begin
```

```
        treg0 <= 12'b0;
        treg1 <= 12'b0;
        treg2 <= 12'b0;
        treg3 <= 12'b0;
    end
    else begin
        if (col[0]) begin
            if (treg0 != 12'hfff)
                treg0 <= treg0_nxt;
        end
        else begin
            treg0 <= 12'b0;
        end

        if (col[1]) begin
            if (treg1 != 12'hfff)
                treg1 <= treg1_nxt;
        end
        else begin
            treg1 <= 12'b0;
        end

        if (col[2]) begin
            if (treg2 != 12'hfff)
                treg2 <= treg2_nxt;
        end
        else begin
            treg2 <= 12'b0;
        end

        if (col[3]) begin
            if (treg3 != 12'hfff)
                treg3 <= treg3_nxt;
```

```
            end
            else begin
                treg3 <= 12'b0;
            end
        end
    end

assign key_pulse[3] = (treg3  !=  12'hfff )  & (treg3_nxt == 12'hfff );
assign key_pulse[2] = (treg2  !=  12'hfff )  & (treg2_nxt == 12'hfff );
assign key_pulse[1] = (treg1  !=  12'hfff )  & (treg1_nxt == 12'hfff );
assign key_pulse[0] = (treg0  !=  12'hfff )  & (treg0_nxt == 12'hfff );

endmodule
```

第三章 总结

本文针对神经网络在嵌入式系统上部署时资源受限的问题，使用基于绝对值的方法对网络进行了剪枝，并以压缩格式进行存储，同时使用 C 语言对算法过程进行建模。为了验证剪枝后网络的运行效率，本文基于 Cortex-M3 处理器核搭建了片上系统，为优化剪枝后网络的运行效率而设计了四个用于推断过程加速的加速器单元通过总线矩阵接入系统，整个片上系统使用 FPGA 实现。最终通过编写软件程序，对比实验得出剪枝后使用稀疏加速器比剪枝后使用纯软件的运行效率提高了约 30%，通过对剪枝后的网络使用压缩存储方式有效地节省了存储空间。

参考文献

[1] Z. Zhang, H. Wang, S. Han, and W. J. Dally, “Sparch: Efficient architecture for sparse matrix multiplication,” in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 261–274.

[2] M. Sadi, G. K. Contreras, J. Chen, L. Winemberg, and M. Tehranipoor, “Design of reliable socs with bist hardware and machine learning,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 11, pp. 3237–3250, 2017.