SEAI_2024_R12

Giulio Capecchi, Jacopo Niccolai

December 2024

1 Introduction

This project focuses on the implementation of a **Multilayer Perceptron (MLP)**, a **Convolutional Neural Network (ConvNet)**, and a **Transformer**. The main objective was to synthesize the forward pass of these networks on an FPGA. To achieve this, the parameters of the networks were first extracted using Python and *PyTorch*. These parameters were then hardcoded into `C` code, enabling hardware synthesis.

2 Project Description

2.1 Parameter Extraction

The neural networks were built and trained using *PyTorch*. The weights and biases were exported in a format compatible with the `C` implementation.

2.2 C Implementation

The `C` code developed includes the forward pass for:

- **MLP**: implementation of propagation through dense layers.
- **ConvNet**: handling of convolution and pooling operations.
- **Transformer**: managing complex operations like attention.

The network parameters (weights and biases) were directly integrated into the code in a hardcoded manner.

3 Code Architecture

3.1 C File Structure

The forward pass is implemented using a sequence of functions for each layer type:

- Activation functions (`relu`, `softmax`, etc.).
- Functions for convolution and pooling operations.
- Functions for attention mechanisms in Transformers.

Example of code for the forward pass:

```
float relu(float x) {
    return x > 0 ? x : 0;
}

void forward_layer(float input[], float output[], ...) {
    // Implementation
}
```

3.2 Hardware Synthesis

The code was designed to be compatible with tools such as Vitis HLS, leveraging specific pragmas to optimize the implementation.

4 Results

The networks implemented in hardware were tested and compared with their software versions. The FPGA synthesis demonstrated advantages in terms of performance and power consumption.

5 Conclusions