EE 271: Final Project Inference Engine on FPGA

Jay Pankaj Patel

March 17, 2025

Contents

1 Building Blocks

For our inference implementation, I am considering using Convolutional Neural Networks (CNNs). They are an algorithm used to recognize patterns in data. I believe the target data should be handwritten digits from 0–9. Neural networks are built using a collection of neurons organized into layers, each with their own weights and biases. The building blocks of CNNs are the following:

- **Tensor** – can be thought of as an n-dimensional matrix. We need to decide how big to make our tensor. I was considering making it smaller, like three dimensions, because our goal is to perform simple classification.
- **Neuron** – is a mathematical function that takes multiple inputs and gives a single output.
- **Layer** – is simply defined as a collection of neurons.
- **Kernel Weights and Biases** – unique to each neuron and are determined during the training phase of our network. We will train the model on a computer and extract the weights and biases to the FPGA for inference.
- **Differentiable Score Function** – represented as class scores on the output layer.

These are the high-level components needed; let's go over what they mean and the equations associated with them.

2 Convolutional Module Design: Neuron

Let's define some terms before we start:

2.1 Background

- **What is a feature?**
  A feature is an individual measurable property that a model uses to make a prediction or classification.
- **What is a feature map?**
  A feature map is the output of all the prevalent features extracted using a filter. In this field, the 'filter' is referred to as a *kernel*. Hence, I will be using "kernel" from this point forward.
- **What is convolution?**
  Convolution, in a mathematical sense, is an operation performed on two functions that produces a third function. The mathematical operation is defined as the integral of the two functions after one of them is reflected about the y-axis and shifted. Mathematically:

2.2 Equations

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

Graphically:

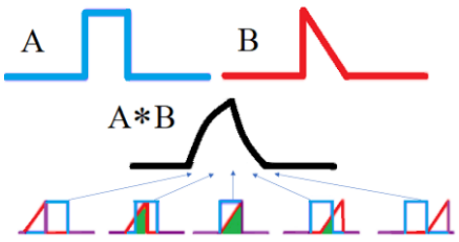

Figure 1: Convolution of two functions A (red) and B (blue) producing a third function describing the overlap (green)

However, this is true for continuous cases. In our case, we are dealing with discrete systems, so the convolution of discrete systems is defined differently. The core definition remains the same; however, the mathematical operation is a summation instead of an integral, as shown:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

In practicality, since our system has a finite response, there exists a support. A support is defined as a range where our function is nonzero. Hence, the following equation can be rewritten as:

$$\boxed{(f * g)[n] = \sum_{m=-M}^{M} f[n-m]\, g[m]}$$

where g has a finite support in the set $\{-M, -M+1, \ldots, M-1, M\}$.

In CNNs, the most computationally expensive operation is the convolution layer. It is used to extract image features. The convolution of the input feature map with the weight kernel results in a feature map of this layer after processing by the activation function [**?**]. The equation is given by:

$$\boxed{y'_j = f\left(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l\right)}$$

- y_j^l : Feature map of the jth convolution kernel result of the lth layer.
- M_j : Represents the selection of the previous input feature map by the current convolution.
- x_i^{l-1}: Represents the previous input feature map.
- k_{ij}^l: Represents the ith weighting coefficient of the jth convolution of the lth layer.
- b_j^l : Represents the bias parameter of the jth convolution kernel of the lth layer.
- f : The nonlinear activation function.

This formula defines a neuron in neural networks, specifically a convolution-based neuron used in CNNs. The weighting coefficients and biases come from the output of a trained model. We will use a pretrained model's weights and biases in our convolution calculations. The weights and biases are given in floating point representation and must be converted to fixed-point. The activation function will most likely be ReLU; however, it may change in the future. Looking at the required computation for each neuron, we can see that this is essentially a multiply-accumulate (MAC) operation. Using this knowledge, we can design a circuit to compute the convolution of each layer efficiently in hardware.