Hardware Implementations of Hyperbolic Tanh

Your Name, Your Professor Name

Your University Name, Department

your_email@gmail.com,

Abstract

The fundamental differences between linear and nonlinear functions in machine learning which is a subset of artificial intelligence is to apply activation functions. Activation functions are used mostly in neural network, due to have a better distinguish and prediction. In some condition it is required to enhance the performance of neural networks algorithms. In some activation functions such as identity and binary step the implementation is designed for digital devices by using fixed point arithmetic which can increase the speed and decrease the hardware resources and therefore it will be energy efficient. The hyperbolic tangent is non-linear functions because contains exponential and division operations so it is not possible to implement in fixed point solutions and it is the dilemma which should be solved for having high performance in neural networks. Some of the most popular approximations are Look-up Tables, piecewise linear, range addressable Look-up Tables and cordic.

Introduction

This article describes that how to compute the hyperbolic tangent of a given real-valued which are using hardware-efficient in variant models. These models are proper for HDL code generation for fixed-point inputs. The algorithm utilizes an architecture that contributes computational and memory units across different steps and iterations, which is useful when deploying to FPGA devices with limited resources. This kind of implementation has a smaller material or items passing through a system than a fully pipelined implementation, but it also has a smaller on-chip footprint, making it appropriate for resource-conscious designs.

However, it is possible to apply fully or piecewise linear functions such as identity and binary step, which is compatible properly with the hardware implementation. The artificial neural networks are mostly deployed on the suitable computer or microprocessor system in floating point arithmetic, for the training and testing. This

computers are enough as long as their power and memory can satisfy our requirements as soon as we need faster systems due to our huge data or heavy computational and operations, we need to implement ANN with the aid of other technologies to gain real-time processing.

Related Works

Fixed-point design methodology of the hyperbolic tangent function using lookup tables

with performance evaluation on FPGA [1]:

VLSI Implementation of Neural Networks [2]:
This paper demonstrated a practical solution to improve two

A NEW DESIGN OF TANGENT HYPERBOLIC FUNCTION GENERATOR WITH APPLICATION TO THE NEURAL NETWORK IMPLEMENTATIONS [3]

Automatic general-purpose neural hardware generator

Background

Artificial Neural Network

GCNN tries to gather information from the

Activation Function

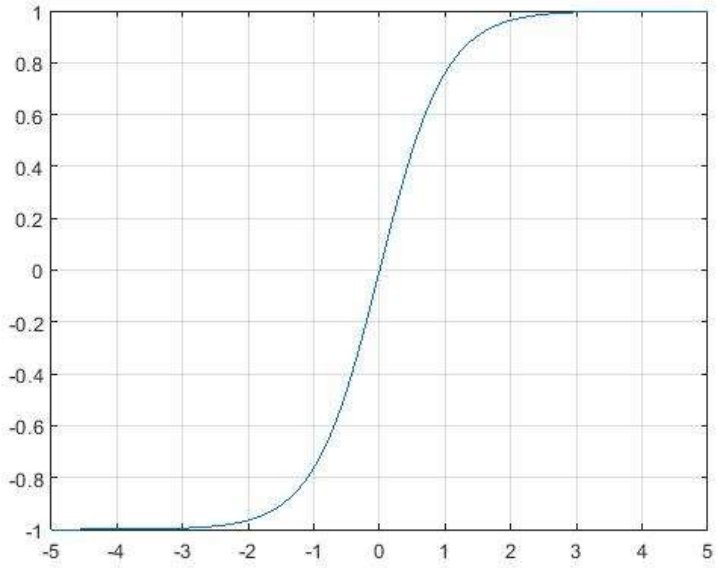
Adam is

Hyperbolic Tangent

Formula:

Tangent hyperbolic curve is S-shaped which varies between -1 and 1.

```
x = -5:0.01:5;
y = tanh(x);
plot(x,y)
grid on
```



$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{2x} - 1}{e^{2x} + 1} \qquad \tanh(x) = -i\tan(ix)$$

FPGA Architecture:

Figure1: FPGA Architecture. []

Look up Table
piecewise linear
range addresable lut
cordic

Methodology

Preprocessing

Fixed-Points Design

Data Generation

Evaluation

Experiments

ID	Memory (bits)	Feasible	Table Size	Intermediate WLs	TableData WL	Error(Max,Current)
0	32768	1	[16 256]	[4 8]	8	6.250000e-02, 3.902460e-03
1	28672	1	[16 256]	[4 8]	7	6.250000e-02, 7.811287e-03
2	24576	1	[16 256]	[4 8]	6	6.250000e-02, 1.561990e-02
3	16384	1	[16 128]	[4 7]	8	6.250000e-02, 6.242016e-02
4	14336	1	[16 128]	[4 7]	7	6.250000e-02, 5.707978e-02
5	12288	1	[16 128]	[4 7]	6	6.250000e-02, 5.870371e-02
6	10240	0	[16 128]	[4 7]	5	6.250000e-02, 8.585766e-02
7	8192	0	[16 128]	[4 7]	4	6.250000e-02, 1.020576e-01

Best Solution

ID	Memory (bits)	Feasible	Table Size	Intermediate WLs	TableData WL	Error(Max,Current)
5	12288	1	[16 128]	[4 7]	6	6.250000e-02, 5.870371e-02

Table 1

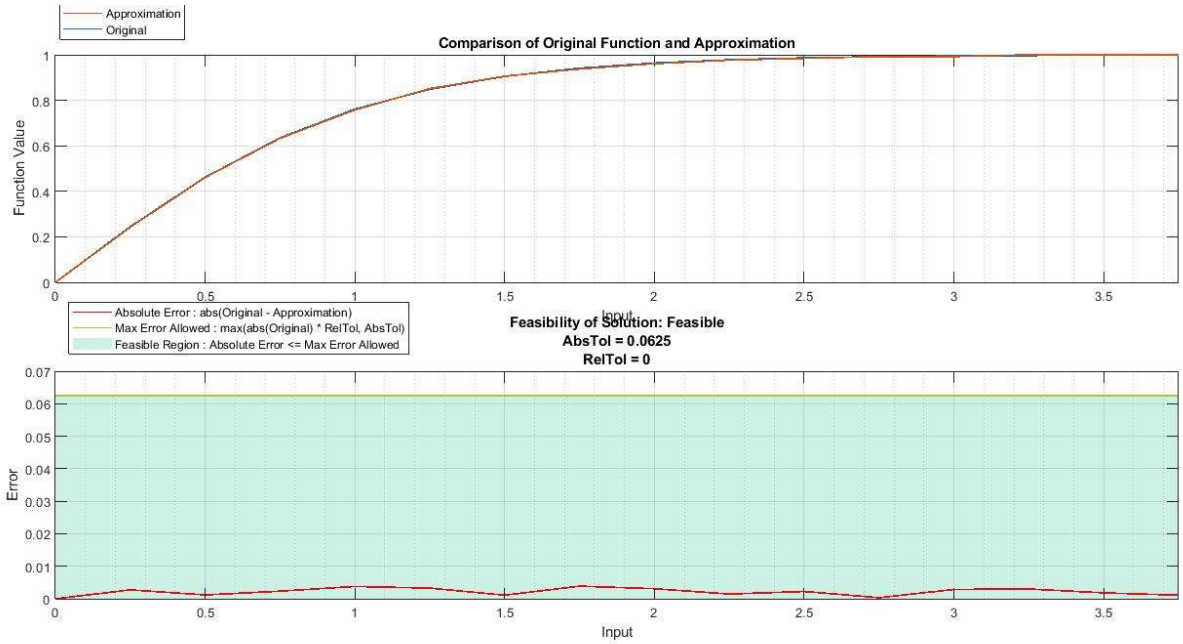


Figure 1:

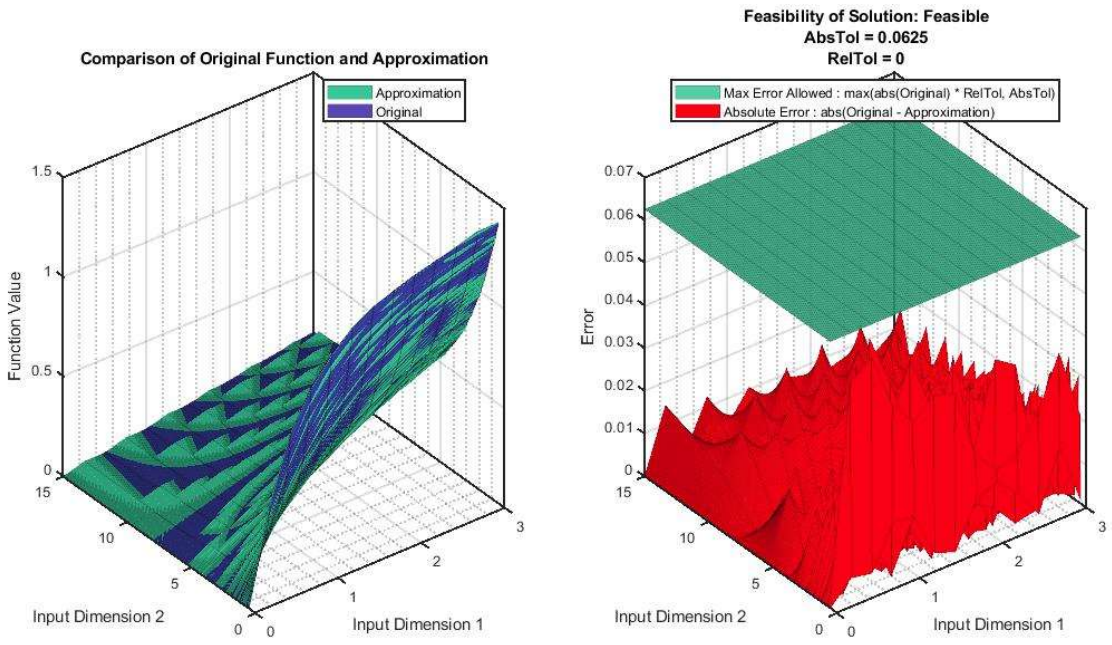


Figure 2

Conclusion and Future Research

References

[1]