

BF16 Performance Evaluation for solving Differential Equations using Neural Network

As the implementation of AI increases exponentially, the use of neural networks to solve the differential equation has recently got a lot of attention [1][2]. In this white paper, we will explore the usage of BF16 number format for training the neural network to solve the differential equations and show that we can achieve a similar level of accuracy with it compared to the FP32 number system.

Introduction:

BF16 Number Format:

BF16 is a 16-bit number system with 8 bits precision and 8-bit exponent. That means it has the same dynamic range of FP32 but with truncated precision bits as shown in Fig 1 (a) & (b). That makes the easy conversion to and from FP32. This format, BF16 (BFloat16, Brain Float 16) has already found wide applications in special hardware accelerators developed by Google, Intel, ARM, etc. [3].

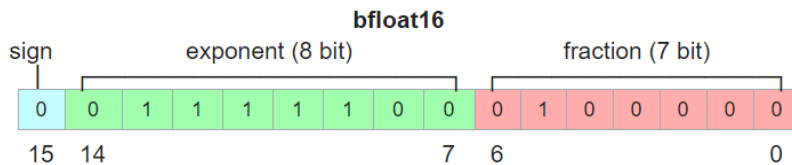


Fig 1 (a) BF16 representation [4]

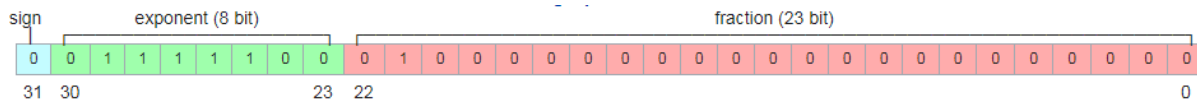


Fig 1 (b) FP32 representation [4]

FP16 has also got a lot of attention in recent times. However, BF16 offers several more advantages than FP16 apart from dynamic range. For example, blocks of hardware multiplier require less chip area for BF16 than FP16. In fact, it is 8 times less compared to FP32 and twice less compared to FP16 [5].

According to the Cybenko theorem [6], neural networks are also function approximators. That means we can use it as an analysis tool to perform classical mathematical problems. As the loss function of a neural network provides a measure of performance of optimization, we can use it to calculate the solution of $F(x) = 0$ by minimizing the loss function of NN. [7] [8] describes this process of function minimization.

In this white-paper, we have four different categories of differential equations. We perform training, testing & metric loss, residue, and solution plot comparison for BF16 with FP32.

As per the neural network, there are three hidden layers for each of these problems with different numbers of neurons ranging from 20 to 50. For activation, "tanh" function is used with "Glorot Uniform" as an initializer.

In this paper, the left figure denotes FP32 system whereas the right one denotes BF16 system.

1)Time Independent Problems: Poisson Equation:

Poisson's equation is an elliptic partial differential equation of broad utility theoretical physics. For example, the solution to Poisson's equation is the potential field caused by a given electric charge or mass density distribution; with the potential field known, one can then calculate electrostatic or gravitational (force) field [9]. This equation is given by,

$$-\Delta u(x, y) = 1, \quad (x, y) \in \Omega, \quad u(x, y) = 0, \quad (x, y) \in \partial\Omega$$

Where, domain is L-shaped given by,

$$\Omega = [-1, 1]^2 \setminus [0, 1]^2$$

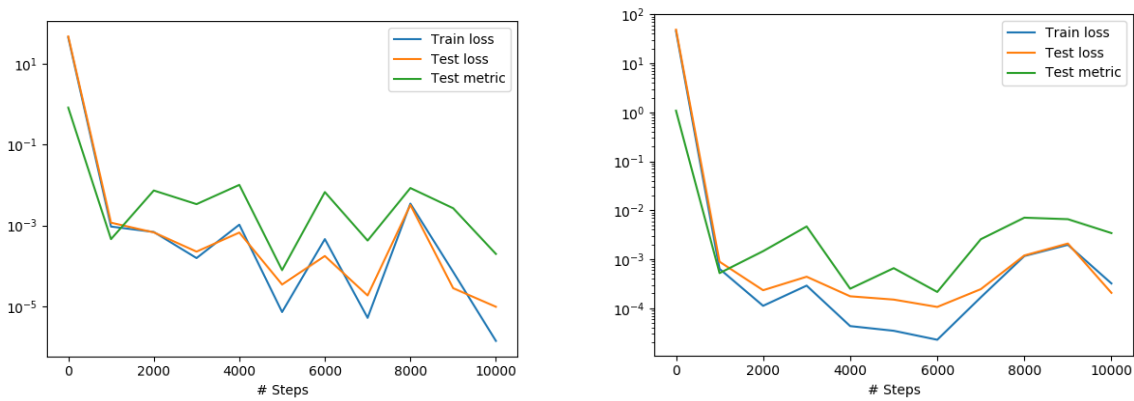


Fig 2. loss plot (FP32 in left: BF16 in right)

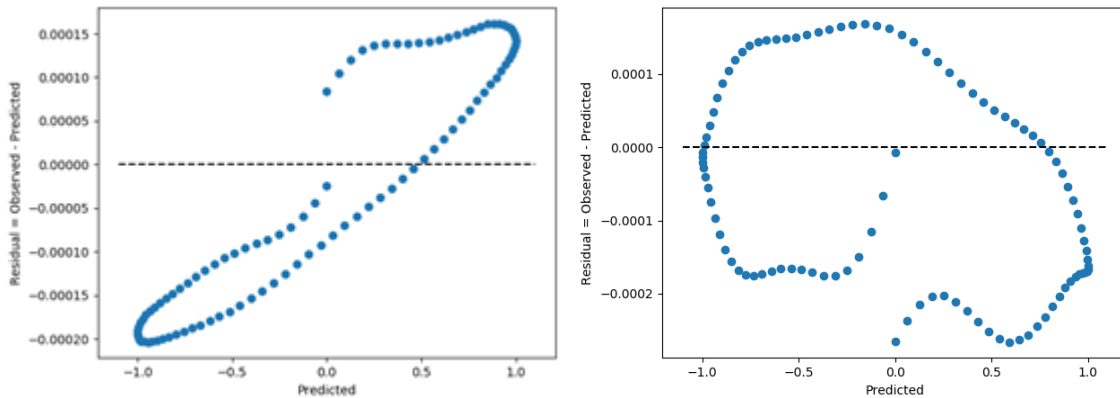


Fig 3. Residual Plot (FP32 in left: BF16 in right)

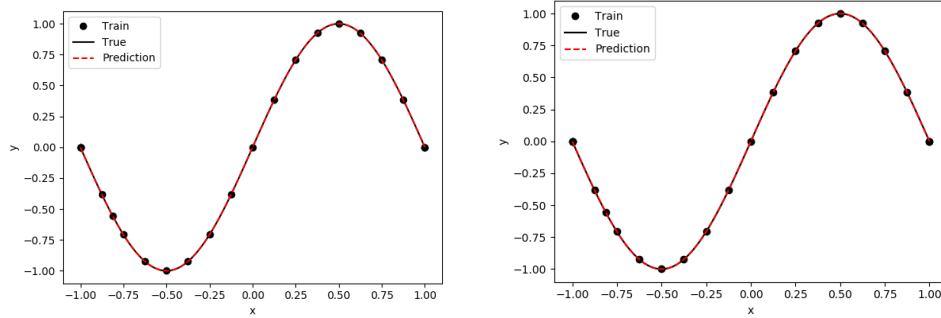


Fig 4. Solution Plot (FP32 in left: BF16 in right)

As we can see, we get similar results for both formats. Accuracy drop is comparable for both. Residue remaining is not too different and the solution looks the same. However minimum loss is less for FP32 from Fig.2.

2)Time-dependent problem: Diffusion equation.

The diffusion equation is a parabolic partial differential equation. In physics, it describes the macroscopic behavior of many micro-particles in Brownian motion, resulting from the random movements and collisions of the particles [10]. This equation is given by

$$\frac{\partial u}{\partial t} = \frac{\partial^2 x}{\partial x^2} - e^{-t}(1 - \pi^2) \sin(\pi x), \quad x \in [-1, 1], t \in [0, 1]$$

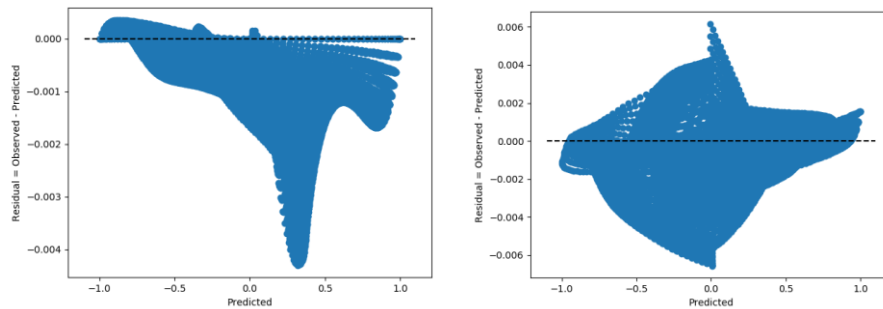


Fig 5. Residual Plot (FP32 in left: BF16 in right)

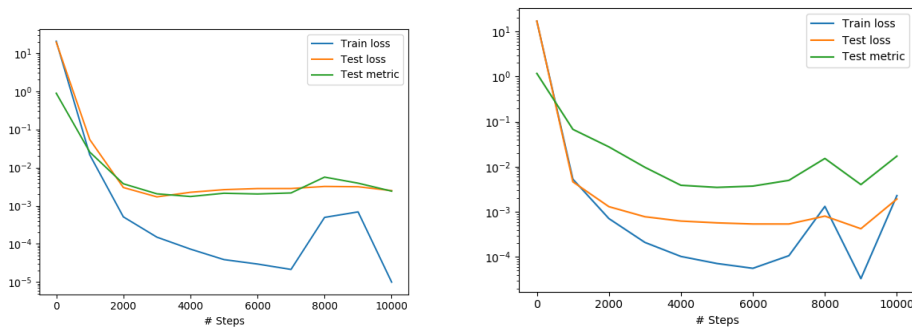


Fig 6. loss plot (FP32 in left: BF16 in right)

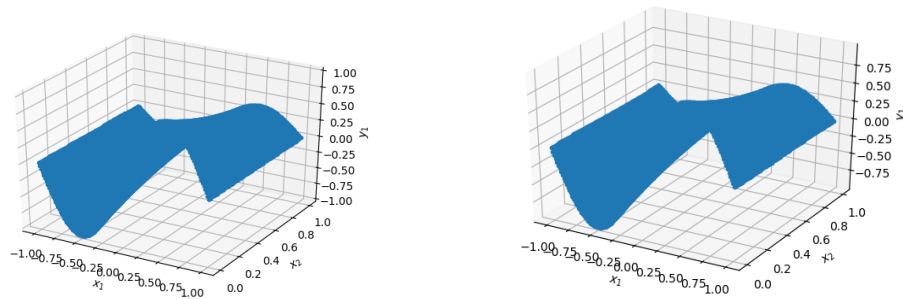


Fig 7. Solution Plot (FP32 in left: BF16 in right)

Again, we have similar performance results though the best model at minimum accuracy drop is slightly better for FP32.

3)ODE/PDE system: Lorentz system.

The Lorentz system is a system of ode having chaotic solutions for certain parameters and initial values resembling the butterfly effect. Meaning small changes in initial condition in the equation leads to increasingly dramatic solutions [11]. Its equation is given by:

$$\frac{dx}{dt} = \rho(y - x), \quad \frac{dy}{dt} = x(\sigma - z) - y, \quad \frac{dz}{dt} = xy - \beta z$$

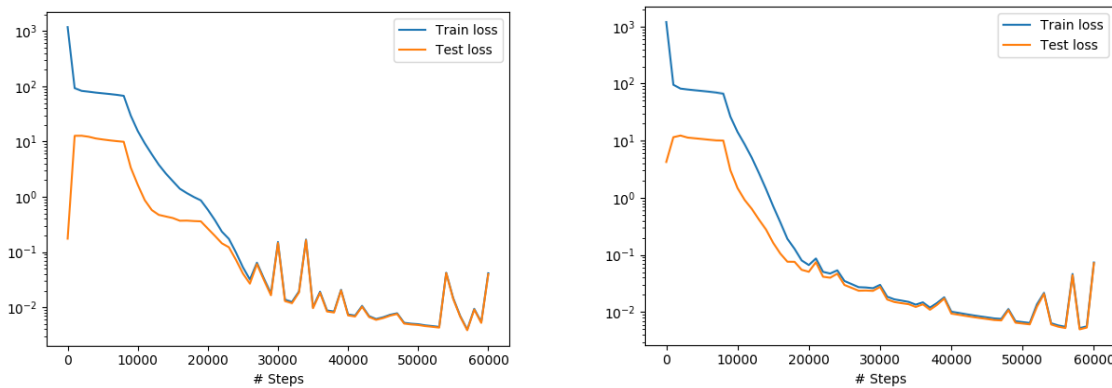


Fig 8. loss plot (FP32 in left: BF16 in right)

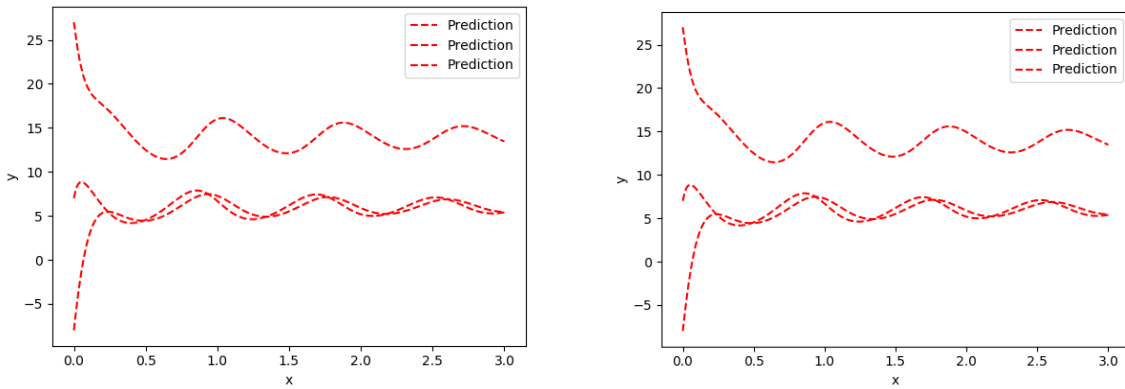


Fig 9. Solution Plot (FP32 in left: BF16 in right)

4) Integro-Differential Equations: Volterra IDE

Volterra integro differential equation is a hybrid of integral and differential equations that is used to model heat and mass diffusion processes, biological species coexisting together with increasing and decreasing rate of growth, electromagnetic theory, and ocean circulations [12]. Its equation is given by:

$$\frac{dy}{dx} + y(x) = \int_0^x e^{t-x} y(t) dt, \quad y(0) = 1$$

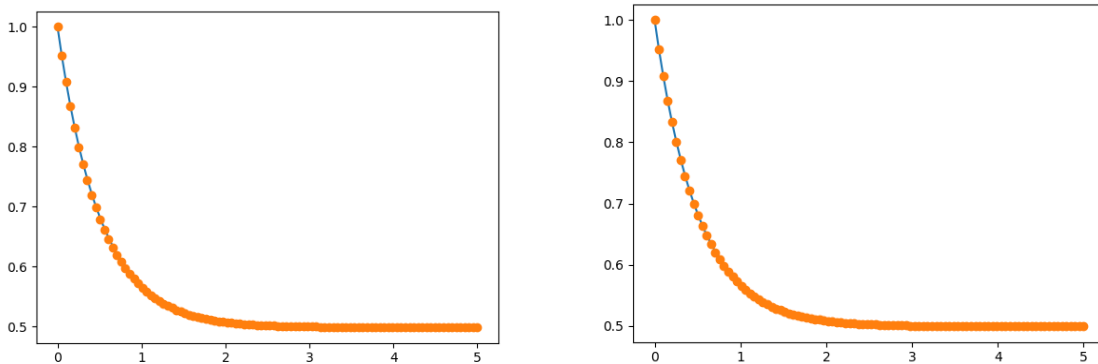


Fig 10. Solution Plot (FP32 in left: BF16 in right)

Conclusion:

Here we can see we have similar levels of loss between both BF16 and FP32 precision however FP32 has slightly better best-model at minimum loss. Partially this is also due to the fact that operations for both formats are operating on the same precision in hardware level. Further research with specialized hardware for BF16 and FP32 will be required for conclusive evidence. In the next white-paper, we'll demonstrate the performance and resource consumption analysis of BF16 and FP32 arithmetic with hardware implementation in HDL.

In this demonstration, we've used an Intel i5-6200U processor being a small problem definition. For network definition and differential equation definition TensorFlow [13] and deepxde libraries [14] are used along with NumPy.

References:

- [1] . Leveraging the bfloat16 Artificial Intelligence Datatype for Higher-Precision Computations
<https://ieeexplore.ieee.org/abstract/document/8877427>
- [2] . Artificial Neural Network for ordinary differential Equations:
<https://arxiv.org/pdf/physics/9705023.pdf>
- [3] . Artificial Intelligence got its own number system:
<https://freenews.live/artificial-intelligence-got-its-own-number-system/>
- [4] . Bfloat16 number format:
https://en.wikipedia.org/wiki/Bfloat16_floating-point_format
- [5] Secret to high Performance on cloud TPUs:
<https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus>
- [6] . Universal Approximation Theorem:
https://en.wikipedia.org/wiki/Universal_approximation_theorem
- [7]. Neural Network based Approximators for Solving Partial Differential Equations.
<https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1640100303>
- [8].Artificial Neural Network for Solving Ordinary and Partial Differential Equations.
<https://ieeexplore.ieee.org/abstract/document/712178>
- [9] . Poisson Equation:
https://en.wikipedia.org/wiki/Poisson%27s_equation
- [10] . Diffusion Equation:
https://en.wikipedia.org/wiki/Diffusion_equation
- [11] . Lorenz System:
https://en.wikipedia.org/wiki/Lorenz_system
- [12] . Solution to first order Volterra Ide:
<https://www.hindawi.com/journals/jam/2017/1510267/>
- [13]. Tensorflow Libraries
<https://github.com/tensorflow/tensorflow>
- [14]. A deep learning Library for Solving Differential Equations
<https://arxiv.org/abs/1907.04502>, <https://github.com/lululxvi/deepxde>

Revision History

The following table shows the revision history of this white paper-WPL061.

Date	Version	Details
July 11 th , 2020	v1.0	Initial Release

About LogicTronix

LogicTronix provide turnkey solutions to customers on accelerating Machine Learning algorithms for various applications including ADAS, Surveillance, Computer Vision, etc.

If you are interested in pruning the machine learning models to meet the performance requirements or to deploy a custom machine learning model using Xilinx DPU contact us at our email.