International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys.13, No.1 (May-23)

http://dx.doi.org/10.12785/ijcds/130199

A Posit based Handwritten Digits Recognition System

Nitish Kumar[1] **and Dr. Vrinda Gupta**[2]

[1]*School of VLSI and Embedded System Design, National Institute of Technology Kurukshetra, Kurukshetra, India*
[2]*Department of Electronics and Communication Engineering, National Institute of Technology Kurukshetra, Kurukshetra, India*

Received 10 May. 2022, Revised 1 May. 2023, Accepted 14 May. 2023, Published 30 May. 2023

Abstract: The automated handwritten digit recognition system has numerous applications. It is required to be performed for address interpretation in postal services, bank cheque processing, or digitization of paper documents. But, for computers to recognize the handwritten numeral images is a challenging task. Various techniques have been utilized for this purpose, like convolutional neural networks architectures. This paper presents a novel design of a Posit-based handwritten digits recognition system, one of the convolutional neural network applications. Posit, a universal number system is a substitute of floating point arithmetic format and is hardware friendly. Herein, LeNet and ResNet-18 based HDRS (Handwritten Digits Recognition System) architecture is used for training and inference of model. The parameters obtained after training were converted to (8,0) Posit number system. Training of LeNet and ResNet-18 based HDRS has been done over the MNIST database, an open-source database for handwritten digits recognition. The proposed Posit (8, 0) based HDRS provides comparable accuracy to traditional floating point and fixed point based HDRS.

Keywords: HDRS(Handwritten Digits Recognition System), POSIT, MNIST Database, Neural networks.

1. INTRODUCTION

During the current scenario, almost everyone uses visual patterns to communicate figures and facts , and it is also common to bring out useful data from them. One of the difficult jobs in the relatively new domain of pattern recognition is the computerized identification of human writing with high accuracy because handwriting differs from person to person [1][2]. Training and inference of LeNet and ResNet-18 based Handwritten Digits Recognition System (HDRS) is used to resolve the accuracy problem because the variance of handwriting does not cause any problem to human beings. Still, teaching computers to recognize common handwriting is difficult [1][3].

Deep learning and machine learning require lots of mathematical calculation during training and inference; deep understanding incorporates different layers. Each layer has its weights and biases and complex linear algebra operations. The colossal complexity of the algebraic process in deep learning affects the computation time. It introduces memory problems when single-precision and double-precision floating-point based modules are implemented on the hardware. The algorithm part optimizes the memory and accuracy problem of deep learning. In the past decade, a fixed point-based model was implemented to replace the single-precision floating-point number system with less precision, but it deteriorated the accuracy during the inference of the model [4][5].

In 2017, Posit number system was introduced by John-Gustafson [6] system which is a category-III variant of universal numbers system (Unum). The Posit number system is a better substitute for IEEE754 standard floating-point and fixed-point arithematic format. . It provides better vital span , tapered accuracy, and parameterized precision superior todeep learning and machine learning application. When implemented on the hardware, it can also reducememory utilization for applying deep learning and machine learning [6][7][8].

The paper focuses on the implementation of (8,0) and (16,1) posit-based handwritten digits recognition system that involves training and inference of LeNet and ResNet-18 (HDRS) model, conversion of obtained different layers parameters from 32-bit floating point to 8-bit and 16-bit posit number system. The outcomes are compared to models based on IEEE 754 floating-point and fixed-point number systems. Training of the model is done on a 32-bit floating-point number system, and its inference is made on a posit (8, 0) number system. Training data is collected from the MNIST database, an open-source database for handwritten digit recognition. The collected information is trained using the tensor-flow-2.0 and PyTorch framework with a jupyter notebook as an editor tool.

The rest of the article is structured as given herein. Section 2 presents the associated efforts, and Section 3 focuses on the related framework study. Section 4 presents

E-mail address: nitishkr8862@gmail.com, vrindag16@nitkkr.ac.in

the implementation of the work. Section 5 discusses the obtained result of the work, and the article is summarized in Section 6, which explains the conclusion, future work, and references related to the work.

2. RELATEDWORK

In the previous decade, single and double-precision floating-point and fixed-point number systems have been used for the training and inference to apply machine learning and deep learning. The work proposed in 2017 by S.Hashemi et al. explains the use of single-precision floating-point and fixed-point number systems by performing deep neural network inference on the LeNet, ConvNet, and Alexnet deep neural networks using 32-bit fixed-point and 32-bit floating-point. It was found that using a fixed-point number system reduced energy consumption by 12% and improved accuracy by less than 1% [9].

Zhejiang Liu et al. in 2016 suggested an HDL-based convolution neural network (CNN) model that used a 16-bit fixed-point for input data and an 18-bit fixed-point for output data. This suggestion [10] focused on prevention of data flooding during the calculation of task. However, fixed-point arithmetic faulted, minimizing calculation loss. In 2020, Jinze Li et al. put forward a handwritten digit recognition system using the convolutional neural network. It introduced an offline detection system built on convolutional neural networks for handwritten digits. This system's implementation can significantly reduce labour costs and increase productivity, which is critical in many disciplines [11].

In 2021, Pu et al. proposed a pool-based algorithm to select data using the least confidence method and entropy sample model. It helps to bring down the training duration and the cost of manual labeling [12]. Li et al. suggested an algorithm to implement the HDRS model, which is further migrated to android. It introduces the concept of multilayer perceptrons in the HDRS model. It introduced the Gradient diffusion problem when the number of hidden layers increases [13]. Whereas, Liu et al. proposed an HDRS model based on BP neural network. It uses input data processing and training of the model. It achieved 85.88 accuracy [14].

In 2022, Faghihi et al. proposed an explainable attribute abstraction technique for handwriting digit classification dependent on keeping image particulars on single neurons as non-synaptic memory. It achieves 75% accuracy for 0.016% of training data and 85% overall accuracy for the MNIST dataset using one epoch [15]. In 2020, Ali et al. suggested a Java-based deeplearning4j framework and a CNN architecture that helps to improve the accuracy problem [16]. Besides this, in 2021, Agrawal et al. suggested another deep-learning technique for handwritten digit classification. It attains 99.06%of training accuracy and 98.08% test accuracy by adding a convolutional layer with pooling and dropout [17].

Chakraborty et al. initiated a handwritten digit string recognition method. It uses a deep auto-encoder based segmentation technique and ResNet architecture to acquire the machine-encoded digit string [18]. Chen et al. proposed a multilayer self-organizing impulse neural network structure. The hidden layer assumes the LIF model. It achieves 92.8% recognition accuracy [19]. Koster et al. in 2014 submitted a Flexpoint data pattern that completely replaces the 32-bit floating-point data format used for training and inference of deep neural-based models [20]. Posit arithmetic is also used in weather and climate models. In 2019, Milan Klowry et al. suggested posits as a substitute to floating for environmental models. It uses a software emulator for testing posit arithmetic for weather and climate simulation [21].

Bryan J Moore et al. in 2020 proposed a peculiar deep-learning proceed towards using a convolutional neural network to design output neural spike activity from input neural spike activity. The proposed model can achieve a high interconnection linking the speculated and actual probability of spiking in the output neuron for data generated with a generalized linear model. It also shows that the kernels accustomed in a synthetic GLM of spike transformation interconnecting layers of neurons can be learned by CNN [22].

Different works have also been reported to analyze the results obtained after implementing deep learning and machine learning applications by replacing floating and fixed-point number systems with the posit number system. It found that posit provides a more comprehensive dynamic range that minimizes the system memory uses. In 2020, N. Buoncristiani et al. presented a healthy learning between 32-bit Posit and 32-bit IEEE 754-2008 depiction. They proposed a conceptual investigation for the two number representations for single-bit flips and double-bit flips and experiments were carried out on machine learning applications. They concluded that posit demonstrates higher robustness than IEEE 754 representations [23].

An arithmetic unit was designed by J. Johnson in 2018, which is used for merging posit addition with logarithmic multiplication for CNN inferences. The author designed a posit-based deep neural network accelerator to represent the weights and activation and combine it with an FPGA softcore for 8-bit posit exact multiply and accumulate (MAC) operation. This model's training is based on the floating-point number, but the inference was made in the low-precision POSIT format. It also depicts that ImageNet categorization using the ResNet-50 deep neural network architecture can be done with less than 1% accuracy loss and that using 8/38-bit Exact log-linear multiply-add (ELMA) instead of an 8/32-bit integer multiply-add and an IEEE-754 float16 fused multiply-add can save around 4% and 41% of power, respectively [24].

Posit arithmetic is also used in self-governing driving practices. M. Cococcioni et al. discussed how to accelerate deep neural network computing in an automotive application. For that, a Posit Processing Unit (PPU) was designed, which is the second to a Floating-point Processing Unit (FPU). These 16-bit posits replace conventional FPU because self-driving cars need 16-bit floating-point depiction for security demanding applications. PPU also saves energy because it uses less number of transistors than FPU [25].

In 2019, De Dinechin et al. explained how the posit is suitable for the machine learning application because activation functions scale their convolutions and other neuron summations. The statical count in each plane is little for the total to hold on in the golden region. It discusses where posits are superior than floating-point and where it is inferior [26]. While, Raul Murillo et al. described an algorithm for multiplying two Posit numbers and integrating them into the FloPoCo framework. Posit is analyzed for inference and training stages of neural networks, and the results while working with the MNIST dataset are promising compared to Floating-Point [27].

Beside this, Yohan Uguen et al. proposed work to analyze the hardware cost for implementing the posit number system. It discusses the operational value of translating a floating-point application to a posit number system. It was found that Posit hardware is overpriced than floating-point hardware, but from an application point of view, posit is more accurate than the floating-point number system for the same size [28].

In 2020, Murillo et al. proposed a DeepPeNSieve structure for training and inference of deep neural networks (DNNs) applications using the Posit number system. It uses an 8-bit Posit and fused operation for the low-precision inference. It is used for training with posit (32, 2) and Posit (16, 1) and carry out post-quantization to Posit (8, 0) with the assistance of the quire and the fused dot product. It provides more than 4% improvements in training compared to the floating-point number system [29].In 2021; G. Raposo et al. proposed a new DNN framework (PositNN) that helps training and inference for any posit precision. It uses quire for accumulation. It found that the DNN framework provides a similar result as float [30].

3. BACKGROUND

A. *MNIST Database*

Modified National Institute of Standards and Technology (MNIST) is an open-source directory generally handled for training and testing in machine learning. It was formed by ”re-mixing” the NIST’s actual files [1][10]. It consists of sixty thousand training data sets and ten thousand test data sets. The sample of the MNIST dataset is shown in fig 1.

B. *Floating Point Number System*

High precision data representation is required to accurately capture sample data, which is fulfilled using a floating-point number system. The standardizing authority, IEEE introduced the technical representation for floating-point arithmetic called IEEE 754. It was revised in 2008 to IEEE 754-2008 [31], later superseded by IEEE-754 2019 [31]. The floating-point number representation standard IEEE 754 is used to represent real numbers effectively. A standard floating-point number representation has four parts. i.e, Precision, Sign bits, Exponent, and mantissa. Based on precision, a floating-point number is categorized into two types: single precision and double precision [31].

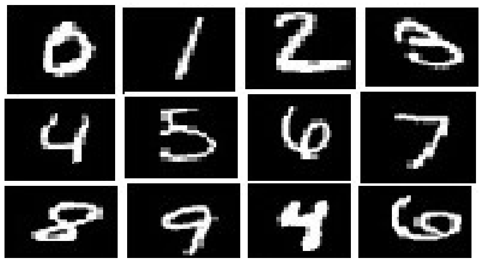
Figure 1. Sample image from MNIST database

The single and double-precision floating-point number system representation is shown in fig 2 and 3.

A Floating Point (FP) number F can be expressed by equation 1

$$F = (-1)^{sf}.(mf).(2)^{ef} \tag{1}$$

Where sf indicates the sign, mf represents the mantissa that is normalized within the range [1; 2), 2 represents the radix, and ef represents the exponent by which mantissa is scaled.

The significant advantage of FP design is that it can represent a more extensive set of characters with high accuracy. However, FP representation has complex techniques for arithmetic operations which create overhead in terms of area, delay, and power. Hence, it is required to convert floating-point number representation to lower precision number representation.

C. *Posit Number System*

The IEEE standard for floating-point arithmetic can be replaced by a POSIT number system that provides better dynamic range, tapered accuracy, parameterized precision, and many more[28]. As most of the network’s weights fall in the range of -1 to 1, and most of the POSIT coding space also falls in that range, using the POSIT system of numeration in a deep learning network is very efficient [6][7]. Mostly 8-bit or 16-bit posit schemes are used in the deep learning system, and 32-bit posit formats are common in scientific arithmetic. Also, the 32-bit posit format is used to replace the IEEE 754 double precision floating-point format.

The floating-point number system has a sign bit, a set of bits to represent the exponent term, and a bunch of bits

Precision	Sign(bits)	Exponent(bits)	Mantissa(bits)
Single	1	8	23

Figure 2. Single precision floating-point number system

Precision	Sign(bits)	Exponent(bits)	Mantissa(bits)
Double	1	11	53

Figure 3. Double precision floating-point number system

called the mantissa, whereas posits add an extra bit known as a regime. Therefore, a posit number has four parts: sign bit, regime, exponent, and fraction bits. The sign and regime bits have higher priority than others [6][31][26].

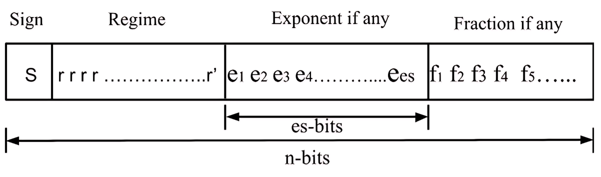

Figure 4. Posit number system representation

Posit is mainly represented as (N,es) format, where N is the total number of bits and es is the number of exponent bits. The first bit of the posit number system represents the sign bit, which means a negative number if set to '1'. The Sign bit is followed by regime bits which can be varied, and for a 'n' bit posit number, the regime can be of length 2 to n-1. It can be running ones followed by zeros or running zeros followed by a one. The exponent bits and fraction bits occupy the rest of the bit positions. The binary representation of the Posit number system is exhibited in fig 4.

A Posit number system X can be expressed by the equation 2

$$X = (-1)^s.((2)^{(2)^{es}})^k.(2)^e.(1+f) \quad (2)$$

Where s represents the sign bits, es represents the maximum exponent bits, k represents the regime bits which depend upon the MSB (Most significant bits) of the regime bits, and the value of m represents the number of identical bits in regime bits. K = -m if the regime has m 0's and m-1 if the regime has m 1's.

The posit Number system has the following advantage over floating-point and fixed-point [6][32].

- • Unique exclusive depiction for zero.
- • No representations wasted for Not-a-Number (NaNs).

4. SYSTEM DESCRIPTION

A LeNet and ResNet-18 architecture-based HDRS (Handwritten Digits Recognition System) is implemented using a Posit number system with low precision compared to a floating and fixed-point number system. The implementation of the model follows different steps like training of the model, extraction of different parameters, and conversion of the parameter from a 32-bit floating number system to an 8-bit Posit number system. The model is trained on the MNIST database, an open-source handwritten digits recognition system.

LeNet architecture comprises seven different layers: two convolution layers, two max-pooling layers, two dense layers, and one dropout layer. The ResNet-18 architecture comprises four different layers: average-pooling and FC (Fully Connected layer). The four layers of ResNet-18 are named layer1, layer2, layer3, and layer4. Layer1 consists of two BasicBlock, and each BasicBlock comprises: two convolutional layers (conv1 and conv2), two BatchNormalisation (bn1 and bn2), and ReLU activation function, the BasicBlock of the remaining layers has one extra layer(downsample), which comprises of convolutional layer and BatchNormalisation.

MNIST database consists of 60,000 training data sets and 10,000 testing data sets. Training data is used to train the model to obtain maximum training accuracy, and testing data is used to test the model for securing test accuracy. The model's training is done by importing different predefined python modules and frameworks. Tensorflow-2.0 is used as a framework for LeNet based HDRS model and PyTorch frameworks for the ResNet-18 based HDRS model, an open-source python software library mainly used for machine learning and deep learning. It permits the developer to craft data flow graph structures. Similarly, Keras is also an open-source library that runs on top of Tensorflow-2.0 and expands the capability of the base of machine learning and deep learning. The functions used in training are Conv2D, MaxPooling2D, BatchNorm2d, AvgPool2d, Dense, and Dropout; these are predefined functions. The shape and size of the model are set according to the LeNet and ResNet-18 architecture before training. Input provided in the grayscale form and having pixel range from 0 to 255 and digit labels data as integers in the range of 0 to 9. The data of pixels in the field of 0 to 255 is normalized to provide better accuracy. The Dropout layer, which aid prevention of overfitting, sets input units to 0 with a rated frequency at each step during training period. Inputs that are not zero are scaled up by 1/(1 - learning rate) so that the total sum continues the same. During training, different kernels are used for convolution and max-pooling operation, and ReLU is used as an activation function for the model. After adding all layers, the model is compiled to create a final model. The loss (categorical cross-entropy) function finds fault or deviation in the learning process during compiling. Adam is an optimizer used to optimize the input weights by comparing with prediction and loss functions. Accuracy is used as a metric to assess the model's performance. After compiling, a model is trained using a different number of epochs that helps to provide stabilized parameters like weights and bias values. Training of the LeNet model is done for 25 epochs, 0.1 validation split, one verbose, one shuffle, and having a batch size of 400, and the training of ResNet-18 model is done for ten epochs, 128 batch size and having a learning rate of 0.001. The training of the ResNet-18 is done over GPU where cuda: 0 is used as a GPU device.

During training, the function adjusts the weights according to the data to achieve better accuracy. The training data set has different training data and corresponding levels used to compare the output with their corresponding level. If an error is found, it rectifies with the back propagation.

During inference of the model, the parameters of different layers like no. of layers, weights values, bias values, and an activation function is calculated. The trained LeNet model is saved to HDF5 format using .h5 as an extension. The ResNet-18 trained model is saved to PTH format, commonly used in PyTorch using .pt as an extension.

Different parameters like weights and bias values are obtained by writing a separate python script. It saves to different files corresponding to a different layer in CSV format using a 32-bit floating-point number system. Keras replica functions are developed for each operation from scratch that can operate on any image of any depths without using the python Keras sequential model. Similarly, posit functions are developed to convert the trained parameter to (8,0), (16,1) Posit, and different Posit operations like Convolution, Max-Pooling, Dense, activation, and dot product. Other parameters stored in the CSV file in 32-bit floating-point numbers are converted to (8,0) and (16,1) posit number system that helps to reduce the precision and provides comparable accuracy.

After performing all the operations on the LeNet based handwritten digits recognition system using the 8-bit and 16-bit posit number system, again, it is tested with the 10,000 MNIST testing data set to obtain the test accuracy based on the low precision 8-bit and 16-bit Posit number system.

5. RESULTS AND DISCUSSION

The LeNet and ResNet-18 architecture used to develop a Posit-based handwritten digits recognition system are shown in fig 5and 6, respectively.

It represents the seven different layers of LeNet architecture. The input feeding to the LeNet model has a dimension of (28*28). The first layer of the neural network is a convolutional layer having 30 kernels of size (5*5*1). The output of the first layer (convolution layers) having dimension (24*24*30) is fed as input to the second layer (Max Pooling layer) having a kernel of size (2*2). The output of the Max Pooling operation having dimension (12*12*30) is fed as input to the third layer (convolutional layer), having 15 kernels of size (3*3*30). Further, the output of the third layer having dimension (10*10*15) is fed as an input to the fourth layer(Max Pooling layer) having a kernel of size (2*2). The output of Max Pooling layers having dimension (5*5*15) is catered as an input to the fifth layer(flatten layer), having a dimension of (375*500). The output is fed as input to the sixth layer (Dense layers), and the output of the sixth layer with a dimension of (500*10) is fed as input to the last layer that provides output as one dimension vector having dimension (10*1). Training is carried out in single-precision floating-point format using the TensorFlow-2.0 python framework, and all the model parameters are in fp32 format. The model obtained a accuracy for training: 98.92%, validation: 98.70%, and test: 97.98%.

The input data fed to ResNet-18 model having dimension (1*28*28) acts as an input to neural networks which is fed to Layer1 of the neural network which is convolutional layer having 64 kernels of size (1*7*7) generates 64 filters and output of first layer (convolution layer) having dimension (64*14*14) is normalized using BatchNormalization and applied ReLU activation and fed as input to the second layer (max-pooling layer) having kernel of size (2*2) and the output of max pooling operation(output1) having dimension (64*7*7) is fed as input to convolutional layer having 64 kernels of size (64*3*3), The output of this convolution layer having dimension(64*7*7) is normalized using BatchNormalization and applied ReLU activation and fed as input to the convolution layer having 64 kernels of dimension(64*3*3), the output of this convolution layer is normalized and added to (output1), the result of addition is applied ReLU activation, The output of ReLU activation(output2) having dimension (64*7*7) is fed to a convolutional layer having 64 kernels of size (64*3*3) generates 64 filters and output of convolution layer having dimension (64*7*7) is normalized using BatchNormalization and applied ReLU activation and fed as input to the convolution layer having 64 kernels of dimension(64*3*3), the output of this convolution layer is normalized and added to (output2), the result of addition is applied ReLU activation, The output of ReLU activation(output3) having dimension (64*7*7) is fed as input to convolutional layer having 128 kernels of size (64*1*1), The output of this convolution layer having dimension(128*4*4) is normalized using BatchNormalization, Let the output is represented as downsample1, Output3 having dimension (64*7*7) is fed as input to a convolutional layer having 128 kernels of size (64*3*3), The output of this convolution layer having dimension(128*4*4) is normalized using BatchNormalization and applied ReLU activation and fed as input to the convolution layer having 128 kernels of dimension(128*3*3), the output of this convolution layer is normalized and added to downsample1. The result of addition is applied ReLU activation, The output of ReLU activation(output4) having dimension (128*4*4) is fed to convolutional layer having 128 kernels of size (128*3*3) generates 128 filters and output of convolution layer having dimension (128*4*4)

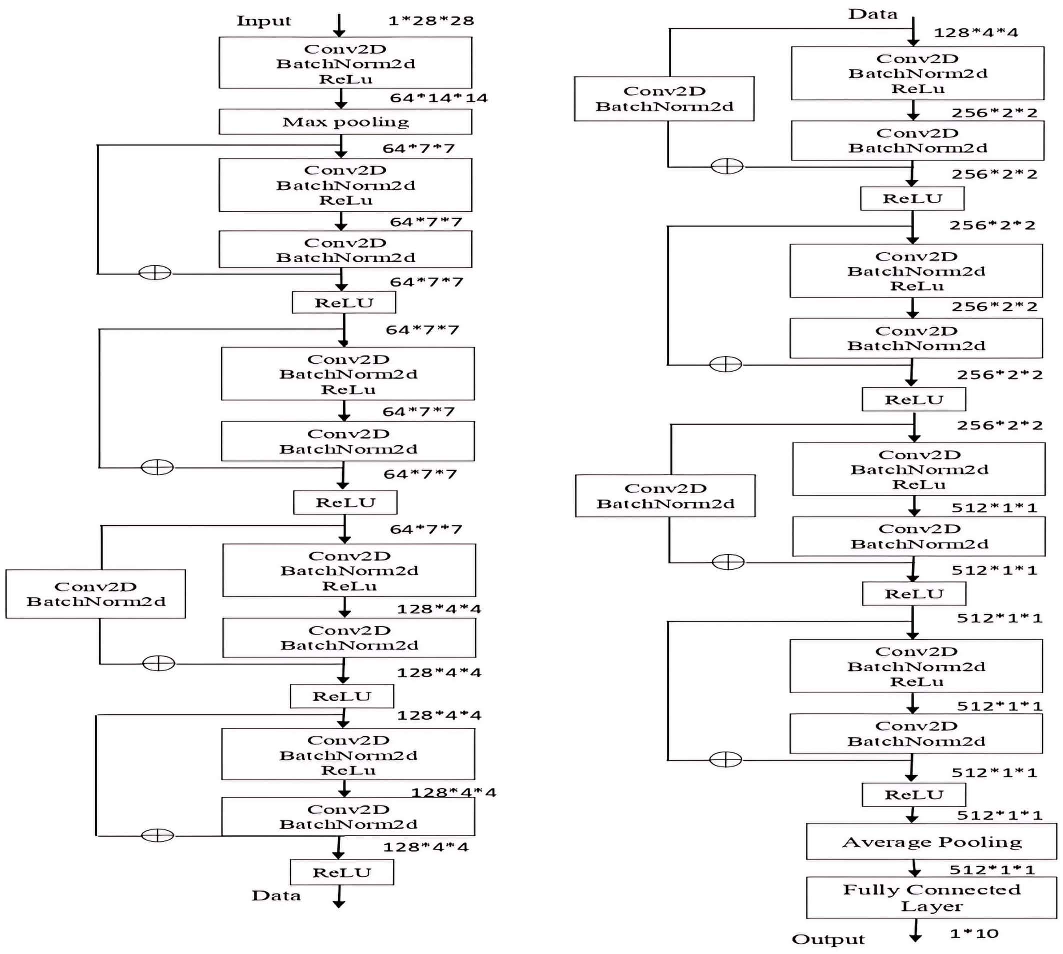

Figure 5. ResNet-18 model architecture

is normalized using BatchNormalization and applied ReLU activation and fed as input to the convolution layer having 128 kernels of dimension(128*3*3), the output of this convolution layer is normalized and added to (output4), the result of addition is applied ReLU activation, The output of ReLU activation(output5) having dimension (128*4*4) is fed as input to convolutional layer having 256 kernels of size (128*1*1), The output of this convolution layer having dimension(256*2*2) is normalized using BatchNormalization, Let the output is represented as downsample2, Output5 having dimension (128*4*4) is fed as input to convolutional layer having 256 kernels of size (128*3*3), The output of this convolution layer having dimension(256*2*2) is normalized using BatchNormalization and applied ReLU activation and fed as input to the convolution layer having 256 kernels of dimension(256*3*3), the output of this convolution layer is normalized and added to downsample2, The result of addition is applied ReLU activation, The output of ReLU activation(output6) having dimension (256*2*2) is fed to a convolutional layer having 256 kernels of size (256*3*3) generates 256 filters and output of convolution layer having dimension (256*2*2) is normalized using BatchNormalization and applied ReLU activation and fed as input to the convolution layer having 256 kernels of dimension(256*3*3), the output of this convolution layer is normalized and added to (output6), the result of addition is applied ReLU activation, The output of ReLU activation(output7) having dimension (256*2*2) is fed as input to convolutional layer having 512 kernels of size (256*1*1), The output of this convolution layer having dimension(512*1*1) is normalized using BatchNormalization, Let the output is represented as downsample3, Output7 having dimension (256*2*2) is fed as input to convolutional layer having 512 kernels of size (256*3*3), The output of this convolution layer having dimension(512*1*1) is normalized using BatchNormalization and applied ReLU activation and fed as input to the convolution layer having 512 kernels of dimension(512*3*3), the output of this convolution layer is normalized and added to downsample3, The result of addition is applied ReLU activation,

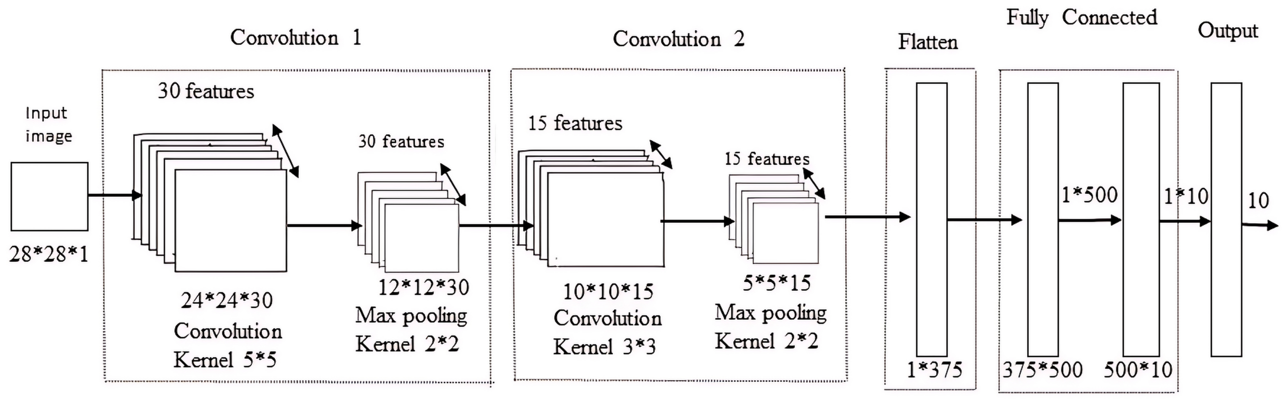

Figure 6. LeNet model architecture

The output of ReLU activation(output8) having dimension (512*1*1) is fed to convolutional layer having 512 kernels of size (512*3*3) generates 512 filters and output of convolution layer having dimension (512*1*1) is normalized using BatchNormalization and applied ReLU activation and fed as input to the convolution layer having 512 kernels of dimension(512*3*3), the output of this convolution layer is normalized and added to (output8), the result of addition is applied ReLU activation, The output of ReLU activation(output9) having dimension(512*1*1) is fed to dense layer having dimension(512*10), The output of dense layer is applied softmax activation function and the final output having dimension(10,1) is obtained which can be used to predict the class of the image.

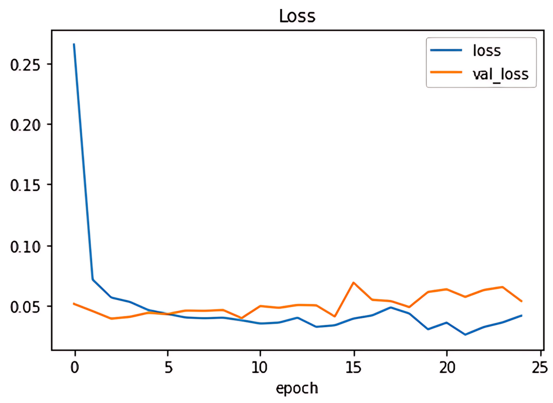

Figure 7. Validation loss and training loss vs epochs

The training accuracy of the handwritten digits recognition system depends on the number of epochs.The model's validation and training losses depend on the number of epochs shown using the graph shown in fig 7. The accuracy result of the model obtained by using predefined Keras and PyTorch function is compared with (8,0), (16,1) posit and float16, float32, 8-bit fixed-point, and 16 bit fixed point-based model. It dispenses almost similar accuracy.

TABLE I. Accuracy Comparison.

Number Format	LeNet(%)	ResNet-18(%)
32-bit floating point	98.98	99.54
16-bit floating point	96.42	97.81
16-bit fixed point	98.02	98.70
8-bit fixed point	91.53	93.21
(8,0) Posit	98.18	98.91
(16,1) Posit	98.98	99.54

The accuracy comparison of the proposed Posit-based LeNet HDRS system and ResNet-18 HDRS system with floating-and fixed-Point system is shown in Table 1. It can be seen that the Posit-based system gives comparable accuracy with the ideal fp32 system with reduced bit precision from 32 to 8, 16 bit and better accuracy compared to fixed-point systems.

6. CONCLUSION AND FUTURE WORK

The proposed system successfully carries out the implementation and analysis of LeNet and ResNet-18 based handwritten digits recognition system using a posit number system.The accuracy level of projects is divided into three classes of accuracy viz. training, validation , and test . Training accuracy was 99.08% and the proposed system provided 98% approx as test accuracy when it tested with 10,000 MNIST datasets. It can be seen that the Posit- based system gives comparable accuracy with the 32-bit floating-point number system. POSIT number system provides a better dynamic range, tapered accuracy, and parameterized precision. The implementation of the ResNet-18based HDRS model takes longer time than the LeNet-based HDRS but provides better accuracy.

When done with a posit unit in the deep neural network, a quantization technique dramatically reduces the memory requirement and computational cost, and it can also be used to improve power efficiency.

The LeNet and ResNet-18 based handwritten digits recognition system using posit number system is further implemented and analyzed on the posit-based AI (Artificial Intelligence) accelerator to verify the efficient utilization of the system's memory. A parallel posit processing engine (PPE) and a systolic array are developed to reduce the time of the extensive computational process.

Acknowledgment

We would like to thank Viswanatham Prashanth, Dr. Pranose J Edavoor, Mr.Aneesh Raveendaran, Mr. Vivian Desalphine and Mr. A Devid Selvakumar of Center for Development of Advance Computing (C-DAC) organization for their active guidance, help and encouragement during the project's development.

References

[1] R. Walid and A. Lasfar, "Handwritten digit recognition using sparse deep architectures," 2014 9th International Conference on Intelligent Systems: Theories and Applications (SITA-14), 2014, pp. 1-6, doi: 10.1109/SITA.2014.6847284.

[2] O. Agaton, S. Kustrin, R. Beresford, "Basic concepts of artificial-neural network (ANN) modeling and its application in pharmaceutical research", Journal of biomedical analysis, vol. 22, no. 5, pp. 717-727,2000.

[3] Luca B. Saldanha, Christophe Bobda, "An embedded system for handwritten digit recognition",Journal of Systems Architecture, vol 61, Issue 10,2015,Pages 693-699,SSN 1383-7621,https://doi.org/10.1016/j.sysarc.2015.07.015.

[4] M. Zhu, Q. Kuang, C. Yang, and J. Lin, "Optimization of convolutional neural network hardware structure based on FPGA," 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2018, pp. 1797-1802, doi: 10.1109/ICIEA.2018.8398000.

[5] Jianhui Han, Zhaolin Li, WeiminZheng, and Youhui Zhang, "Hardware Implementation of Spiking Neural Networks on FPGA",TSINGHUA SCIENCE AND TECHNOLOGY ISSNll1007-0214 04/10 pp479–486 DOI: 10.26599/TST.2019.9010019Volume 25, Number 4, August 2020.

[6] R. Chaurasiya, J. Gustafson, R. Shrestha, J. Neudorfer, S. Nambiar, K. Niyogi, and R. Leupers, "Parameterized posit arithmetic hardware generator," in Proc. IEEE 36th Int. Conf. Comput. Design (ICCD), Oct. 2018, pp. 334–341.

[7] M. Jaiswal and H.K.H. "Universal number posit arithmetic generator on FPGA". In Proceedings of the Design, Automation & amp; Test in Europe Conference and amp, Exhibition, Dresden, Germany, 19–23 March 2018;pp. 1159–1162.

[8] J.L. Hennessy and D. A. Patterson, "A new golden age for computerarchitecture," Communications of the ACM, vol. 62, no. 2, pp. 48–60, Jan 2019.

[9] S. Hashemi, N. Anthony, H. Tann, R. Bahar, and S. Reda, "Understanding the impact of precision quantization on the accuracy and energy of neural networks," in Proceedings of the Conference on Design, Automation & Test in Europe. European Design and Automation Association, 2017, pp. 1478–1483.

[10] Zhiqiang Liu, Yong Dou, Jingfei Jiang, JinweiXu,"Automatic code generation of convolutional neuralnetworks in FPGA implementation," in 2016 International Conference on Field-ProgrammableTechnology (FPT), Xi'an, China, 2016.

[11] Jinze Li, Gongbo Sun and LeiyeYi,"Handwritten Digit Recognition System Based on Convolutional Neural Network," IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), 2020.

[12] Pu, Tiantian. "Application of active learning algorithm in handwriting recognition numbers," Journal of Physics: Conference Series. vol. 1861. No. 1. IOP Publishing, 2021.

[13] Li, Yiqing. "Deep learning based handwritten digit recognition in Android," Journal of Physics: Conference Series. vol. 2010. No. 1. IOP Publishing, 2021.

[14] Liu, Jianghai, and Jie Hong. "Design of Handwritten Numeral Recognition System Based on BP Neural Network," Journal of Physics: Conference Series. vol. 2025. No. 1. IOP Publishing, 2021.

[15] Faghihi, Faramarz, et al. "A Nonsynaptic Memory Based Neural Network for Hand-Written Digit Classification Using an Explainable Feature Extraction Method," Proceedings of the 6th International Conference on Information System and Data Mining. 2022.

[16] Ali, Saqib, Zareen Sakhawat, Tariq Mahmood, Muhammad Saqlain Aslam, Zeeshan Shaukat, and Sana Sahiba. "A robust CNN model for handwritten digits recognition and classification," In IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), pp. 261-265. IEEE, 2020.

[17] Agrawal, Ayush Kumar, A. K. Shrivas, and Vineet kumar Awasthi. "A Robust Model for Handwritten Digit Recognition using Machine and Deep Learning Technique," 2nd International Conference for Emerging Technology (INCET). IEEE, 2021.

[18] Chakraborty, Anuran, Rajonya De, Samir Malakar, Friedhelm Schwenker, and Ram Sarkar. "Handwritten digit string recognition using deep autoencoder based segmentation and resnet based recognition approach," In 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 7737-7742. IEEE.

[19] Chen, Hongzhou, Bin Hu, Long Chen, Dingxue Zhang, and Zhihong Guan. "Multilayer Self-Organizing Impulse Neural Network For Handwritten Digit Recognition," In 2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS), 2021, pp. 1123-1127. IEEE.

[20] Köster, Urs, Tristan Webb, Xin Wang, Marcel Nassar, Arjun K. Bansal, William Constable, Oguz Elibol et al. "Flexpoint: An adaptive numerical format for efficient training of deep neural networks," Advances in neural information processing systems 30 (2017).

[21] Klöwer, Milan, Peter D. Düben, and Tim N. Palmer. "Posits as an alternative to floats for weather and climate models," In Proceedings of the Conference for Next Generation Arithmetic 2019, pp. 1-8. 2019.

[22] B. J. Moore, T. Berger and D. Song, "Validation of a Convolutional Neural Network Model for Spike Transformation Using a Generalized Linear Model", 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 2020, pp. 3236-3239.

[23] N. Buoncristiani, S. Shah, D. Donofrio and J. Shalf, “Evaluating the Numerical Stability of Posit Arithmetic”, in IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, USA, 2020.

[24] J. Johnson, “Rethinking floating point for deep learning,” arXiv e-prints, Nov 2018. [Online]. Available: http://arxiv.org/abs/1811.01721

[25] M. Cococcioni, E. Ruffaldi, and S. Saponara, “Exploiting posit arithmetic for deep neural networks in autonomous driving applications,” in 2018 International Conference of Electrical and Electronic Technologies for Automotive. IEEE, 2018, pp. 1–6

[26] F. de Dinechin, L. Forget, J.-M. Muller, and Y. Uguen, “Posits: the good, the bad and the ugly,” in Proceedings of the Conference for Next Generation Arithmetic 2019. New York, NY, USA: ACM, March 2019, pp. 1–10.

[27] Raul Murillo Montero, Alberto A. Del Barrio, Guillermo Botella, “TemplateBased Posit Multiplication for Training and Inferring in Neural Networks”, 2019, arXiv:1907.04091v1.

[28] Y. Uguen, L. Forget and F. de Dinechin, ”Evaluating the Hardware Cost of the Posit Number System,” 2019 29th International Conference on Field Programmable Logic and Applications (FPL), 2019, pp. 106-113, doi: 10.1109/FPL.2019.00026.

[29] R. Murillo, A. A. Del Barrio, and G. Botella, “Deep PeNSieve: A deep learning framework based on the posit number system,” Digital SignalProcessing, vol. 102, Jul 2020, p. 102762.

[30] G. Raposo, P. Tom´as and N. Roma, “Positnn: Training Deep Neural Networks with Mixed Low-Precision Posit”, ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 7908-7912

[31] ”IEEE Standard for Floating-Point Arithmetic,” in IEEE Std 754-2008 , vol., no., pp.1-70, 29 Aug. 2008, doi: 10.1109/IEEESTD.2008.4610935.

[32] S. H. F. Langroudi, T. Pandit, and D. Kudithipudi, “Deep learning inference on embedded devices: Fixed-point vs posit,” in 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2), March 2018, pp. 19–23.

[33] De Dinechin, Florent, Luc Forget, Jean-Michel Muller, and Yohann Uguen, “Posits: the good, the bad and the ugly”, In Proceedings of the Conference for Next Generation Arithmetic, 2019, pp. 1-10.

Nitish Kumar received his BE degree in electronic and communication engineering from RGPV University Bhopal, India. He is currently pursuing the M.Tech degree in Embedded System Design from NIT Kurukshetra, India and working as Intern in Center for Development of Advance Computing (C-DAC) Bangalore, India.

Dr. Vrinda Gupta received her Ph.D. and M.Tech degrees (Electronics and Communication Engineering) from National Institute of Technology (NIT) Kurukshetra and BE degree from Nagpur University, India. She has worked with Banasthali Vidyapith for about two years starting in 1988. She is presently working as Associate Professor and Head of Department of Electronics and Communication Engineering, NIT Kurukshetra. She has been teaching subjects in the area of communication networks, and wireless communication. She has to her credit two Book Chapter, 26 Journal and 45 Conference publications. She is the member of IEEE, IETE, IE(I), and ISTE.