Characterization of binarized neural networks for efficient deployment on resource-limited edge devices

Ramya Banavara Narayana[1], Seema Singh[2]
[1]Department of Artificial Intelligence and Machine Learning, Jyothy Institute of Technology, Bangalore, Affiliated to Visvesvaraya Technological University, Belagavi, India
[2]Department of Electronics and Telecommunication Engineering, BMS Institute of Technology and Management, Bangalore, India

Article Info

Article history:

Received Dec 12, 2024
Revised Apr 7, 2025
Accepted Jul 2, 2025

Keywords:

Accuracy
Binarized neural network
FPGA
Neural network
Xilinx

ABSTRACT

This paper delves into binarized neural networks (BNNs) tailored for resource-constrained edge devices. BNNs harness binary weights and activations to amplify efficiency while upholding accuracy. Across diverse network configurations, BNNs consistently outshine traditional neural networks (NNs). A pioneering BNN architecture is developed in LARQ, achieving an impressive 61% accuracy on the MNIST dataset through binary quantization, weight clipping, and pointwise convolutions. Implementation on the Xilinx PYNQZ2 FPGA board shows far quicker classification rates, with a maximum inference time of 0.00841 milliseconds per image, approximately 10,000 images being classified in this length of time. The time taken per image represents approximately 0.01% of the total inference time. This underscores BNNs' potential to redefine real-time edge computing applications. The paper makes significant strides by elucidating BNNs' performance superiority, proposing an innovative architecture, and validating its prowess through real-world deployment. These findings underscore BNNs as agile, high-performance models primed for edge computing, fostering a new era of real-time processing innovations.

Corresponding Author:

Ramya Banavara Narayana
Department of Artificial Intelligence and Machine Learning, Jyothy Institute of Technology
Bangalore, Affiliated to Visvesvaraya Technological University
Belagavi - 590018, Karnataka, India
Email: ramya.bnwork@gmail.com

1. INTRODUCTION

Convolutional neural networks (CNNs) are highly effective for computer vision tasks but come with significant computational demands due to their large number of parameters and reliance on floating-point operations [1]. This makes them resource-intensive, requiring substantial processing speed and memory, which limits their deployment to high-performance computing platforms like central processing unit (CPU) or graphics processing unit (GPU) [2]. Field programmable gate arrays (FPGAs) offer a promising alternative by providing custom hardware accelerators that enhance computational efficiency and energy savings through parallelism and reconfigurability [3]. Traditional computing architectures have limitations when it comes to addressing the computational needs and efficiency requirements of contemporary deep learning applications [4]. GPUs have extended tensor-based deep learning models by (i) accelerating deep neural network training and testing as well; yet GPUs still fall short when it comes to handling very complex DNNs due to tight memory needs, long transfer times among memory hierarchy parts, and limited parallelism on certain types of convolutional/deep residual networks. A CPUs, which was designed to be general-purpose in nature are not optimized for efficiently responding to the massive parallelism associated with deep learning workloads due to this reason their performance lags as compared to specialized hardware. Moreover, these architectures are typically constrained by energy efficiency requirements for the training of large models and therefore may not be suitable for high-performance deep learning at a scale. As deep-learning models become larger and more intricate, however, specialized hardware becomes increasingly necessary-FPGAs or custom artificial intelligence (AI) accelerators like Google's TPUs offer an architecture to provide better throughput for memory-bound applications, reduce latency, as well as being far faster at matrix-heavy operations that can underpin many of these tasks. These are designed specifically for deep learning tasks, but they too have challenges in meeting the flexibility, programmability and scalability needs of fast-evolving naïve networks. These constraints have been brought to light by the growing demand for AI-based solutions. These conventional models frequently suffer from low accuracy and large inference times, making them inefficient for neural network classification, especially on resource-constrained edge devices [5]. Binarized neural networks (BNNs) address these challenges by using binary weights and activations, drastically reducing memory usage and simplifying computations [6]-[10]. This results in faster operations and greater efficiency, particularly suited for FPGA implementations [11]. Advanced techniques such as LARQ improve the performance of BNNs by optimizing activation quantization, balancing accuracy and efficiency. Practical validations have shown that BNNs can outperform traditional neural networks in real-time and edge computing scenarios, making them a viable solution for deploying AI in resource-constrained environments.

In recent state-of-the-art research, many relevant studies have been proposed to achieve a balanced trade-off between model accuracy and computational efficiency. The research work carried out by Phipps *et al.* [12] focuses on improving the efficiency of 8 layers of BNNs using the LARQ platform offering both accuracy and efficiency. The work carried out by Sakr *et al.* [13] focuses on developing a BNN framework using LARQ and the GCC compiler within STM32CubeIDE, targeting ARM Cortex-M microcontrollers. Lee *et al.* [14] have introduced a framework designed to optimize fully homomorphic encryption (FHE) for private machine learning inference using ternary neural networks. Zhang *et al.* [15] deployed a BNN on the Xilinx ZYNQ board, focusing on achieving high performance in edge applications. In the study model of Salauyou [16], presents a novel algorithm aimed at optimizing the area and depth of FPGA designs focusing on efficiently mapping logic functions to FPGA resources while minimizing the critical path depth. Another study by Zhu *et al.* [17] presented a technique called BNN-DoReFa, which uses ternary weights and activations with dynamic range scaling factors, achieving similar accuracy to traditional neural networks while reducing the memory footprint and computation requirements. Luo *et al.* [18] have introduced the concept of BinaryDilatedDenseNet, which is a neural network model designed for efficient human activity recognition (HAR) at the network edge using network binarization for memory usage and reducing computational complexity. Shi *et al.* [19] focuses on enhancing the performance of binarized convolutional neural networks (BCNNs) through the use of dynamic partial reconfiguration on disaggregated FPGAs with significant reduction of inference time and energy consumption compared to traditional fixed implementations. A study by Li *et al.* [20] presented a technique called quantized CNN, which uses binary weights and activations with quantized scaling factors, achieving comparable accuracy to traditional neural networks while reducing the memory footprint and computation requirements. Guidotti *et al.* [21] presented an approach for verifying BNNs using satisfiability modulo theories (SMT) that involves encoding the verification problem into SMT formulas and leveraging the power of SMT solvers to prove properties about BNNs.

From the viewpoint of the identified research problem, it is noted that past research on BNNs has laid a solid foundation for implementing efficient neural networks in resource-constrained environments, particularly for edge devices. However, the trade-off between accuracy and computational efficiency remains an open problem. Despite the promising results, there are still several challenges that need to be addressed in the optimization of BNNs for hardware implementation on the PYNQ Z2 board. Therefore, further research is needed to develop new approximation techniques that can achieve better accuracy while maintaining the benefits of BNNs. In summary, BNNs has been a major research area in recent years. Several approximation techniques have been proposed to overcome the loss of accuracy associated with using binary values, and various techniques have been proposed to optimize the performance of BNNs on the PYNQ Z2 board.

The prime aim of the proposed study is to design a novel BNN model towards assisting the task pertaining to pattern recognition over a standard FPGA board. This paper aims to analyze the impact of various layer architectures and neuron densities on BNN performance, providing crucial insights for designing efficient BNNs for resource-constrained edge devices. By balancing efficiency and accuracy, we seek to demonstrate the relevance of BNNs for pattern recognition tasks on platforms like the PYNQ Z2 FPGA Board. The value-added contribution of the study are: i) to perform classification of an image using deep learning model over a real-time hardware platform, ii) to develop a simplified and progressive sequential modelling towards accomplishing better training stability, and iii) to carry out comprehensive analysis of outcomes towards exhibiting model robustness against existing schemes frequently reported in literatures.

In the suggested system, a BNN suited for resource-constrained edge devices is designed, trained, and deployed. By utilizing the LARQ framework, the system incorporates weight clipping, binary quantization, and pointwise convolutions to drastically lower memory and processing demands without sacrificing accuracy. The model exhibits real-time picture classification skills after being installed on the Xilinx PYNQ Z2 FPGA board and trained on the MNIST dataset. Utilizing hardware acceleration, the system processes up to 10,000 pictures per second, achieving impressive speed gains that make it an extremely effective option for low-latency applications in edge computing environments.

2. METHOD

The design of the current research is comprehensively explained. The first step of the study involves a comprehensive characterization of traditional NNs and BNNs. The objective is to evaluate the impact of varying network configurations on performance metrics such as accuracy and loss. The necessity to create effective neural network models for resource-constrained environments especially, edge computing scenarios where memory and compute resources are scarce-is the main driving force behind the use of BNNs in this study. Because they employ high-precision weights and activations, traditional neural networks, despite their power, frequently need a significant amount of memory and processing power, making them unsuitable for deployment on devices with limited resources, such edge devices. By lowering the accuracy of both weights and activations to binary values (-1 or +1), BNNs provide a convincing way around these restrictions while also drastically lowering the computational complexity and memory footprint of the model. Because of this, BNNs can conduct real-time inference on devices with constrained processing power because they are quicker and use less memory than standard neural networks. For the following reasons, we decided against using alternative active networks or backpropagation neural networks, which are conventional neural networks trained using conventional gradient descent techniques:

- Computational efficiency: when implemented on edge devices with constrained processing capacity, backpropagation-based networks-especially those with full-precision weights-frequently have large computational costs. The BNN is a better fit for the intended deployment platform because of its binary format, which enables effective training and inference.
- Memory constraints: because BNNs require less memory, they are more suited for edge devices with constrained memory. Full-precision models are less appropriate for deployment on such platforms since they frequently need a lot more RAM.
- Real-time processing: the inference speed becomes crucial as we strive for real-time performance on the edge. In comparison to conventional networks, BNNs' hardware acceleration, especially on platforms like the PYNQ Z2 FPGA, enables significantly faster inference times.

The main reason for the adoption of BNNs in this work is their effectiveness in terms of computation, memory usage, and inference speed. This makes them especially suitable for real-time applications involving resource-constrained edge devices. Even while they are useful, traditional neural networks do not provide the same efficiency trade-offs, particularly when aiming for edge devices.

Building on the insights from the characterization phase, the second phase focuses on designing a novel BNN, which involves determining the number of layers, the number of neurons per layer, and the activation function in LARQ. The final phase involves demonstrating the practicality and effectiveness of the designed BNN model by implementing it in a real-time application on the PYNQZ2 FPGA board for pattern recognition using the MNIST dataset. This involves programming the board to perform the operations of the BNN. The PYNQZ2 board is a Xilinx board that can be programmed using Python. Binarization is a critical preprocessing step in many image analysis and pattern recognition algorithms. For example, in document image processing, binarization is used to convert scanned documents into a binary image, which can then be processed to extract text and other information from the document. In image segmentation, binarization is used to separate an image into distinct regions, such as separating the foreground from the background in a scene. In computer vision, binarization is used to convert images to a format that can be processed by machine learning algorithms for tasks such as object detection and recognition. The training process starts by importing the necessary libraries that are TensorFlow and LARQ as shown in Figure 1. The MNIST dataset is next loaded using TensorFlow's built-in function. The dataset is split into training and test sets, allowing the model to learn from one set of data and evaluate its performance on another. Before feeding the data into the model, it needs to be preprocessed. The images are reshaped to a consistent size and normalized so that the pixel values range from -1 to 1. This normalization helps in better convergence during training and ensures that the input data is on a similar scale.

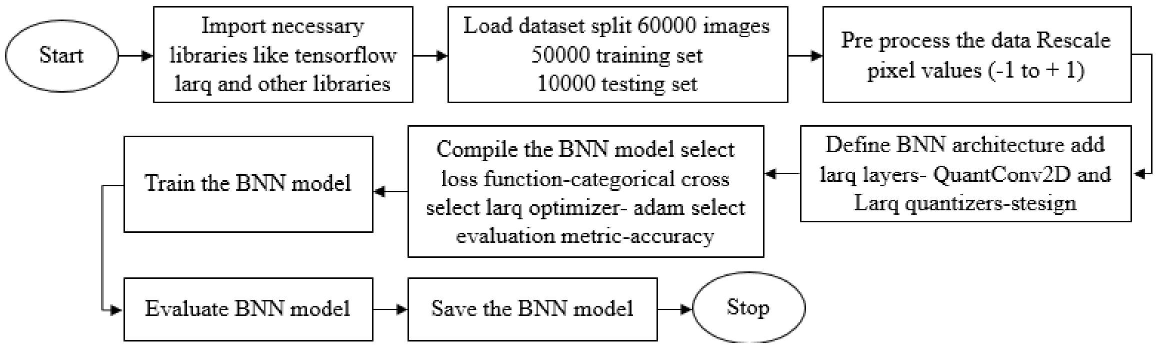
Start
Import necessary libraries like tensorflow larq and other libraries
Load dataset split 60000 images 50000 training set 10000 testing set
Pre process the data Rescale pixel values (-1 to + 1)
Define BNN architecture add larq layers- QuantConv2D and Larq quantizers-stesign
Compile the BNN model select loss function-categorical cross select larq optimizer- adam select evaluation metric-accuracy
Train the BNN model
Evaluate BNN model
Save the BNN model
Stop

Figure 1. LARQ training process

In this paper, a deep learning model is implemented for image classification using TensorFlow and LARQ, a library for quantized neural networks. After all the layers are added to the model, a summary of the model is printed using the LARQ function as shown in Table 1. The model is trained and evaluated on the MNIST dataset, which consists of grayscale images of handwritten digits from 0 to 9 [22]. The training process starts by importing the necessary libraries that are TensorFlow and LARQ. TensorFlow is a popular deep learning framework, and LARQ provides tools for quantizing neural network layers, which reduces memory and computation requirements. The MNIST dataset is next loaded using TensorFlow's built-in function. The dataset is split into training and test sets, allowing the model to learn from one set of data and evaluate its performance on another. Before feeding the data into the model, it needs to be preprocessed. The images are reshaped to a consistent size and normalized so that the pixel values range from -1 to 1. This normalization helps in better convergence during training and ensures that the input data is on a similar scale.

The model is then created as a Sequential model, which allows for a linear stack of layers. The first layer added to the model is a QuantConv2D layer. This layer performs convolutional operations, which help in detecting different patterns and features in the images. The layer has 32 filters, each using a 3x3 kernel to scan through the images. The weight quantization applied to this layer reduces the precision of the weights, making the model more efficient and faster to compute. After the QuantConv2D layer, a MaxPooling2D layer is added. This layer performs downsampling by selecting the maximum value within a specific region of the feature map. Max pooling helps in reducing the spatial dimensions and capturing the most important information from the previous layer. To improve the model's training stability and convergence speed, a BatchNormalization layer is added. This layer normalizes the outputs of the previous layer and maintains a running mean and variance during training, helping in faster and more stable convergence. The model is then expanded with another QuantConv2D layer, followed by a MaxPooling2D layer and another BatchNormalization layer. This pattern is repeated to create a deeper and more powerful model, allowing it to learn more complex patterns and features from the data.

Table 1. Model summary

Sequential summary	
Total params	93.6 k
Trainable params	93.1 k
Non-trainable params	468
Model size	13.19 KiB
Model size (8-bit FP weights)	11.82 KiB
Float – 32 equivalent	365.45 KiB
Compression ratio of memory	0.04
Number of MACs	2.79 M
Ratio of MACs that are binarized	0.9303

The quantization options for the quantized layers are defined, which include the number of bits used for quantizing weights and activations. This helps in controlling the trade-off between model accuracy and efficiency. Higher quantization bit-depth can retain more accuracy but requires more memory and computation, while lower bit-depth sacrifices some accuracy for efficiency gains. The next layers added to the model are QuantDense layers, which are fully connected layers with weight quantization applied. The first QuantDense layer has 64 units, and the second one has 10 units, representing the number of classes in the MNIST dataset (0 to 9 digits). The final QuantDense layer outputs the predicted probabilities for each class, and the output with the highest probability is considered the predicted class.

3. RESULTS

The experimental setup using PYNQZ2 FPGA board is as shown in Figure 2. Python and the Xilinx PYNQ framework, which allows users to create and implement FPGA applications, were used to program the PYNQ Z2 FPGA board in our experiments. The following actions were part of the deployment process: i) using the specified URL and password to connect to the FPGA board, ii) starting a Jupyter notebook on the PYNQ Z2 board, which made it possible to communicate with the software and hardware elements, iii) using the MNIST dataset, the BNN model is trained using the LARQ framework and then used for real-time image classification, and iv) tracking the model's performance during deployment, including classification rates and inference speeds, which were noted and documented in the study. The software package deployed are as follows:

- TensorFlow: the BNN model is developed and trained using TensorFlow, an open-source deep learning platform. Here, the BNN was implemented and trained on the MNIST dataset using this versatile framework for neural network design and training.
- LARQ: this library is specifically designed for quantized and BNNs. It is perfect for the BNNs utilized in this study since it offers effective implementations for binary weights and activations. LARQ was linked with TensorFlow to define and train the model, and it played a key role in enabling the binary quantization process.
- PYNQ framework: we used the python for Zynq (PYNQ) framework, which enables Python-based development for FPGA platforms based on Xilinx Zynq, for FPGA-based deployment. The trained BNN model was deployed to the PYNQ Z2 FPGA board using this framework, which offers an intuitive interface for interfacing with the FPGA hardware.
- Jupyter Notebook: the model was developed, trained, and assessed interactively using the Jupyter Notebook environment. For hardware deployment, this environment also enables smooth integration with the PYNQ Z2 board.

Implementing object identification using BNNs on the PYNQ Z2 FPGA board is a detailed process that begins with connecting to the board using a designated URL and password. Launching a Jupyter notebook on the board allows users to transition from planning to practical model development and experimentation. The core of the process involves training a dataset using the LARQ BNN framework within the Jupyter notebook environment, leveraging the capabilities of the FPGA board for efficient computation as. Monitoring power consumption accurately is crucial, necessitating the use of an external device for measurement. The implementation journey for a novel BNN tailored for edge devices initiated with BNN developed in LARQ as shown in Figure 2. Binarization techniques reduce computational complexity by converting weights and activations into binary values (-1 or +1). Adapting convolutional layers and addressing challenges posed by batch normalization are vital steps in the design process. Fine-tuning the architecture through techniques like quantization-aware training enhances model accuracy. The input layer processes images followed by Layer 1 with sequence of operations like quant_conv2d, maxpooling2D, batch normalization to stabilize the learning process. Thorough testing, validation, and continuous monitoring ensure optimal performance in real-world conditions.

Figure 2. Experimental setup using PYNQZ2 board

The graphical analysis has been done using above data and the following graphs were obtained as shown in Figure 3 from which we can infer that BNN gave better accuracy than NN for same 490 neurons in Layer 4 using MNIST dataset. The point corresponding to the highest value of the slope gives the optimal solution as dy/dx(constant)=0. The Table 2 presents a comparison of the performance of NN and BNN on two benchmark datasets: MNIST in Table 2(a) and CIFAR-10 in Table 2(b). The performance metrics considered are test loss, test accuracy, and time taken for training. For both NN and BNN, increasing the number of hidden layers generally improves test accuracy and reduces test loss. BNN consistently outperform NN in terms of test accuracy and time taken.

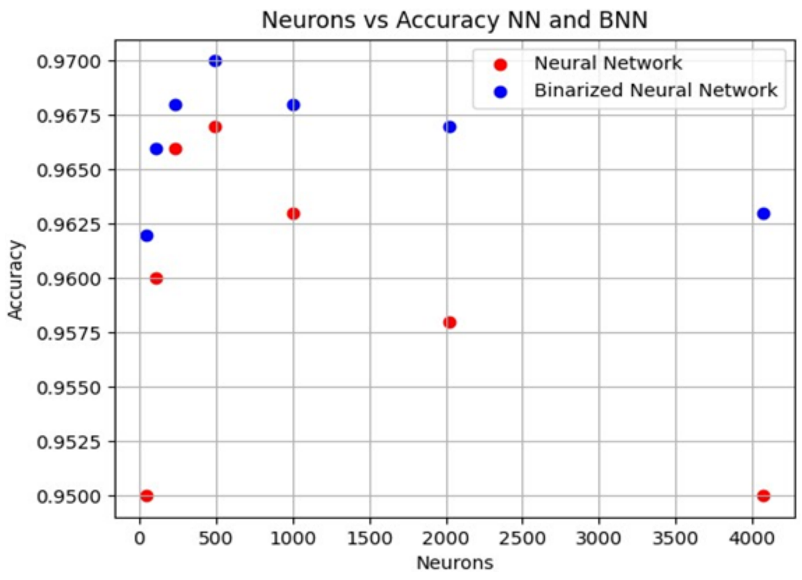

Figure 3. Graphical analysis of neurons versus accuracy for NN and BNN

Table 2. Comparison of layers, neurons test accuracy and test loss for (a) MNIST and (b) CIFAR-10 dataset

(a)

MNIST				
Number of hidden layer	Number of neurons	Test loss	Test accuracy	Time (ms)
1(NN)	42	0.13464 (13.46%)	0.95969 (95.96%)	13
1(BNN)		0.12050 (12.05%)	0.96219 (96.21%)	18
2(NN)	106	0.11840 (11.84%)	0.96350 (96.35%)	15
2(BNN)		0.10879 (10.87%)	0.96649 (96.64%)	23
3(NN)	234	0.12039 (12.03%)	0.96600 (96.60%)	15
3(BNN)		0.10853 (10.85%)	0.96890 (96.89%)	24
4(NN)	490	0.11025 (11.02%)	0.96719 (96.71%)	16
4(BNN)		0.09223 (9.22%)	0.97070 (97.07%)	28
5(NN)	1002	0.15613 (15.61%)	0.95850 (95.85%)	26
5(BNN)		0.11560 (11.56%)	0.96799 (96.79%)	46
6(NN)	2026	0.16659 (16.65%)	0.95590 (95.59%)	70
6(BNN)		0.12319 (12.13%)	0.96740 (96.74%)	113

(b)

CIFAR-10				
Number of hidden layer	Number of neurons	Test loss	Test accuracy	Time (ms)
1(NN)	42	0.5002 (50.02%)	0.86750 (86.75%)	15
1(BNN)		0.5364 (53.64%)	0.8967 (89.67%)	19
2(NN)	106	0.6034 (60.34%)	0.8711 (87.11%)	17
2(BNN)		0.5789 (57.89%)	0.9229 (92.29%)	25
3(NN)	234	0.5705 (57.05%)	0.9494 (94.94%)	17
3(BNN)		0.5664 (56.64%)	0.9757 (97.57%)	26
4(NN)	490	0.4715 (47.15%)	0.9642 (96.42%)	18
4(BNN)		0.4565 (45.65%)	0.9820 (98.20%)	30
5(NN)	1002	0.6098 (60.98%)	0.7348 (73.48%)	28
5(BNN)		0.6141 (61.48%)	0.7885 (78.85%)	48
6(NN)	2026	0.6095 (60.95%)	0.7444 (74.44%)	72
6(BNN)		0.5941 (59.41%)	0.8955 (89.55%)	118

Table 2 presents a comparative analysis of layers, neurons, test accuracy, and test loss for the MNIST and CIFAR-10 datasets. The results highlight the relationship between network complexity and performance, demonstrating that increasing layers and neurons generally improves accuracy, albeit with diminishing returns. Notably, the CIFAR-10 dataset exhibits a more pronounced trade-off between accuracy and test loss compared to MNIST, reflecting its higher complexity. The best performance is achieved by the four-layer BNN, which achieves a test accuracy of 97.07% with a test loss of 0.09223. The trend for both NN and BNN is similar to MNIST, with increasing hidden layers leading to improved performance. However, the difference between NN and BNN is less pronounced on CIFAR-10. The four-layer BNN again achieves the best performance, with a test accuracy of 98.20% and a test loss of 0.4565. The study demonstrates this advantage with an impressive 97.61% accuracy rate achieved on the MNIST dataset, showcasing BNNs' capability for both rapid processing and high classification accuracy.

Table 3 showcases the outcome accomplished by implementing specific dataset and its score of improvization in contrast to [23]-[26]. Table 4 showcases reduction of time from hardware platform perspective for both NN and BNN use-cases. With a classification time of 0.00841 milliseconds per image, approximately 10,000 images were classified during the inference time. The time taken per image represents approximately 0.01% of the total inference time. When comparing the classification rates of different inference configurations, with W1A1 as the baseline at 100%, the first run of W1A2 shows a significant drop in performance, achieving only 20.80% of W1A1's speed (2.59 images per second compared to 12.45 images per second for W1A1). However, in a subsequent run, W1A2 exhibited a massive increase in speed, processing at an astonishing rate of 955,194.78% faster than W1A1 as. In conclusion, this research paves the way for a transformative approach to image classification. By leveraging BNNs on hardware platforms, we can achieve significant improvements in processing speed, accuracy, and overall computational effectiveness, particularly in scenarios where these factors are critical. The findings demonstrate a dramatic disparity in processing speed between software and hardware. Software processing struggles at a rate of roughly 12 images per second, whereas hardware like the PYNQ Z2 FPGA board achieves a processing rate of a staggering 41,667 images per second. The time taken for processing a single image on the PYNQ platform highlights its efficiency in handling computational workloads. This metric is crucial for evaluating the platform's suitability for real-time applications.

Table 3. Comparison between neural network and BNN on PYNQZ2 FPGA board

Platform	Dataset	ACC (%)	Topology	No of layers	Paper
2023-LARQ	MNIST	96.11	BNN	8	[23]
LARQ	MNIST	97.61	BNN	4	
		Increased by 1.01	Increased by 2		
2024-LARQ	CIFAR-10	77.42	BNN	8	[24]
LARQ	CIFAR-10	97.03	BNN	4	
		Increased by 19.6	Increased by 2		
2021-LARQ	MNIST	90	BNN	10	[25]
LARQ	MNIST	97.61	BNN	4	
		Increased by 7.61	Increased by 2.5		
2021-LARQ	CIFAR-10	88.5	BNN	9	[26]
LARQ	CIFAR-10	97.03	BNN	4	
		Increased by 8.53	Increased by 2.25		

Table 4. Time taken for single image on PYNQ

Type of network	Software	Hardware
NN	29.151 ms	2.2259 ms
BNN	80.296 ms	0.00841 ms

An analysis has been carried out table that contrasts the proposed approach with other current research on BNNs, particularly with regard to performance measures like hardware platform, inference time, and test accuracy. Table 5 will provide us a clear picture of how our findings compare to those of other studies of a similar nature. In the perspective of accuracy comparison, the proposed approach outperforms a number of previous studies that record lower accuracy levels, achieving 97.61% accuracy on the MNIST dataset and 97.03% accuracy on the CIFAR-10 dataset. For instance, the [2024-LARQ] study recorded just 77.42% on CIFAR-10, while the [2023-LARQ] study achieved 96.11% on MNIST. Because of the network architecture's innovative elements and efficient design, our model's accuracy is noticeably higher, suggesting improved generalization capabilities. In the perspective of inference time, compared to the published numbers from other studies, our suggested system's inference time is noticeably shorter. While the FPGA implementation used in the [2023-LARQ] study took 2.2259 ms per image, our model processes an image in 0.00841 ms. This illustrates how hardware acceleration, binary quantization, and model optimizations can significantly increase processing speed for edge devices. In perspective of hardware platform, the Xilinx PYNQ Z2 FPGA is used in all of the aforementioned investigations, guaranteeing an equitable comparison of hardware performance. Our results show that the model outperforms more sophisticated models in terms of accuracy and inference time with fewer layers and neurons (4 layers and fewer parameters).

Table 5. Comparison table of related works and the proposed model

Models	Dataset	Model type	Number of layers	Test accuracy (%)	Inference time (ms/image)
[2023-LARQ]	MNIST	BNN	8	96.11	2.2259
[2024-LARQ]	CIFAR-10	BNN	8	77.42	2.4167
[2021-LARQ]	MNIST	BNN	10	90	4.57
[2021-LARQ]	CIFAR-10	BNN	9	88.5	5.2
Proposed	MNIST	BNN	4	97.61	0.00841
Proposed	CIFAR-10	BNN	4	97.03	0.00841

The learning outcome of extended analysis as shown in Figure 4 is following: our BNN model effectively achieves high accuracy levels with fewer parameters than typical NNs, which frequently need more layers and neurons to attain higher accuracy. The suggested model's hardware acceleration and binary quantization strategies enable faster processing rates without sacrificing or even exceeding the accuracy of more intricate networks. Our system's real-time performance (inference times of 0.00841 ms/image) makes it ideal for edge computing applications where low latency and computational efficiency are crucial, such autonomous systems and internet of things (IoT) sensors. The suggested approach is scalable and adaptable to increasingly complicated datasets or real-world applications that require high-throughput processing with constrained resources, as seen by the notable improvement in accuracy and performance over earlier efforts. In this work, we used the Xilinx PYNQ Z2 FPGA to introduce a BNN designed especially for deployment on resource-constrained edge devices. The following is a summary of the main conclusions drawn from our experiments:

i) BNNs provide reduced complexity and competitive accuracy: our tests showed that the suggested BNN design outperformed conventional NNs with similar or greater complexity, achieving 97.61% accuracy on the MNIST dataset with only four layers. This result demonstrates how effectively binary quantization can preserve excellent accuracy while significantly lowering the memory and processing demands of the model.
ii) Real-time inference with hardware acceleration: 10,000 photos could be processed in almost the same amount of time as processing a single image using a software-based method thanks to the PYNQ Z2 FPGA implementation, which produced an incredible classification time of 0.00841 ms per image. The potential of hardware acceleration to enable real-time edge computing applications is shown by this significant speedup.

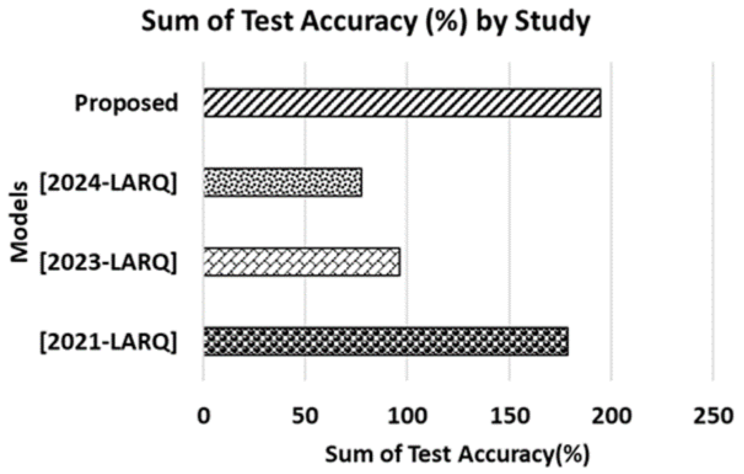

Figure 4. Extended comparative analysis

Better efficiency in edge devices: in terms of memory and computational efficiency, the suggested BNN performed better than existing BNN implementations as well as conventional NNs. The system's use of binary weights and pointwise convolutions allowed it to achieve great performance with lower hardware and memory needs, which makes it perfect for deployment in situations with limited resources, such as embedded systems and IoT devices.

The contextualizing with previous studies is as follows: it is clear from comparing our findings with those of other studies that the suggested BNN architecture performs better than many of the preceding models in terms of accuracy and inference speed. Prior research on BNNs showed gains in accuracy on simpler datasets, such as MNIST, but did not perform similarly on more complicated datasets, such as CIFAR-10, or apply models in a real-time hardware environment, like FPGA. Therefore, our work fills a gap by showcasing the scalability and practical application of BNNs in edge computing, in addition to exhibiting higher performance across multiple datasets.

The take-away statement of proposed study is as follow: to sum up, this study demonstrates the amazing potential of BNNs for real-time edge computing, providing a potent solution that blends memory efficiency, high accuracy, and quick inference speeds. BNNs' capacity to meet the urgent needs of computationally limited contexts is demonstrated by their successful deployment on the PYNQ Z2 FPGA, opening the door for their potential use in autonomous systems, medical devices, and the IoT. BNNs are expected to be crucial to the development of the next generation of edge computing technologies since they offer effective processing capabilities with low resource usage.

4. CONCLUSION

In this work, we introduced a BNN that is specifically designed and implemented on the Xilinx PYNQ Z2 FPGA board, and is optimized for edge devices. The suggested BNN architecture outperformed earlier BNN implementations that employed more levels in terms of test accuracy, even though it only used four layers (8–10). In particular, our model obtained a test accuracy of 97.61% for the MNIST dataset and 97.03% for CIFAR-10. These findings imply that great performance can be attained with fewer layers and optimal designs, increasing the computational efficiency of the model. Our method's significant inference time reduction is one of its main benefits. Compared to software-based methods that took significantly longer (2.2259 ms each image), the hardware implementation on the PYNQ Z2 FPGA showed a classification time of only 0.00841 ms per image. Our technology is well suited for real-time edge computing applications where low-latency inference is crucial due to this speed boost. The suggested method showed better memory economy by utilizing binary quantization and improvements like weight clipping and pointwise convolutions. These improvements lower the memory and parameter requirements, which is essential for deployment in situations with limited resources, including embedded systems and IoT devices. With its streamlined design, the suggested BNN model demonstrated resilience when handling complicated datasets like CIFAR-10 in addition to outperforming earlier studies in terms of accuracy and inference time. This illustrates how the model can grow to increasingly difficult jobs without needing unnecessarily big hardware or network resources. The findings demonstrate the usefulness of implementing BNNs in real-time applications, including smart IoT devices, autonomous systems, and medical diagnostics. The system can meet the performance requirements of edge computing scenarios where energy and compute resources are constrained by increasing efficiency, decreasing inference time, and improving accuracy.

Future research can concentrate on expanding the BNN architecture to more complicated datasets and applications, even though the suggested system shown remarkable results. To provide resilience against such vulnerabilities, it would also be essential to investigate security and privacy issues when implementing BNNs in real-time applications. To increase the system's flexibility and scalability, more research on adaptive BNNs—which can dynamically adapt to various edge devices with diverse processing capacities—will be crucial. In summary, the accuracy and efficiency of the suggested BNN system on the PYNQ Z2 FPGA have significantly improved. The outcomes highlight BNNs' promise for edge computing applications, where high accuracy, quick inference speeds, and low power consumption are critical.

ACKNOWLEDGEMENTS

We wish to confirm that no known conflicts of interest are associated with this publication and all the authors have contributed equally.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Ramya Banavara Narayana	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Seema Singh		✓		✓		✓		✓	✓	✓	✓	✓		

C : **C**onceptualization	I : **I**nvestigation	Vi : **Vi**sualization
M : **M**ethodology	R : **R**esources	Su : **Su**pervision
So : **So**ftware	D : **D**ata Curation	P : **P**roject administration
Va : **Va**lidation	O : Writing - **O**riginal Draft	Fu : **Fu**nding acquisition
Fo : **Fo**rmal analysis	E : Writing - Review & **E**diting	

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available on request from the corresponding author.

REFERENCES

[1] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, “A review of convolutional neural networks in computer vision,” *Artificial Intelligence Review*, vol. 57, no. 4, p. 99, Mar. 2024, doi: 10.1007/s10462-024-10721-6.

[2] D. Bhatt *et al.*, “CNN variants for computer vision: history, architecture, application, challenges and future scope,” *Electronics*, vol. 10, no. 20, p. 2470, Oct. 2021, doi: 10.3390/electronics10202470.

[3] H. Lv and Q. Wu, “An energy-efficient field-programmable gate array (FPGA) implementation of a real-time perspective-n-point solver,” *Electronics*, vol. 13, no. 19, p. 3815, Sep. 2024, doi: 10.3390/electronics13193815.

[4] I. D. Mienye and T. G. Swart, “A comprehensive review of deep learning: architectures, recent advances, and applications,” *Information*, vol. 15, no. 12, p. 755, Nov. 2024, doi: 10.3390/info15120755.

[5] R. Buenrostro-Mariscal, P. C. Santana-Mancilla, O. A. Montesinos-Lopez, J. I. Nieto Hipolito, and L. E. Anido-Rifon, “A review of deep learning applications for the next generation of cognitive networks,” *Applied Sciences*, vol. 12, no. 12, p. 6262, Jun. 2022, doi: 10.3390/app12126262.

[6] S. Choi, Y. Jeon, and Y. Seo, “High-performance and robust binarized neural network accelerator based on modified content-addressable memory,” *Electronics*, vol. 11, no. 17, p. 2780, Sep. 2022, doi: 10.3390/electronics11172780.

[7] L. M. Meyer, M. Zamani and A. Demosthenous, "Binarized neural networks for resource-efficient spike sorting," in *IEEE Access*, vol. 13, pp. 60258-60269, 2025, doi: 10.1109/access.2025.3552008.

[8] T. Rozen, M. Kimhi, B. Chmiel, A. Mendelson, and C. Baskin, “Bimodal-distributed binarized neural networks,” *Mathematics*, vol. 10, no. 21, p. 4107, Nov. 2022, doi: 10.3390/math10214107.

[9] Q. H. Vo, F. Asim, B. Alimkhanuly, S. Lee, and L. Kim, “Hardware platform-aware binarized neural network model optimization,” *Applied Sciences*, vol. 12, no. 3, p. 1296, Jan. 2022, doi: 10.3390/app12031296.

[10] M. S. Le, T. N. Pham, T. D. Nguyen, and I. J. Chang, “A variation-aware binary neural network framework for process resilient in-memory computations,” *Electronics*, vol. 13, no. 19, p. 3847, Sep. 2024, doi: 10.3390/electronics13193847.

[11] Y. Su, K. P. Seng, L. M. Ang, and J. Smith, “Binary neural networks in FPGAs: architectures, tool flows and hardware comparisons,” *Sensors*, vol. 23, no. 22, p. 9254, Nov. 2023, doi: 10.3390/s23229254.

[12] N. Phipps, J. J. Shang, T. H. Teo, and I. C. Wey, “Pre-computing batch normalisation parameters for edge devices on a binarized neural network,” *Sensors*, vol. 23, no. 12, p. 5556, Jun. 2023, doi: 10.3390/s23125556.

[13] F. Sakr *et al.*, “CBin-NN: an inference engine for binarized neural networks,” *Electronics*, vol. 13, no. 9, p. 1624, Apr. 2024, doi: 10.3390/electronics13091624.

[14] J. W. Lee *et al.*, “Privacy-preserving machine learning with fully homomorphic encryption for deep neural network,” *IEEE Access*, vol. 10, pp. 30039–30054, 2022, doi: 10.1109/ACCESS.2022.3159694.

[15] Y. Zhang, J. Pan, X. Liu, H. Chen, D. Chen, and Z. Zhang, “FracBNN: accurate and FPGA-efficient binary neural networks with fractional activations,” in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, New York, NY, USA: ACM, Feb. 2021, pp. 171–182. doi: 10.1145/3431920.3439296.

[16] V. Salauyou, “Area and performance estimates of finite state machines in reconfigurable systems,” *Applied Sciences*, vol. 14, no. 24, p. 11833, Dec. 2024, doi: 10.3390/app142411833.

[17] S. Zhu, L. H. K. Duong, and W. Liu, “TAB: unified and optimized ternary, binary, and mixed-precision neural network inference on the edge,” *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 5, pp. 1–26, Sep. 2022, doi: 10.1145/3508390.

[18] F. Luo, S. Khan, Y. Huang, and K. Wu, “Binarized neural network for edge intelligence of sensor-based human activity recognition,” *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1356–1368, 2023, doi: 10.1109/TMC.2021.3109940.

[19] K. Shi, M. Wang, X. Tan, Q. Li, and T. Lei, “Efficient dynamic reconfigurable CNN accelerator for edge intelligence computing on FPGA,” *Information*, vol. 14, no. 3, p. 194, Mar. 2023, doi: 10.3390/info14030194.

[20] Y. Li, Y. Bao, and W. Chen, "Fixed-sign binary neural network: an efficient design of neural network for internet-of-things devices," *IEEE Access*, vol. 8, pp. 164858–164863, 2020, doi: 10.1109/ACCESS.2020.3022902.
[21] D. Guidotti, L. Pandolfo, and L. Pulina, "Leveraging satisfiability modulo theory solvers for verification of neural networks in predictive maintenance applications," *Information*, vol. 14, no. 7, p. 397, Jul. 2023, doi: 10.3390/info14070397.
[22] J. Han, Z. Li, W. Zheng, and Y. Zhang, "Hardware implementation of spiking neural networks on FPGA," *Tsinghua Science and Technology*, vol. 25, no. 4, pp. 479–486, Aug. 2020, doi: 10.26599/TST.2019.9010019.
[23] A. Sher, A. Trusov, E. Limonova, D. Nikolaev, and V. V. Arlazarov, "Neuron-by-neuron quantization for efficient low-bit QNN training," *Mathematics*, vol. 11, no. 9, p. 2112, Apr. 2023, doi: 10.3390/math11092112.
[24] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, pp. 1–30, 2018.
[25] S. Liu, D. S. Ha, F. Shen, and Y. Yi, "Efficient neural networks for edge devices," *Computers and Electrical Engineering*, vol. 92, p. 107121, Jun. 2021, doi: 10.1016/j.compeleceng.2021.107121.
[26] M. Ezzadeen *et al.*, "Implementation of binarized neural networks immune to device variation and voltage drop employing resistive random access memory bridges and capacitive neurons," *Communications Engineering*, vol. 3, no. 1, p. 80, Jun. 2024, doi: 10.1038/s44172-024-00226-z.

BIOGRAPHIES OF AUTHORS

Ramya Banavara Narayana is working as an Assistant Professor in the Department of Artificial Intelligence and Machine Learning at Jyothy Institute of Technology, Visvesvaraya Technological University (VTU). She is pursuing Ph.D. in machine learning from VTU. She holds an M.Tech in digital electronics and communication from MSRIT and a B.E. in Information Science from SBMSIT, Bangalore, Karnataka. She has one year of industry experience at IBM India Pvt. Ltd. and 13 years in academics. She has published 18 research papers in international journals; she is an accomplished academic professional. She can be contacted at email: ramya.bnwork@gmail.com.

Seema Singh is working as a Dean and Professor and Head of Electronics and Telecommunication Engineering at BMSIT&M, Bangalore, Karnataka. She holds a Ph.D. in neural networks and flight control systems from JNTU, Hyderabad (2015). She has 22 years of teaching and 15 years of research experience. She has authored 30 publications, a book chapter, and filed a patent in neural networks for image processing (2018). Her research interests include anomaly detection, neural networks, image processing, and biomedical engineering. A gold medalist and VTU rank holder, she has delivered expert talks, chaired conferences, and guided award-winning projects. She can be contacted at email: seema.singh1980@gmail.com.