# GaZmusino: An extended edge RISC-V core with support for Bayesian Neural Networks

Samuel Pérez Pedrajas[1]*, Javier Resano[1] and Darío Suárez Gracia[1]

[1]Department of Computer Science and Systems Engineering (DIIS), Aragon Institute for Engineering Research (I3A), University of Zaragoza.

**Abstract**

*As the demand for more transparent artificial intelligence models grows, Bayesian Neural Networks (BNN) offer a solution by enabling prediction uncertainty estimation. However, their computational requirements exceed those of traditional neural networks. This work introduces GaZmusino, a low-cost RISC-V core extended with instructions to accelerate 8.93× BNN inference.*

## Introduction

As Machine Learning (ML) models grow in complexity, their energy demands increase, raising concerns about sustainability. Shifting ML inference from data centers to edge devices reduces energy consumption and minimizes data transmission, making this change a more sustainable alternative. This paradigm is commonly known as TinyML. In addition, there is a growing demand for transparency and trust in ML models, especially in safety-critical applications where uncertainty estimation is essential. Autonomous systems, medical diagnostics, and industrial monitoring are examples where decision making under uncertainty is crucial, yet classic neural networks (NN) often show incorrect overconfident predictions [1].

Bayesian Neural Networks (BNN) provide calibrated uncertainty estimates, improving their reliability [2]. However, BNN inference is more computationally expensive than conventional NNs mainly because of their probabilistic nature. These overheads make efficient BNN deployment on low-power microcontrollers (MCU) challenging. Prior work on BNN acceleration has focused on high-end FPGA accelerators with fixed model architecture support [3]. On the contrary, this work targets running Bayesian models on the edge.

We introduce GaZmusino, an open source RISC-V processor[1] extended with instructions designed to optimize BNN inference. Additionally, we introduce an open source software toolchain[2] that can efficiently run any BNN model trained in BayesianTorch [4] taking advantage of GaZmusino. The toolchain automatically applies software optimizations, such as Batch Norm folding, and produces C code that benefits from our proposed extension when available.

*Corresponding author: samuel.perez@unizar.es
Funded by PDC2023-145851-I00 AEI/10.13039/501100011033, NextGenerationEU and PID2022-136454NB-C22
[1] https://github.com/samuprpd/GaZmusino
[2] https://github.com/samuprpd/BNN4C

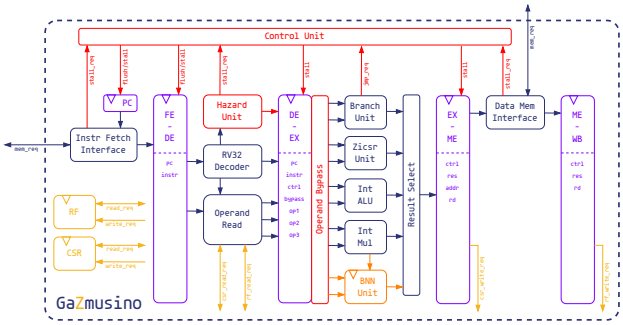


**Figure 1:** *5-stage BNN-extended pipeline of GaZmusino.*

## Optimizing BNN Inference

Since most models are trained and represented in floating point format, the first step in optimization is to transform them into fixed-point representations. A profiling analysis of our BNN inference code reveals that the processor spends approximately 80% of execution cycles sampling Gaussian distributions.

Related work has shown that simpler distributions can be used instead of Gaussian [3]. We observed that using uniform random number generation (RNG) instead of Gaussian achieves similar model performance while reducing the sampling cost. To implement this optimization efficiently, we introduce a dedicated uniform RNG functional unit in the GaZmusino execution stage. This functional unit internally uses a 39 bit 32 step look-ahead linear feedback shift register.

Additionally, since BNNs heavily rely on fixed-point arithmetic, we also add a fixed-point arithmetic unit capable of executing multiply accumulate (MAC) operations. BNN inference requires two MAC operations where traditional NNs require only one. Scaling and centering the weight sample requires one, and the standard forward pass computations requires a second one. Figure 1 shows a diagram of the extended processor pipeline. Both functional units are combined into a new functional unit, called BNN unit, drawn in orange.

These optimizations lead to the introduction of two key new instructions: one for uniform RNG, which generates uniform samples at a given fixed-point scale, and the other for fixed-point MAC operations. To complement these instructions, we add an instruction to change the generator seed. The complete list of instructions is provided below.

- `fxgen.unif rd, I`. Generates a uniform random sample and shifts it by I bits.
- `fxgen.seed ra`. Sets the seed of the generator using the value of a register.
- `fx.madd rd, ra, rb, rc, I`. Performs a fixed-point MAC operation, right-shifting by I bits.

## Evaluation

To evaluate BNN performance, we used the following metrics: accuracy (Acc ↑), reliability error (RE ↓), which measures how well model outputs can be interpreted as probabilities, and uncertainty calibration error (UCE ↓) which measures how well prediction uncertainty is related to the probability of the prediction being wrong [5].

The results shown in Table 1 demonstrate that applying the proposed optimizations does not result in significant degradation in any metric. The slight variations can be attributed to the precision loss introduced by fixed-point arithmetic or the inherent probabilistic nature of BNNs. We tested a diverse set of model architectures recognized to be suitable for TinyML or relevant in the BNN literature [6].

**Table 1:** *Optimization evaluation results. The BayesianTorch (BT) results are shown as values, the GaZmusino (GZ) results are shown as differences with BT.*

| | ↑ Acc % | | ↓ RE % | | ↓ UCE % | |
|---|---|---|---|---|---|---|
| Model | BT | GZ | BT | GZ | BT | GZ |
| HYPER | 89.46 | 0.03 | 3.93 | 0.00 | 3.31 | -0.11 |
| LENET | 62.61 | -0.38 | 2.62 | -0.75 | 4.09 | 1.35 |
| B2N2 | 75.77 | 0.17 | 2.13 | -0.54 | 2.72 | 1.86 |
| RESNET | 81.01 | -1.34 | 2.23 | -0.74 | 2.24 | 0.71 |
| Average | | -0.38 | | -0.51 | | 0.95 |
| Std. Dev. | | 0.29 | | 0.39 | | 1.02 |

The performance improvements achieved in GaZmusino are summarized in Figure 2. To analyze the cost of our implementation, we deployed GaZmusino on a Zynq UltraScale+ ZCU104 Evaluation Board FPGA, measuring resource utilization and energy consumption. Table 2 presents these results, showing that our extensions introduce only a modest increase in hardware cost while reducing the image processing energy consumption by 87.12% in average and increasing the image/J efficiency by 8.19×.

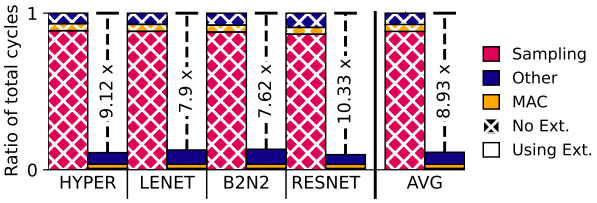


**Figure 2:** *Execution profiling of a BNN forward pass using and not using the custom extension.*

**Table 2:** *FPGA implementation results. The number of hardware components are shown as values and percentage of total available resources. BNN Ext. column shows the cost of extending the pipeline.*

| | GaZmusino | | BNN Ext. |
|---|---|---|---|
| LUTs | 2532 | 1.10% | 308 |
| Registers | 1745 | 0.38% | 44 |
| Power (W) | | 0.036 | 0.003 |

## Conclusions

Our work proposes an open source low-cost RISC-V processor alongside a lightweight ISA extension that achieves an average 8.93× speedup and 8.19× image/J efficiency increase for BNNs. Shifting the primary performance bottleneck from weight sampling and computation to control overhead. Moving forward, we aim to continue refining our approach, further narrowing the performance gap between BNNs and traditional NNs, allowing the deployment of uncertainty-aware models on the edge.

## References


[1] Chuan Guo et al. "On Calibration of Modern Neural Networks". In: (June 2017). DOI: 10.48550/ARXIV.1706.04599. arXiv: 1706.04599 [cs.LG].

[2] Charles Blundell et al. *Weight Uncertainty in Neural Networks*. 2015. DOI: 10.48550/arxiv.1505.05424. arXiv: 1505.05424 [stat.ML].

[3] Hiromitsu Awano and Masanori Hashimoto. "B2N2: Resource efficient Bayesian neural network accelerator using Bernoulli sampler on FPGA". In: *Integration* 89 (2023), pp. 1–8. ISSN: 0167-9260. DOI: 10.1016/j.vlsi.2022.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0167926022001523.

[4] Ranganath Krishnan, Pi Esposito, and Mahesh Subedar. *Bayesian-Torch: Bayesian neural network layers for uncertainty estimation*. 2022. DOI: 10.5281/ZENODO.5908307.

[5] Max-Heinrich Laves et al. "Well-calibrated Model Uncertainty with Temperature Scaling for Dropout Variational Inference". In: *CoRR* abs/1909.13550 (2019). arXiv: 1909.13550. URL: http://arxiv.org/abs/1909.13550.

[6] Colby Banbury et al. "MLPerf Tiny Benchmark". In: (June 2021). DOI: 10.48550/ARXIV.2106.07597. arXiv: 2106.07597 [cs.LG].