Formal Verification of Quantized Neural Network

Wahiba Bachiri, Yassamine Seladji, Pierre-Loïc Garoche

Formal Verification of Quantized Neural Network

Wahiba Bachiri
Department of Computer Science
Abou Bakr Belkaid University
Tlemcen, Algeria
wahiba.bachiri@univ-tlemcen.dz

Yassamine Seladji
Department of Computer Science
Abou Bakr Belkaid University
Tlemcen, Algeria
yassamine.seladji@univ-tlemcen.dz

Pierre-Loïc Garoche
National School of Civil Aviation
University of Toulouse
Toulouse, France
pierre-loic.garoche@enac.fr

***Abstract*—Neural networks have become a crucial element in modern artificial intelligence. However, despite their advancements, they often act as black boxes and can produce unexpected and incorrect results. This is why it is important to formally verify the properties of neural networks. Our work focuses on using quantization as an optimization method to accelerate the verification process. We propose a verification method for a quantized neural network (QNN) based on rational approximation of the neural network and set-based theory. We evaluate our method using the iris and HIGHWAY-ENV benchmarks, with z3 as a Satisfiability modulo theories (SMT) solver.**

***Keywords*—Formal Verification, Fixed-point arithmetic, Satisfiability Modulo Theories (SMT), Artificial Neural Network, Quantized neural network.**

I. INTRODUCTION

Great effort is currently being made to use neural networks (NNs) as controllers for safety-critical cyber-physical systems (SCCPSs) such as drones, and autonomous cars. Certifications for both security and reliability are essential in these domains. However, numerous studies have demonstrated that NNs, due to their architecture and non-linear nature, often make dangerous mistakes and are vulnerable to adversarial perturbations [1]. To mitigate this issue, various works have proposed methods for testing neural networks [2], [3]. These methods treat the neural network as a white box and generate test cases to optimize different coverage criteria, including neuron coverage, and condition/decision coverage. Testing alone does not formally guarantee that a neural network satisfies a formal specification, even when maximizing test coverage.

As a result, there is a growing interest and attention to formal verification as a means to meet this requirement. Formal verification involves ensuring the robustness and correctness of a neural network by formally capturing its correct behaviour based on its properties.

Common methods that have been widely employed to verify neural network are SMT-based approaches [4], [5] which consist of transforming the structure and properties of neural networks into propositional formulas, and solving them using SMT solvers. Consequently, most of these approaches encounter numerous challenges when dealing with large artificial neural networks (ANNs) and their complexities [5]. Another common approach is to use a method that guarantees soundness but not completeness, such as abstract interpretation. [6], [7], which aims to overapproximate the behavior of the neural network. For example, in [8], they combined symbolic interval analysis and linear relaxation to verify various safety properties of large neural networks using a tool called Neurify.

The SCCPSs based on neural network often have limited resources and, in some configurations, no floating-point units. As a result, they used quantization as an optimization technique [9] to create a quantized neural network (QNN) that compresses real neural network weights and biases and processes them using fixed-point arithmetic. While it has been empirically shown that this approach results in a small loss of accuracy, it optimizes the inference of the initial neural network [10].

The analysis of QNN remains limited and restricted to improving the efficiency of the neural network while optimizing the computations [11], [12]. However, it is not used as an optimization step to accelerate the formal verification of the neural network.

Indeed, because of the use of integers and bit-vector encoding for SMT-based verification, the exact model verification becomes a PSPACE-hard problem [13]. Instead of just reducing the encoding as proposed by [13], or transforming the verification of QNN to a mixed-integer linear problem as discussed in [14], [15], we rather use an approach inspired by the combination of set theory and SMT-based verification for verifying NN before and after quantization and improving the impact of quantization as an optimization method to accelerate the verification process.

Contributions: More specifically, we present here the following contributions:

(1) Proposing a formal verification method for verifying quantized neural networks based on rational approximation and set-based theory, using SMT-based model checking.
(2) Identifying the impact of quantization as an optimization method for accelerating the verification process of neural networks.

Outline: We summarized the structure of this paper as follows: in Section II, we highlighted the principal concepts of our work, Then, in Section III, we presented our verification method. Finally, we demonstrated our method in experimentation and results in Section IV, and we concluded by mentioning our future perspectives in Section V.

II. Background and Preliminaries

A. Fixed-point Arithmetics

Fixed-point arithmetic is used to represent real numbers by separating them into integral and fractional parts with a binary point. Both the integer and fractional parts have a predetermined length, and the position of the binary point can be adjusted as necessary [16]. $u_{\langle w,f\rangle}$ represents a fixed-point number. The total width of $u_{\langle w,f\rangle}$ is denoted by w, $w = k+f$. where k and f represent the number of bits for integer and fractional parts respectively.

We focus on the binary format of fixed-point arithmetic because it is widely used in hardware and software implementations. Additionally, developers utilize both signed and unsigned fixed-point formats depending on their intended usage. The signed or unsigned format determines whether the bit pattern representing the fixed-point number should be interpreted as a signed or unsigned integer, respectively [17]

For instance, Fig. 1 shows a 16-bit register storing a fixed-point number $u_{\langle 16,10\rangle}$. In this case, 6 bits are allocated for the integer part, including 1 bit for the sign, and 10 bits are allocated for the fractional part.

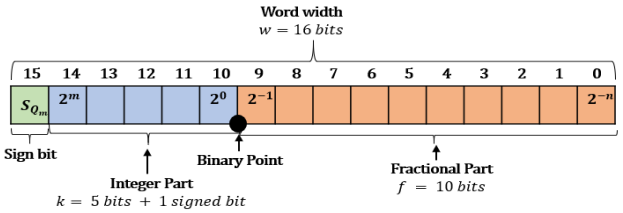

Fig. 1: Representation of Fixed-point Number $u_{\langle 16,10\rangle}$ on 16-bit.

For example, in fixed-point addition, it involves aligning the lengths of the fractional parts, denoted as f and f', of the fixed-point numbers $u_{\langle w,f\rangle}$ and $u'_{\langle w',f'\rangle}$, as shown in Fig. 2. If $f > f_c$, we truncate $f - f_c$ bits. Otherwise, we append $f_c - f$ zeros to the right of $u_{\langle w,f\rangle}$. The same procedure is applied to $u'_{\langle w',f'\rangle}$. The length of the integer part k_c should be the greater value between k and k'. If a carry occurs, we increase the number of bits in the integer part by 1. Otherwise, the result will be incorrect [18]. The detailed explanation of fixed-point operations is available in [19].

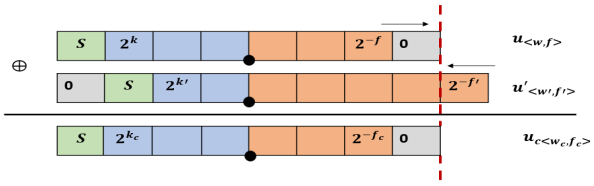

Fig. 2: Addition of two fixed-point numbers.

B. Quantized Neural Network

Among the most effective ways to reduce the computational time and energy consumption of neural networks is through quantization. Neural network quantization involves storing the weights and activation tensors with lower bit precision compared to the usual 16 or 32-bit precision used during training [19]. By converting all the neural network parameters to fixed-point numbers using a specified format of fractional and word width bits, and performing fixed-point elementary operations, quantization has been proven to minimally impact the accuracy of neural networks. This allows for the use of lower bit-widths which may lead to a decrease in accuracy. Moreover, neural network quantization can often be combined with other common optimization methods such as neural architecture search, compression, and pruning [12].

Simplistic View of QNN: we define in Eq. (1) the corresponding formula for the i-th quantized Fully-Connected layer with precision of $\langle w, f\rangle$:

$$X_i^{<w,f>} = \sigma_i^{<w,f>}(W_i^{<w,f>} \otimes X_{i-1}^{<w,f>} \oplus b_i^{<w,f>}) \quad (1)$$

where $X_{i-1}^{<w,f>}$, $W_i^{q<w,f>}$, $b_i^{<w,f>}$, and $\sigma_i^{<w,f>}$ represent quantized Input layer, weights, bias, and activation functions respectively; $\otimes$ and $\oplus$ indicates the fixed-point addition and the multiplication [20].

C. Satisfiability Modulo Theories (SMT)

In order to formally verify the initial and quantized neural network, which should always satisfy a safety property P, several approximate techniques exist for this purpose. However, this paper focuses on the formal verification of neural networks both before and after quantization using satisfiability modulo theories (SMT).

The SMT problem is a decision-making problem that involves a logic formula combining the theory and equation expressed in classical first-order logic. It can also be seen as a form of constraint satisfaction problem. To solve this problem, a verification problem is converted into a logic formula, which is then checked for its satisfiability [21].

SMT extends the Boolean satisfiability problem (SAT) by allowing more complex formulas with real numbers, integers, and data structures like lists, arrays, bit vectors, and strings. An SMT solver is a tool used to solve SMT problems. It is a constraint solver that deals with propositional logic involving predicate logic with quantifiers. The solver aims to find a solution that satisfies a given property. If a solution exists, the formula is considered satisfactory. If not, the property is deemed unsatisfiable.

III. SMT Verification of Quantized Neural Network

In this section, we highlight our method for verifying quantized neural networks in order to identify the impact of quantization on accelerating the verification of neural networks.

A. General View of Formal Verification of QNN

In this paper, we present a verification method for the quantized neural network, NN_q. Quantization is an optimization method applied on real neural network using the

$quantize(w, f)$ method. This method requires a word width and fractional bit size, as explained in Section II. Instead of directly analyzing the validity of the property P for NN_q, we verify it on its rational approximation, NN_r. This is due to the complexity of performing fixed-point elementary operations using integer or bit vector theories in SMT solvers. The error between the output sets of NN_q and NN_r is evaluated using the p-norm distance, denoted as ϵ_p (e.g. ϵ_1 for $||\cdot||_1$, ϵ_∞ for $||\cdot||_\infty$). Once we ensure the validity of P over NN_r, we add bounded noises to the output set of NN_r to create $\widetilde{NN}_r$. This perturbed rational neural network must satisfy the property P and includes both the output sets of NN_r and NN_q. If these conditions are satisfied, it guarantees that NN_q verifies P as illustrated in Fig. 3.

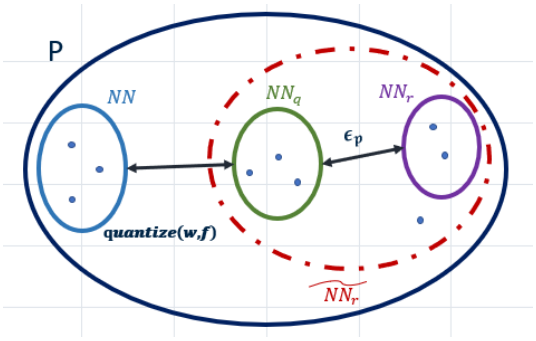

Fig. 3: General View of SMT verification of QNN

B. SMT Verification of QNN using Rational Approximation and Set-based Reasoning

a) Rational Approximation of QNN: In this paper, we have opted to verify P using an approximate rational neural network instead of directly verifying it on the NN_q. This choice is motivated by the complexity involved in performing elementary operations such as addition and multiplication using fixed-point arithmetic on the NN_q. This complexity arises from the need to incorporate shifts and avoid overflows for each operation result.

To approximate the NN_q as a rational neural network, each variable, neural net weight, or bias is associated with a format $\langle w, f\rangle$, where w represents the number of bits for the word width and $f \leq w$ represents the fractional part. Each NN_q can be expressed as a NN_r by representing all integers x in the format $\langle w, f\rangle$ using the rational scalar $x * 2^{-f}$. Similarly, all fixed-point arithmetic operations, including the use of shifts, are replaced with their rational counterparts.

b) Characterization of the Error between NN_q and NN_r and Verifying $\widetilde{NN}_r$: The approximation of NN_q by NN_r produces numerical errors between the two functions. To over-approximate the distance ϵ_p between the two image sets as illustrated in Fig. 3, we need first to select a relevant norm and then use a method to compute such distance.
While static analysis such as interval or affine arithmetics could provide sound over-approximations of such distance, in this early approach, we rely on test method to compute ϵ_p, the maximal k point-wise p-norm distance between the two neural networks. which is defined in Eq. (2):

Definition 1 (ϵ_p approximation)**.** *Let $I \subseteq \mathbb{F}^n$ be a finite set of input vectors and $card(I) = k$. ϵ_p is the maximum distance for the set of k inputs I.*

$$\epsilon_p \triangleq max_{x \in I}(||NN_r(x) - NN_q(x)||_p). \tag{2}$$

In our experiments, we used either the 1-norm and ∞-norm.

First, we check that the property holds for NN_r for all input domain I.

$$\forall x_r \in I, P(NN_r(x_r)) \tag{3}$$

We introduce bounded perturbations to the rational approximate neural network, denoted as NN_r, in order to define the perturbed rational neural network, denoted as $\widetilde{NN}_r$. This is further elaborated in Equation (4):

$$\widetilde{NN}_r \triangleq \left\{ \begin{array}{l} \forall x_r \in \mathbb{Q}^n, \exists \tilde{y}_r \in \mathbb{Q}^m, \lambda_i \in \mathbb{R}^m \text{ s.t.} \\ \tilde{y}_r = NN_r(x_r) + \lambda_i \end{array} \right\} \tag{4}$$

The sufficient condition that guarantees the validity of the property P for $\widetilde{NN}_r$ and encompasses both NN_q and NN_r output sets is defined in Eq. (5):

$$\forall x_r \in I, ||NN_r(x_r) - \widetilde{NN}_r(x_r)|| \leq \epsilon_p \implies P\left(\widetilde{NN}_r(x_r)\right) \tag{5}$$

The proof of Eq. (5) is performed with the SMT solver by searching for a model of its negation. An UNSAT result ensures its validity: P holds for NN_q.

IV. Experimentations and Results

A. Setup

We evaluated the experimentation using the SMT solver z3 [21] and performed it on the Apple M1 Ultra with 20 CPU cores and 128 GB RAM.

To evaluate our approach, we first quantized the initial neural network of the two benchmarks by setting the integer part to be 4 bits, as defined in [22, Eq. (1), page 8]. This ensures that the maximum value in the operations performed on both neural networks is represented in 3 bits for real arithmetic, with an additional 1 bit for the sign.

B. Benchmark

*a) **Iris datasets**:* The dataset comprises 50 samples for each of the three Iris species. To classify the Iris species, we utilized the MLP (Multilayer Perceptron) model with 4 inputs, 3 outputs, a hidden layer with 10 neurons, and a RELU activation function. Softmax is the activation function for the output layer. we aim to verify the property ϕ_1 as defined in Eq. (6):

$$\begin{array}{ll} \phi_1 : \forall x \in D, D \subseteq \mathbb{Q}^n & \exists y \in \mathbb{Q}^m, y = \textbf{\textit{MLP}}(x), \\ s.t. & \forall i \in \{2, \ldots, m\}, y_1 > y_i \end{array} \tag{6}$$

where D represents the ranges of each feature , n and m represent the numbers of inputs and output respectively. In this case $n = 4$, and $m = 3$. y_1 refers to the prediction value of iris-setosa.

b) **HIGHWAY-ENV***:* HIGHWAY-ENV [23] is a collection of environments specifically created for autonomous driving tasks. In our study, we use the DQN reinforcement algorithm with the highway environment, which consists of the ego vehicle (autonomous vehicle) (EV) and normal vehicles (NV), resulting in 25 observations (5 input kinematic variables x 5 vehicles) as illustrated in Fig. 4.

Fig. 4: A scene of a video recorder for `faster` action of HIGHWAY-ENV environment.

We performed our experiments on MLP policy function with two hidden layers of 80 neurons and ReLU activation function, with the five actions representing the outputs of the MLP. we aim to verify the property ϕ_2 where the MLP policy function return always `faster` Action according to a defined range feature D as illustrated in Fig. 4 and defined in Eq. (7).

$$\phi_2 : \forall x \in D, D \subseteq \mathbb{Q}^n, \exists y \in \mathbb{Q}^m, y = MLP(x), \quad s.t. \quad \forall i \in \{0, \ldots, m\}, y_3 > y_i \tag{7}$$

Where y_3 represents the `faster` action.

C. Results and Discussion

In this paper, we verify the properties ϕ_1 and ϕ_2 of the iris and HIGHWAY-ENV benchmarks, respectively, using our verification method. Our goal was to determine the impact of quantization as an optimization method for accelerating the verification process.

Before verifying the perturbed rational neural network $\widetilde{NN}_r$, we first validate the properties ϕ_1 and ϕ_2 on the rational versions NN_r.

Table I demonstrates the validity of our approach to verify quantized neural networks from 8 to 16 word-width versions, for the 1-norm and ∞-norm. We observe that the ∞-norm is less resource-intensive in terms of execution time and memory usage compared to the 1-norm, with only minor differences between their values.

TABLE I: Evaluation of our method for the IRIS benchmark using the 1-norm and ∞-norm based on the verification result (V.R.), time and memory usage. w, and f represent word width bits and fractional part bits respectively.

w	f	1-norm			∞-norm		
		V.R.	T(s)	M(mb)	V.R.	T(s)	M(mb)
8	4	✓	15.08	120.08	✓	15.08	120
9	5	✓	15.08	149.61	✓	15.05	131.35
10	6	✓	15.08	149.08	✓	15.07	149.09
11	7	✓	15.10	149.46	✓	15.09	149.48
12	8	✓	15.09	147.63	✓	15.09	154.16
13	9	✓	15.08	118.06	✓	15.07	121.48
14	10	✓	15.08	122.05	✓	15.07	128.57
15	11	✓	15.07	128.67	✓	15.08	128.68
16	12	✓	15.08	138.69	✓	15.08	138.59

In Table II, our method is considered valid with a quantized word width of 18-21 bits for the ∞-norm and 19-21 bits for the 1-norm. However, we were unable to verify the quantized neural network from 8 to 18 for the 1-norm and from 8 to 17 for the ∞-norm due to the large error ϵ_p.

TABLE II: Evaluation of our approach through 1-norm and ∞-norm of HIGHWAY-ENV benchmark.

w	f	$\widetilde{NN}_{\mathbf{r}}$ (1-norm)	ϵ_1	$\widetilde{NN}_{\mathbf{r}}$ (∞-norm)	ϵ_∞
8	4	✗	19.6473	✗	4.6103
9	5	✗	15.0099	✗	3.4724
10	6	✗	9.6597	✗	2.2583
11	7	✗	5.0505	✗	1.1793
12	8	✗	2.7093	✗	0.6356
13	9	✗	1.409	✗	0.3271
14	10	✗	0.7307	✗	0.1689
15	11	✗	0.3694	✗	0.0873
16	12	✗	0.1916	✗	0.0447
17	13	✗	0.0944	✗	0.0225
18	14	✗	0.046	✓	0.0108
19	15	✓	0.0242	✓	0.0058
20	16	✓	0.0118	✓	0.0028
21	17	✓	0.006	✓	0.0014

According to Table III, it can be observed that the ∞-norm outperforms the 1-norm in terms of both execution time and memory consumption for HIGHWAY-ENV benchmark.

TABLE III: Time and memory evaluation of our approach through 1-norms, and ∞-norms of HIGHWAY-ENV MLP.

Width bit	Frac-bit	1-norm time (s)	1-norm memory (mb)	∞-norm time (s)	∞-norm memory (mb)
18	14	-	-	314.97	78.63
19	15	565.76	79.77	269.98	79.62
20	16	744.57	81.83	312.72	81.89
21	17	1487.99	89.51	741.52	89.29

To determine the effect of quantization on accelerating the verification process, we conduct a comparison between the original neural networks of the Iris and HIGHWAY-ENV benchmarks prior to quantization, using our proposed verification method. We apply the ∞-norm and choose the word width format that results in the least execution time and memory usage as defined in Table IV.

TABLE IV: Identification of the impact of quantization on the verification process for Iris and HIGHWAY-ENV benchmarks.

Before Quantization				
Benchmarks		V. Results	Time (s)	Memory (mb)
IRIS (ϕ_1)		✓	15.13	133.75
HIGHWAY-ENV (ϕ_2)		✓	1656.25	125.30
After Quantization				
Benchmarks	p-norm	V. Results	Time (s)	Memory (mb)
IRIS (ϕ_1)	1-norm	✓	15.08	120.08
	∞-norm	✓	15.07	121.48
HIGHWAY-ENV (ϕ_2)	1-norm	✓	565.76	79.77
	∞-norm	✓	269.98	79.62

We noticed that verifying quantized neural networks using our method is sound, efficient, and less time-consuming and memory-usage compared to the SMT verification of the initial neural network before quantization. This holds true for both norms used and for both benchmarks. After quantization, the verification time is more than 5 times faster for the highway benchmarks than the verification of the initial neural network. However, there is a consistent difference of approximately 50 (MB) in terms of memory usage.

Finally, we can assume that quantization accelerates the verification process and protects the verification results for medium to large word widths, even for large neural networks.

V. Conclusion

The formal verification of neural networks using SMT is one of the major challenging tasks that we addressed in this paper. Our work consists of using quantization as an optimization method for accelerating the SMT verification process. Additionally, we proposed a formal verification of quantized neural networks based on rational approximation and set-based theory. Our method improves the soundness and efficiency of verification time and memory consumption compared to the verification of the initial neural network using SMT.

From our perspective, we aim to improve our method by exploring more complex architectures and activation functions of large neural networks.

References

[1] X. Sun, H. Khedr, and Y. Shoukry, "Formal verification of neural network controlled autonomous systems," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems:Computation and Control, Montreal, QC, Canada*. ACM, 2019, pp. 147–156.

[2] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, "Concolic testing for deep neural networks," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE, Montpellier, France*. ACM, 2018, pp. 109–119.

[3] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," *GetMobile Mob. Comput. Commun.*, vol. 22, no. 3, pp. 36–38, 2018.

[4] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification - 29th International Conference, CAV Heidelberg, Germany*. Springer, 2017, pp. 3–29.

[5] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Computer Aided Verification - 29th International Conference, CAV Heidelberg, Germany*. Springer, 2017, pp. 97–117.

[6] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. T. Vechev, "AI2: safety and robustness certification of neural networks with abstract interpretation," in *IEEE Symposium on Security and Privacy, SP , San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 3–18.

[7] O. E. Mellouki, M. I. Khedher, and M. A. El-Yacoubi, "Abstract layer for leakyrelu for neural network verification based on abstract interpretation," *IEEE Access*, vol. 11, pp. 33 401–33 413, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3263145

[8] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Efficient formal safety analysis of neural networks," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December Canada*, 2018, pp. 6369–6379.

[9] S. Chen, W. Wang, and S. J. Pan, "Deep neural network quantization via layer-wise optimization using limited training data," in *The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI, Honolulu, Hawaii, USA*. AAAI Press, 2019, pp. 3329–3336.

[10] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference," in *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops, Seoul, Korea (South)*. IEEE, 2019, pp. 3009–3018.

[11] N. Hernández, F. Almeida, and V. B. Pérez, "Optimizing convolutional neural networks for iot devices: performance and energy efficiency of quantization techniques," *J. Supercomput.*, vol. 80, no. 9, pp. 12 686–12 705, 2024.

[12] A. Kuzmin, M. Nagel, M. van Baalen, A. Behboodi, and T. Blankevoort, "Pruning vs quantization: Which is better?" in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems LA, USA*, 2023.

[13] T. A. Henzinger, M. Lechner, and D. Zikelic, "Scalable verification of quantized neural networks," in *The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 3787–3795.

[14] Y. Zhang, F. Song, and J. Sun, "Qebverif: Quantization error bound verification of neural networks," in *Computer Aided Verification - 35th International Conference, CAV, Paris, France*. Springer, 2023, pp. 413–437.

[15] Y. Zhang, Z. Zhao, G. Chen, F. Song, M. Zhang, T. Chen, and J. Sun, "QVIP: an ilp-based formal verification approach for quantized neural networks," in *37th IEEE/ACM International Conference on Automated Software Engineering, ASE, Rochester, MI, USA*. ACM, 2022, pp. 82:1–82:13.

[16] B. Parhami, "Number representation and computer arithmetic," in *Encyclopedia of Information Systems*, H. Bidgoli, Ed. Academic Press, 2002, pp. 317–333.

[17] M. S. Baranowski, S. He, M. Lechner, T. S. Nguyen, and Z. Rakamaric, "An SMT theory of fixed-point arithmetic," in *Automated Reasoning - 10th International Joint Conference, IJCAR Paris, France*. Springer, 2020, pp. 13–31.

[18] H. Benmaghnia, M. Martel, and Y. Seladji, "Fixed-point code synthesis for neural networks," *CoRR*, vol. abs/2202.02095, 2022. [Online]. Available: https://arxiv.org/abs/2202.02095

[19] M. A. Najahi, "Synthesis of certified programs in fixed-point arithmetic, and its application to linear algebra basic blocks. (synthèse de programmes certifiés en arithmètique à virgule fixe, et son application à des briques de base d'algèbre linéaire)," Ph.D. dissertation, University of Perpignan, France, 2014. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01158310

[20] S. Zhou, Y. Wang, H. Wen, Q. He, and Y. Zou, "Balanced quantization: An effective and efficient approach to quantized neural networks," *J. Comput. Sci. Technol.*, pp. 667–682, 2017.

[21] L. M. de Moura and N. S. Bjørner, "Z3: an efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS Budapest, Hungary*. Springer, 2008, pp. 337–340.

[22] P. Novac, G. B. Hacene, A. Pegatoquet, B. Miramond, and V. Gripon, "Quantization and deployment of deep neural networks on microcontrollers," *Sensors*, vol. 21, no. 9, p. 2984, 2021. [Online]. Available: https://doi.org/10.3390/s21092984

[23] E. Leurent, "An environment for autonomous driving decision-making," https://github.com/eleurent/highway-env, 2018.