Journal of Signal Processing Systems (2024) 96:301–312
https://doi.org/10.1007/s11265-024-01911-0

Floating-Point Quantization Analysis of Multi-Layer Perceptron Artificial Neural Networks

Hussein Al-Rikabi[1] · **Balázs Renczes**[1]

Received: 7 September 2023 / Revised: 30 January 2024 / Accepted: 18 February 2024 / Published online: 25 March 2024

Abstract
The impact of quantization in Multi-Layer Perceptron (MLP) Artificial Neural Networks (ANNs) is presented in this paper. In this architecture, the constant increase in size and the demand to decrease bit precision are two factors that contribute to the significant enlargement of quantization errors. We introduce an analytical tool that models the propagation of Quantization Noise Power (QNP) in floating-point MLP ANNs. Contrary to the state-of-the-art approach, which compares the exact and quantized data experimentally, the proposed algorithm can predict the QNP theoretically when the effect of operation quantization and Coefficient Quantization Error (CQE) are considered. This supports decisions in determining the required precision during the hardware design. The algorithm is flexible in handling MLP ANNs of user-defined parameters, such as size and type of activation function. Additionally, a simulation environment is built that can perform each operation on an adjustable bit precision. The accuracy of the QNP calculation is verified with two publicly available benchmarked datasets, using the default precision simulation environment as a reference.

Keywords Numerical accuracy · Artificial Neural Networks · Quantization noise · Coefficient quantization · Floating-Point arithmetic

1 Introduction

In the last couple of decades, Artificial Neural Networks (ANNs) have been playing an increasingly important role in solving digital signal processing tasks. They are nonlinear statistical data models that replicate the role of biological neural networks [1]. As ANNs show considerably high performance, besides conventional model-based methods, they have become obvious candidates for solving problems, among other applications, in the area of image processing [1, 2], regression [3–5], classification [6], and time series prediction [7–9].

During the hardware design step of ANNs, the main applied optimization criteria are accuracy, size, and speed. The number of bits representing each signal and each operation in ANNs critically affects these properties. From the introduction of the Floating-Point Double-Precision using 64-bit (FP64) number representation [10], until recently, the effect of quantization error was neglected in most cases. However, decreasing bit precision has become of increased importance in hardware design to obtain cheap and/or fast solutions [11]. Recent hardware platforms are aiming to use reduced-precision arithmetic, simplifying the data path and reducing memory storage, bandwidth, and energy requirements [9]. In parallel with this tendency, the size of ANNs is continuously growing to achieve high learning rates. The network architectures consist of hundreds or even thousands of neurons arranged in several layers [12].

Both the reduction of precision and the increase in size contribute to the phenomenon that the effect of quantization has become more significant. This makes the analysis of quantization noise a state-of-the-art issue again, which yields the motivation of this paper. In particular, we focus on Floating-Point (FP) evaluation, considering the following. Compared to the fixed-point number representation, FP is often advantageous due to the wide range it can cover. Furthermore, as FP calculations become faster and less expensive to implement, the trend toward using FP num-

✉ Hussein Al-Rikabi
rhussein@mit.bme.hu

Balázs Renczes
renczes@mit.bme.hu

[1] Department of Measurement and Information Systems, Budapest University of Technology and Economics, Műegyetem rkp. 3, Budapest H-1111, Hungary

bers is accelerating [3]. However, the analysis of quantization noise is not straightforward, as it will be demonstrated in the following sections.

There are few studies in the literature showing the quantization error effect. It was investigated extensively for systems such as the fast Fourier transform in [13], the sine-fitting algorithms for analog-to-digital converter testing in [14], and the filter-bank multicarrier transmitters in [15]. However, there is an obvious gap in the area of Multi-Layer Perceptron (MLP) ANNs. In this study, MLP ANNs will be investigated in detail from this point of view.

This paper deals with two types of errors: The Quantization Noise Power (QNP) and the Coefficient Quantization Error (CQE). The QNP is generated due to the quantization of each operation. On the other hand, the CQE yields when quantizing the constants. Both types will be thoroughly described and analyzed in Sections 3, 4, and 5. The main contributions of this study can be summarized by the following:

1. A thorough FP quantization analysis of MLP ANNs is presented.
2. A generic tool is coded capable of estimating the quantization error before the hardware implementation phase with user-defined parameters.
3. precisely calculating and compensating the CQE to achieve accurate QNP estimation.
4. The presented procedure of evaluating the QNP and the CQE can be used for other nonlinear systems.

In order to provide a concise overview of the novelty of this work, we offer an in-depth theoretical examination that delves into each operation within the network, specifically focusing on the generation and propagation of quantization error. Existing literature primarily concentrates on experimentally evaluating different quantization schemes for various neural network architectures. Some of these schemes employ minimal bit usage, such as binary or ternary formats [16, 17], while others utilize integer formats [18]. However, it is important to note that, to the best of our knowledge, the proposed analysis and methods have not yet been explored. In other words, none of the previous methods have theoretically investigated the phenomenon of quantization across a wide range of FP resolutions.

It is important to highlight that the proposed algorithm yields a theoretical bound on the quantization errors, contrary to the experimental approach of state-of-the-art solutions, as highlighted in the following.

Recently, there has been a growing interest in the performance of ANNs with respect to the applied bit resolution. Shah et al. [12] designed an MLP ANN for speech recognition applications using five- and six-bit fixed-point representation. After training the models using FP representation, they used a Post-Training Quantization (PTQ) scheme for the implementation. They experimentally observed the weight sensitivity and operation output range to decide the resolution of each weight and operation. Kristian et al. [19] implemented an MLP ANN on Field Programmable Gate Array (FPGA) using the 32-bit Single-Precision Floating-Point (FP32) representation. The implemented model was compared to a 16-bit fixed-point implementation in terms of quantization error. It was concluded that the fixed-point implementation was thirteen times smaller in logic size than the FP32 implementation. Peric et al. [20] presented a CQE analysis by comparing the FP32 and the 32-bit fixed-point number representations for the weights of the MLP ANN implementations. It was shown that both number representations yield the same quantization error. Huang et al. [5] used the 32-bit fixed-point representation to implement an MLP ANN on FPGA for a tomography application. They observed from their experiments that the fixed-point implementation slightly affects the performance of their system.

There are also studies improving quantized ANNs by performing the quantization before and during the training [21]. In our study, we perform the analysis after training the network with full resolution; that is, we face PTQ.

A quantization error analysis for a small MLP ANN containing three neurons with TanSig Activation Functions (AFs) in a single layer was presented in [22]. Only the FP32 representation was considered in that study. It was concluded that the FP32 representation is valid for the presented application, but it is not a general conclusion. The idea of generalizing this approach and creating a generic tool for arbitrary MLP ANN size and precision with different AFs has motivated us for this study.

It is essential to mention that we are analyzing MLP neural networks in this paper. Though the presented ideas can be applied to other network structures like deep convolutional neural networks and recurrent neural networks, this extension exceeds the limits of this paper and is a topic of future investigation.

The remainder of this paper is organized as follows. Section 2 gives a brief theoretical background on MLP ANNs and quantization noise. Section 3 introduces a simulation environment so that the quantization error is simulated step by step along the evaluation of network calculations on limited FP precision. Section 4 explains the proposed method to compute the approximated QNP theoretically. The CQE analysis is given in Section 5. Section 6 presents the verification of the method through the comparison of the simulated quantization error and the theoretically predicted quantization noise. This is performed on two case studies: a regression prediction model of a weather station and a classification model in cancer diagnosis. Finally, Section 7 concludes the paper.

2 Theoretical Background

This section gives a brief overview of the MLP ANNs and the quantization theorem that is necessary to characterize the QNP of the investigated networks.

2.1 Multi-Layer Perceptron (MLP) Artificial Neural Networks

An MLP ANN architecture consists of an input layer, an output layer, and hidden layers, which are fully connected. Each layer contains neurons that are inspired by the structure of the human brain [12]. A neuron is the core object in the architecture that weights the incoming inputs from a previous layer and sums the resulting weighted products. The sum of products is biased and passed to an AF as an argument to give the neuron output [23]. The AF is a critical part of the ANN because of its enormous impact on the training operation [24]. The neuron can be best described by the following equation:

$$y = F\left(b + \sum_{j=1}^{m} w_j \cdot x_j\right) , \tag{1}$$

where F denotes the AF, b is the bias, x_j is the input of the neuron, w_j is the corresponding weight, and m is the number of inputs [25].

2.2 Quantization Error

Quantization is the process of approximating a continuous signal by a set of discrete symbols or integer values. Physical quantities have infinite precision, but when digitized and inserted into a computer or a digital system, they are rounded to the nearest representable value. This process introduces a quantization error. The nearest representable value is defined by the type of the number representation. The FP representation has been commonly used for decades. It has been issued by the IEEE standard-754 [10]. An FP representation defines a value with a bit vector containing three parts: sign, exponent, and mantissa. The FP number resolution is defined by the precision, which is the number of mantissa bits, while the exponent defines the range [26].

The rounding of the FP arithmetic is considered to be an operation that follows the infinitely precise operation. The quantization error can be described with the following difference [13]:

$$v = Q(y) - y , \tag{2}$$

where v denotes the quantization error, and $Q(.)$ is the quantization function, hence $Q(y)$ is the quantized value of the operation output. As the equation shows, quantization is a deterministic process. However, after the operation, the original value cannot be retrieved anymore. Consequently, the exact quantization error remains unknown, as well.

To characterize the process, [13] suggested a model that describes quantization error as an additive noise source. This source adds a uniformly distributed quantization noise to the exact value. The width of the uniform distribution is determined by the quantizer. For the FP representation, the noise level depends on the corresponding QNP, which can be approximated as follows [13]:

$$E\{v_{\mathrm{FP}}^2\} = 0.18 \cdot 2^{-2p} \cdot E\{y^2\} , \tag{3}$$

where $E\{v_{\mathrm{FP}}^2\}$ is the expected value of the squared quantization error, that is, the QNP of the FP quantizer, $E\{y^2\}$ is the signal power at the quantizer input, and p is the precision, which is the mantissa bits. For instance, $p = 24$ for the single-precision format. It follows that having one hidden mantissa bit in the representation, the significand is 23-bit [10].

This approximation becomes handy since it is a continuous function of the input level, contrary to the exact value, which is a staircase function. During the analysis, we, therefore, apply this approximation. A comparison between the approximate and the exact characteristics is shown in Fig. 1. Since the quantization noise is modeled as a random variable, we will perform statistical analysis to determine the QNP. Based on this result, error bounds will be presented to characterize output inaccuracy.

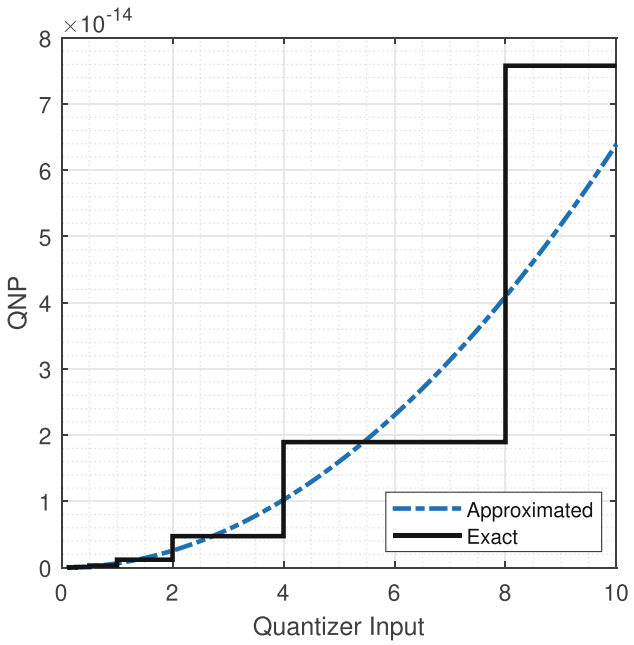

Figure 1 A comparison between the approximated and the exact characteristics of the QNP in the FP representation with $p = 24$.

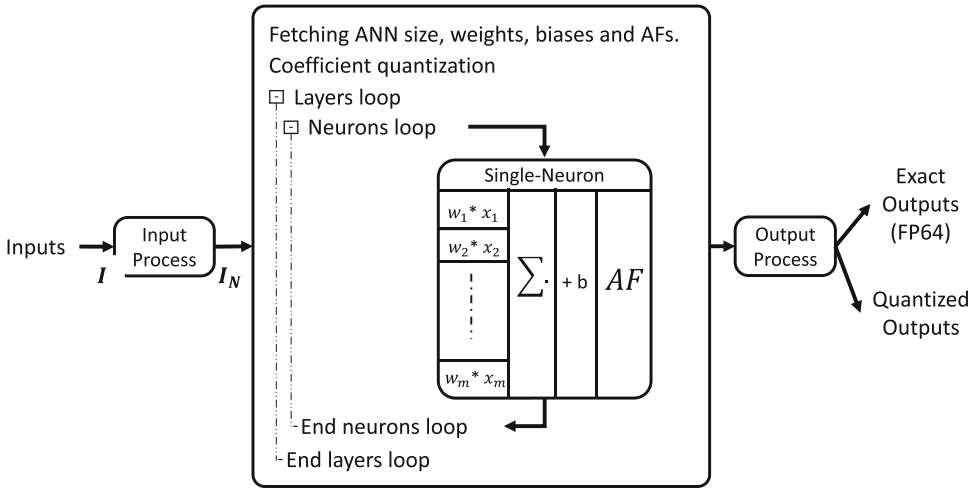

Figure 2 Block diagram of the proposed method simulation.

It is important to mention here that it is not possible to change the precision of an FP number representation using the built-in MATLAB toolbox, so we used the toolbox provided in [13] for the FP quantization with variable precision. The quantizer resolution is basically defined by the precision, while the exponent does not affect the quantization error in the case of no overflow; therefore, the exponent only defines the range. The rounding direction used in this work is the default round to the nearest method. Besides the proposed method being the default setting for most practical applications, analyzing other rounding techniques, like rounding towards infinity or towards zero, would also require new quantization approximations, other than (3). This exceeds the limits of this study.

3 Computation of Quantization Noise in MLP ANNs

After the brief theoretical overview, the QNP in MLP ANNs is simulated in this section. Our aim is to provide a flexible solution that can handle MLP models with different sizes, precisions, and AFs. As we have described in Section 2, this exact value is not available in quantized systems due to the irreversible nature of quantization. However, during simulations, FP64 arithmetics can be applied as a reference, and we can follow the propagation of the error in the system by comparing the FP64 (exact) and limited precision results.

The block diagram of the proposed algorithm is depicted in Fig. 2. It can be seen that three main stages are to be performed, which are the input process, processing of layers, and output process. The input process is a normalization step that maps the input dataset I to normalized values I_N of the range $[-1; +1]$ as follows:

$$I_N = \left[(I - I_{\min}) \cdot \frac{2}{I_{\max} - I_{\min}} \right] - 1 \,, \tag{4}$$

where $I_{\max}$ and $I_{\min}$ represent the range of the input training dataset I.

The output process has the same operations in the reverse order that denormalizes the data back to the original range. By this means, the parameters and operations between the input/output process blocks are not restricted to the normalization range. However, after normalization, the values may increase depending on the parameters but will not increase massively as the input values are normalized.

In between these processes, the values of the exact and quantized neuron outputs are calculated and saved to the next layer. By this means, we obtain the final network quantized and exact outputs, and the difference between them is the quantization error. The square of that error is the simulated QNP to be compared with the theoretical approximation presented in Section 4.

Figure 3 shows how each operation in the single-neuron block is quantized, where Q-denoted blocks represent the quantization function described in (2). Similarly, all other operations in the model are quantized. By this means, the algorithm can be used generically to simulate and analyze MLP networks with an arbitrary size. The simulated quantization error of a neuron is computed by performing the following actions.

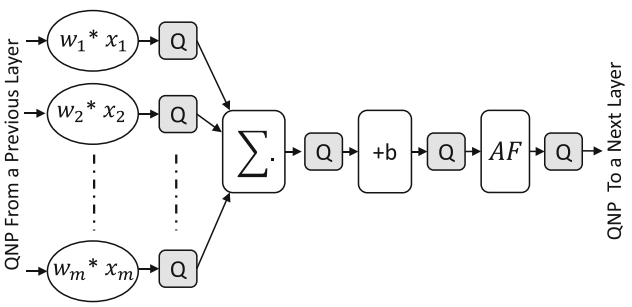

Figure 3 Single-neuron block diagram showing the quantization process after each operation.

Table 1 List of AFs included in this study and the corresponding number of quantizers.

Activation function	Explanation of abbreviation	Number of quantizers
TanSig	Hyperbolic Tangent Sigmoid	4
LogSig	Log Sigmoid	3
RadBas	Radial Basis	2
TriBas	Triangular Basis	1
PureLin	Linear	0
SatLin	Saturating Linear	0
SatLins	Symmetric Saturating Linear	0
PosLin	Positive Linear (ReLU)	0
HardLim	Hard Limit	0
HardLims	Symmetric Hard Limit	0

First, we retrieve characteristic data of the network, and we quantize the coefficients of the network to the investigated precision. The CQE is a rather different type of error that will be investigated in Section 5. Then, in two embedded cycles, neurons are processed layer by layer. In accordance with (1), the inputs are weighted, and the weighted sum is calculated and added to the bias. Finally, the result is led through an AF. After each operation, the result is quantized to the targeted precision. In parallel, the operations are also performed on FP64 as a reference. By this means, we gain control of the quantization error so that we will be able to verify the effectiveness of the theoretical approximation that will be provided in Section 4.

During the simulation, the algorithm can handle ten commonly used AFs delineated in Table 1. The table also includes the number of quantization operations needed to simulate the behavior of the network on a limited precision. It is important to mention that the Rectified Linear Unit (ReLU) AF is, by definition, a PosLin AF. The block diagrams of three different AFs are depicted in Fig. 4. Since the other AFs have much simpler diagrams which are mostly saturation operations, we did not present them in this figure. Note that the number of blocks might be higher than the corresponding number of quantizers. For instance, the TanSig function consists of six blocks. Among these, the multiplication by ± 2 does not inject an extra QNP since these operations only affect the exponent and the sign of the FP number. This decreases the number of needed quantizers to four. However, these two blocks also contribute to the propagation of quantization errors. As a part of the theoretical QNP analysis, this will be discussed in detail in Section 4.

4 Theoretical Analysis of QNP

In this section, we analyze the QNP of an MLP network from a theoretical point of view. To this aim, the QNP is computed after each operation as a sum of two noise sources. The first noise source originates from the QNP of the previous operations. This source is shaped by the actual operation. In other words, it is propagated. It will be discussed in more detail later in this section.

The second noise source models the quantization behavior of the current operation as a uniformly distributed random variable, as discussed in Section 2. During the analysis, as the quantization noise variables are independent of one another, the variance after each operation can be determined as the sum of the variances of these two sources.

Since the quantization noise is zero-mean, after each operation, the expected value will be the true value. Due to this property, the variance of the quantization noise and the QNP, which is the squared expected value, also coincide. If we propagate all the quantization noise sources through the network, at the output, we will obtain the cumulative QNP. This is beneficial since we will be able to define an error bound for

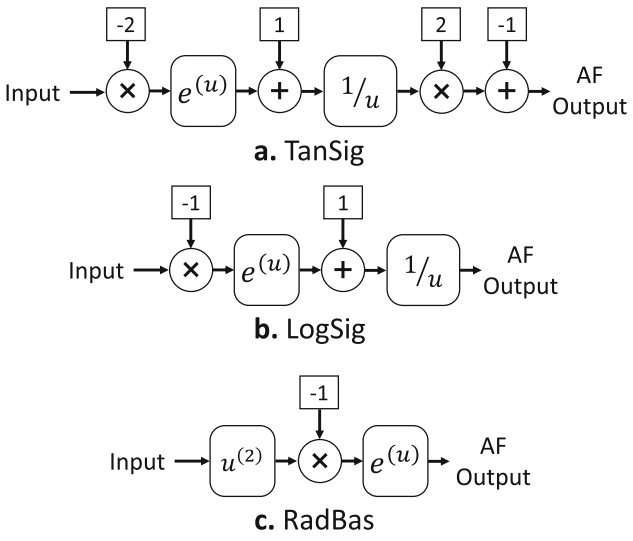

Figure 4 Block diagrams of **a**. TanSig, **b**. LogSig and **c**. RadBas AFs, where u represents an input of the corresponding operation.

the quantized network output that was computed in Section 3, considering the following. As we have several operations in the network, we can assume that the resulting QNP will follow Gaussian distribution according to the Central Limit Theorem. The standard deviation equals the square root of the QNP. With this assumption, we can claim with 99.7% confidence that the exact output is in the $\pm 3\sqrt{\text{QNP}}$ range of the quantized output.

In the calculations, the QNP injected by the current quantizer is approximated by (3). This is the first noise source. As described above, the QNP of the previous operations is shaped by the current operation, which is the second source. In order to calculate the propagated quantization noise through the network, let a general function G be a function of input u. With the operating point linearization [27, 28], we have that

$$\Delta G(u) \approx \frac{\partial G(u)}{\partial u} \cdot \Delta u \ . \tag{5}$$

In our case, u is a random variable with an expected value of the true value and an additive quantization noise from previous stages. Let us denote the standard deviation of this noise by σ_u. As the linear assumption applies, the standard deviation of $G(u)$ will also be in a linear relationship with the standard deviation of the input, and accordingly, the variance is

$$\sigma^2_{G(u)} \approx \left(\frac{\partial G(u)}{\partial u}\right)^2 \cdot \sigma_u^2 \ . \tag{6}$$

Utilizing this approach, the variance propagation formulas are derived for the used operations in this study, and they are listed in Table 2.

To demonstrate the calculation of the QNP at the output of the network, let us investigate the computation steps for a single neuron. For the needed operations, we refer to (1) and Fig. 2. The evaluation steps are as follows:

1. Variance after weighting. Since the neuron receives inputs from either a previous layer or from the normalization stage, propagated error variances are obtained, and we propagate the error according to the multiplication propagation formula. The output variance is computed as the sum of the variance due to error propagation and the variance due to the quantization after multiplication. This variance is propagated to step 2.
2. Variance after summation. For every operand to be summed, we obtain the variance from step 1 and propagate the error according to the addition propagation formula. Furthermore, after each addition, the variance due to quantization is added. The resulting accumulated variance is propagated to step 3.
3. Variance after adding the bias. We obtain the variance from step 2 and propagate the error according to the addition propagation formula. Variance due to quantization is added, and the resulting accumulated variance is propagated to step 4.
4. Variance after the AF. We obtain the variance from step 3 and lead it to the input of the first block of the AF. We propagate the error through this block with the corresponding error propagation formula in Table 2. Variance due to quantization is added, and the resulting accumulated variance is propagated to the next block in the AF. This is repeated until the last block is evaluated. The accumulated variance is propagated to the next layer and will be an input for step 1.

Table 2 List of operations along with the corresponding propagation formula.

Operation	Variance Propagation Formula
$G(u) = \text{constant}$	$\sigma^2_{G(u)} = 0$
$G(u) = u + \text{constant}$	$\sigma^2_{G(u)} = \sigma_u^2$
$G(u) = u \cdot \text{constant}$	$\sigma^2_{G(u)} = (\text{constant})^2 \cdot \sigma_u^2$
$G(u) = u^2$	$\sigma^2_{G(u)} = 4G(u) \cdot \sigma_u^2$
$G(u) = e^u$	$\sigma^2_{G(u)} = G^2(u) \cdot \sigma_u^2$
$G(u) = 1/u$	$\sigma^2_{G(u)} = G^4(u) \cdot \sigma_u^2$

Following these steps, the computations can also be demonstrated using mathematical equations. Referring to (3), Fig. 3, and Table 2, the theoretical estimation of the QNP can be computed for the neuron. Let the outputs of the first three stages be; $out_{1,i}$, out_2 and out_3 respectively, then the QNP is propagated as follows:

$$\text{QNP}_{1,i} = w_i^2 \cdot \text{QNP}_{x,i} + 0.18 \cdot 2^{-2p} \cdot E\{out_{1,i}^2\} \tag{7}$$

$$\text{QNP}_2 = \sum_{i=1}^{m} \text{QNP}_{1,i} + 0.18 \cdot 2^{-2p} \cdot E\{out_2^2\} \tag{8}$$

$$\text{QNP}_3 = \text{QNP}_2 + 0.18 \cdot 2^{-2p} \cdot E\{out_3^2\} \tag{9}$$

where m is the number of inputs to this neuron, hence the number of multiplications.

It can be seen from (8) that the summation operation produces a quantization error only once at the final result of the summation, but this depends on the type of the accumulator. Some accumulators in different micro-controllers accumulate the summation step by step. Because ANNs are mostly implemented in parallel hardware platforms like FPGAs, we assume it to be a parallel accumulator. However, the analysis and derivation of the QNP can be performed using the sequential accumulator.

For the fourth stage in the neuron, the AF is a generic block. Let us assume that it is a LogSig AF, referring to Fig. 4b, let the outputs of the AF operations be AF_{op1}, AF_{op2}, AF_{op3}, and AF_{op4}, then the QNP is propagated through the AF respectively as follows:

$$\mathrm{QNP}_{\mathrm{LogSig},1} = \mathrm{QNP}_3 \tag{10}$$

$$\begin{aligned}\mathrm{QNP}_{\mathrm{LogSig},2} &= \mathrm{QNP}_{\mathrm{LogSig},1} \cdot AF_{op2}^2 \\ &\quad + 0.18 \cdot 2^{-2p} \cdot E\{AF_{op2}^2\}.\end{aligned} \tag{11}$$

$$\begin{aligned}\mathrm{QNP}_{\mathrm{LogSig},3} &= \mathrm{QNP}_{\mathrm{LogSig},2} \\ &\quad + 0.18 \cdot 2^{-2p} \cdot E\{AF_{op3}^2\}\end{aligned} \tag{12}$$

$$\begin{aligned}\mathrm{QNP}_{\mathrm{LogSig},4} &= \mathrm{QNP}_{\mathrm{LogSig},3} \cdot AF_{op4}^4 \\ &\quad + 0.18 \cdot 2^{-2p} \cdot E\{AF_{op4}^2\}\end{aligned} \tag{13}$$

It should be mentioned that due to the error propagation, the QNP is not necessarily growing after every operation. On the one hand, there are steps that keep the accumulated QNP constant. These are the error propagation due to multiplication by ± 1 and when the AF is linear. On the other hand, the QNP due to error propagation can decrease, as well, depending on the input operand. For instance, multiplication by a constant with an absolute value below one will decrease the accumulated QNP. Additionally, the QNP at the output of an AF can be zero when the AF has constant output or, in other words, saturated. Namely, the derivative of a constant is zero.

5 Coefficient Quantization Error Analysis

The QNP was analyzed and approximated in the previous sections, yet the impact of the coefficient errors was never properly addressed. This section delves into the CQE; hence, a method of calculating and neutralizing that error will be presented.

This type of error contributes to the total error at the output by shifting or biasing each sample. This biasing behavior happens due to the nature of the CQE; that is, the error between the coefficient exact and quantized values is fixed when different input samples are introduced. These small biasing errors are accumulated and propagated to the output from each coefficient in the model. By this means, each sample at the model output is biased. It is crucial to preserve the sign of that biasing so that this phenomenon can be compensated. As described in the literature, this type of error can not be approximated theoretically for nonlinear systems and can only be calculated using simulation [13].

To calculate the CQE, a model has been configured so that only the coefficients are quantized while operations are performed in default resolution (FP64). By this means, the output of this model can be compared to the exact model, and the difference is simply the CQE. Unfortunately, this method can not be used when operation quantization is considered, but it is still useful to verify the next method.

A more systematic and general method is presented in this paper. Each coefficient generates a CQE that propagates through different operations to affect the final output. It is required to compute the propagated CQE after each operation; hence, error propagation formulas are derived for all operations used in this study, and they are presented in Table 3. The first three formulas in the table are special propagation formulas that apply only to quantization error propagation [13], while the rest of the formulas are derived using (5) [27, 28].

To verify the formulas derived in Table 3, the results from the two methods are compared, and the error between them is negligible, as can be seen in Fig. 5. The CQE for the two methods in this experiment is in the range of 10^{-4}, while the difference between them for all samples is in the range of 10^{-7}. Hence, the accuracy is 99.99%, which has been achieved in all performed experiments with different simulation parameters. This observation confirms the applicability of the general method with high accuracy.

The general method presented has been used when fully quantized MLP ANNs are analyzed. By this means, the CQE at the output is known; hence, it can be compensated at the model output.

After discussing the QNP and the CQE, part of the code for the RadBas AF is shown in Fig. 6 to demonstrate how the algorithm works. There are three operations in this AF presented in Fig. 4c. For each operation, the quantized out-

Table 3 Error Propagation formulas used in the CQE calculation.

Operation	Error propagation
$G(u) = \pm u \pm \text{constant}(c)$	$\sigma_{G(u)} = \pm\sigma_u \pm \sigma_c$
$G(u) = \pm u_1 \pm u_2 \pm \ldots$	$\sigma_{G(u)} = \pm\sigma_{u1} \pm \sigma_{u2} \pm \ldots$
$G(u) = u \cdot \text{constant}(c)$	$\sigma_{G(u)} = (c \cdot \sigma_u) + (u \cdot \sigma_c)$
$G(u) = e^u$	$\sigma_{G(u)} = G(u) \cdot \sigma_u$
$G(u) = u^2$	$\sigma_{G(u)} = 2 \cdot G(u) \cdot (\sigma_u / u)$
$G(u) = 1/u$	$\sigma_{G(u)} = -1 \cdot (\sigma_u / u^2)$
$G(u) = \lvert u \rvert$	$\sigma_{G(u)} = \sigma_u$, when(u > 0)
	$\sigma_{G(u)} = -\sigma_u$, when(u < 0)
$G(u) = PosLin(u)$	$\sigma_{G(u)} = \sigma_u$, when(u > 0)
	$\sigma_{G(u)} = 0$, when(u < 0)
$G(u) = SatLin(u)$	$\sigma_{G(u)} = \sigma_u$, when(0 < u < 1)
	$\sigma_{G(u)} = 0$, Otherwise
$G(u) = SatLins(u)$	$\sigma_{G(u)} = \sigma_u$, when(|u| < 1)
	$\sigma_{G(u)} = 0$, Otherwise
$G(u) = HardLim(u)$	$\sigma_{G(u)} = 0$
$G(u) = HardLims(u)$	$\sigma_{G(u)} = 0$

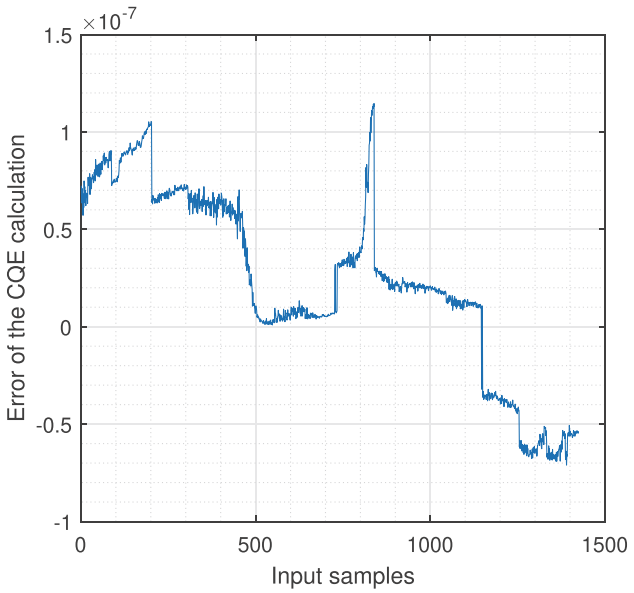

Figure 5 The difference between the calculated CQE based on FP64 operations and the proposed error propagation.

put is calculated then the QNP is estimated along with the CQE calculation. By this means, the values are saved to the next layer.

6 Case Studies

In this section, the proposed analysis will be carried out on two publicly available datasets. To verify the applicability of the method, the results of the theoretical analysis, described in Section 4 will be compared to the simulated quantization errors, as it was demonstrated in Section 3. For this comparison, the CQE, which was presented in Section 5, is also considered.

6.1 Case Study 1: Weather Station

The first dataset is taken from ThingSpeak™ channel 12397 of a weather station in Natick, Massachusetts, USA. As this channel stores new readings every day, we used the data stored for one week, starting from (Jul 1, 2023) to (Jul 8, 2023). The data can be accessed using MATLAB's "thingSpeakRead" command, specifying the channel ID, duration, and attributes. In this dataset, the humidity, temperature, atmospheric pressure, and wind speed are the attributes used to calculate the dew point [29].

Different MLP ANN training setup experiments have been used for this analysis, including the number of neurons, the number of layers, and the type of AF in each layer. The objective is to monitor the behavior of quantization noise for different MLP structures. So, at first, a small network was trained that has ten neurons in one hidden layer with a LogSig AF, in addition to the default output layer of a single neuron with PureLin AF.

The dataset consists of 8000 samples. It has been divided into two equal parts: the first half was used for training, validation, and test purposes with (80,10,10) percentages, respectively. The second half was kept for the quantization error analysis. FP quantization has been carried out with the single-precision, that is, with 24-bit precision, 8-bit exponent, and the default rounding to the nearest method. The theoretical and simulated QNP mean values at the final model output are $2.8 \cdot 10^{-11}$ and $2.95 \cdot 10^{-11}$, respectively.

More illustration of the quantization behavior can be seen from the histograms shown in Fig. 7. The figure plots two histograms for the number of occurrences of the mean square of the quantization error as a function of the theoretical standard deviation. To interpret the results, let us recall the assumption that the quantization noise at the output is of Gaussian distribution with zero means. For each input instance, the quantization error at the output (ν) is normalized by the standard deviation of the Gaussian random variable ($\sqrt{\text{QNP}}$). In other words, the random variables are standardized. The first histogram, with the circle marker, is plotted when the CQE is neutralized. The figure shows that the actual errors are mostly in the ($\pm 3 \cdot \nu/\sqrt{\text{QNP}}$) range, and the shape of the curve is Gaussian with a mean around zero. On the other hand, the second histogram is presented to emphasize the impact of the CQE, where it can be seen that the histogram is shifted due to the CQE at each output sample. For this reason, it is crucial to compensate for that phenomenon.

To observe the effect of the precision on the QNP, the same trained ANN has been simulated with different precisions from 3 to 52 bits shown in Fig. 8. Recall that FP64, which has 53-bit precision, is used as a reference. Thus, for $p = 53$, the error would be exactly 0. At each precision, the means of the actual and approximated QNPs are shown. As a side note, we can observe that, as expected, the QNP decreases to the fourth of the original value if the precision is increased by one. This is a natural consequence of the properties of quantization. As the vertical scale is logarithmic, the QNP, as a function of p, can be characterized by a straight line.

In order to address the effect of quantization in a large MLP ANN, a model with nine layers was trained using the same dataset. By definition, this is a deep model in which each layer contains neurons organized as follows: (5, 10, 20, 40, 30, 20, 10, 5, 3) respectively, and all of these layers have TanSig AFs. This model has been simulated with the half-precision FP representation that has 11-bit precision, including the hidden bit, and the mean values of the simulated and estimated QNP are both equal to $1.27 \cdot 10^{-2}$.

The relation between the number of layers and quantization error can not be exactly calculated because many other parameters influence the quantization phenomena, like the

```
% Definitions:
% The (q) letter denotes the quantizaed cycle.
% Input(q) / Output(q) are AF terminals.
% QNPin and CQEin are from previous operations.
% roundq is the quantizer function.
% j is the neuron index in the layer loop.
% Q is the predefined quantizer of:
% p is Targeted reduced precision.

% Cycle 1: Defult bit resolution (FP64)
op1   = Input.^2;
op2   = (-1)*op1;
op3   = exp(op2);
Output(j,:) = op3; % Saving neuron output to next layer.

% Cycle 2: Reduced bit resolution
op1q   = Inputq.^2;     % First RadBas operation.
QNP1 = (4.*op1q.*QNPin) + 0.18*(2^(-2*p))*((op1q.^2));
CQE1 = 2*op1q.*(CQEin./Inputq);
op1q   = roundq(op1q,Q);
op2q   = (-1)*op1q;      % Second RadBas operation.
CQE2 = CQE1.*(-1);
op3q   = exp(op2q);     % Third RadBas operation.
QNP3 = QNP1.*((op3q.^2)) + 0.18*(2^(-2*p))*((op3q.^2));
CQE3 = op3q.*CQE2;
op3q   = roundq(op3q,Q);
Outputq(j,:) = op3q;     % Saving neuron output, QNP
QNPout(j,:) = QNP3;    % and CQE to next layer.
CQEout(j,:) = CQE3;
```

Figure 6 Part of the algorithm code showing the calculation of the two cycles, the approximated QNP, and the calculation of the CQE, for the RadBas AF.

number of neurons, the values of the coefficients, and the type of AF. Using the proposed method can be quite beneficial because it can characterize the QNP of the investigated network structure precisely.

6.2 Case Study 2: Cancer Diagnosis

The second dataset was collected to perform classification in cancer diagnosis. Nine input parameters (size, shape, etc.) and two classes embedded in one output, which decides if the case is a benign or malignant type of cancer, were considered in this dataset. The measured data is publicly available in [30]. It contains 698 measured cases divided into two equal parts, similarly as in Section 6.1.

An ANN of ten neurons in one hidden layer with a Tan-Sig AF was trained to perform the classification. As this case study deals with a classification problem, it is essential to see when the ANN misclassifies the data and gives a faulty diagnosis. Figure 9 shows the number of faulty diagnosis samples out of the 349 samples. At 5-bit precision or less, the network misdiagnosed at least one case. Keeping in mind that the number of exponent bits does not affect the quantization error as described in Section 2, for that reason, we focus only on the precision. In this experiment, we employed 4-bit for the exponent, which is sufficient to avoid an overflow while changing the precision. For example, at three precision bits, there are two misclassified samples. To explain how these two samples have been misclassified, Fig. 10 is presented, in

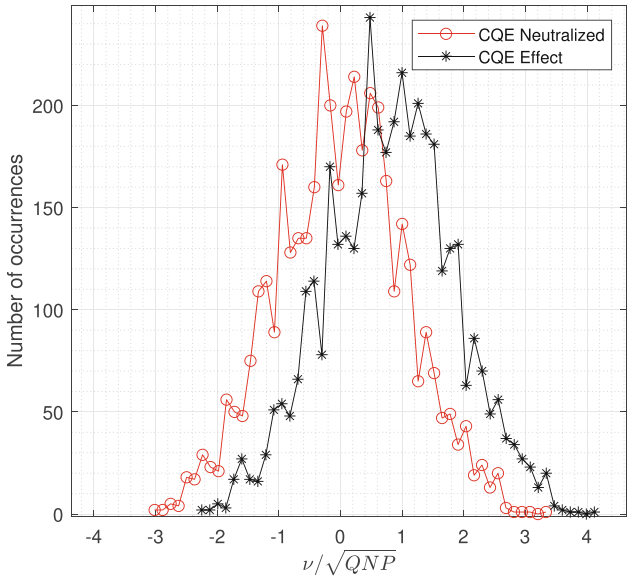

Figure 7 Weather station case study: Histograms of the mean square of the exact simulated QNP and the theoretical standard deviation for two cases with and without the compensation for the CQE.

which the closest four samples to the threshold are observed. It can be seen that for the second sample, the exact value is 0.4, but due to quantization, it has become 0.5, which is on the threshold, and in the classification operation, it is rounded up to one. A similar situation can be seen in the fourth sample, while the other samples are correctly classified. This demonstration clearly shows the importance of the analysis: with the error bounds, the user can be notified that the result is prone to classification error just due to quantization.

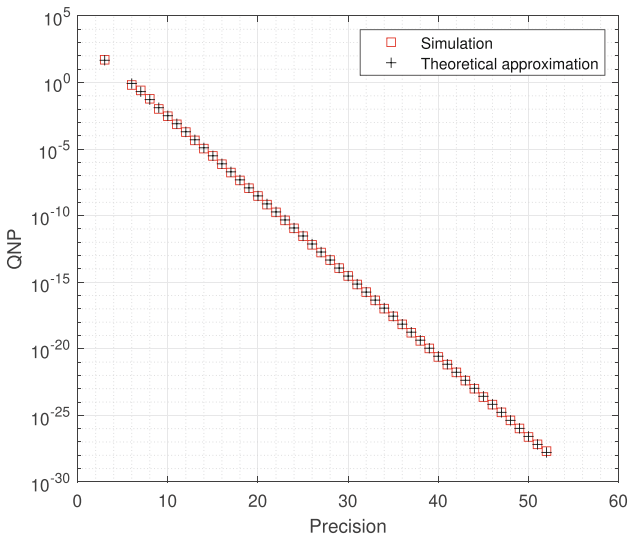

Figure 8 Weather station case study: The effect of decreasing the precision on the QNP.

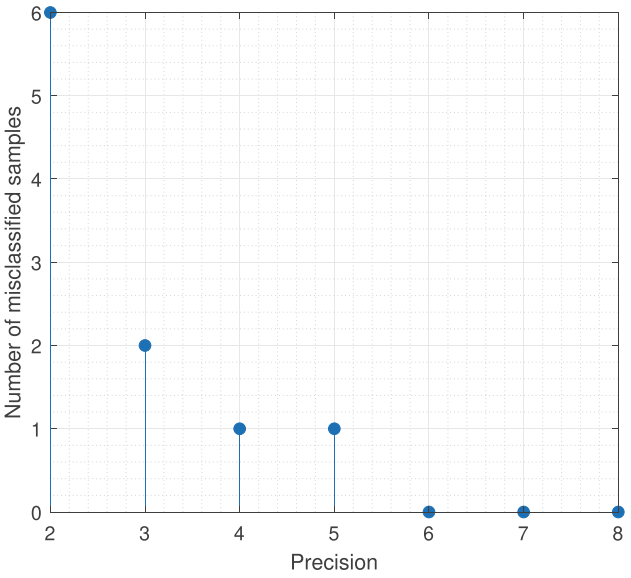

Figure 9 Cancer diagnosis case study: The effect of decreasing the precision on the classification.

In these two case studies, TanSig, LogSig, RadBas, and PureLin AFs have been used and analyzed. The other AFs listed in Table 1 are less involved. Their behavior can be described as follows. The TriBas has only one addition operation that yields QNP and propagates errors from previous stages, but it is saturated when the input exceeds the range ± 1.

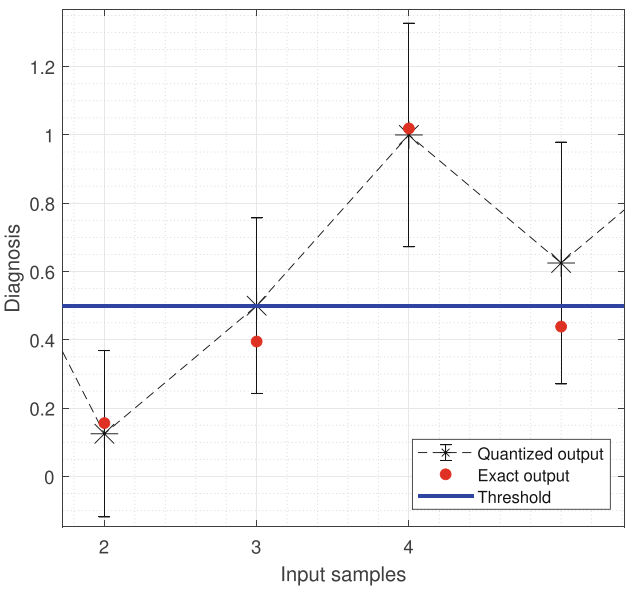

Figure 10 Cancer diagnosis case study: Output response for four samples to illustrate the quantization error effect on the classification decision when using three precision bits. The standard deviation error bar is also shown at each point.

Hence, at saturation, the QNP at the output of the AF is zero, as mentioned earlier. The Satlin, SatLins, and PosLin (ReLU) AFs are linear with saturation. In the saturation phase, they produce zero quantization error. In the linear phase, they do not produce any additional quantization error, and only the input error is propagated. Finally, the HardLim and HardLims are always at saturation, which makes the QNP at their output zero. These saturation AFs also contribute to the propagation of the CQE, as listed in Table 3.

7 Conclusions

In this study, MLP ANNs have been analyzed in terms of FP quantization. The analysis includes a statistical approximation of the QNP and a precise calculation of the CQE. Hence, the effect of the CQE can be observed and handled for better QNP approximation.

A generic algorithm was presented to calculate the QNP and the CQE of a reduced precision network with FP number representation. The proposed algorithm covers ten AFs that are commonly used in recent applications. In operation-wise computations, the QNP and CQE were propagated through the model, employing the mentioned necessary formula at each step.

Two case studies were considered for regression and classification models. The coded algorithm could precisely predict the QNP in both cases in a wide range of applied precision bits. In general, MLP networks can show high nonlinear behavior, which makes it hard to determine the quantization error at the output of the network. The algorithm described in this paper aims to facilitate the hardware design in a sense that, on a given precision, it yields the error bounds of the output. This can be performed before the actual implementation. By this means, the hardware designer can select the sufficient resolution appropriate for the application in the designing phase.

Author Contributions The authors have confirmed their contributions to the paper as follows: writing of manuscript, coding and design: Al-Rikabi, Hussein; Study conception, analysis and interpretation of results, draft manuscript preparation and the whole project was supervised by Dr. Renczes, Balázs. All authors have reviewed the results and have given their approval for the final version of the manuscript.

Funding Open access funding provided by Budapest University of Technology and Economics. Project no. TKP2021-EGA-02 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-EGA funding scheme.

Availability of Data and Code The source code and datasets generated during and/or analysed during the current study are available from the corresponding authors on reasonable request.

Declarations

Ethics Approval I declare the following: the manuscript is not published nor under consideration elsewhere; it is not part of another published paper; it has no Plagiarism; All data and resources used are approved; and the software used is licensed.

Competing Interests The authors declare that there are no relevant financial or non-financial competing interests to report.

References

1. Chao, Z., & Kim, H. J. (2020). Brain image segmentation based on the hybrid of back propagation neural network and AdaBoost system. *Journal of Signal Processing Systems, 92*, 289–298.
2. Sahoo, M., Dey, S., Sahoo, S., Das, A., Ray, A., Nayak, S., & Subudhi, E. (2023). MLP (multi-layer perceptron) and RBF (radial basis function) neural network approach for estimating and optimizing 6-gingerol content in Zingiber officinale Rosc. in different agro-climatic conditions. *Industrial Crops and Products, 198*, 116658.
3. Yin, P., Wang, C., Liu, W., Swartzlander, E. E., & Lombardi, F. (2018). Designs of approximate floating-point multipliers with variable accuracy for error-tolerant applications. *Journal of Signal Processing Systems, 90*, 641–654.
4. Barrachina, J. A., Ren, C., Morisseau, C., Vieillard, G., & Ovarlez, J. P. (2023). Comparison between equivalent architectures of complex-valued and real-valued neural networks-application on polarimetric SAR image segmentation. *Journal of Signal Processing Systems, 95*(1), 57–66.
5. Huang, A., Cao, Z., Wang, C., Wen, J., Lu, F., & Xu, L. (2021). An FPGA-based on-chip neural network for TDLAS tomography in dynamic flames. *IEEE Transactions on Instrumentation and Measurement, 70*, 1–11.
6. Garg, M., Arora, A., & Gupta, S. (2021). An efficient human identification through Iris recognition system. *Journal of Signal Processing Systems, 93*, 701–708.
7. Yu, H., Shou, G., Zhang, X., Li, H., Liu, Y., & Hu, Y. (2023). Application of neural networks for predicting UTC local time-scale with clock ensemble. *IEEE Transactions on Instrumentation and Measurement, 72*, 1–9.
8. Chinatamby, P., & Jewaratnam, J. (2023). A performance comparison study on PM2.5 prediction at industrial areas using different training algorithms of feedforward-backpropagation neural network (FBNN). *Chemosphere, 317*, 137788.

9. Hassan, O., Paul, T., Shuvo, M. H., Parvin, D., Thakker, R., Chen, M., Mosa, A. S. M., & Islam, S. K. (2022). Energy efficient deep learning inference embedded on FPGA for sleep apnea detection. *Journal of Signal Processing Systems*, *94*(6), 609–619. https://doi.org/10.1007/s11265-021-01722-7, https://link.springer.com/article/10.1007/s11265-021-01722-7
10. IEEE. (2019). *Standard for binary floating-point arithmetic*. Std 754-2019, IEEE.
11. Liang, S., Yin, S., Liu, L., Luk, W., & Wei, S. (2018). FP-BNN: Binarized neural network on FPGA. *Neurocomputing, 275*, 1072–1086.
12. Shah, M., Arunachalam, S., Wang, J., Blaauw, D., Sylvester, D., Kim, H. S., Seo, J. S., & Chakrabarti, C. (2018). A fixed-point neural network architecture for speech applications on resource constrained hardware. *Journal of Signal Processing Systems, 90*, 727–741.
13. Widrow, B., & Kollár, I. (2008). *Quantization noise: roundoff error in digital computation, signal processing, control, and communications*. Cambridge, UK: Cambridge University Press.
14. Renczes, B. (2017). Accurate floating-point argument calculation for sine-fitting algorithms. *IEEE Transactions on Instrumentation and Measurement, 66*(11), 2988–2996.
15. Alrwashdeh, M., & Kollár, Z. (2022). Analysis of quantization noise in FBMC transmitters. *Digital Signal Processing, 131*, 103760.
16. Huang, K., Ni, B., & Yang, X. (2019). Efficient quantization for neural networks with binary weights and low bitwidth activations. *Proceedings of the AAAI Conference on Artificial Intelligence, 33*(1), 3854–3861.
17. Zhu, C., Han, S., Mao, H., & Dally, W. J. (2016). Trained ternary quantization. Preprint retrieved from http://arxiv.org/abs/1612.01064. International Conference on Learning Representations (ICLR) (2017).
18. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). *Quantization and training of neural networks for efficient integer-arithmetic-only inference* (pp. 2704–2713). Salt Lake City, UT, USA: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2018.00286, https://arxiv.org/abs/1712.05877
19. Nichols, K. R., Moussa, M. A., & Areibi, S. M. (2002). *Feasibility of floating-point arithmetic in FPGA based artificial neural networks* (pp. 8–13). San Diego, California: Proc. CAINE, 15th International Conference on Computer Applications in Industry and Engineering. URL: Scopus.
20. Perić, Z. H., Savic, M. S., Dincic, M. R., Vučić, N. J., Djosic, D., & Milosavljevic, S. (2021). *Floating point and fixed point 32-bits quantizers for quantization of weights of neural networks* (pp. 1–4). Bucharest, Romania: 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE). https://doi.org/10.1109/ATEE52255.2021.9425265
21. He, L., Zheng, S., Chen, W., Ma, Z. M., & Liu, T. Y. (2019). OptQuant: Distributed training of neural networks with optimized quantization mechanisms. *Neurocomputing, 340*, 233–244.
22. Al-Rikabi, H., & Renczes, B. (2022). *Floating-point roundoff error analysis in artificial neural networks* (pp. 79–83). Brescia: 25th IMEKO TC-4 International Symposium on Measurement of Electrical Quantities, IMEKO TC-4 2022 and 23rd International Workshop on ADC and DAC Modelling and Testing, IWADC 2022. https://doi.org/10.21014/tc4-2022.15
23. Alsadi, N., Gadsden, S. A., & Yawney, J. (2023). Intelligent estimation: A review of theory, applications, and recent advances. *Digital Signal Processing, 135*, 103966.
24. Al-Rikabi, H., Al-Ja'afari, M. A., Ali, A. H., & Abdulwahed, S. H. (2020). Generic model implementation of deep neural network activation functions using GWO-optimized SCPWL model on FPGA. *Microprocessors and Microsystems, 77*, 103141.
25. Shanmuganathan, S. (2016). *Artificial Neural Network Modelling: An Introduction* (pp. 1–14). Cham: Springer International Publishing.
26. Russinoff, D. M. (2019). *Formal verification of floating-point hardware design* (1st ed.). Cham: Springer.
27. BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, OIML. Evaluation of measurement data — Guide to the expression of uncertainty in measurement. Joint Committee for Guides in Metrology, JCGM 100:2008.
28. Dieck, R. H. (2007). *Measurement uncertainty: methods and applications* (4th ed.). ISA-The Instrumentation, Systems, and Automation Society.
29. Wetjen, E. *Thingspeak channel 12397; MathWorks weather station, Natick, USA*. https://thingspeak.com/channels/12397. Accessed July 2023.
30. Garza-Ulloa, J. (2022). Applied biomedical engineering using artificial intelligence and cognitive models - chapter 5 - dataset - deep learning models principles applied to biomedical engineering. https://doi.org/10.17632/nc9m5zm8st.1. Accessed July 2023.

Hussein Al-Rikabi received his M.S. degree in electronics engineering from the University of Technology, Baghdad, Iraq, in 2017. Subsequently, in 2021, he embarked on his Ph.D. studies in electrical engineering at the Budapest University of Technology and Economics, Hungary. His current areas of research include quantization error investigation of artificial intelligence models, quantization aware training and training optimization.

Balázs Renczes received his M.S. and Ph.D. degrees in Electrical Engineering from the Budapest University of Technology and Economics, Budapest, Hungary in 2013 and 2017, respectively. Since 2023, he has been an Associate Professor at the Department of Measurement and Information Systems, Budapest University of Technology and Economics. He currently investigates signal processing problems in roughly quantized systems and the optimization of nonlinear system identification techniques.