Beyond Base-2 Logarithmic Number Systems (WiP Paper)

Syed Asad Alam*

Research Fellow

Software Systems Group, Computer Science
School of Computer Science and Statistics, Trinity
College Dublin, The University of Dublin
Dublin 2, Dublin, Ireland
syed.asad.alam@tcd.ie

David Gregg†

Professor

Software Systems Group, Computer Science
School of Computer Science and Statistics, Trinity
College Dublin, The University of Dublin
Dublin 2, Dublin, Ireland
david.gregg@scss.tcd.ie

Abstract

Logarithmic number systems (LNS) reduce hardware complexity for multiplication and division in embedded systems, at the cost of more complicated addition and subtraction. Existing LNS typically use base-2, meaning that representable numbers are some (often fractional) power of two. We argue that other bases should be considered. The base of the LNS determines the distribution of values and may reduce representation errors when converting inputs to LNS in domain-specific embedded hardware accelerators. Further, LNS addition and subtraction are normally implemented with lookup tables whose properties may be a function of the base.

We show that other bases can lower both representation and addition and subtraction error. We consider the case of 8-bit LNS, when converting from 8-bit floating point (FP). We find that base-1.984 significantly reduces the conversion error. Where we can scale values, base-1.851 reduces the error to just 0.011 units of least precision (ULP). A suitable base can also reduce average arithmetic errors. For example, base-1.802 LNS has an average error of 0.242 ULP and 0.212 ULP as compared to 0.243 ULP and 0.226 ULP for addition and subtraction, respectively, for base-2.

CCS Concepts • Computer systems organization → Embedded systems; • Mathematics of computing;

Keywords Logarithmic number system; floating point number system; digital arithmetic; embedded devices.

*S. A. Alam is a member of Lero – the Irish Software Research Center

†D. Gregg is a member of Lero – the Irish Software Research Center

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

LCTES '20, June 16, 2020, London, United Kingdom

© 2020 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-7094-3/20/06.

<https://doi.org/10.1145/3372799.3394368>

ACM Reference Format:

Syed Asad Alam and David Gregg. 2020. Beyond Base-2 Logarithmic Number Systems (WiP Paper). In *Proceedings of the 21st ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '20)*, June 16, 2020, London, United Kingdom. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3372799.3394368>

1 Introduction

Logarithmic number systems (LNS) have been explored as an alternative number system for embedded applications such as signal processing [11], video processing [3] and neural networks [15]. LNS provides a non-uniform distribution of numbers, and operations like multiplication, division, and square root are reduced to addition, subtraction and shift.

In deep neural networks (DNN), 32-bit floating point numbers have been used predominantly for data and parameter representation [12] while for embedded applications like FIR filters, fixed-point data representation has been the common choice [13]. However, 8-bit floating point — also known as mini-float — and 8-bit fixed point number representation have been explored for embedded DNN models like MobileNet and ResNet with negligible accuracy degradation [20, 23]. Shorter word length will lead to reduced memory requirements and memory throughput required thus helping improve the power efficiency for resource-constrained embedded devices [24].

Logarithmic number systems are particularly suitable for low-precision types, such as 8-bit as they require only small lookup tables for addition and subtraction. This aids fast execution of these operations, even on small embedded systems, as lookup tables size grow rapidly with the bit-width.

Existing LNS generally use a base of two. We propose to increase the accuracy of low-precision LNS arithmetic by investigating other bases. Major contributions are:

- Analysis of different bases in the LNS for **representation error** when quantizing **8-bit mini-float** numbers to LNS, both for all **representable numbers** and those between ± 1 . It is shown that the quantization error can be reduced for bases other than base-2

- Similar analysis is carried out for **arithmetic error**, minus the representation error, and it is further shown that base-2 does not provide the optimal solution.

This paper is organized as follows: Section 2 provides a literature survey of related work while in Section 3, a brief introduction is given for LNS representation along with the quantization of a floating point number to an LNS number. Section 4 describes the LNS addition and subtraction and tables required for performing these operations while Section 5 provides results for arithmetic error when computing those tables. Finally, Section 6 concludes the paper while outlining future directions of the contribution.

2 Previous Work

Logarithmic number systems (LNS) have been used as an alternative to more common number representations like fixed and floating point. Kingsbury and Rayner [10] show that they have superior numerical properties like higher dynamic range when compared to fixed-point number representation. They also provide better round-off noise performance as compared to floating point numbers for the same number of bits [4]. Paliouras and Stouratis [16] show that LNS can lead to lower switching activity, while Basetas et al. [5] show that LNS requires a reduced word length when implementing FIR and infinite impulse response (IIR) filters. Alam and Gustafsson design an FIR filter in the log space, showing a significant reduction of approximation error for the same coefficient word length [1]. Furthermore, for 32-bit word length, LNS may be the better technique than corresponding single precision floating point [22].

Low precision DNNs, using various number representations, have been proposed to reduce the memory footprint of its data and parameter storage in order to execute them on small embedded edge devices to truly enable technologies like driverless cars, edge computing and smart wearable devices. A neural network can have as high as 138 M parameters, for e.g., for VGG-16 [18]. It has been shown that half precision, 16-bit floating point number (PF16) has no impact on the inference error rates [7]. The mini-float, 8-bit floating point number representation, has been explored for models like MobileNet and Resnet with negligible accuracy degradation [20] while being shown to outperform 16- and 8-bit fixed point counterparts [6].

Fixed point number systems with primarily 8- and 16-bit quantization, have also been evaluated as an alternative, providing promising results with some degradation in error rates [17]. Extremely low-bit quantization has also proved useful in significantly reducing the network size, enabling them to be run on low-power embedded systems [8].

Research on LNS for neural networks has focused on the non-uniform distribution of weights and activations. Arnold et al. [2] explore various bases using Mitchell's method for LNS addition without tables while Lee and Miyashita [12]

propose an inference engine, named LogNet, with more accurate encoding of the parameters and data. Miyashita et al. have also considered bases of $\sqrt{2}$ and $\sqrt[3]{2}$ for representing data and parameters in a neural network [15]. They find that that the total quantization error on the weights for a base of $\sqrt{2}$ is about $2\times$ smaller than base-2 LNS*.

3 Logarithmic Number System (LNS)

In the logarithmic number system (LNS), a number can be represented using a triplet $\mathcal{X}$ as follows [21]:

$$\mathcal{X} = (z_x, s_x, m_x), \quad (1)$$

where z_x indicates whether $\mathcal{X}$ is zero, $s_x = \text{sign}(X)$ and $m_x = \log_b |X|$. z_x may be represented as an extra bit or with the most negative value of m_x [9].

The base- b of an LNS number influences the representation capabilities and complexity of computations and conversions [19] and m_x is typically represented using fixed-point representation, with i and f integer and fractional bits (mathematically represented as $Q(i, f)$), respectively [1, 22].

3.1 Quantization of 8-bit Precision Floating Point Number Representation (mini-float)

To evaluate the effectiveness of non-base-2 LNS, we analyze the effect of base by calculating the representation error when quantizing a mini-float, with four-bit exponent (bias = 7[§]) and three-bit mantissa, to an LNS distribution.

We quantize by first constructing a vector whose each value is the mean of two adjacent LNS numbers. The vector acts as a bin to find the indices (using Matlab's `discretize` function) that contain each mini-float value which is then used to find the LNS value that best quantizes the corresponding mini-float value. Figure 1 shows an example for base-2 with the arrow showing corresponding pairs of mini-float and LNS numbers to which they are quantized.

This quantization results in an average representation error, shown in Fig. 2(a), where a base of 1.984 provides the least average relative error, as a percentage of unit of least representation, commonly known as ULP, where $\text{ULP} = b^{2^{-f}}$. The average relative error, as a percentage of one ULP, is calculated as:

$$e_u = \frac{e_r}{\text{ULP} - 1} \times 100 \quad (2)$$

where e_u is the average relative error as a percentage of ULP and e_r is the average relative error. The relative error, e_r is:

$$e_r = \frac{\sum |f - \hat{f}|}{N} \quad (3)$$

*LNS with base $\sqrt{2}$ or $\sqrt[3]{2}$ are already represented within base-2 fixed-point LNS. By selecting the position of the fixed point, standard base-2 LNS can represent implicit bases such as $\sqrt{2}$, $\sqrt[3]{2}$, $\sqrt[4]{2}$, 2, 4, 8, etc.

§The exponent bias was set to 7 according to IEEE-754 rules for half (15) and full precision (127) floating point number.

Figure 1. Quantization of 8-bit floating point numbers (mini-float) to 8-bit LNS (without scaling).

Figure 2. Average relative error as a percentage of the *ULP* when the full distribution of mini-float numbers are quantized to different (a) **unscaled** and (b) **scaled** LNS bases.

where f is the 8-bit floating point number, $\hat{f}$ is the quantized floating point number and N is total number of floating point values where the relative error is not equal to zero.

3.1.1 Scaling of LNS Numbers

The LNS numbers to which the mini-float was quantized had a different range than floating point numbers. This will result in a significant error, specially for smaller bases. For example, for base- $\sqrt{2}$, the minimum/maximum values are ± 15.3034 , the number to which the minimum/maximum values of mini-float (± 240) will be quantized to.

In order for the ranges of the mini-float and LNS numbers of all bases to be equal, it is important to scale the LNS numbers before quantization by the ratio of the maximum floating point and LNS number for each base. The representation error profile with scaling is shown in Fig. 2(b). The minimum error was given when base = 1.851.

In a number of applications, the range of numbers is concentrated in a very small range, for e.g., it is between ± 1 for an FIR filter. Even for a neural network, the weights are typically concentrated within the same range [15]. For this purpose, the mini-float numbers within this range were quantized to LNS (without scaling[‡]). However, since now the range is small, for LNS only 3 integer bits were enough with 3 fractional bits and base-1.741 gave the minimum relative error, as a percentage of *ULP*.

[‡]Scaling is not needed as both the mini-float and LNS numbers have the same maximum and minimum value of +1 and -1, respectively

4 LNS Addition and Subtraction

Addition and subtraction of two numbers in the LNS domain are given by [9]:

$$m_{\text{add/sub}} = \max(m_x, m_y) + \Phi(x) \quad (4)$$

where

$$\Phi(x) = \begin{cases} \Phi^+(x) = \log_b |1 + 2^{-x}| & S_\Phi = 0 \\ \Phi^-(x) = \log_b |1 - 2^{-x}| & S_\Phi = 1 \end{cases} \quad (5)$$

where $S_\Phi = s_x \oplus s_y \oplus op$ (op is 0 for addition and 1 for subtraction) and $x = -|m_x - m_y|$ (where $x \leq 0$). The functions $\Phi^+(x)$ and $\Phi^-(x)$ are non-linear.

It is the computation of the function $\Phi(x)$ that is non-trivial and has the potential to undo any advantages gained by using LNS number representation. It requires either look-up tables, iterative method, CORDIC method, Taylor series expansion or function approximation [9, 14]. Among the techniques mentioned, the look-up table based approach is suitable for word lengths up to 20-bits [22], which has a time and space complexity of N and 2^N , respectively.

The two functions in (5) were realized as a single table, for different bases, where values in the table were quantized to the given word length and compared against non-quantized table values to calculate the arithmetic error as a percentage of *ULP*. The bases ranged from $\sqrt{2}$ and 2 because any base beyond this range finds an alias within this range. For e.g., base- $\sqrt{2}$ provides the same quantization and range as base-2 (and, as an extension, base-4 as well), but with one fractional bit less for the same total word length. Or, for the same number of fractional bits and total word length, base- $\sqrt{2}$ provides higher precision for a smaller range.

5 Experimental Evaluation

The maximum error for rounding to nearest is expected to be 0.5ULP and the average around 0.25ULP, i.e., an error of half a bit and quarter of a bit, respectively. The error generated when quantizing the table values for $\Phi(x)$, when represented as a percentage of one ULP is shown in Fig. 3. This shows that the difference in arithmetic errors for various bases do not differ significantly and for the choice of the base, thus, one will have to strongly consider the representation error, as given in Section 3.1.

For nearly all bases, the average relative error is around a quarter of a bit. The bases which give the maximum and minimum arithmetic error for the two tables along with different interesting bases together with their representation error, are shown in Table 1. Also included is base-2 and all lowest values highlighted in bold for convenience.

These tables show that for the two functions of $\phi^+(x)$ and $\phi^-(x)$, no one base gives the global minimum. However, the representation error, shown in Figs. 2(a) and 2(b) earlier, needs to be taken into account to determine the optimal base. Although base-1.851 gives an arithmetic error that's higher by

Figure 3. Average relative error as a percentage of one ULP for (a) $\phi^+(x)$ and (b) $\phi^-(x)$.

Table 1. Percentage average relative error for various bases.

Base	Perc. avg. rel. error (%)		Repr. error (Avg.) [★]		Repr. error (Avg.) [‡]
	$\phi^+(x)$	$\phi^-(x)$	$S = 1^*$	$S = n^\dagger$	
2.0	24.296	22.615	34.816	22.113	34.816
1.741	24.713	24.301	83.619	2.771	14.885
1.802	24.186	21.205	63.693	1.564	23.495
1.851	24.466	23.725	44.502	1.105	20.118
1.853	24.018	23.574	43.508	1.314	21.742
1.918	25.517	24.489	32.064	3.824	24.328
1.976	24.468	27.654	24.942	9.486	23.047
1.984	24.504	26.635	23.865	11.062	24.787

[★] For full range of 8-bit floating point number

[‡] Range limited to ± 1

^{*} Scaling factor, $S = 1$

[†] Scaling factor, $S = n \implies S = \frac{\max(fp8)}{\max(int8)}$

2% and 12% than the best cases for $\phi^+(x)$ and $\phi^-(x)$, respectively, it is lower by 29% and 15% for the average representation error than the bases which provide the best cases for $\phi^-(x)$ & $\phi^+(x)$, respectively, which is significant.

6 Conclusion & Future Work

Representing data using a shorter word length is important for running applications on embedded systems and the aim of this work was to analyze different bases other than base-2 systematically for 8-bit numbers. Bases were identified for which the distribution of numbers provided the best quantization of an 8-bit floating point number, with and without scaling, and then addition and subtraction tables of LNS were realized in these different bases.

We show that base-2 does not necessarily give the least average error. Apart from errors in the LNS addition and subtraction tables, one must consider the representation error when representing floating point numbers in the LNS

domain. It was shown that base-1.851 gives the minimum relative representation error when 8-bit floating point numbers were quantized to LNS numbers. It was also shown that the arithmetic errors in the tables for different bases were around a quarter of a ULP with base-1.851 providing the best compromise over both types of error. However, the selection of base with minimum arithmetic and representation error is dependent on the numbers of bits being used to represent the two number systems.

This work lays down an important mathematical foundation towards realizing different systems where the logarithmic number system (LNS) has been reported to provide improvements, like signal processing and neural networks. LNS simplifies the computationally intensive arithmetic calculations into simple ones and combining it with an optimal base for a shorter word length will allow complex systems to be implemented in low-powered and resource-constrained embedded devices. Furthermore, reduced memory size is important to offset the complexity of logarithmic addition and subtraction and sustain the advantages provided by LNS. Additionally, one can further work on identifying the base to be chosen for different ranges of numbers in a real system. Finally, it is also important to study how accumulation of arithmetic errors across multiple computation affect the overall error, thus favoring one base over another.

Acknowledgments

This material is based upon work supported, in part, by Science Foundation Ireland under Grant No. 13/RC/2094 and, in part, by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement and Grant No. 754489. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Science Foundation Ireland and European Union's Horizon 2020 programme.

References

- [1] Syed Asad Alam and Oscar Gustafsson. 2014. Design of Finite Word Length Linear-Phase FIR Filters in the Logarithmic Number System Domain. *VLSI Des.* 2014 (2014), 14.
- [2] M. Arnold, E. Chester, J. Cowles, and C. Johnson. 2019. Optimizing Mitchell's Method for Approximate Logarithmic Addition via Base Selection with Application to Back-Propagation. In *Proc. IEEE Nordic Circuits Syst. Conf. (NorCAS): NORCHIP and Int. Symp. of Syst.-on-Chip (SoC)*. 1–6.
- [3] M. G. Arnold. 2002. Reduced power consumption for MPEG decoding with LNS. In *Proc. IEEE Int. Applicat.-Specific Syst. Arch. Processors Conf.* 65–67.
- [4] M. G. Arnold, T. A. Bailey, J. R. Cowles, and M. D. Winkel. 1992. Applying features of the IEEE 754 to sign/logarithm arithmetic. *IEEE Trans. Comput.* 41, 8 (Aug. 1992), 1040–1050.
- [5] C. Basetas, I. Kouretas, and V. Paliouras. 2007. Low-Power Digital Filtering Based on the Logarithmic Number System. In *Proc. Int. Workshop Power Timing Modeling Optimization Simulation*. Vol. 4644. 546–555.

- [6] E. Chung and J. Fowers et al. 2018. Serving DNNs in Real Time at Datacenter Scale with Project Brainwave. *IEEE Micro* 38, 2 (March 2018), 8–20.
- [7] M. Courbariaux, Y. Bengio, and J. P. David. 2014. Training deep neural networks with low precision multiplications. [arXiv:1412.7024](https://arxiv.org/abs/1412.7024)
- [8] Itay Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. 2017. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *ACM J. Mach. Learn. Res.* 18, 1 (Jan. 2017), 6869–6898.
- [9] K. Johansson, O. Gustafsson, and L. Wanhammar. 2008. Implementation of elementary functions for logarithmic number systems. *IET Comput. Digit. Tech.* 2, 4 (June 2008), 295–304.
- [10] N. G. Kingsbury and P. J. W. Rayner. 1971. Digital Filtering using Logarithmic Arithmetic. *Electron. Lett.* 7, 2 (Jan. 1971), 56–58.
- [11] I. Kouretas, Ch. Basetas, and V. Paliouras. 2013. Low-Power Logarithmic Number System Addition/Subtraction and Their Impact on Digital Filters. *IEEE Trans. Comput.* 62, 11 (2013), 2196–2209.
- [12] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong. 2017. LogNet: Energy-efficient neural networks using logarithmic computation. In *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.* 5900–5904.
- [13] Y. C. Lim. 1990. Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude. *IEEE Trans. Circuits Syst.* 37, 12 (1990), 1480–1486.
- [14] A.M. Mansour, A.M. El-Sawy, M.S. Aziz, and A. T. Sayed. 2015. A new hardware implementation of base 2 logarithm for FPGA. *Int. J. Signal Proc. Syst.* 3, 2 (2015), 171–181.
- [15] Daisuke Miyashita, Edward H. Lee, and Boris Murmann. [n. d.]. Convolutional neural networks using logarithmic data representation. [arXiv:cs.NE/1603.01025v2](https://arxiv.org/abs/cs.NE/1603.01025v2)
- [16] V. Paliouras and T. Stouraitis. 2000. Logarithmic number system for low-power arithmetic. In *Proc. Int. Workshop Power Timing Modeling Optimization Simulation*, Vol. 1918. 285–294.
- [17] M. Shah, S. Arunachalam, S. Wang, D. Blaauw, D. Sylvester, H. S. Kim, J. s. Seo, and C. Chakrabarti. 2018. A fixed-point neural network architecture for speech applications on resource constrained hardware. *J. Signal Process. Syst.* 90, 5 (May 2018), 727–741.
- [18] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. [arXiv:1409.1556](https://arxiv.org/abs/1409.1556)
- [19] T. Stouraitis and V. Paliouras. 2002. Considering the alternatives in low-power design. *IEEE Circuits Syst. Mag.* 17, 4 (Aug. 2002), 22–29.
- [20] X. Sun, J. Choi, C. Y. Chen, N. Wang, S. Venkataramani, V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan. 2019. Hybrid 8-bit Floating Point (HFP8) Training and Inference for Deep Neural Networks. In *Proc. Advances in Neural Information Processing Syst.* 4901–4910.
- [21] E.E. Swartzlander Jr. and A.G. Alexopoulos. 1975. The Sign/Logarithm Number System. *IEEE Trans. Comput.* C-24, 12 (Dec. 1975), 1238–1242.
- [22] J. N. Coleman et al. 2008. The European Logarithmic Microprocessor. *IEEE Trans. Comput.* 57, 4 (April 2008), 532–546.
- [23] V. Vanhoucke, A. Senior, and M. Z. Mao. 2011. Improving the speed of neural networks on CPUs. In *Proc. Deep Learning and Unsupervised Feature Learning Workshop, NIPS*.
- [24] W. Wolf and M. Kandemir. 2003. Memory system optimization of embedded software. *Proc. IEEE* 91, 1 (Jan. 2003), 165–182.