IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 7, 2021, accepted May 23, 2021, date of publication June 4, 2021, date of current version June 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3086669

Analysis of Posit and Bfloat Arithmetic of Real Numbers for Machine Learning

ALEKSANDR YU. ROMANOV[1], (Member, IEEE), ALEXANDER L. STEMPKOVSKY[2], (Member, IEEE), ILIA V. LARIUSHKIN[3], GEORGY E. NOVOSELOV[4], ROMAN A. SOLOVYEV[2], VLADIMIR A. STARYKH[1], (Member, IEEE), IRINA I. ROMANOVA[1], (Member, IEEE), DMITRY V. TELPUKHOV[2], AND ILYA A. MKRTCHAN[2], (Member, IEEE)

[1]National Research University Higher School of Economics (HSE University), 101000 Moscow, Russia
[2]Institute for Design Problems in Microelectronics of Russian Academy of Sciences, 124681 Moscow, Russia
[3]Advanced System Technologies, 124681 Moscow, Russia
[4]NTProgress, 115280 Moscow, Russia

Corresponding author: Aleksandr Yu. Romanov (a.romanov@hse.ru)

The work of Aleksandr Yu. Romanov was supported by the Basic Research Program of the National Research University Higher School of Economics. The work of Alexander L. Stempkovsky, Roman A. Solovyev, Dmitry V. Telpukhov, and Ilya A. Mkrtchan was supported in part by the Institute for Design Problems in Microelectronics of Russian Academy of Sciences, under Project FFNZ-2021-0001.

ABSTRACT Modern computational tasks are often required to not only guarantee predefined accuracy, but get the result fast. Optimizing calculations using floating point numbers, as opposed to integers, is a non-trivial task. For this reason, there is a need to explore new ways to improve such operations. This paper presents analysis and comparison of various floating point formats – float, posit and bfloat. One of the promising areas in which the problem of using such values can be considered to be the most acute is neural networks. That is why we pay special attention to algorithms of linear algebra and artificial intelligence to assess efficiency of new data types in this area. The research results showed that software implementations of posit16 and posit32 have high accuracy, but they are not particularly fast; on the other hand, bfloat16 is not much different from float32 in accuracy, but significantly surpasses it in performance for large amounts of data and complex machine learning algorithms. Thus, posit16 can be used in systems with less stringent performance requirements, as well as in conditions of limited computer memory; and also in cases when bfloat16 cannot provide required accuracy. As for bfloat16, it can speed up systems based on the IEEE 754 standard, but it cannot solve all the problems of conventional floating point arithmetic. Thus, although posits and bfloats are not a full fledged replacement for float, they provide (under certain conditions) advantages that can be useful for implementation of machine learning algorithms.

INDEX TERMS Machine learning, floating point, posit, IEEE 754, benchmark.

I. INTRODUCTION

With the growth of computer systems computing power, machine learning advanced significantly over the past decades, enabling developers from various subject areas to create stable and high-performance systems based on artificial intelligence technologies. Despite the fact that computers have become powerful enough to train large neural networks for solving real practical problems, the industry as a whole shows significant decrease in growth rate of processor performance [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

Existing floating point calculation, which is the base of computational tools used in almost all areas of computer science, can be considered as one of the causes of the potential crisis. The IEEE 754 standard, formally approved in 1985 and updated 12 years ago, has been widely used as the main floating point format for over 30 years. Despite a number of shortcomings of this format, among which we can accentuate large memory requirements and substantial limitations in the accuracy of calculations, until recently engineers have not been able to find a worthy replacement for it. In 2017, John Gustafson introduced posit format, the third variant of unum (''universal numbers'') data type [2]. Combining different approaches to floating point computation has resulted in

a unique structure that addresses most of the fundamental problems of the IEEE 754 standard. According to Gustafson, posits provide better computational performance and accuracy, take less memory, avoid problems with Not-a-Number type and rounding, and overflow errors [3]. Posit's breakthrough computing power is based on the term ''quire''. This data type allows developers to perform reproducible linear algebra based on posits that cannot be achieved using standard floats. The quire mechanism eliminates rounding errors and number overflow resulting from dot products and sums, which is especially important for machine learning problems [4]. Posits attracted attention of a wide range of developers, which led to a lot of new research. Although it is assumed that the proposed data type could replace floats, the level of understanding of all the advantages and disadvantages of using posits in machine learning systems is still insufficient.

II. POSIT

Posit data type resulted from the development of unum format, ''universal numbers'', is presented in [5]. The main features of this data type implementation are described in [3], which first of all, compares computational capabilities of posits and floats in the framework of ordinary operations with one argument (such as reciprocals, powers of 2, square roots, logarithms and exponents) and two arguments (addition, subtraction, multiplication). For all the simple examples, posits demonstrated their superiority in computational accuracy, dynamic range, and closure properties of number sets. In this work, it was also suggested that 8-bit posits can be faster in training neural networks than half-precision (16-bit) floats. Researchers suppose that quarter-precision posits can be 2–4 times faster than half-precision floats, occupying 2 times less memory and increasing total performance of network training process. This assumption was demonstrated for sigmoid function, but further studies of this issue were not presented. Moreover, based on the work of Gustafson, it can be concluded that posits are highly resistant to errors and undefined values, which can also be an advantage for artificial intelligence systems.

The format structure is shown in Fig. 1. According to the standard, 16-bit posits have 1 exponent bit; and regime and mantissa bits are allocated during calculations [2]. 32-bit and 64-bit posits have 2 and 3 exponent bits, respectively. Optionally, the number of exponent bits can be increased.

The IEEE 754 float format, unlike posit, contains only exponent and mantissa bits (let alone the sign). Their number is always fixed, which makes it easier to implement and analyze this type [6]. The standard float32 consists of 8 exponent bits and 23 mantissa bits (Fig. 2). The extended version of float64 consists of 11 exponent bits and 52 mantissa bits.

In the years since posit presentation, developers have managed to offer a significant number of format implementations. Thus, in work [7], representatives of Lombiq Technologies described their software and hardware implementations of Type I unums and posits. Alternatively, researchers from

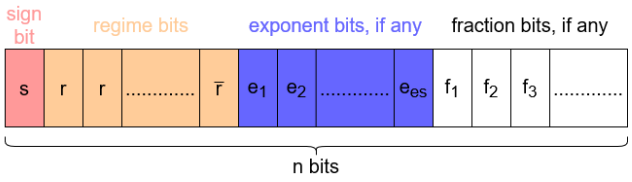


FIGURE 1. Generic posit format [2].

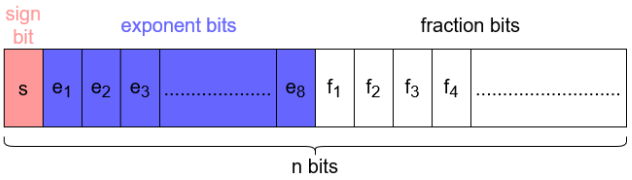


FIGURE 2. Float32 format [2].

Tokyo Institute of Technology completely concentrated on the hardware implementation of the format [8]. The efficiency of multiplying posit numbers was also demonstrated in the work by engineers from Complutense University of Madrid [9]. One more study was presented by developers from Harvard University, who used this data type for recurrent neural network for speech recognition [10].

Based on the publications above, we can state that the posit format is actively investigated in various recent works on finding efficient solutions in such problems as matrix calculations, as well as training and implementation of neural networks. At this, many authors have shown that 8-bit posits cannot fully satisfy requirements of existing high-load networks, so it is 16-bit representation of the new data type that should be considered as a replacement for float32.

III. BFLOAT

While some researchers concentrate on comparing posits with numbers in the IEEE 754 standard, others focus on finding new and more efficient floating point solutions.

The main competitor of float (along with posit types) is bfloat or Brain Floating Point [11], originally proposed as a special solution for artificial intelligence systems, which is the basis of Google TPU (Tensor Processing Unit) cloud computing [12]. In addition to hardware optimization for a certain class of tasks, using bfloats helps to move to half-precision numbers (in case of basic 16-bit implementation). This solution was applied in paper [13]. Unlike 16-bit numbers of the IEEE 754 standard, which can represent values in the range from $\sim 5,96 \times 10^{-8}$ to 65504, bfloat16 covers the interval between $\sim 10^{-38}$ and $\sim 3 \times 10^{38}$, while classical floats can only achieve that when using 32 bits. Bfloat16 structure uses 8 exponent bits and 7 fraction bits, which leads to better compactness, but, at the same time, to decrease in the accuracy of stored numbers (Fig. 3). Although TPU is the target platform for bfloat, simple idea of the format allows it to be used not only in tensor systems, but also for other hardware platforms.

Currently, there are many possibilities for emulating bfloat behavior, since its structure is simple to understand and implement. For example, Intel and Facebook engineers used

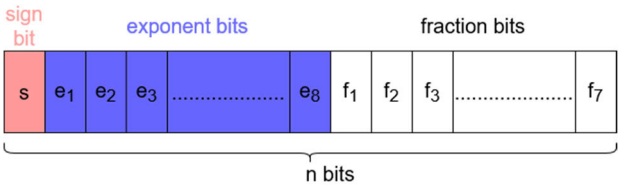


FIGURE 3. Bfloat16 format [2].

a similar approach to float and bfloat conversion, analyzing performance capabilities of low-precision bfloat in deep learning networks [14]. For such systems as IntelCaffe, Caffe2, Neon, and Tensorflow frameworks were developed to work with bfloat16. To emulate bfloat behavior, their developers set to zero the lower 16 bits of float32 numbers and rounded off incoming operands. The experimental results, characterized by test accuracy and losses, show that bfloat16 covers a wide range of tensors in completely different networks and compares favorably to float32 in information processing accuracy, halving the memory space. Nevertheless, this data type does not provide significant improvements in accuracy when rounding data for recognition, which is emphasized by posit format developer [3].

Matrix calculations based on floating point numbers and bfloats have been investigated by Intel engineers using GEMM (General matrix multiply) algorithms as the basis for exploring prospects of low precision computations in comparison with standard data types using more than 16 bits [15]. The authors claim that systolic arrays based on bfloat16 and float16 provide many times faster computations than standard 32-bit floats.

However, bfloat is still not the best solution for the entire spectrum of machine learning problems. Many researchers agree that 16-bit posits and bfloats are optimal for working with neural networks, since they strike a balance between feasible volumes of processed data and memory requirements. However, as for performance of these formats, the question is still open.

IV. TEST SUITE DEVELOPMENT

To compare the data types, we had to generate a set of tests, which were then included into the test system. In addition, the rules and principles for test execution and analysis of results were specified. Among many different variants of tests and approaches to computation performance estimation, the most suitable algorithms and methods were identified, which made it possible to evaluate performance for the data types under consideration.

In [16] a non-standard mathematical apparatus, including various recurrent formulas, polynomials, and infinite series for evaluating accuracy of calculations and stability of data types is presented. Some of these tests were implemented in this study.

To analyze the use of data types in the general case, it is not specific neural networks that deserve attention, but the algorithms necessary for their implementation, such as the BLAS (Basic Linear Algebra Subprograms) specification [17]. Also, one of the most common tools, used in the studies of other authors [14], [15], is the GEMM algorithm [18]. This algorithm is an extended version of the standard matrix multiplication.

Also, the algorithms of more limited, but voluminous library LAPACK (Linear Algebra PACKage) [19] were implemented in the test base. Its implementation for C++ (LAPACK++ [21]) was chosen among all similar libraries specializing in linear algebra (such as Intel MKL [20], AMD ACML and Sun Performance Library) because it has an open source code and is not bound to a specific operating system or processor architecture.

One of the largest and most important test packages for analyzing floating point numbers is the SPEC suite, which provides a wide set of benchmarks for processors and cloud computing [22]. Stable versions of SPEC 2006 and 2017 include tests for integers and floating point numbers (SPECint and SPECfp, respectively). For this study, examples from SPECfp set were used. It should be noted that in SPEC CPU 2017, tests based on physical problems are mainly offered for real numbers, therefore, we used SoPlex [23], the open source library for solving linear programming problems from SPECfp CPU 2006.

Among the tools presented, a set of test operations, most widely used in implementations of neural networks, was identified; on their basis, further comparison of data types was carried out. The set of test operations includes matrix and scalar addition and multiplication, recursive function for determinant calculation, QR- and LU- matrix decomposition, Gauss-Lagrange algorithm, calculation of Frobenius and L1 norms, Moore-Penrose pseudo-inversion, matrix normalization and transposition, and two sigmoid calculator functions.

For software implementation of testing infrastructure [24], the C++ language was chosen. All studied data types were defined, and the set of tests were split into two classes: for validation (necessary to check correctness of plug-in formats; allows detailed comparison of the results obtained in calculations using built-in 32-bit float and external data types) and performance evaluation (run times of the program are evaluated using Google Benchmark library). For each iteration, dataset was created based on random generation. The testing system itself [24] emulates operation of the data types (i.e., maintains data at software rather than hardware level). Therefore, such implementations will naturally operate much slower than built-in data types; however, they can be compared with each other, since they are implemented in consistent way. Hence, we can claim that their performance ratio will remain the same for hardware implementation. In addition, the test suites are designed to cover the most commonly used machine learning operations and algorithms. This allows us to draw conclusions about the efficiency of using the data types under consideration for different variants of neural networks, since they are mainly based on combinations of the same-type operations and algorithms.

V. EVALUATION

On the basis of the developed testing system, tests were run for the studied data types, namely, posit16, bfloat16, float32, and also posit32. Results were additionally compared with tests for C++ built-in float (orange 'float' on figures). Built-in float usually much faster than other methods because it is not emulated, but computed directly. Therefore, difference in execution time between external and embedded programming language formats was demonstrated. From Fig. 4, it follows that posit32 is the fastest to cope with scalar addition. Float32 and bfloat16 (external to the testing program) show significant computation delays. But as expected, all methods are much slower comparing to C++ built-in float type.

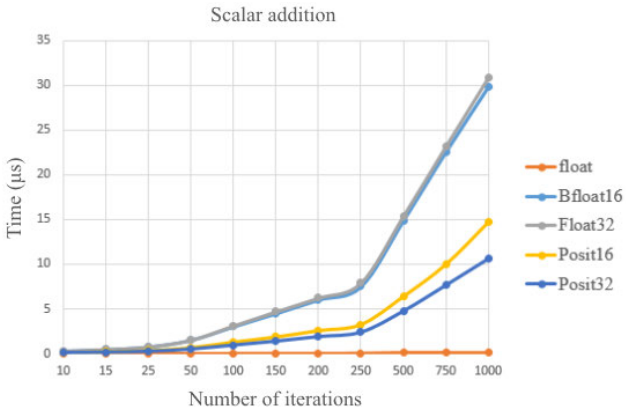


FIGURE 4. Scalar addition test.

In case of scalar multiplication, the results are different. Because of the peculiarities of their implementation, posits produce additional delays during complicated calculations, and thus, lag behind both bfloat16 and float32, which is clearly seen in Fig. 5.

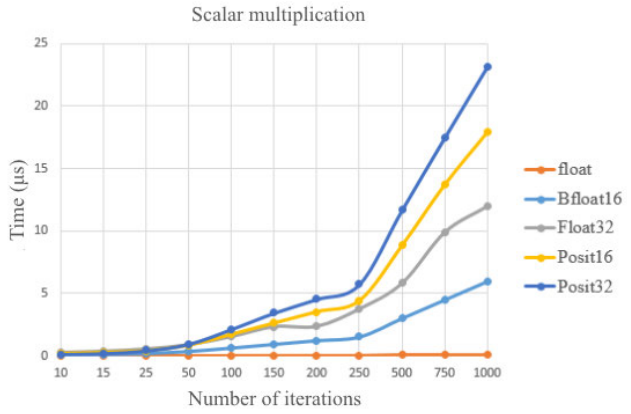


FIGURE 5. Scalar multiplication test.

When the amount of processed data grows significantly, all the formats demonstrate approximately the same results for addition of square matrices (Fig. 6). It is noteworthy that running time increases faster for posit16 than for other studied formats, including posit32.

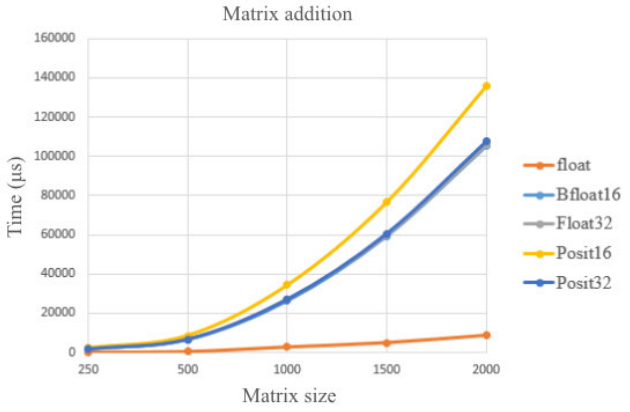


FIGURE 6. Matrix addition test.

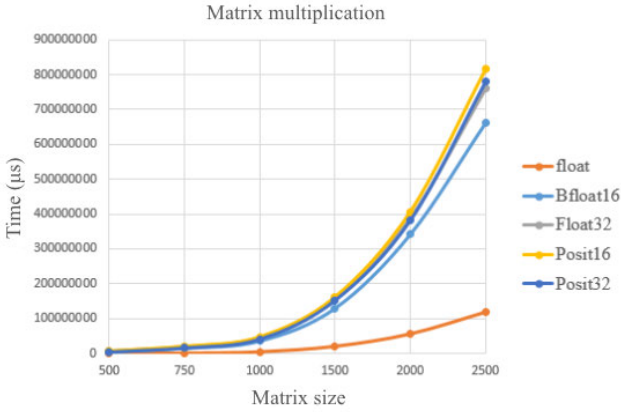


FIGURE 7. Matrix multiplication test.

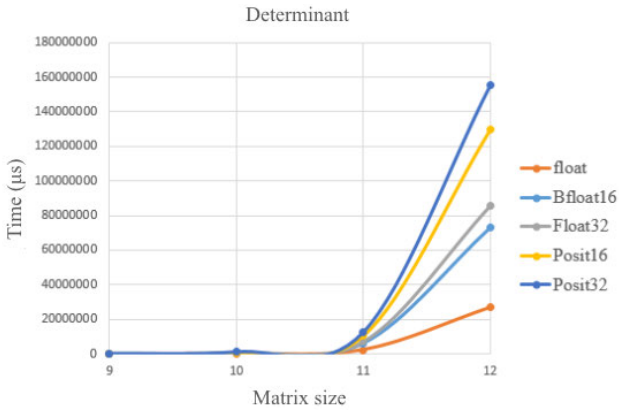


FIGURE 8. Matrix determinant test.

Also, efficiency of bfloat16 when performing multiplication is shown for matrix multiplication example (Fig. 7).

Although posits manage to outperform float32, the alternative version of the IEEE 754 standard is much faster in complex matrix calculations. This is also demonstrated by the results of such highly loaded tests as determinant calculation and QR matrix decomposition (Fig. 8, 9).

One more test is calculating approximate value of pi; according to the test results, float32 is best suited, but immediately after it comes bfloat16, with significant gap from posit

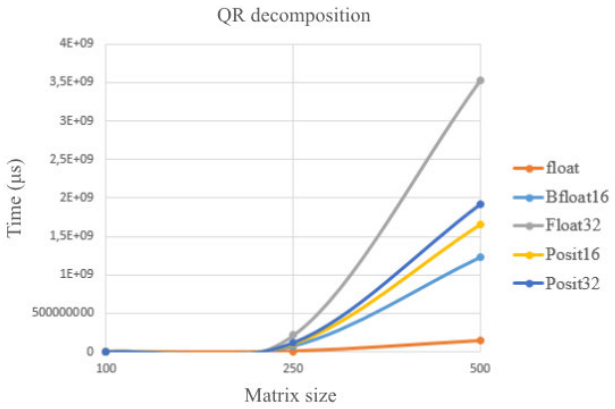


FIGURE 9. QR matrix decomposition test.

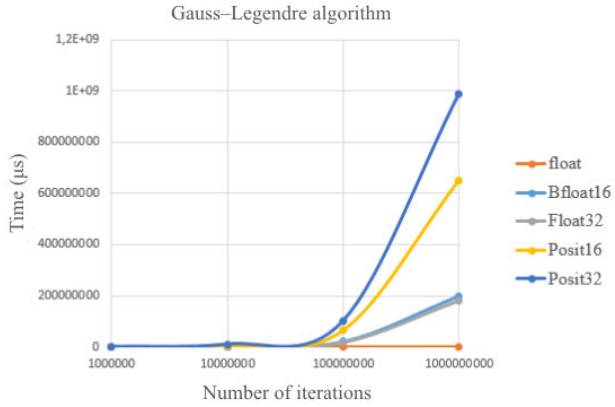


FIGURE 10. Gauss-Lagrange algorithm test.

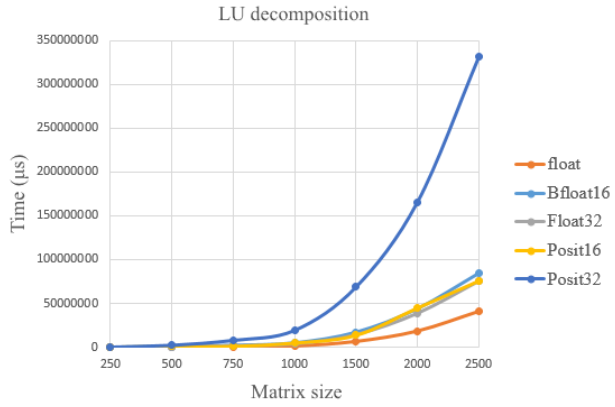


FIGURE 11. LU matrix decomposition test.

(Fig. 10). Despite the fact that built-in float almost instantly solves this problem for any number of iterations, execution time of the test constantly grows when the other formats are used.

The result obtained for LU matrix decomposition test is highlighted. Fig. 11 shows that posit16 is much faster than posit32, not being inferior to IEEE 754 standard formats.

Since LU decomposition function has no points of premature exit from the program, and all formats worked with the same random values, correctness of the test is beyond doubt. This result is explained by specifics of the format, which also manifested itself for float32 and bfloat16 in QR decomposition test (Fig. 9).

VI. CONCLUSION

As part of this work, custom software package [24] was developed to emulate behavior of posit, float and bfloat formats. Additionally, test infrastructure was created for evaluating performance of real numbers storage formats based on conventional algorithms, used in machine learning systems. Using this package, we studied performance of posits and bfloats as compared with floating-point numbers in the IEEE 754 standard.

The software implementations of posit16 and posit32 partly have fallen short of expectations; they are not very fast in complex algorithms which use large amount of data. The main disadvantage of posits with regard to speed is dynamic calculation of the structure of numbers. But it allows for greater accuracy at less memory usage and wide dynamic range. Posit format can be recommended for neural networks and other systems, which are less demanding in terms of performance but require higher accuracy and operate in conditions of limited memory.

The bfloat16 format, which practically does not differ from float32, significantly surpasses it in speed for large amounts of data and complex machine learning algorithms. However, bfloat16, based on the IEEE 754 standard, has a number of drawbacks, including reduced calculation accuracy and vast deal of undefined states.

Indeed, this format can speed up systems based on the IEEE 754 standard and currently looks like the best choice for neural net applications. However, bfloat16 cannot solve all the problems of conventional floating point arithmetic; so, we need to continue research for better formats.

REFERENCES

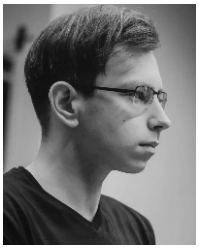
[1] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. San Mateo, CA, USA: Morgan Kaufmann, 2012, p. 1357.
[2] J. L. Gustafson, *Posit Arithmetic*. 2017, p. 137.
[3] J. L. Gustafson and I. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomput. Frontiers Innov.*, vol. 4, no. 2, pp. 71–86, 2017, doi: 10.14529/jsfi170206.
[4] U. W. Kulisch and W. L. Miranker, "The arithmetic of the digital computer: A new approach," *SIAM Rev.*, vol. 28, no. 1, pp. 1–40, Mar. 1986, doi: 10.1137/1028001.
[5] J. L. Gustafson, *The End of Error: Unum Computing*. Boca Raton, FL, USA: CRC Press, 2017, p. 416.
[6] *IEEE Standard for Floating-Point Arithmetic*, Standard IEEE Standard 754-2008, doi: 10.1109/IEEESTD.2008.4610935.
[7] Z. Lehóczky, A. Retzler, R. Tóth, Á. Szabó, B. Farkas, and K. Somogyi, "High-level.NET software implementations of unum type I and posit with simultaneous FPGA implementation using hastlayer," in *Proc. Conf. Next Gener. Arithmetic*, Mar. 2018, pp. 1–7, doi: 10.1145/3190339.3190343.
[8] A. Podobas and S. Matsuoka, "Hardware implementation of POSITs and their application in FPGAs," in *Proc. IPDPSW*, May 2018, pp. 138–145, doi: 10.1109/IPDPSW.2018.00029.
[9] R. M. Montero, A. A. D. Barrio, and G. Botella, "Template-based posit multiplication for training and inferring in neural networks," Jul. 2019, *arXiv:1907.04091*. [Online]. Available: http://arxiv.org/abs/1907.04091
[10] Z. Wan, E. Mibuari, E.-Y. Yang, and T. Tambe, "Study of posit numeric in speech recognition neural inference," Harvard Univ., Cambridge, MA, USA, Tech. Rep. CS247r, 2018.

[11] *Using Bfloat16 With TensorFlow Models*. Accessed: Apr. 05, 2021. [Online]. Available: https://cloud.google.com/tpu/docs/bfloat16
[12] *Google Supercharges Machine Learning Tasks With TPU Custom Chip*. Accessed: Apr. 05, 2021. [Online]. Available: https://cloud.google.com/blog/products/gcp/google-supercharges-machine-learning-tasks-with-custom-chip
[13] J. Dean, "The deep learning revolution and its implications for computer architecture and chip design," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 8–14, doi: 10.1109/ISSCC19947.2020.9063049.
[14] D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen, J. Yang, J. Park, A. Heinecke, E. Georganas, S. Srinivasan, A. Kundu, M. Smelyanskiy, B. Kaul, and P. Dubey, "A study of BFLOAT16 for deep learning training," 2019, *arXiv:1905.12322*. [Online]. Available: http://arxiv.org/abs/1905.12322
[15] G. Henry, P. T. P. Tang, and A. Heinecke, "Leveraging the bfloat16 artificial intelligence datatype for higher-precision computations," in *Proc. IEEE 26th Symp. Comput. Arithmetic (ARITH)*, Jun. 2019, pp. 69–76, doi: 10.1109/ARITH.2019.00019.
[16] E. Morancho, "Unum: Adaptive floating-point arithmetic," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2016, pp. 651–656, doi: 10.1109/DSD.2016.39.
[17] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, "Basic linear algebra subprograms for fortran usage," *ACM Trans. Math. Softw.*, vol. 5, no. 3, pp. 308–323, Sep. 1979, doi: 10.1145/355841.355847.
[18] B. Kågström, P. Ling, and C. van Loan, "GEMM-based level 3 BLAS: high-performance model implementations and performance evaluation benchmark," *ACM Trans. Math. Softw.*, vol. 24, no. 3, pp. 268–302, Sep. 1998, doi: 10.1145/292395.292412.
[19] J. Demmel, "LAPACK: A portable linear algebra library for high-performance computers," *Concurrency, Pract. Exper.*, vol. 3, no. 6, pp. 655–666, Dec. 1991, doi: 10.1002/cpe.4330030610.
[20] E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu, and Ya, *Wang, High-Performance Computing on the Intel Xeon Phi: How to Fully Exploit MIC Architectures*. Cham, Switzerland: Springer, 2014, p. 338, doi: 10.1007/978-3-319-06486-4.
[21] J. J. Dongarra, R. Pozo, and D. W. Walker, "LAPACK++: A design overview of object-oriented extension for high performance linear algebra," in *Proc. ACM/IEEE Conf. Supercomputing*, Nov. 1993, pp. 162–171, doi: 10.1145/169627.169680.
[22] J. L. Henning, "SPEC CPU2000: Measuring CPU performance in the new millennium," *Computer*, vol. 33, no. 7, pp. 28–35, Jul. 2000, doi: 10.1109/2.869367.
[23] A. L. Pérez, "Linear programming with SoPlex, SoPlex complexity and SCIP complexity," *Universidad de Valladolid*, p. 72, 2016.
[24] *Testing System for Floating Point Numbers*. Accessed: Apr. 05, 2021. [Online]. Available: https://github.com/RomeoMe5/posit-bfloat-testsystem

ALEKSANDR YU. ROMANOV (Member, IEEE) received the B.S. and M.S. degrees in computer systems and networks from National Technical University Kharkov Polytechnic Institute (NTU KhPI), Kharkov, Ukraine, in 2007 and 2009, respectively; and the Ph.D. degree in technical sciences from the Federal State-Funded Institution of Science Institute for Design Problems in Microelectronics of Russian Academy of Sciences (IPPM RAS), Zelenograd, Russia, in 2015. He is currently an Associate Professor with National Research University Higher School of Economics (HSE University), Moscow; the HSE Tikhonov Moscow Institute of Electronics and Mathematics. He is also a University Teacher of the following subjects: SoC Development, System Design of Electronic Devices. He is the author of the book *Digital Design: Practical Course*, and more than 100 scientific articles. His research interests include SoC, NoC, embedded systems, FPGA, and neural networks. He is an IEEE Industrial Electronics Society Member. He has received numerous awards as the supervisor of students' teams taking part in international competitions in programming, electronics and robotics, as well as the HSE University Best Lecturer Award, in 2018, 2019, and 2021.

ALEXANDER L. STEMPKOVSKY (Member, IEEE) graduated from the National Research University of Electronic Technology (MIET). He defended his Ph.D. thesis in 1981, and then his doctoral thesis in 1991. Since 1992, he has been working as the Director of the Institute for Design Problems in Microelectronics of Russian Academy of Sciences (IPPM RAS). Since 2018, he has been the Scientific Director of the Institute. He is the author of over 200 research articles. His research interests are related to the problems of integrated circuit design methodology, automation of design procedures, and hardware implementation of digital signal processing devices. He has been a member of IEEE Communications Society for over ten years.

ILIA V. LARIUSHKIN received the B.S. degree in computer systems and networks from National Research University Higher School of Economics (HSE University), Moscow, Russia, in 2020. He is currently a Software Developer Specialist with Advanced System Technologies (AST). His research interests include the study, performance, and robotic systems modeling. He was a Regional Finalist of the Innovate FPGA (Intel) Design Contest 2019. He was also a Finalist in Student Research Paper Competition 2020 in field technical science held by HSE University.

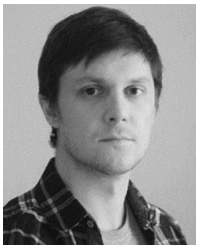


GEORGY E. NOVOSELOV received the B.S. degree in computer systems and networks from National Research University Higher School of Economics (HSE University), Moscow, Russia, in 2020. He is currently a Software Developer with NTProgress. His research interests include the study, artificial intelligence, and robotic systems modeling. He was a Regional Finalist of the Innovate FPGA (Intel) Design Contest 2019 and a Finalist of the Student Research Paper Competition 2020 held by HSE University in the field of technical science.

ROMAN A. SOLOVYEV received the Specialist degree in software for computer technology and automated systems from National Research University of Electronic Technology (MIET), Moscow, Russia, in 2004, and the Ph.D. degree in technical sciences from the Federal State-Funded Institution of Science Institute for Design Problems in Microelectronics of Russian Academy of Sciences (IPPM RAS), Zelenograd, Russia, in 2007 (stage 1) and 2018 (stage 2). He is currently a Professor with MIET, and a University Teacher of the following subject: methods for processing and analyzing big data. He is also working as a leading Researcher with IPPM RAS. He is the author of over 80 research articles.

VLADIMIR A. STARYKH (Member, IEEE) graduated from the Moscow Institute of Electronic Engineering (MIEE). He received the Ph.D. degree in technical sciences with MIEE, in 1978. He is currently the Head of the School of Computer Engineering, National Research University Higher School of Economics (HSE University), Moscow. He is also a professor and a university teacher of the scientific seminar for masters of the educational program ''Computer systems and Networks''. He is the author of four books and more than 120 scientific articles. His research interests include intelligent systems, embedded systems, systems design automation, neural networks, and distributed information resource processing technologies. He is an IEEE Industrial Electronics Society Member. He is the winner of the Russian Government Award in Education, the Honorary Employee of Higher Education of Russian Federation, and the Honorary Employee of Science and Technology of Russian Federation.

IRINA I. ROMANOVA (Member, IEEE) received the B.S. and Speciliast degrees in biomedical and chemical devices and systems from National Technical University of Ukraine ''Igor Sikorsky Kyiv Polytechnic Institute'' (NTUU ''KPI''), Kyiv, Ukraine, in 2007 and 2009, respectively, and the M.S. degree in applied mathematics and computer science from National Research University Higher School of Economics (HSE University), Moscow, in 2018. She is currently a Senior Lecturer with HSE University (Department of HSE Tikhonov Moscow Institute of Electronics and Mathematics), and a University Teacher of the following subjects: Informatics, Computer Workshop ''Programming Tools''. She is the author of more than 25 scientific articles. Her research interests include mathematics, embedded systems, neural networks, and engineering education.



DMITRY V. TELPUKHOV received the degree in informatics and computer engineering from the National Research University of Electronic Technology (MIET). Since 2008, he has been working as an Engineer–Researcher with the Institute for Design Problems in Microelectronics of Russian Academy of Sciences (IPPM RAS), and later as a Junior Researcher and a Researcher. Since 2015, he has been the Head of the Department with IPPM RAS. His research interests include the problems of integrated circuit design methodology, automation of design procedures, and hardware implementation of digital signal processing devices and neurochips. He defended his Ph.D. thesis, in 2012, and then his doctoral thesis, in 2018.

ILYA A. MKRTCHAN (Member, IEEE) received the B.S. degree in electronics and nanoelectronics from National Research University Moscow Institute of Electronic Technology (MIET), Moscow, Russia, in 2020, where he is currently pursuing the degree in informatics and computer engineering. Since 2019, he has been working as an Intern Researcher with the Federal State-Funded Institution of Science Institute for Design Problems in Microelectronics of Russian Academy of Sciences (IPPM RAS), where he is currently as an Engineer Researcher. He is the coauthor of five articles. His research interests include modular arithmetic and hardware implementations of neural networks.