A Novel Posit-based Fast Approximation of ELU Activation Function for Deep Neural Networks

Marco Cococcioni, Federico Rossi, Emanuele Ruffaldi, Sergio Saponara

Introduction
▶ Cost efficiency of Deep Neural Networks (DNNs) is critical
▶ Industry and academia push towards reduction of arithmetic complexity [1, 2]
• Posit number system [3] is a new promising compressed floating point format
▶ Typical bottlenecks in DNNs are: a) massive use of small-filter convolutions and matrix-vector multiplications b) computation of activation function over big amount of data
▶ a) can be addressed with vector or graphics processing units.
▶ b) involves non-linear operators and knowledge of underlying data distribution

Posit number system
▶ The posit format is represented by a 2’s complement integer and is configurable in the total number of bits and exponent bits
• posit(nbits, esbits)
▶ Maximum of 4 fields
• sign (1-bit)
• regime (run-length encoded value)
• exponent (variable length with esbits maximum)
• mantissa (variable length)
▶ Given a posit represented by the integer \( l \), the correspondent real value is:
\[
\text{sign}(l) \times \text{useed}^k \cdot 2^{e \cdot (1 + f)}
\]
• Where useed = 2\(^{esbits}\), \( k \) is the value extracted from the regime and \( f \) is the value of the mantissa

cppPosit Library
▶ Developed in Pisa by MMI spa and University of Pisa
▶ Modern C++ with templatization and traits for posit configuration
▶ Operations are classified into four different levels: L1 to L4
• L1 operations are the most efficient ones, involving only manipulation of the representing integer
▶ Posit emulation is supported by different backends (e.g. float backend, ALU backend or fixed backend)

Extended Linear Unit (ELU) function
▶ S-shaped functions like hyperbolic tangent or sigmoid suffer from vanishing gradients
• ELU-like functions solve this problem
\[
\text{ELU}(x) = e^x - 1, \text{ if } x \leq 0, \ x \text{ otherwise} \quad (1)
\]
• ELU function can be expressed as a function of the sigmoid one:
\[
2 \cdot \left[ \frac{1}{2} \cdot \text{Sigmoid}(-x) - 1 \right]
\]

Fast approximation: fastELU
▶ If we substitute the Sigmoid function in the previous equation with its posit approximated version we obtain a L1 version of the ELU function

Benchmark and experimental environment
▶ The benchmark used for experimental analysis is an image classification task on the GTRSB (German Traffic Road Sign Benchmark) dataset.
▶ The LeNet-5 deep neural network model has been used during the experimental phase.
▶ Benchmarks are executed on a 7-th generation Intel i7-7560U processor, running Ubuntu Linux 18.04, equipped with GCC 8.3.

Discussion
▶ As reported therein, Float32 accuracy is easily matched by Posits with 16 down to 10 bits, and, in particular, for GTRSB similar performance are obtained even with a Posit8,0. According to these results the adoption of Posit and ELU can lead to nearly the same processing accuracy of Float32 but with a remarkable reduction, up to a factor of 4, of the data storage.

Results

| Activation   | FastELU (this paper) | ELU | ReLU |
|--------------|----------------------|-----|------|
| %            | ms                   | %   | ms   | %   |
| SoftFloat32  | -                    | -   | 94.2 | 15.86 | 92.7 | 8.2 |
| Pos16,0      | 94.0                 | 5.8 | 94.2 | 6.37 | 92.7 | 5.0 |
| Pos14,0      | 94.0                 | 4.6 | 94.2 | 5.21 | 92.7 | 4.3 |
| Pos12,0      | 94.0                 | 4.6 | 94.2 | 5.08 | 92.7 | 4.3 |
| Pos10,0      | 94.0                 | 4.6 | 94.2 | 5.03 | 92.7 | 4.3 |
| Pos8,0       | 92.0                 | 4.6 | 91.8 | 5.0  | 86.8 | 4.0 |

Table: Benchmark results on the GTRSB dataset.

Conclusions
▶ In this work we have introduced a fast way to approximate the well-known ELU activation function in DNNs, when using the novel Posit format for representing the reals, instead of classic IEEE-754 Floats.

Acknowledgements
▶ Work funded by the H2020 European Processor Initiative project

Bibliography
[1] U. Köster et al. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. In Advances in Neural Inf. Processing Systems, pages 1742–1752, 2017.
[2] A. Malossi et al. The transprecision computing paradigm: Concept, design, and applications. In 2018 Design, Automation Test in Europe Conference Exhibition (DATE), pages 1105–1110, 2018. doi: 10.23919/DATE.2018.8342176.
[3] John L Gustafson and Isaac T Yonemoto. Beating floating point at its own game: Posit arithmetic. Supercomputing Frontiers and Innovations, 4(2):71–86, 2017.