Sound Efficient Quantization of Neural Networks

Context Neural networks are being used more and more in areas where safety really matters, such as self-driving cars, medical devices, and robotics. To ensure these systems behave correctly, researchers have developed techniques for formally verifying neural networks — but usually only for "ideal" versions that work with real numbers.

In real-world applications, however, networks must be implemented with finite-precision numbers (using floating-point or fixed-point arithmetic), which can introduce roundoff errors. If we do not account for these errors, a system that seems safe during verification could behave incorrectly when finally deployed in a resource-constrained embedded platform.

To make neural networks safe and efficient in practice, thus, we need to carefully choose the precision used in the implementation: the precision must be small enough to save memory and speed up computations, but it must also be high enough to keep roundoff errors within safe limits. Finding such a tradeoff manually is very difficult and tedious for anything beyond tiny networks.

State-of-the-art A recent technique [1] addressed this problem for regression models, which are often used as controllers in safety-critical systems. The tool Aster formulates the quantization task as an integer linear programming (ILP) problem and solves it to find the optimal precision assignment for all variables. It then generates an efficient C++ implementation of the trained regression model that guarantees a safe error bound and can be directly synthesized onto FPGAs for deployment.

However, Aster has some important limitations: 1) It only supports very simple activation functions (ReLU and linear); 2) Aster only works for basic feedforward regression networks — not classifiers or CNNs. 3) The underlying analysis tends to be very conservative, assigning higher precisions than actually necessary, which can waste resources.

Goal of the Thesis The goal of this thesis is to improve and extend the existing approach, making it more practical, scalable, and applicable to a wider range of real-world neural networks:

1. We aim to support a broader set of activation functions. By exploring linear approximation techniques [2], the tool can be extended to handle commonly used nonlinear activations, such as sigmoid, tanh, and others, going beyond its current limitation to ReLU and linear activations. This will enable it to work with a wider variety of network architectures.

2. We plan to expand the tool's applicability to different types of models and architectures. In particular, we aim to make the tool support not only regression tasks but also classification models and convolutional neural networks (CNNs), which are widely used in practical machine learning applications.

3. We explore ways to reduce unnecessary over-approximations in the static analysis without sacrificing scalability. Thus, the tool will be able to generate code that uses fewer bits and fewer hardware resources while still guaranteeing error bounds.

Through this project, you will gain hands-on experience with formal methods, program optimization, and static roundoff error analysis. The existing tool, Aster, is written in Scala, a language similar to Java, and you will be extending and building on top of this tool.

There are no strict prerequisites for the project. However, prior experience with program analysis, verification, automated reasoning, or familiarity with Scala or Java would be helpful.

References

[1] Debasmita Lohar, Clothilde Jeangoudoux, Anastasis Volkova, and Eva Darulova. Sound Mixed Fixed-Point Quantization of Neural Networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 2023.

[2] Zhiwu Xu, Yazheng Liu, Shengchao Qin, and Zhong Ming. Output Range Analysis for Feed-Forward Deep Neural Networks via Linear Programming. *IEEE Transactions on Reliability*, 2022.

Contact: Debasmita Lohar (Office: 50.34, R202)
Email: debasmita.lohar@kit.edu