Automatic Hardware-aware Design and Optimization of Deep Learning Models

Deep Learning represents one of the major technological breakthroughs of recent years. Typically, the training and inference phases of Deep Learning models, known as Deep Neural Networks (DNNs), happen on powerful cloud hardware. Nonetheless, evidence exists about the potential advantages of directly deploying DNNs on resource-constrained edge devices. These advantages range in several directions, including reduced latency, reduced energy consumption, improved privacy, and enhanced reliability.

Nonetheless, deploying DNNs on edge devices represents a non-negligible challenge due to resource constraints regarding memory, operating frequencies, and supported instructions.

This thesis will present several optimization techniques that can consider the target hardware's characteristics, such as the memory footprint or the latency, to enable DNN inference at the edge. Moreover, all the novel techniques presented in this manuscript are framed as gradient-based optimization problems aimed at minimizing a combination of task-specific loss and hardware-aware cost metrics. The optimization knobs are the standard and the architectural parameters of the DNN which are learned through gradient descent. In this way, the DNN can be trained and optimized in one shot.

The first optimization technique presented will be the structured pruning approach known as PIT. This algorithm allows exploring the key hyperparameters of Temporal Convolutional Networks (TCNs), a state-of-the-art DNN typically used to process time-series data such as audio and bio-signals.

Then, we will show how to jointly explore structured pruning with mixed-precision search in Convolutional Neural Networks (CNNs). With this algorithm, it is possible to completely remove or quantize each weight channel to different precisions in each CNN layer.

Third, we will propose a novel optimization problem formulation that introduces multiple hardware-related constraints. This formulation allows us to discover DNNs in the design space that perform well on the task at hand while simultaneously satisfying multiple constraints imposed by the target hardware platform.

The last optimization tool discussed will be ODiMO, a mapping tool capable of splitting the execution of DNNs with fine-grain over multiple compute units available in System on Chips (SoCs) that accelerate different DNN layer alternatives or with operands with incompatible precision.

Finally, the manuscript will conclude with three real-world applications where DNNs are optimized using some of the techniques discussed above and deployed on edge devices. These applications include PPG-based heart rate estimation on wearables, visual-pose estimation on nano-drones, and people counting on low-resolution infrared arrays.