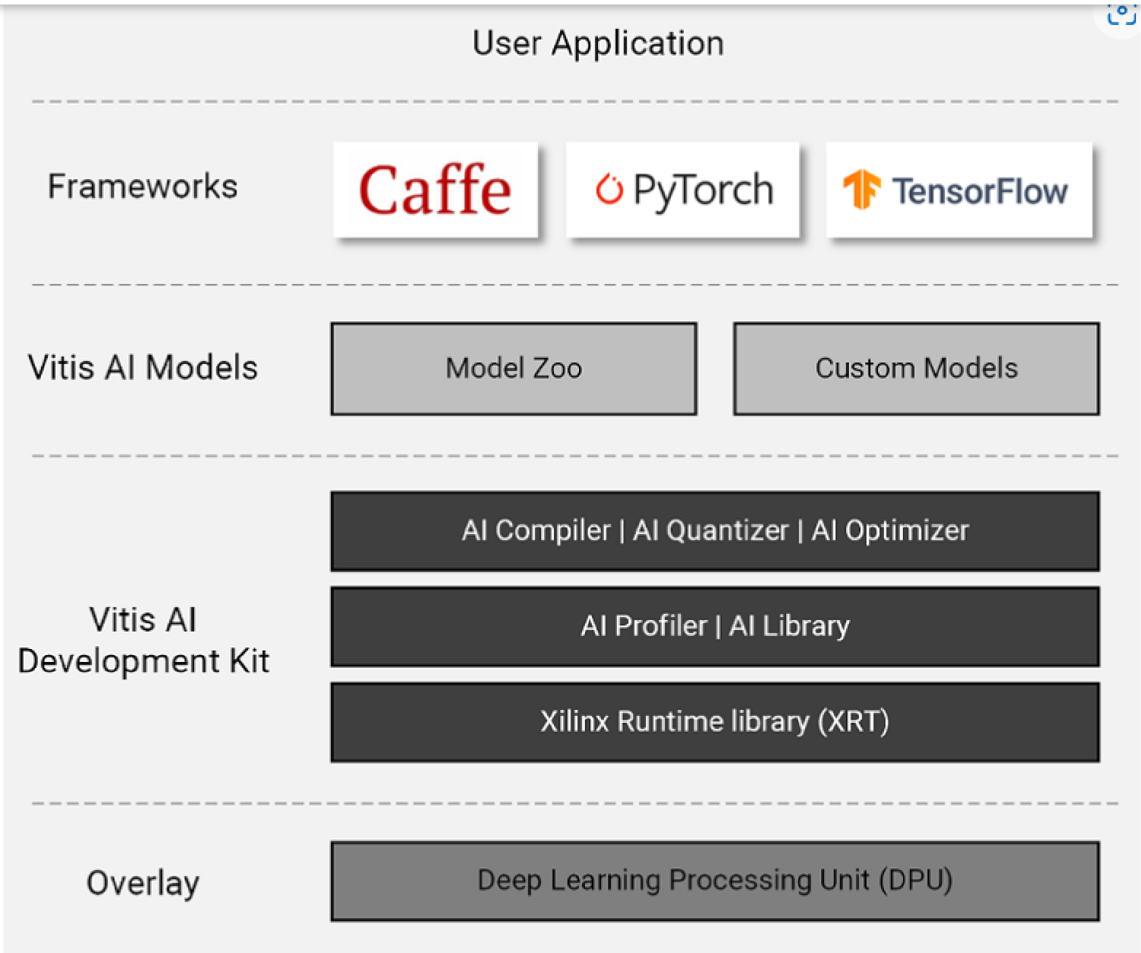
Vitis AI and Xilinx DPU

Vitis AI tool is a development environment created to access the full potential of IA acceleration on FPGA or acceleration platforms. Vitis AI contains optimized IP cores and Deep Learning Processing Units (DPU), specialized tools and libraries, models and design examples.

Vitis AI Tools description


User Application

Frameworks: Caffe | PyTorch | TensorFlow

Vitis AI Models: Model Zoo | Custom Models

Vitis AI Development Kit: AI Compiler | AI Quantizer | AI Optimizer; AI Profiler | AI Library; Xilinx Runtime library (XRT)

Overlay: Deep Learning Processing Unit (DPU)

Figure 1 : Vitis AI stack, extracted from Vitis AI user Guide on xilinx documentation portal

Above we can see the Vitis AI stack, we'll describe each layers:

Frameworks: It's possible to import into Vitis AI any machine learning model from a framework like *Caffe, Pytorch* or *TensorFlow*. Once imported the project can be evaluated and optimized.

Vitis AI models:There are many models of optimized deep learning which can be found in *Custom models* or *Vitis AI Model Zoo*. Using these models allows to speed up the deployment and to perform acceleration of AI on Xilinx platforms.

Vitis AI development Kit: In this kit there are

- *Vitis AI Optimizer* is a compression technology where the model complexity can be reduced between 5 and 50 times, and with a minimal accuracy degradation by running (setting parameters to 0).

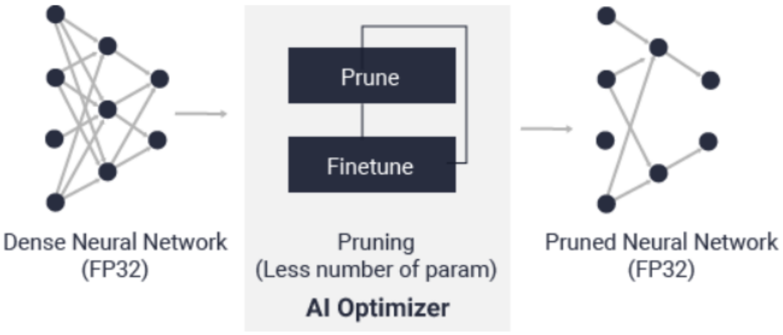


Figure2: AI Optimiser compression, extracted from Xillinx Vitis AI Overview

- *Vitis AI Quantizer* is another compression technology, the computing complexity is reduced by converting 32-bit floating point to fixed point like int8. then the model would need less memory bandwidth, faster speed and higher power efficiency.

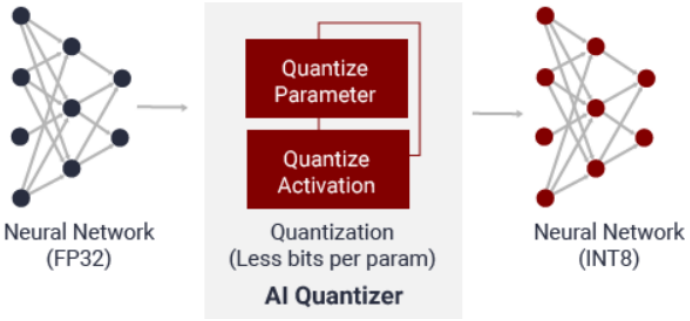


Figure3: AI Quantizer compression, extracted from Xilinx Vitis AI Overview

- *Vitis AI Compiler* is used to map the AI model to an instruction set with optimizations like layer fusion, reuses on chip memory or instruction scheduling.

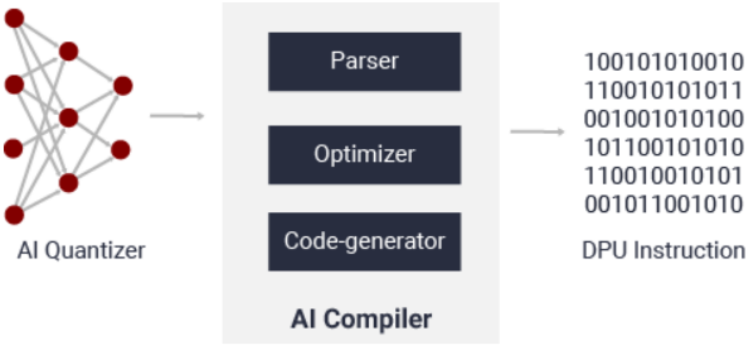


Figure4: AI Compiler, extracted from Xilinx Vitis AI Overview

- *Vitis AI profiler* is used to analyse the neural network and so to trace function calls or run time or to collect pieces of information such as CPU utilisation , hardware ressources or memory utilization.

- *Vitis AI Library* simplifies the use of neural networks providing easy to use and unified interface.

- *Xilinx Runtime library (XRT)* is implemented as a combination of user-space and kernel driver components. and provides a software interface to Xilinx programmable logic devices. XRT has to be install to use acceleration implementation flow in Vitis, therefore in our case we don't need to install it as long as we target Arm based embedded platforms (Vitis compiler for hardware generation and the XRT from the sysroot for the software compilation)

Deep Learning processing Unit (DPU)

The DPU is a software core IP that is optimized for deep neural networks. The DPU is a group of pre-implemented parameterizable IP cores on the hardware. The design is done in order to relieve, lighten deep learning algorithms, mostly during the learning process.

Compared with classic programmable logic using such DPU allows most of the computing tasks to be improved in terms of performances. The DPU configurations can be changed and specialized instructions are available on Vitis to facilitate an efficient implementation. To illustrate how DPU can be used, let's see a configuration proposed by Xilinx to a multiply accumulation operation, doing that we'll use the work *Convolutional neural network implementation using Vitis AI* by Akihiko Ushiroyama, Minoru Watanabe, Nobuya Watanabe, and Akira Nagoya 2 Graduate School of Natural Science and Technology Okayama University.
The multiply accumulate operation method is for 2 int8 multiplication executed concurrently, the example is to calculate the result of (a+b) x c, here the steps to calculate in parallel a x c and b x c:

- variables a and b are packed in 27 bits port through the pre adder
- a is shifted to 18 bis left and joined to b
- c is fit to correspond to the DPU multiply port size
- the a and b joined is multiplied to the c
- the post adder is use to additionate, accumulate the product

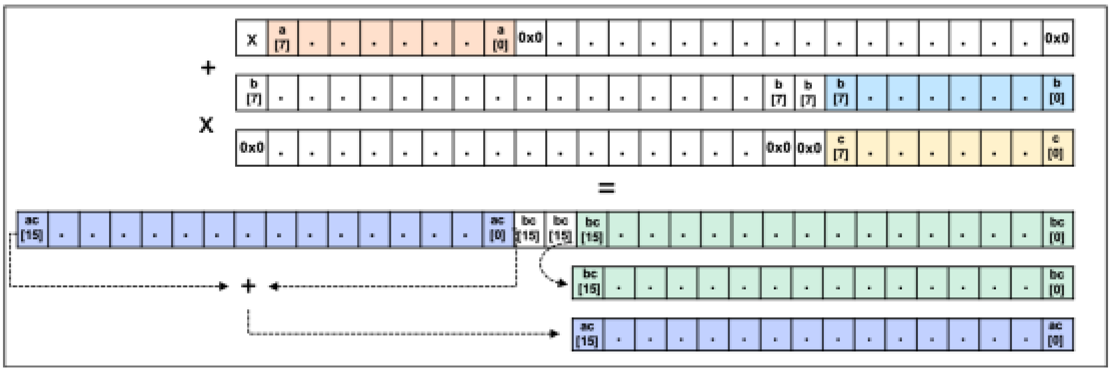


figure5: method for concurrent execution cumulative multiplication, extracted from *Convolutional neural network implementation using Vitis AI*

Each DPU architecture has a degree of parallelism, and so different performances in terms of resources, power consumption or timing performances.