Development of Edge AI Device Platform for Application Developer

Sungjae Yoon, Jeon Seung Hyup
Regional ICT Convergence Research Lab
Electronics and Telecommunications Research Institute (ETRI)
Seongnam, Korea
sj.yoon@etri.re.kr

***Abstract*—** Using neural processing unit (NPU), artificial intelligence (AI) technology can be applied to various products. There are many types of NPU and AI model. Application developers select the appropriate AI model, optimization method, and NPU for the product. we propose an edge AI device platform to support application developer. The proposed platform provides them with a variety of development environments. They can develop applications on the target HW platform.

Keywords—edge AI device, deep learning, NPU, FPGA

I. INTRODUCTION

As GPU performance and data volume increased, artificial intelligence (AI) technology developed rapidly. Early AI technology focused on increasing accuracy. In particular, ResNet won the ILSVRC'15 with 96.4% accuracy [1]. (For your information, human accuracy is known to be 95%.) ResNet consists of 152 layers, requiring high-performance GPUs for real-time operations. Therefore, high-accuracy models were difficult to apply to products used in daily life.

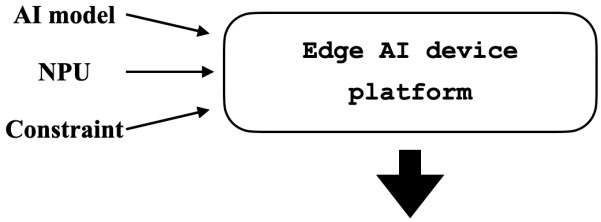
AI technology is increasingly being applied to everyday life. In order to use AI in real world, two ways of research are needed. First, the computational cost and parameter size of the AI model should be reduced. For example, there are optimization methods such as quantization, pruning, etc. When applying optimization methods, it is important to minimize the reduction in accuracy of AI models. Second, specialized HW is needed to process the AI model. Samsung, Google, and Apple are developing HW accelerators that are customized for their products. Recently, as the research described above has been actively conducted, AI technology is being applied in daily life.

A neural processing unit (NPU) is specially designed to accelerate AI operations. AI model consists of various types (convolution, LSTM, dense, etc.) of layers. Each layer looks complicated, but in reality, it is a structure in which simple operations are repeated. It is not suitable for processing by the CPU due to its large amount of computation. In order to solve this problem, NPUs such as TPU and VTA are in the spotlight [2-3]. The architecture of NPU is simple. but the performance and cost vary depending on the number of ALUs, the size of the memory, etc. Therefore, NPU developers have to select NPU architectures that meet the constraints required by the product.

Many companies are developing their own NPUs to apply AI to products. Each NPU has a different compiler and library. Therefore, engineers should setup development environments as many as the number of NPUs. To overcome this problem, AI compilers such as TVM, Glow, etc. being developed [4-5]. When the AI compiler is successfully developed, various NPUs can be used in a single development environment.

Application developers should choose the appropriate AI model, optimization method, and NPU for their product. Each product has the required inference time and cost. In order to consider all of these constraints, experienced engineers with background knowledge in various fields are needed. However, it is practically difficult for all companies to hire such engineers.

In this paper, we propose a platform for application developers to select suitable AI models and NPUs. The platform sends the bitstream and libraries to the FPGA board. This feature can be utilized to develop applications on actual FPGA boards. Developers can focus on developing application without considering AI models, NPUs, etc. Currently, it supports three NPUs. In the future, we plan to provide developers with an environment where they can use more NPUs.



Num.	AI model	NPU	Infer. time(ms)	Opt. Method
1	ResNet	Ⓐ	34.47	Pruing(25%)
2	VGG	Ⓐ	12.94	Qaunt.(8bits)
…	…	…	…	…
N	ResNet	Ⓓ	57.23	Qaunt.(16bits)

Fig. 1. Inputs and outputs of the edge AI device platform

II. EDGE AI DEVICE PLATFORM

A. Overall Architecture

Fig. 2 shows the architecture of the edge AI device platform proposed in this paper.

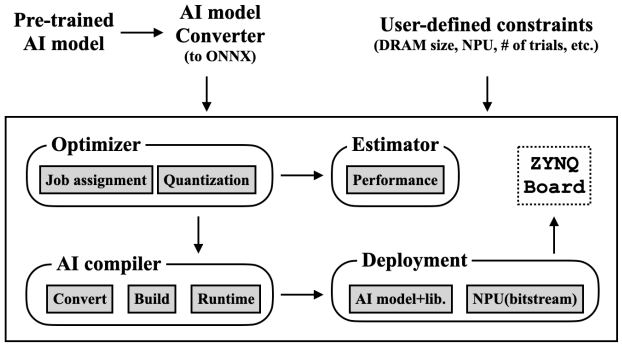


Fig. 2. The diagram of edge AI device platform

The input data for this platform is as follows. *1)Pre-trained AI model*. When training AI models, various AI frameworks such as TensorFlow, Keras, and Pytorch are used. Supporting all AI frameworks takes a lot of time and cost to develop a platform. To address these challenges, the proposed platform converts the AI model into ONNX format. *2)User-defined constraints*. In order to apply AI technology to products, various constraints must be satisfied. Developers can select DRAM size, NPU, number of trials, etc. to meet constraints.

B. Optimizer

The proposed platform supports job assignment and quantization as optimization method.

The job assignment is a process of determining where to process each layer, CPU or NPU. This process consists of three steps as follows. *1)Assigning each layer to NPU or CPU.* The NPU cannot process all operations that make up the AI model. Therefore, operations that cannot be processed by the NPU should be set to be processed by the CPU. *2)Splitting each layer to NPU's buffer size.* If the NPU's internal buffer is large enough, each layer of the AI model may be sequentially processed. However, in reality, the NPU's internal buffer is not large enough. Therefore, each layer should be splitted and processed according to the NPU's buffer size. *3)Selecting layers that are more efficient to handle by the CPU.* It is more efficient to process specific layers in the CPU than NPU. Usually, it is recommended to assign a layer with a large memory usage and a small computation to the CPU.

Quantization is to change the type of data used in AI processing. In general, AI frameworks use 32-bit floating-point data in the training process. It has the disadvantage of increasing the size of the AI model and computational cost. To solve this problem, the data type is converted to 16-bit or 8-bit integers. When the number of bits used to express data is decreased, the accuracy of the AI model may be decreased. Therefore, quantization method that satisfy constraints such as accuracy, memory usage, etc. should be applied. The proposed method provides an environment in which various quantization methods can be applied to test.

C. AI compiler

AI compilers generate what is needed to process AI models in NPUs. In order to process the AI model in NPU, the architecture and parameters of the AI model and a library to control NPU are required. The AI compiler consists of three stages. *1)convert:* it is the process of converting an ONNX model into a model that can be easily interpreted by a compiler. *2)build*: It extracts the architecture and parameters of the AI model and generates a library. *3)runtime*: It provides an API that allows users to easily develop applications. This three-step process allows the selected AI model to be processed by the NPU.

D. Deployer

The Deployer provides an environment in which the application developer can test in the selected environment. It deploys the results of AI compiler and NPU to be used on the target HW platform (e.g. Xilinx ZYNQ). NPU refers to a bitstream synthesized according to the target HW platform. Once a development environment is setup in the target HW platform, application developers can remotely access the target HW platform to develop and test their applications.

E. Estimator

Testing all environments on NPU is inefficient. The proposed platform uses an estimator to predict the approximate performance. The estimator utilizes clock frequency, number of operations, and number of DRAM accesses to estimate performance. The clock frequency uses values allocated to CPU, AXI bus, NPU, etc. AI models are analyzed to calculate the number of operations and memory usage by layer. When a new NPU is added, the estimator is tuned by comparing the inference time between the NPU and the estimator. After tuning is completed, the developer can perform various tests without a target HW platform.

F. Experimental Results

Open source TVM and VTA were used to test the functionality of the proposed platform. TVM is an AI compiler, and VTA is an HW accelerator specialized in general matrix-matrix multiplication (GEMM). The target HW platform used ZCU106. An experimental environment was established by sending libraries and a VTA bitstream to the ZCU106 board.

TABLE I. EXPERIMENTAL ENVIRONMENT

	HW	Clock Freq.	Opt. Method
CPU	ARM Cortex-A53	666 MHz	32bit FP
NPU	VTA	100 MHZ	8bit INT

ResNet18 was processed to CPU and NPU respectively to test the proposed platform functionality. The inference time

used the average value of the time when 1000 images were inferred. The experiment result shows that it took 2378.60ms for ARM CPU and 371.05ms for VTA.

III. Conclusion

In this paper, we propose an edge AI device platform for application developers. The proposed platform targets application developers who do not have expertise in AI and NPU. Using this platform, they can select AI model, optimization methods, and NPU suitable for their products. In order to upgrade the platform, we plan to develop it in two ways.

1) Unified AI compiler: The proposed platform uses an AI compiler specialized for each NPU individually. We plan to develop a unified AI compiler that can easily interface with various NPUs.

2) Improving the performance of estimator: The proposed platform uses simple parameters(clock frequency, operation counts, etc.) to estimate performance. We plan to develop a module that integrally predicts the performance of CPU and NPU including other parameter information. We plan to develop an estimator that integrally estimate the performance of CPU and NPU, including other parameters.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00454, Technology development of smart edge device SW development platform)

References

[1] Kaiming H, Xiangyu Z, Shaoqing R, Jian S, "Deep residual learning for image recognition," CVPR, 2016.

[2] Jouppi, N. P. et al. "In-datacenter performance analysis of a tensor processing unit," In Proc. International Symposium on Computer Architecture(ISCA) 1–12 (IEEE, 2017).

[3] T. Moreau, T. Chen, L. Vega, J. Roesch, E. Yan, L. Zheng, J. Fromm, Z. Jiang, L. Ceze, C. Guestrin, and A. Krishnamurthy, "A hardware-software blueprint for flexible deep learning specialization," IEEE Micro, vol. 39, no. 5, pp. 8–16, 2019.

[4] T. Chen et al., "TVM: An automated end-to-end optimizing compiler for deep learning," in Proc. 13th USENIX Symp. Oper. Syst. Design Implement. (OSDI), Carlsbad, CA, USA, 2018, pp. 578–594.

[5] Nadav Rotem, Jordan Fix, Saleem Abdulrasool, Garret Catron, Summer Deng, Roman Dzhabarov, Nick Gibson, James Hegeman, Meghan Lele, Roman Levenstein, et al., "Glow: Graph lowering compiler techniques for neural networks". arXiv:1805.00907 (2018).