AIRV: Enabling Deep Learning Inference on RISC-V

Yangyang Kong

State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
kongyangyang@iie.ac.cn

Abstract. Recently the emerging RISC-V instruction set architecture (ISA) has been widely adopted by both academia and industry. Meanwhile, various artificial intelligence (AI) applications have been extensively deployed in cloud, edge, mobile and IoT devices due to latest breakthroughs in deep learning algorithms and techniques. Therefore, there is an increasing need for enabling deep learning inference on RISC-V. However, at present mainstream machine learning frameworks have not been ported to RISC-V, which poses challenges to deep learning application developers. In this paper, we explore approaches to enabling deep learning inference on RISC-V. Experimental results show that in our work, there is a great gap between the performance of deep learning inference on RISC-V and that on x86; thus compared with direct compilation on RISC-V, cross-compilation on x86 is a better option to significantly improve development efficiency.

Keywords: RISC-V · deep learning · machine learning framework · cross-compile.

1 Introduction

RISC-V [1] is a clean-slate, open and free instruction set architecture (ISA), which allows anyone to develop both open-source and proprietary implementations. Its modular design for extensibility and specialization makes itself suitable for computing systems ranging from microcontrollers to supercomputers. Hence RISC-V is attracting extensive attention from both academia and industry. Meanwhile, as deep learning has become state-of-the-art in domains such as computer vision, voice recognition, natural language processing, etc., artificial intelligence (AI) is experiencing a renaissance, and a large variety of AI applications are being deployed in a wide range of computing platforms, including cloud, edge, mobile and IoT devices. Therefore, deep learning is expected to be enabled on RISC-V for both research and commercial purposes. However, mainstream machine learning frameworks (e.g. TensorFlow [2]) do not have a full-fledged RISC-V port for now, posing challenges to the development of AI applications targeting RISC-V.

On the other hand, since server and desktop markets are dominated by Intel x86, and mobile and IoT markets dominated by Arm, machine learning frameworks and applications have to be developed for different ISAs respectively, in order to cover various computing platforms. However, this may induce a large amount of development efforts. Fortunately, RISC-V would provide an unique opportunity for both machine learning frameworks and applications to be developed for only one ISA and to be deployed on a wide range of computing platforms. Therefore, machine learning frameworks and applications are especially worth porting to RISC-V.

In this paper, AIRV stands for "AI on RISC-V". Our vision is to enable a large variety of AI applications on a wide range of RISC-V platforms. At present, we focus on enabling deep learning inference on RISC-V, and evaluate the performance of deep learning inference on multiple platforms. Our contributions are summarized as follows:

- We explore approaches to enabling deep learning inference on RISC-V;
- We evaluate the performance of deep learning inference both on RISC-V and on x86;
- We show that in our work, compared with direct compilation on RISC-V, cross-compilation on x86 is a better option to reduce the development cycle of deep learning inference applications targeting RISC-V.

The paper is organized as follows. Section 2 summarizes related work. Section 3 briefly introduces the ResNet-20 deep neural network (DNN) and the CIFAR-10 dataset. Section 4 illustrates the architecture for deep learning inference applications in our work. Section 5 implements the architecture using TensorFlow Python, TensorFlow C and TensorFlow Lite for Microcontrollers, respectively. Section 6 compares two compilation methods: direct compilation and cross-compilation. Section 7 presents experimental results. Section 8 concludes the paper.

2 Related Work

Most prior work [3–5] focused on offloading deep learning inference tasks especially key kernels (e.g. convolution, activation, pooling) from RISC-V cores to a dedicated hardware accelerator to achieve high performance and/or energy efficiency for deep learning inference applications targeting mobile and/or IoT devices. They usually use a hardware/software co-design method which features an SoC incorporating RISC-V cores, a dedicated hardware accelerator for deep learning inference, and a specialized software toolchain comprised of customized libraries, compilers, etc.

In the aforementioned solutions, most deep learning inference tasks are actually executed on a deep learning accelerator, rather than a RISC-V core which is usually used as a microcontroller. Therefore, their solutions rely on dedicated hardwares and specialized softwares, which potentially incur additional programming efforts for application developers. Moreover, it may not be able to port the applications developed using their software tools to other RISC-V platforms especially those without a required hardware accelerator.

In this paper, we focus on directly running all deep learning inference tasks on a RISC-V based processor, instead of a dedicated hardware accelerator. One of our goals is to enable a large variety of deep learning inference applications on a wide range of RISC-V platforms using mainstream machine learning frameworks. Furthermore, we plan to accelerate deep learning inference on RISC-V using a pure hardware solution to improve performance and energy efficiency without incurring extra programming efforts for application developers, and we leave this optimization for future work.

3 Background

We briefly introduce the ResNet-20 deep neural network and the CIFAR-10 dataset, in preparation for illustration of the architecture for deep learning inference applications in our work.

3.1 ResNet-20

The ResNet [6] deep neural network originates from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [7], which provides millions of labeled images for the contestants to train and evaluate their deep neural networks. On the ILSVRC 2015 classification task, ResNet achieved 96.4% accuracy on the ImageNet test set and won the first place, which outperformed human accuracy (94.9%) [7] on image classification.

The ResNet-20 deep neural network in this paper is a pre-trained model from the 2019 BenchCouncil International AI System and Algorithm Challenges [8], which consist of four tracks: International AI System Challenge based on RISC-V, International AI System Challenge based on Cambricon Chip, International AI System Challenge based on X86 Platform, and International 3D Face Recognition Algorithm Challenge. The topics of the tracks are derived from AIBench [9, 10], the source code of which is publicly available from http://www.benchcouncil.org/benchhub/AIBench/ (sign up to get access). The ResNet-20 model is used to predict the most possible category of an image: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, or truck. The input of the model is a color image with three channels (i.e. red, green and blue); each channel is comprised of 32x32 pixels and each pixel is an integer ranging from 0 to 255. The output of the model is a vector consisting of ten floating point numbers (with each number ranging from 0.0 to 1.0), indicating the possibilities of an image belonging to the aforementioned categories, respectively. Thus we can predict the most possible category of the image by finding the maximum element of the vector.

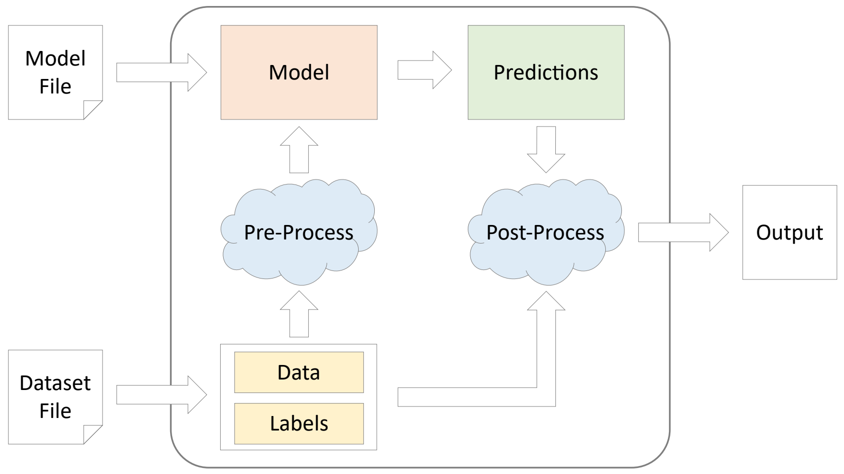

Fig. 1. Architecture for deep learning inference applications.

3.2 CIFAR-10

The CIFAR-10 [11] dataset consists of 60,000 color images and their labels; each image shares the same attributes with that described in section 3.1. CIFAR-10 is divided into two parts: a training dataset containing 50,000 labeled images for training deep neural networks and a test dataset containing 10,000 labeled images for evaluating deep neural networks. The label is used to judge whether a deep neural network has made a right prediction on the corresponding image. In other words, a right prediction has been made if the category of the image predicted by the deep neural network is the same as the corresponding label; otherwise, it's a wrong prediction.

The ResNet-20 deep neural network model used in our work, is trained on the CIFAR-10 training dataset, and achieves 84.03% accuracy on the CIFAR-10 test dataset.

4 Architecture

In this section, we illustrate the architecture for deep learning inference applications developed in our work. As depicted in Fig. 1, the architecture involves loading a pre-trained deep neural network model and a labeled dataset from corresponding files, pre-processing the data before input into the model (e.g. reshaping, transforming, normalization), predicting on the data using the model, post-processing the predictions (e.g. checking the predictions using the labels, calculating prediction accuracy), and outputting useful information (e.g. logs, final results).

5 Implementations

We implement the architecture illustrated in section 4 by using the ResNet-20 deep neural network model, the CIFAR-10 dataset and the TensorFlow machine learning framework. Three applications are developed using three TensorFlow libraries, i.e. TensorFlow Python, TensorFlow C and TensorFlow Lite for Microcontrollers, respectively.

The first application developed using the TensorFlow Python library can be successfully run on x86 and achieves 84.03% accuracy as expected. However, the application can not be successfully run on RISC-V because the TensorFlow Python library does not have a RISC-V version (for now). Although the library could be ported to RISC-V using Bazel, an open-source build tool, it may require a lot of developing efforts. For example, all machine-specific source code in the library must be properly modified. Thus we leave this port for future work.

Similarly, the second application developed using the TensorFlow C library can be successfully compiled and run on x86, and achieves 84.03% accuracy as well. However, the application suffers a similar problem as described above. Also, we leave the solution for future work.

The third application is developed using TensorFlow Lite for Microcontrollers, which is designed to have fewer dependencies. The application can be successfully compiled and run on RISC-V as well as on x86, and achieves 84.03% accuracy on both platforms.

6 Compilation Methods

In order to implement the architecture on RISC-V, as discussed in section 5, the application developed using TensorFlow Lite for Microcontrollers needs to be compiled to produce a binary which can be executed on RISC-V. There are two compilation methods: direct compilation and cross-compilation. As illustrated in Fig. 2, direct compilation means compiling the application on RISC-V using the RISC-V toolchain to produce a RISC-V executable, while cross-compilation means compiling the application on x86 using the RISC-V toolchain to produce a RISC-V executable.

Direct compilation seems to be the simplest way to compile the application, since the required RISC-V toolchain has already been pre-installed on the host platform which is exactly the target platform as well. However, we show that direct compilation may not be a good option because there are resource and performance challenges when directly compiling the application on RISC-V in our work.

Resource challenge. The application developed using TensorFlow Lite for Microcontrollers can not be successfully compiled on a RISC-V based FPGA board used in our work, due to limited memory on the board. Fortunately, this problem can be solved by augmenting the virtual memory.

Performance challenge. Although the aforementioned problem can be solved, it takes more than three hours to complete the direct compilation process

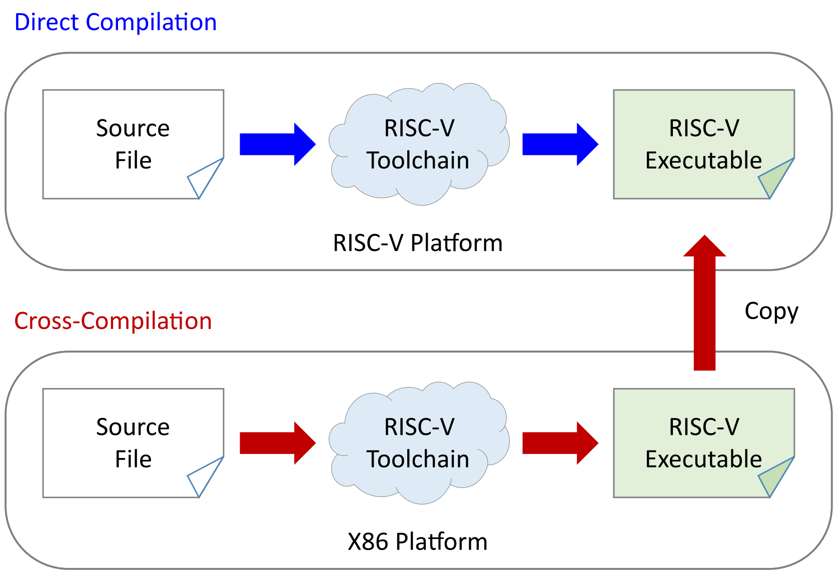

Fig. 2. Direct Compilation vs. Cross-Compilation.

on the board, which is about 300x slower than the cross-compilation process on an x86 server used in our work. Therefore, we prefer to cross-compile the application on the server so that we can save a large amount of time, despite it also takes a little time to copy the produced RISC-V executable from the server to the board.

7 Experimental Results

We evaluate the performance of compiling and running the application illustrated in section 5 on three different platforms: an Xilinx PYNQ Z2 FPGA board based on RISC-V, RISCV-QEMU [12] (a QEMU emulator for RISC-V), and an x86 server. The environment configurations of the platforms are listed in Table 1. The real time of compiling and running the application on the platforms is listed in Table 2 and illustrated in Fig. 3. Experimental results show that compiling the application on RISCV-QEMU and the RISC-V based FPGA board is about 16x and 293x slower than that on the x86 server, respectively; running the application on the two RISC-V platforms is about 28x and 620x slower than that on the x86 server, respectively. Therefore, high performance RISC-V based platforms are expected to be used to reduce the development cycle of applications targeting RISC-V.

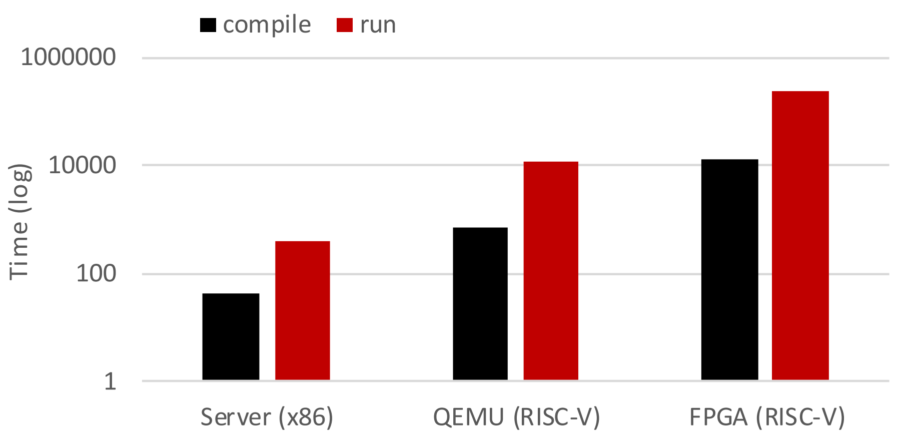

Fig. 3. Real time of compiling and running the application on the three platforms.

Table 1. Environment configurations of the three platforms.

	FPGA (RISC-V)	QEMU (RISC-V)	Server (x86)
OS	Linux 4.19.0	Linux 4.19.0	Linux 4.4.0
CPU	SiFive Rocket0, RV64GC, Sv39, 50 MHz, 1 core	RISC-V, RV64GCSU, Sv48, 8 cores	Intel Xeon E5-2697 v4 @ 2.30 GHz, 18 cores
Memory	180 MB	16 GB	64 GB
gcc	gcc 9.2.1	gcc 7.3.1	gcc 7.4.0
g++	g++ 9.2.1	g++ 7.3.1	g++ 7.4.0
python	Python 2.7.16+	Python 2.7.14	Python 2.7.6
python3	Python 3.7.4+	Python 3.6.4	Python 3.6.8
make	GNU Make 4.2.1	GNU Make 4.2.1	GNU Make 4.2.1

Table 2. Real time of compiling and running the application on the three platforms.

	FPGA (RISC-V)	QEMU (RISC-V)	Server (x86)
Compile Time (s)	12744.30	713.40	43.55
Run Time (s)	250601.78	11429.58	404.21

8 Conclusions

In this paper, we explore approaches to enabling deep learning inference on RISC-V by implementing the architecture for deep learning inference applications using ResNet-20, CIFAR-10 and three TensorFlow libraries. At present, since TensorFlow does not have a full-fledged RISC-V port, only the application developed using TensorFlow Lite for Microcontrollers can be successfully compiled and run on RISC-V and achieves the expected accuracy. Still, we show that there are resource and performance challenges when directly compiling and running the application on two RISC-V platforms in our work, and cross-compiling the application on a high performance x86 platform is a better option for us to save a large amount of time. We evaluate the performance of compiling and running the application on three platforms: an Xilinx PYNQ Z2 FPGA board based on RISC-V, RISCV-QEMU, and an x86 server. Experimental results show that directly compiling the application on the two RISC-V platforms is about 16x and 293x slower than that on the x86 server, respectively; running the application on the two RISC-V platforms is about 28x and 620x slower than that on the x86 server, respectively. Therefore, high performance RISC-V based platforms are expected to be used to speed up the development of applications targeting RISC-V.

Future work involves implementing the architecture on RISC-V using TensorFlow Python, TensorFlow C, and other mainstream machine learning frameworks such as PyTorch. Furthermore, we plan to accelerate deep learning inference using a pure hardware solution to improve performance and energy efficiency without incurring extra programming efforts for application developers.

9 Acknowledgements

This work is supported by the Strategic Priority Research Program of Chinese Academy of Sciences, under Grant No. XDC02010200.

References

1. RISC-V, https://riscv.org.
2. TensorFlow, https://www.tensorflow.org.
3. S. Davidson et al.: The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric: Fast Architectures and Design Methodologies for Fast Chips. in IEEE Micro, vol. 38, no. 2, pp. 30–41, Mar./Apr. 2018.
4. E. Flamand et al.: GAP-8: A RISC-V SoC for AI at the Edge of the IoT. 2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP), Milan, 2018, pp. 1–4.
5. M. S. Louis et al.: Towards Deep Learning using TensorFlow Lite on RISC-V. Third Workshop on Computer Architecture Research with RISC-V (CARRV). 2019.
6. K. He, X. Zhang, S. Ren and J. Sun: Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770–778.

7. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision 115.3 (2015): 211–252.
8. 2019 BenchCouncil International AI System and Algorithm Challenges, http://www.benchcouncil.org/competitions.html.
9. Wanling Gao, Fei Tang, Lei Wang, Jianfeng Zhan, Chunxin Lan, Chunjie Luo, Yunyou Huang, Chen Zheng, Jiahui Dai, Zheng Cao, Daoyi Zheng, Haoning Tang, Kunlin Zhan, Biao Wang, Defei Kong, Tong Wu, Minghe Yu, Chongkang Tan, Huan Li, Xinhui Tian, Yatao Li, Gang Lu, Junchao Shao, Zhenyu Wang, Xiaoyu Wang, and Hainan Ye: AIBench: An Industry Standard Internet Service AI Benchmark Suite. Technical Report 2019.
10. Wanling Gao, Chunjie Luo, Lei Wang, Xingwang Xiong, Jianan Chen, Tianshu Hao, Zihan Jiang, Fanda Fan, Mengjia Du, Yunyou Huang, Fan Zhang, Xu Wen, Chen Zheng, Xiwen He, Jiahui Dai, Hainan Ye, Zheng Cao, Zhen Jia, Kunlin Zhan, Haoning Tang, Daoyi Zheng, Biwei Xie, Wei Li, Xiaoyu Wang, Jianfeng Zhan: AIBench: Towards Scalable and Comprehensive Datacenter AI Benchmarking. In BenchCouncil International Symposium on Benchmarking, Measuring and Optimizing (Bench18), 2018.
11. CIFAR-10, https://www.cs.toronto.edu/˜kriz/cifar.html.
12. RISCV-QEMU, https://hub.docker.com/r/crva/riscv-qemu.