USING AUTOTVM TO AUTOMATICALLY GENERATE DEEP LEARNING LIBRARIES FOR MOBILE DEVICES

Eddie Yan [1] **Tianqi Chen** [1] **Lianmin Zheng** [2] **Ziheng Jiang** [1] **Thierry Moreau** [1] **Carlos Guestrin** [1] **Luis Ceze** [1] **Arvind Krishnamurthy** [1]

ABSTRACT

Providing automated performance portability remains an important challenge for deep learning systems. While popular networks enjoy high performance on popular hardware targets such as server-class GPUs and TPUs, porting even off-the-shelf models with reasonable performance to mobile and embedded systems remains challenging. In this demo, we will demonstrate the AutoTVM system, which provides an automated end-to-end solution for optimizing deep learning operator implementations from computational graph level model specification to specialized operator libraries.

1 INTRODUCTION

Developers of deep learning models have enjoyed tremendous advancements in productivity in recent years. Cutting edge GPU and TPU hardware allow for ever-faster model training and deployment, while operator and framework support continues to improve for increasingly exotic model architectures and operators. On the other hand, support on edge devices, where deep learning deployment stands to have the largest impact on communication bandwidth and energy savings, remains limited. Our demo aims to show an automated end-to-end solution for optimizing and deploying deep learning models on edge hardware, specifically single-board-computer and mobile phone form-factor devices. We demonstrate automation from model ingestion (via standard frameworks such as MxNet (Chen et al., 2015) and PyTorch (Paszke et al., 2017)) to operator implementation tuning (for specific hardware devices such as mobile CPUs and GPUs) to deployment (compilation of device-specific libraries).

2 BACKGROUND

Traditional deep learning frameworks leverage vast libraries of highly optimized deep learning operator kernels backed by intensive engineering effort to provide performance on commonly used hardware devices such as server and desktop class GPUs. Support for mobile devices has also adopted this approach. In place of GPU libraries such as cuDNN (Chetlur et al., 2014) and MIOpen (mio), mobile devices rely on libraries such as NCNN (ncn), ARMComputeLibrary (arm), and TensorFlow Lite (tfl). However, the sheer diversity of mobile device SoCs (different CPU and GPU configurations) in addition to the number of possible model architectures used by application developers means that library support is in a perpetual race against the clock as new models emerge and older models fall out of fashion. In fields such as computer vision, model turnover can be especially rapid for mobile devices as improved features (e.g., object detection vs. classification), improved accuracy (e.g., ResNet (He et al., 2016) vs. VGG (Simonyan & Zisserman, 2014)), and power-savings (e.g., ResNet vs. MobileNet (Howard et al., 2017) or quantized models) are all highly valued.

Instead of relying on handcrafted libraries of deep learning operators that require vast engineering resources to develop, we will demonstrate an automated approach. Our demo demonstrates the application of the AutoTVM (Chen et al., 2018) system to an edge-device hardware target. AutoTVM extracts operator requirements automatically from a neural network computation graph, and then leverages a machine-learning guided optimizer to tune operator implementations for performance on real hardware. This process is made possible by the TVM language, which allows us to programmatically define a rich search space of implementation options analogous to what a human engineer can explore manually. At the end of this process, AutoTVM produces executable libraries customized to the specific hardware platform and model architecture, without laborious manual effort.

[1]Paul G. Allen School of Computer Science and Engineering, Seattle, Washington, USA. http://sampl.ai. [2]Shanghai Jiao Tong University, Shanghai, China. Correspondence to: Eddie Yan <eqy@cs.washington.edu>.

Proceedings of the 2^{nd} SysML Conference, Palo Alto, CA, USA, 2019.

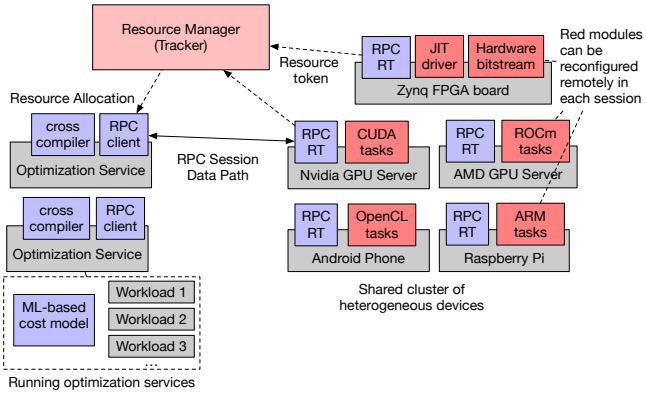

Figure 1. Overview of the TVM RPC system used by AutoTVM. AutoTVM optimization services request hardware resources from a centralized tracker that allows a wide range of hardware devices to be shared across optimization tasks. This system allows optimization algorithms to use arbitrary hardware devices, regardless of the algorithm's compute platform.

3 System Overview

Our demo will leverage the TVM RPC system, which allows hardware devices to be shared across different optimization tasks. Using this system, optimization algorithms request hardware resource tokens from a centralized tracker, which provide access to devices ranging from GPUs to single board computers to remote cloud instances. This access allows optimization algorithms to profile device performance on real hardware devices, independently of their own computation platform. Figure 1 shows an example TVM RPC system setup containing different hardware devices.

4 Demo Details

Our demo will show the example of a user porting a popular computer vision model to a single board computer (e.g., Raspberry Pi 3B, RK3399, Ultra96 FPGA board) or comparable mobile phone. Initially, after extracting the required operators from the models computation graph, AutoTVM will be able to achieve a fallback level of performance (without optimization). As AutoTVM completes more iterations of optimization and finds faster and faster implementations of each operator, we will see the run time of the model improve until it exceeds the speeds achieved by out-of-the-box library solutions, such as Tensorflow Lite. We will use a Python tutorial script to highlight the ease-of-deployment of AutoTVM during this portion, and encourage audience interruptions and questions regarding the use of our stack. At this point of the demo, we can leverage distributed remote infrastructure to speed up the optimization pipeline, as it involves both performance measurement on real hardware (which can be outsourced to a cluster of devices), and model fitting and implementation proposal (which can also be done remotely). If the venue does not have an internet connection that is suitable for using distributed infrastructure, we can reliably fall back on local hardware without sacrificing the required performance for demo purposes. Our demo hardware (excluding remote devices if connected) will be a single board computer and a laptop. As demo time is limited, we will also show the maximum performance achieved by our system after hours of optimization time. Note that both the processes of importing a model from a framework and switching the optimization target from one device to another are push-button, and AutoTVM is portable and performance-competitive across a wide range of model architectures. Additionally, whereas our demo focuses on the case of mobile and edge devices, our stack is portable across many hardware types (e.g., x86 CPU, NVIDIA GPU, AMD GPU, embedded FPGA).

References

Arm-software/computelibrary. https://github.com/ARM-software/ComputeLibrary. Accessed: 2019-02-09.

Rocmsoftwareplatform/miopen. https://github.com/ROCmSoftwarePlatform/MIOpen. Accessed: 2019-02-09.

Tensorflow lite — tensorflow. https://www.tensorflow.org/lite. Accessed: 2019-02-09.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015. URL http://arxiv.org/abs/1512.01274.

Chen, T., Zheng, L., Yan, E. Q., Jiang, Z., Moreau, T., Ceze, L., Guestrin, C., and Krishnamurthy, A. Learning to optimize tensor programs. *CoRR*, abs/1805.08166, 2018. URL http://arxiv.org/abs/1805.08166.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759, 2014. URL http://arxiv.org/abs/1410.0759.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets:

Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.