RSTensorFlow: GPU Enabled TensorFlow for Deep Learning on Commodity Android Devices

Project 6, Task 2 (Deep Learning for Multi-Layer Situational Understanding)

Moustafa Alzantot*, Yingnan Wang†, Zhengshuang Rene‡ and Mani Srivastava§
University of California, Los Angeles
Los Angeles, CA, USA
Email: *malzantot@ucla.edu, †yingnanwang@ucla.edu, ‡zhengshuangren@gmail.com, §mbs@ucla.edu

***Abstract*—Small networked devices with low computing power (such as phones, surveillance cameras, and IoT devices) have become an essential part of our daily lives. They collect a lot of data and apply machine learning to make useful inferences from it. Although these devices can process its collected by sending it to a remote server, For security and privacy constraints it is, sometimes, preferred to process the data locally on the device. However, popular and commonly used tools and frameworks for machine intelligence are still lacking the ability to make proper use of the available heterogeneous computing resources on these low-end devices. In this paper, we study the benefits of utilizing the heterogeneous (CPU and GPU) computing resources available on commodity Android devices while running deep learning models. We leveraged the heterogeneous computing framework `RenderScript` to accelerate the execution of deep learning models on commodity Android devices. Our system is implemented as an extension to the popular open-source framework `TensorFlow`. By integrating our acceleration framework tightly into `TensorFlow`, machine learning engineers can now easily make benefit of the heterogeneous computing resources on mobile devices without the need of any extra tools. We evaluate our system on different android phones models to study the trade-offs of running different neural network operations on the GPU. Our result shows that although GPUs on the phones are capable of offering substantial performance gain in matrix multiplication on mobile devices. Therefore, models that involve multiplication of large matrices can run much faster (approx. *3 times faster* in our experiments) due to GPU support.**

I. Introduction

Recent developments in artificial intelligence and machine learning have made huge leaps in the accuracy of machine perception algorithms in different domains such as object detection, and speech recognition. A lot of this progression comes due to the renaissance of deep neural networks (a.k.a. deep learning [1]) methods. Running the deep learning model locally - on device - saves the time and money spent on sending data to remote servers and reinforces the user privacy. However, running deep learning models involves a massive amount of calculations. Therefore, a lot of applications prefer to send the data from the mobile device to remote servers where the model runs and sends the result back to device despite the obvious benefits of running the models locally on mobile devices. Although popular deep learning frameworks (e.g. Caffe, Torch, Theano, and TensorFlow) accelerate the computation of deep learning models by utilizing heterogeneous hard-aware (CPU / GPU) resources and even custom hardware accelerator such as tensor processing units (TPU) used in Google data centers. When running the mobile device versions of these frameworks run entirely on the device CPU. In this paper, we introduce `RSTensorFlow` an extended version of `TensorFlow` that supports heterogeneous computing resources for commodity Android devices. `RSTensorFlow` is implemented by modifying the *kernels* of `TensorFlow` operations to leverage the `RenderScript` heterogeneous computing framework on Android devices.

In this paper, we make the following contributions: we introduce and implement `RSTensorFlow` a modified version for `TensorFlow` that supports both CPU and GPU on commodity android devices, and benchmark and evaluate the trade-offs of running common deep learning operations (namely matrix multiplication and convolution) on CPU vs GPU on commodity android phones.Also we provide our framework `RSTensorFlow` as an open-source project[1] for the research community.

We evaluate the performance of our system on different Android devices (Nexus 5x, Nexus 6). Our results show that we achieve *up to 3 times* speedup in running the inception model on Nexus 5X phone.

II. System Design

Running inferences using neural network model requires executing the forward pass of the model which involves different operations. To accelerate the model inference, we focus our efforts on the matrix multiplication, and convolution operations because they represent the biggest share in the model inference time. The following subsections discuss our approach to modify `TensorFlow` to run these operations using `RenderScript` instead of the default `Eigen` ARM `NEON`-based implementation.

A. Matrix Multiplication (`MatMul`)

Matrix multiplication operation (`MatMul`) is an essential ingredient in all kinds of deep learning models as fully connected layers require matrix multiplication between the input matrix and the weight matrix. We modified `TensorFlow` to

[1] https://nesl.github.io/RSTensorFlow/

Batch size	Nexus 6			Nexus 5X		
	Original TF	TF + RS MatMul & RS Conv2D	TF + RS MatMul	Original TF	TF + RS MatMul & RS Conv2D	TF + RS MatMul
1	0.453	1.765	0.312	0.699	2.775	0.351
2	0.718	1.757	0.370	1.235	2.782	0.471
3	0.979	1.879	0.475	1.785	2.811	0.649

TABLE I

COMPARISON OF THE TIME (IN SECONDS) REQUIRED TO RUN THE FORWARD PASS OF INCEPTION MODEL USING THE ORIGINAL TENSORFLOW V1.0.1, TENSORFLOW WITH RENDERSCRIPT MATRIX MULTIPLICATION AND CONVOLUTION OPERATIONS, AND TENSORFLOW WITH RENDERSCRIPT MATRIX MULTIPLICATION ONLY.

make use of the `RenderScript` implementation of matrix multiplication instead of the default `MatMul` implementation that uses `Eigen` library.

B. Convolution Operation (`Conv2D`)

Convolution operations are the core building block of CNN models such as the inception model [2] which has 22 convolution layers. Convolution layer consists of a number of filters (their values are learned during the training phase). The input and output of each convolution layer are represented as volumes where the depth of volume represents the number of feature maps. In order to parallelize the convolution operation using `RenderScript`, we developed a render script kernel file to execute the computation of output values in the convolution output volume in parallel.

III. EXPERIMENTS

We implemented `RSTensorFlow` by extending the `TensorFlow` v1.0.1 implementation. We used Nexus 6 and Nexus 5X mobile phones.

A. Matrix Multiplication Operation Results

We benchmark the running time of the modified matrix multiplication operation and compare it against the running time of the default `Eigen`-based implementation on the two different phones. We perform matrix multiplication between square-matrices of different sizes and measure the time for each multiplication. The timing result are shown in Figure 1.

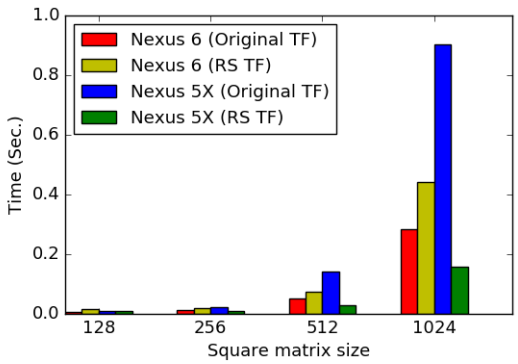

Fig. 1. Time of matrix multiplication between square matrices using Original TensorFlow and RenderScript TensorFlow

The results of our matrix multiplication experiments show that on Nexus 5X the `RenderScript` implementation becomes significantly faster as the matrix size increases. When square matrix size = 1024, matrix multiplication using `RenderScript` took 158 milli-seconds which is *6 times* faster than the default implementation using `Eigen` library that took 904 milli-seconds.

B. Convolution Operation Results

We also benchmarked the running time of both our RenderScript-based implementation of the convolution operation and compared it against the running of the default `Eigen`-based implementation. Unfortunately, we have not noticed speed up in the performance of convolution operation.This might be due to the memory overhead associated with using `RenderScript` that required copying data from/to special buffers (referred to as allocations in `RenderScript`).

C. ConvNet Model Results

Table I shows how `RSTensorFlow` can accelerate the inference running time of inception model approximately *3 times* on Nexus 5X.

IV. CONCLUSION

In this paper, we introduced `RSTensorFlow` an accelerated deep learning framework on commodity android devices using the heterogeneous computing framework `RenderScript`. Although, we noticed that GPU was not used by `RenderScript` on all phone models. When GPU is used, `RSTensorFlow` improves matrix multiplication operations a lot and therefore we observed significant speedup in running different models on Nexus 5X phones.

ACKNOWLEDGEMENT

This research was supported in part by the NIH Center of Excellence for Mobile Sensor Data-to-Knowledge (MD2K) under award 1-U54EB020404-01, and by the U.S. Army Research Laboratory and the UK Ministry of Defence under Agreement Number W911NF-16-3-0001. Any findings in this material are those of the author(s) and do not reflect the views of any of the above funding agencies. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning, 2016.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.