ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/328546129

ECRT: An Edge Computing System for Real-Time Image-based Object Tracking

Conference Paper · November 2018

DOI: 10.1145/3274783.3275199

CITATIONS
2

READS
530

5 authors, including:

Zhihe Zhao
Duke University
4 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Xian Shuai
The Chinese University of Hong Kong
2 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project

Research on IoT-motivated cyber-physical platform using wireless sensor network View project

All content following this page was uploaded by Zhihe Zhao on 29 October 2018.

The user has requested enhancement of the downloaded file.

Demo Abstract: ECRT: An Edge Computing System for Real-Time Image-based Object Tracking

Zhihe Zhao[1,2], Zhehao Jiang[2], Neiwen Ling[2], Xian Shuai[2] and Guoliang Xing[2]
[1]Xi'an Jiaotong-Liverpool University
[2]The Chinese University of Hong Kong

ABSTRACT

Real-time image-based object tracking from live video is of great importance for several smart city applications like surveillance, intelligent traffic management and autonomous driving. Although recent deep learning systems can achieve satisfactory tracking performance, they incur significant compute overhead, which prevents them from wide adoption on resource-constrained IoT platforms. In this demonstration, we present an Edge Computing system for Real-time object Tracking (ECRT) for resource-constrained devices. The key feature of our system is that it intelligently partitions compute-intensive tasks such as inferencing a convolutional neural network(CNN) into two parts, which are executed locally on an IoT device and/or on the edge server. Moreover, ECRT can minimize the power consumption of IoT devices while taking into consideration the dynamic network environment and user requirement on end to end delay.

CCS CONCEPTS

• **Computer systems organization → Real-time system architecture**; • **Computing methodologies** → *Tracking*;

KEYWORDS

Edge computing, Real-time embedded system, Computer vision

ACM Reference Format:
Zhihe Zhao[1,2], Zhehao Jiang[2], Neiwen Ling[2], Xian Shuai[2] and Guoliang Xing[2]. 2018. Demo Abstract: ECRT: An Edge Computing System for Real-Time Image-based Object Tracking. In *Proceedings of ACM Conference (SenSys'18).* ACM, Shenzhen, China, 2 pages. https://doi.org/10.1145/3274783.3275199

1 INTRODUCTION

Several approaches have been proposed to reduce the compute overhead of deep learning algorithms for IoT devices, including quantization of convolutional layers and fully connected layers [2, 7]. However, these methods inevitably sacrifice the quality of solution. On the other hand, several existing systems can offload compute-intensive tasks like CNNs to the Cloud. However, such Cloud-based solutions are ill-suited for real-time applications due to the unpredictable communication delays in transmitting live video streams.

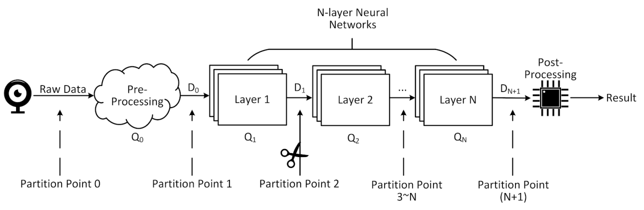


Figure 1: The processing pipeline of ECRT which consists N+2 possible partition points in a N-layer Neural Network model.

In this demo, we present an ECRT that can partition a deep learning algorithm for object-tracking dynamically between an IoT device (e.g., a battery-powered wireless camera) and the edge server. Different from the existing offloading approach [6], ECRT intelligently partitions the execution of a CNN into two subsets of layers. Such a fine-grained runtime partition mechanism enables the system to minimize the power consumption of the IoT device while meeting the application requirement on latency in the presence of dynamic network bandwidth.

We now use a real-world example to motivate the design of ECRT. Consider a real-time video surveillance system that consists of a battery-powered Raspberry Pi 3B+ which can transmit images captured by its camera wirelessly to an edge server, NVIDIA Jetson TX2 with 256 CUDA cores[1, 4]. YOLO[5], a widely used object recognition model which has 24 convolutional layers and 3 full connect layers, takes nearly 20 seconds to process a single image frame on Raspberry Pi while taking only about 0.3 seconds on NVIDIA Jetson TX2. However, it consumes considerable energy and causes a communication delay when transmitting a live video stream to the edge. In such a scenario, when the user requires an end to end delay below 20 seconds, it is thus desirable to partition YOLO into two subsets of layers to execute on Raspberry Pi and edge server respectively, which reduces the power consumption of Raspberry Pi compared with executing YOLO locally or offloading the video entirely to the edge. In other words, such a fined-grained partition mechanism can achieve a desirable trade-off between power consumption and application delay under a wide range of settings.

2 SYSTEM DESIGN

ECRT contains two components, the IoT device and the edge server. A CNN-based pipeline consists of preprocessing like image resizing, CNN model, and post-processing like bounding box. The system will partition this pipeline into two parts and execute them on the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SenSys'18, November 4-7, 2018, Shenzhen, China
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5952-8/18/11…$15.00
https://doi.org/10.1145/3274783.3275199

IoT device and edge server respectively. The objective is to minimize power consumption of the IoT device while meeting the latency required by user. Depending on where the partition point lies, the resulted compute architecture is one of the following modes:

(i) Local mode: When the IoT device receives data from cameras, it processes the images and outputs the detected objects.
(ii) Edge mode: When the IoT device receives data from cameras, it sends the raw data to the edge server, which then executes the whole CNN model and sends output to the IoT device.
(iii) Collaboration mode: When the IoT device receives data from cameras, it executes a subset of CNN layers. The edge server executes the rest CNN layers and sends output to the IoT device.

Our design explores the trade-off between latency and power consumption, while accounting for different user requirements on latency and dynamic network environments. The end to end latency includes the task execution, data transmission and serialization (the process of converting data to be transmitted to a stream of bytes).

We now formally formulate the problem of task partitioning in ECRT. Define $i \in \{0, 1, ..., n+1\}$, where i is the i^{th} partition point in our model and n is the number of layers in an N-layer deep neural network (DNN). The corresponding i^{th} partition point is shown in Fig. 1. The serialization and transmission latency mainly depends on the size of output data at the i^{th} partition point. Define $s(i)$ as the function of serialization and transmission latency, $Q_n(i)$ as the computing delay on IoT device and $Q_e(i)$ as the computing delay on edge with respect to the partition point i. Hence the total computing delay is:

$$c(i) = \sum_{k=0}^{i-1} Q_n(k) + \sum_{k=i}^{n+1} Q_e(k) \tag{1}$$

The end to end delay $T(i)$ including the transmission delay is

$$T(i) = s(i) + c(i) \tag{2}$$

From equation 2, if $i = 0$, $T(i) = s(i) + \sum_{k=0}^{n+1} Q_e(k)$, which represents the latency of edge mode. When $i = n+1$, $T(i) = s(i) + \sum_{k=0}^{n+1} Q_n(k)$, which represents the latency of local mode. Thus, the local and edge modes can be regarded as the special cases of collaboration mode.

$$\begin{aligned} &i = \operatorname{argmin} P(i),\ i \in \{0, 1, 2, ...n+1\}, \\ &s.t. \quad T(i) \le \Delta, \end{aligned} \tag{3}$$

$P(i)$ represents the total power cost of the IoT device when the i^{th} partition point is chosen and Δ is the end to end delay specified by the user. In ECRT, $P(i)$ and $T(i)$ are updated in real time.

3 INTERACTIVE DEMONSTRATION

As is shown in Fig. 2, We implement the demo using an Raspberry Pi 3B+ as an IoT device, An NVIDIA Jetson TX2 as the edge server and an Arduino Uno board for measuring real-time current through the IoT device [3]. We deploy YOLOv2-Tiny, a real-time object detection system including a pre-trained 9-layer convolutional network as the inference model. It divides the input image into a 13×13 grid and predicts a (13,13,125) tensor including the features of the object class and bounding box [5]. A web GUI is designed for users to set the bound of delay and to show the system parameters discussed in section 2 as well as the real-time object tracking results from three Raspberry Pis, each runs in local, edge or collaboration mode.

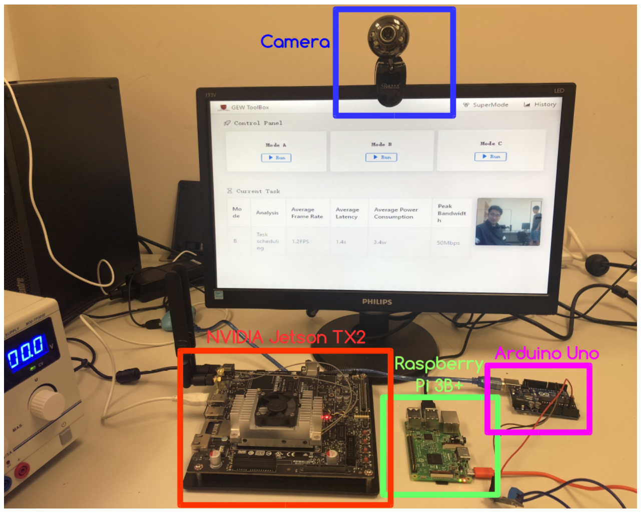
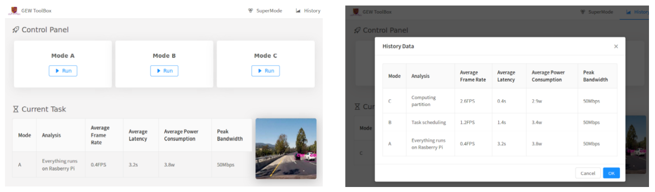


Figure 2: ECRT demonstration setup: A USB camera connected to a Rapberry Pi 3B+, An NVIDIA Jetson TX2 as the edge server, A Pi as an IoT device, An Arduino Uno board used to measure real-time current through the IoT device and a monitor that shows online running results of ECRT.



Mode	Analysis	Average Frame Rate	Average Latency	Average Power Consumption	Peak Bandwidth
A	Everything runs on Rasberry Pi	0.4FPS	3.2s	3.8w	50Mbps

Mode	Analysis	Average Frame Rate	Average Latency	Average Power Consumption	Peak Bandwidth
C	Computing partition	2.6FPS	0.4s	2.9w	50Mbps
B	Task scheduling	1.2FPS	1.4s	3.4w	50Mbps
A	Everything runs on Rasberry Pi	0.4FPS	3.2s	3.8w	50Mbps

Figure 3: Screenshots of the web GUI of ECRT where the user can select one of three modes. It also shows the history records of the tasks as well as their concerning data.

During the demonstration, ECRT will detect and track objects in front of the camera including moving people in the hall. A monitor will be used to show the online running results of ECRT via the web GUI. The detection bounding boxes will be shown when ECRT tracks different objects. Participants can interact with the demo by setting different bounds of system delay and power consumption.

REFERENCES

[1] Raspberry Pi Foundation. 2018. Raspberry Pi (2018). https://www.raspberrypi.org.
[2] Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *CoRR* abs/1510.00149 (2015). arXiv:1510.00149 http://arxiv.org/abs/1510.00149
[3] Arduino Inc. 2018. Arduino Uno (2018). https:https://www.arduino.cc/.
[4] NVIDIA Inc. 2018. NVIDIA TX2 (2018). https://www.nvidia.com/zhcn/autonomous-machines/embedded-systems-dev-kits-modules/.
[5] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-Time Object Detection. *CoRR* abs/1506.02640 (2015). arXiv:1506.02640 http://arxiv.org/abs/1506.02640
[6] Siri Team. 2017. Hey Siri: An On-device DNN-powered Voice Trigger for AppleâĂŹs Personal Assistant. https://machinelearning.apple.com/2017/10/01/hey-siri.html.
[7] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. 2016. Quantized Convolutional Neural Networks for Mobile Devices. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4820–4828. https://doi.org/10.1109/CVPR.2016.521

View publication stats