An opensource framework for edge-to-cloud inference on resource-constrained RISC-V systems

Mehdi Akeddar*, Thomas Rieder, Guillaume Chacun,
Bruno Da Rocha Carvalho and Marina Zapater

REDS. School of Engineering and Management Vaud, HES-SO University of Applied Sciences and Arts Western Switzerland

Abstract

In recent years we are witnessing an increasing adoption of RISC-V based systems to run Artificial Intelligence (AI) inference tasks. This trend spans to visual navigation, where major players start adopting RISC-V for autonomous driving. Still, RISC-V based edge devices fall short in providing the performance requirements of complex AI inference. Our work tackles the previous challenges by proposing an opensource framework for transparent distribution of visual navigation inference tasks between edge and cloud for resource-constrained RISC-V edge devices. Our framework automates the partitioning of ONNX and TFLite models between a RISC-V accelerated nanodrone equipped a GAP8 system-on-chip and a cloud server. Our results showcase how partial inference improves the performance achieve by drone-only inference.

Introduction

RISC-V is gaining popularity in academia and industry, particularly in emerging domains. The autonomous driving market is one such domain experiencing high growth, and open-source software and RISC-V are becoming essential to fuel innovation. Major players like SiFive, Intel, and ARM have recently adopted RISC-V for autonomous driving. However, most current RISC-V systems and do not possess the necessary performance to execute complex inference tasks.

Our work tackles the performance challenge of RISC-V platforms running Artificial Intelligence (AI) inference in autonomous driving by proposing an open-source framework enabling edge-to-cloud partial inference. Our framework is generic enough to run partial inference in out-of-the-box hardware such as embedded CPUs (Rapsberry Pi) and GPUs (Jetson Nano) on arbitrary models described on ONNX, while at the same time being flexible enough to run on RISC-V resource-constrained devices.

We automate the transition from edge to cloud of deep neural network models (DNNs) running on a GAP8 multicore RISC-V. Using a Crazyflie 2.1 drone equipped with a GAP8 accelerator, we showcase how our approach allows offloading computation from the drone to a server in an efficient way, improving the performance of inference tasks. More specifically, our contributions are as follows:

- We implement an edge-to-cloud partial inference solution that enables running ONNX and TFLite models on drone and server sides.
- We develop an algorithms based on the work by [1] allowing to choose the best cutting point of DNNs, and showing how partial inference can enhance the performance of edge-only inference.

*Corresponding Author: mehdi.akeddar@heig-vd.ch

Edge-to-cloud AI inference

The main blocks (hardware and software stack) comprising our edge-to-cloud framework for AI inference are shown in Fig. 1. The backend is comprised of an AI inference server running NVIDIA Triton which can execute arbitrary ONNX models, either head-to-tail (i.e., from the input to the output layer) or tail-only (i.e., receives a partial inference from a head model that is run on the edge). A monitoring and visualization tool based on graphite/prometheus and Graphana logs several performance parameters (inferences per second at edge and cloud, network connection speed, power consumption of the edge nodes).

Arbitrary nodes (in our specific setup we tested a Raspberry Pi, a smartphone and a Jetson Nano) can act as edge device and execute head models, provided that they are specified in ONNX for compatibility with the server backend.

To showcase autonomous navigation on resource-constrained RISC-V systems our platform provides support for running partial inference on the Crazyflie 2.1 nanodrone. The hardware configuration of our nanodrone is similar to the one proposed by [2]. The nanodrone is an ARM-based system equipping an STM32 processor. An AI-deck 1.1 which contains a GAP8 8-core RISC-V accelerator, a Himax camera and a WiFi module is connected via UART to the nanodrone. AI inference is accelerated on the AI-deck's GAP8, whereas the very limited STM32 equipped on the nanodrone solely takes care of computing flight commands and reading inference results via the UART.

Given the format expected as input from the Triton Inference Server and to unload computing from the AI-deck, we implement a bridge that receives the partial inference result from the nanodrone via TCP and translates that payload into the server expected input.

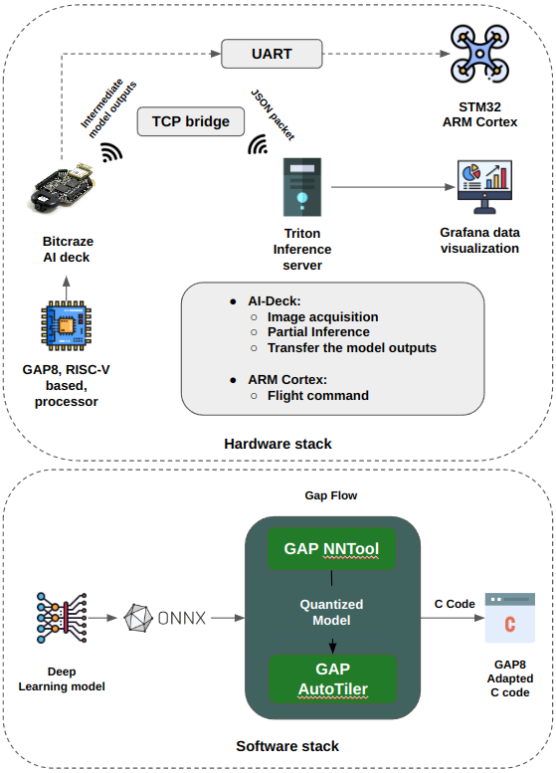

Figure 1: *Hardware (top) and software (bottom) stack of our edge-to-cloud framework for partial AI inference*

The main challenge towards the generality of our framework is the deployment of AI inference on the drone, while maintaining compatibility with the server backend. We use the GAP8 toolchain to convert the DNN models saved in ONNX to C code executable on the drone. Two tools are used for this purpose: (a) NNTool: quantizes and converts the ONNX input model, and (b) the AutoTiler, in charge of generating GAP8-compatible code for memory management and data transfers between all levels.

NNTool and quantization are of extreme importance given that the RISC-V cores of the GAP8 do not have a floating point unit, therefore, the DNNs inputs, weights and biases need to be quantized to either 8-bit integer or 16-bit fixed-point values.

Experimental setup and results

We showcase the benefits of our framework by running three DNNs with different characteristics:

- MobileNet v2, provided on Pytorch. The model takes as input an RGB image of shape 3x224x224 and outputs a probability for each of the 1001 classes.
- The Dronet proposed in [3] provided in TFLite.
- A custom CNN model implemented in ONNX with 3 2D conv layers, 2 linear, and 1 dropout. The model takes as input a grayscale image of shape 1x200x200 and outputs 2 features for collision or not provided on ONNX.

Our proposed algorithm for edge-to-cloud workload partitioning utilizes the Autodidactic Neurosurgeon (ANS) algorithm from [1]. We test different configurations, including cloud-only, edge-only, and partial inference, by deploying MobileNet v2 on a Jetson Nano and Triton Server. Due to the drone's limited memory capacity, which cannot hold intermediate model outputs, we are limited to testing only on the Triton-Jetson setup. Additionally, MobileNet v2 cannot be directly translated, as not all model layers can be quantized by NNtools. The other two DNNs (Dronet and our model) are deployed on the drone using NNTool.

Our CNN puts together the benefits of the former two experiments. This model is transformed into ONNX and only NNTool compatible layers are used. We avoid skip and feedback connections to enable further cutting points. Fig 2 shows how we translate the output of our CNN into the drone, by taking the index of the highest value among the two outputs of the model (0 - no collision; 1 - collision). In terms of performance, edge-only inference runs at 6Hz. An improvement of 50% is observed when we transit to cloud-only, the inference runs at 9Hz, encouraging the use of partial inference.

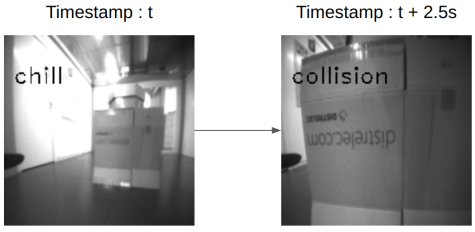

Figure 2: *Examples of custom model inference values in a test environment*

Conclusions

Our proposed framework for partial inference is currently being used for teaching an optional course on autonomous navigation using the Crazyflie in a BSc in Computer Engineering, and will be released open-source as an out-of-the-box solution.

Acknowledgements

This work has been funded by the ECO4AI project granted by HES-SO, under the call for young researchers 2021.

References

[1] Letian Zhang, Lixing Chen, and Jie Xu. "Autodidactic neurosurgeon: Collaborative deep inference for mobile edge intelligence via online learning". In: *WWW 2021* (2021).

[2] Palossi et al. "A 64mW DNN-based Visual Navigation Engine for Autonomous Nano-Drones". In: *IEEE IoT* (2018).

[3] Antonio Loquercio et al. "DroNet: Learning to Fly by Driving". In: *IEEE Robotics and Automation Letters* (2018).