CacheInf: Collaborative Edge-Cloud Cache System for Efficient Robotic Visual Model Inference

Abstract

Placeholder Placeholder Placeholder Placeholder

1 Introduction

Visual information is vital for various robotic tasks deployed on real-world edge devices (typically mobile robots), such as TODO. And as a major visual information processing method, fast visual model inference is important for the real-world robotic tasks to timely respond to environment changes. Unfortunately, the mobile robots often suffer slow visual model inference, because they are typically limited in computation power and suffer from limited and unstable wireless network bandwidth, making both local computation slow and naively offloading the computation to GPU servers slow.

To tackle this problem, we have the following two observations: 1. the robotic visual model mostly leverages local operators (i.e., computation results relies on local geometries); 2. the images for robotic visual model inference are typically continual. These imply that between visual model inference on consecutive images in visual robotic tasks, part of the previous computed results of local operators can be cached and reused, providing opportunity to both reduce local computation time and reduce transmission data volume, accelerating the overall visual model inference.

Based on these observations, we propose CacheInf, a collaborative edge-cloud cache system for efficient robotic visual model inference. Given a continuous stream of visual input in a robotic visual task, CacheInf analyses the overlapping area between consecutive inputs; based on the portion of overlapping area (reusable cache) and the current estimated wireless network bandwidth, CacheInf schedules on the action between reusing local cache to reduce local computation time and reusing the remote cache (e.g., the cache at the GPU server side from the robot's perspective) to reduce transmission time, to ultimately reduce the overall visual model inference time.

The design of CacheInf is non-trivial. The first challenge is to transform cached results to local computation acceleration. While the cached result of overlapping areas can be reused and the corresponding computation can be skipped, the sparse and fragmented remaining area can not be computed on local operators which are typically highly optimized for dense local geometries and outperform their counterparts for sparse local geometries (e.g., TODO). To make use of these highly optimized local operators, in CacheInf we search for a minimal set of rectangles that covers the uncached areas, split and combine them to a new rectangle while preserving local geometries and use it as a temporary input for the local operators whose computed result can be broken up and combined with the cache to form the correct result. We also profile the cost of this process and would choose to ignore cache when the cached portion is too small to bring acceleration.

The second challenge is how to reduce the cache memory consumption, especially at the robot side which typically has a tight GPU memory budget. In visual models, the computation results of local operators typically consumes significantly more memory that the parameters of local operators and naively caching all the computation result can lead to XX times memory consumption in our evaluation.

We observe that in visual models, the computation result of a local operator often serves as the input of another local operator where local geometries are kept. Thus in the cached scenario, the computation result of the combined rectangle of uncached areas can be directly passed to the following local operators without loss of information, and cache of the previous computation result between the two operators can be abandoned to reduce memory consumption. We greedily search for a set of continual local operators whose starting and ending operators incur the least memory consumption when caching their computation results, so as to minimize cache memory consumption.

The third challenge is under various distribution scenarios of cache (e.g., all cache is located at local or the remote GPU server), how to fully utilize the cache for acceleration. For example, when all cache is located at local due to previous limited wireless network bandwidth and we currently have a suitable wireless network bandwidth to offload computation to the remote GPU server, directly offloading all of the input means abandoning all local cache, damaging the potential gain of cache.

To tackle this problem, we integrate a recent offloading paradigm named Intra-DP (TODO: cite arxiv name): during visual model inference on an image, Intra-DP enables splitting of the input of local operators at the dimension of columns and assign different splits to the local robot and the remote GPU server for computation, so that local computation and data transmission of one image can be parallelized. Under this paradigm, we can partially leverage the local cache to compute on a split of the input which leads to faster local computation and also reduced transmission data volume, to better utilize the existing cache. We extend the scheduler of Intra-DP to further consider the potential

acceleration with cache and combine the heuristics about the previous two challenges into a new scheduling algorithm named XXX.

We implemented CacheInf using python and pytorch on Ubuntu20.04. To efficiently combine uncached areas into a new rectangle and recover this rectangle to correct result, we implemented these two operations as cuda kernel using taichi for faster parallel computation on GPU. Our baselines include plain local computation, a state-of-the-art computation offloading system named TODO and our self-modified counterpart with cache enabled and Intra-DP. Our datasets include the standard video datasets of TODO from torchvision. We evaluated CacheInf on a four-wheel robot equipped with a Jetson NX Xavier that is capable of computing locally with the low power consumption GPU and a PC equipped with an Nvidia 2080ti GPU as the remote GPU server. Extensive evaluation on various visual models and wireless network bandwidth circumstances shows that:

- CacheInf is fast. Among the baselines, CacheInf reduced the end-to-end inference time by XX% to XX%.
- CacheInf saves energy. Among the baselines, CacheInf reduced the average energy consumed to complete inference on one image by XX% to XX%.
- CacheInf is also memory-efficient. The above advantages were obtained by only incurring XX% to XX% increase in memory consumption for CacheInf.

The major contribution of this paper is our new paradigm of using cached computation results to accelerate visual modified inference on a continuous stream of input images on robots and the corresponding scheduling algorithm TODO. The resulting system, CacheInf, accelerates both local computation on the robot and the data transmission while offloading the computation to the remote GPU server by collaboratively considering and reusing cached computation results on both the robot and the server. The accelerated visual model inference and the reduced power consumption will make real-world robots more performant on various robotic tasks and nurture more visual models to be deployed in real-world robots. The source code and evaluation logs of CacheInf is available at TODO.

The rest of the paper is organized as follows TODO.

2 Background

3 System Overview

The chapter presents an overview of the design of CacheInf.

4 Implementation

5 Evaluation

6 Conclusion

7 Conclusions

References