EdgeCam: A Distributed Camera Operating System for Inference Scheduling and Continuous Learning

Yuqi Dong, Guanyu Gao
Nanjing University of Science and Technology
Nanjing, China
Email: dongyuqi@njust.edu.cn, gygao@njust.edu.cn

***Abstract*—Deep Neural Networks (DNNs) are commonly used in camera systems for video surveillance. However, the computational demands of DNN inference pose challenges for on-edge video analytics due to potential delay. Additionally, edge cameras typically employ lightweight models, which are susceptible to data drift. In this demo, we present EdgeCam, an open-source distributed camera operating system that incorporates inference scheduling and continuous learning for video analytics. EdgeCam comprises multiple edge nodes and the cloud, enabling collaborative video analytics. Edge nodes also collect drift data to support continuous learning and maintain recognition accuracy. We have implemented essential functionalities and algorithms, ensuring modularity and ease of configuration. The source code of EdgeCam is at https://github.com/MSNLAB/EdgeCam.**

***Index Terms*—Video analytics, edge/cloud computing, machine learning inference, computer vision applications**

I. Introduction

Deep neural network (DNN) models are popular for computer vision tasks due to high accuracy, making them the first choice for video analytics. Deploying DNN models in camera systems faces challenges, including significant bandwidth consumption when transmitting raw video content to the cloud. To address this, one approach is deploying models on edge nodes near the camera, reducing bandwidth consumption but potentially leading to long inference delay due to limited computing power. Moreover, DNN models on edge nodes have limited recognition capabilities and are susceptible to data drift in dynamic environments, resulting in a decrease in recognition accuracy over time. Furthermore, edge cameras have limited processing capabilities and uneven workloads, which can benefit from collaboration between edge-to-edge and edge-to-cloud to enhance overall performance.

The open-source community for video analytics is developing rapidly in recent years. SmartEye [1], [2] provided edge-cloud collaboration platforms for camera video systems, making full use of both the edge and cloud resources. However, these platforms do not consider the impact of data drift. Hence, Ekya [3] and RECL [4] integrated continuous learning for on-edge camera systems to mitigate the adverse effects of data drift. These projects, however, did not consider multi-edge cooperation in distributed camera systems.

In this demo, we introduce EdgeCam, an open-source distributed camera operating system. EdgeCam focuses on inference scheduling, and continuous learning for video analytics. Each edge node filters redundant video frames, collaboratively distributing frames based on workloads to improve performance. EdgeCam also addresses data drift by periodically retraining the model, and adapting to evolving data patterns.

Corresponding author: Guanyu Gao

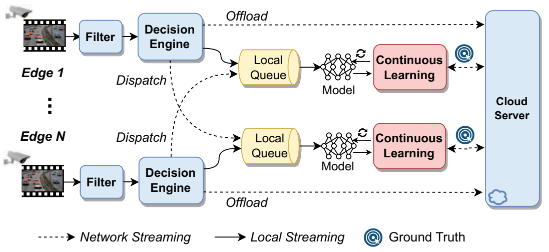


Fig. 1. The system architecture of EdgeCam.

II. System Design

Inference Scheduling. As depicted in Fig. 1, our system architecture, EdgeCam, consists of multiple edge nodes and one cloud server. The filter determines whether to discard video frames from edge cameras based on content redundancy, thus saving downstream resource costs. The decision engine can intelligently select offloading strategies and adjust resolution and encoding quality as needed. The video frame is then processed through one of three video analytics pipelines. 1) *On-edge (Standalone):*Process video frames locally and offload uncertain regions to the cloud. 2) *Edge-to-edge:*Send frames to less loaded edge nodes and process them as in Case 1. 3) *Edge-to-cloud*: Directly offload frames to the cloud, adjust resolution for lower bandwidth costs.

Continuous Learning. EdgeCam implements continuous learning on edge nodes, periodically identifying and sending low-confidence frames to the cloud for ground truth data. These data are used for retraining the edge models, enhancing accuracy to address the data drift issue.

III. System Implementation

We provide a detailed description of the system's modules and their interactions, as illustrated in Fig. 2.

A. Edge Node

Video Reader: This module supports various input sources by encapsulating the OpenCV interface.

979-8-3503-7025-6/24/$31.00 ©2024 IEEE
DOI 10.1109/IoTDI61053.2024.00028

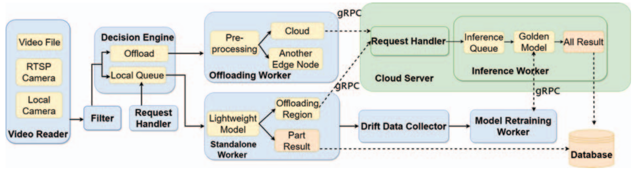


Fig. 2. The implementation of the system modules.

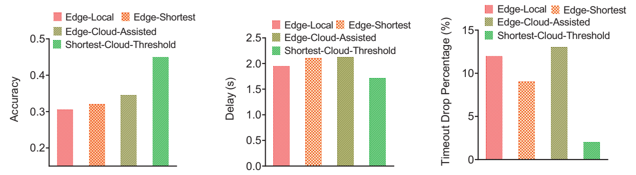
Filter: This module filters redundant video frames by calculating feature differences between neighboring video frames, including pixel, edge, area, and corner differences.
Decision Engine: This module encapsulates various offloading and video processing strategies to achieve workload balance between edge nodes and the cloud server. 1) *Edge-Local:* The edge node independently handles the received requests. 2) *Edge-Shortest:* Dispatching frames to the edge node with the shortest inference queue. 3) *Shortest-Cloud-Threshold:* Offload frames to the cloud if all edge queues exceed a threshold; otherwise, dispatch to the edge node with the shortest queue. 4) *Edge-Cloud-Assisted:* Initial inference occurs at the edge using a lightweight DNN, and low-confidence areas are sent to the cloud for processing.
Standalone Worker: This module, implemented by a thread, extracts tasks from the local queue, performs inference using the local DNN model, and offloads low-confidence regions based on the decision engine.
Offloading Worker: This module utilizes a thread pool to allocate offloading tasks to threads and dispatches them to edge servers or the cloud through gRPC.
Drift Data Collector: This module periodically selects low-confidence video frames from local inference for retraining.
Model Retraining Worker: This module, run by a single thread, sends selected frames to the cloud for inference using the golden model to obtain ground truth data, assisting in retraining models on edge nodes.

B. Cloud

Request Handler: It is a thread that listens to requests from edge nodes and places them into the cloud's inference queue.
Inference Worker: This module is implemented with a thread that extracts tasks from the inference queue and performs inference using the golden DNN model.

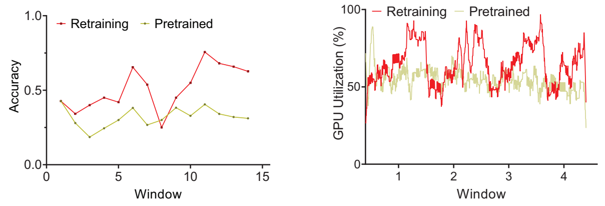
IV. Performance Evaluation

Testbed. The edge layer has two NVIDIA Jetson TX2 devices, and the cloud runs on Tencent Cloud's GN7.2XLARGE32, separated by 80 kilometers. The system performs object detection on traffic surveillance videos. The recognition results of the golden DNN model on the cloud are used as ground truth to evaluate system performance. If a video frame is discarded or exceeds the queue timeout, we reuse the recognition results from the last frame. The default settings: a 4s queue timeout, a queue threshold of 5 for offloading to the cloud, and a 240s retraining window size.
Evaluation on Different Methods. In this experiment, edge nodes compute Edge feature differences. Edge node 1 has a 0.015 difference threshold with a light workload, while edge node 2 has a 0.005 difference threshold with a heavy workload. We compare the performance of *Edge-Local*, *Edge-Shortest*, *Edge-Cloud-Assisted* and *Shortest-Cloud-Threshold* in Fig. 3. *Edge-Shortest* surpasses *Edge-Local* in accuracy and timeout percentage due to workload balancing, despite the additional transmission delay. *Edge-Cloud-Assisted* excels in accuracy but results in higher delay and dropped frames due to transmission delay and server potential overload. Moreover, *Shortest-Cloud-Threshold* achieves optimal performance and reduces edge workload by efficiently distributing tasks between edge nodes and the cloud.



(a) Average accuracy. (b) Average overall delay. (c) Timeout percentage.

Fig. 3. The comparison of the average accuracy, overall delay, and timeout drop percentage under different methods.



(a) Average inference accuracy. (b) GPU utilization.

Fig. 4. The comparison of the average accuracy and GPU utilization under model retraining.

Evaluation on model retraining. We compared the system's performance between model retraining and using pretrained models in Fig. 4. Model retraining improves average accuracy and GPU utilization. Continuous retraining enhances adaptability to dynamic environments, increasing accuracy but consuming more GPU resources.

V. Conclusion

This paper presented the design and implementation of EdgeCam and illustrated some preliminary results evaluated in a real-world testbed. For more details, please visit GitHub at https://github.com/MSNLAB/EdgeCam.

References

[1] X. Wang and G. Gao, "Smarteye: An open source framework for real-time video analytics with edge-cloud collaboration," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 3767–3770.
[2] X. Wang, G. Gao, X. Wu, Y. Lyu, and W. Wu, "Dynamic dnn model selection and inference off loading for video analytics with edge-cloud collaboration," in *ACM NOSSDAV*, 2022, pp. 64–70.
[3] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *NSDI 22*, 2022, pp. 119–135.
[4] M. Khani, G. Ananthanarayanan, K. Hsieh, J. Jiang, R. Netravali, Y. Shu, M. Alizadeh, and V. Bahl, "Recl: Responsive resource-efficient continuous learning for video analytics," in *NSDI 23*, 2023, pp. 917–932.