Traffic Sign Recognition IoT-based Application

Narges Mehran, Dragi Kimovski, Zahra Najafabadi Samani, Radu Prodan

Institute of information technology, Alpen-Adria-Universitaet Klagenfurt (AAU), Klagenfurt, Austria

email: firstname@itec.aau.at

1. Introducing the IoT application

International data corporation predicts that 21.5[1] billion connected Internet of Things (IoT) devices will generate 55% of all data by 2025 [1]. Nowadays, camera sensors can be embedded in most devices. Therefore, we designed an application to receive a video stream from a camera sensor and perform a video processing function. In the beginning, our designed application pre-processes the sensed data by two high-quality video encoding and framing frameworks. Afterward, we apply the machine learning (ML) model based on the low and high training accuracies. Because the user devices cannot often perform high-load machine learning training operations, we consider the ML inference operation acting as a lightweight trained ML model.

This application follows the microservice-oriented architecture that helps to isolate the management and maintenance of the components [2]. In addition, following each microservice's independent development kit, we aim at designing an application that utilizes the Docker container tool[2]. With the help of this tool, every microservice of the application can be separately containerized, and run on any platform executing the Docker engine v.19 or later.

2. Designing the IoT application

Following road safety inspection concerns, we designed a representative traffic management application. This application consisting of eight microservices, is represented in Figure 1. Every microservice communicates with its upstream microservice through a message queuing service [3]. The application receives a raw video, encodes it with a video CoDec in high resolution, divides it into its frames, and detects the sign inside the video based on the applied ML model. We describe the microservices and other components of the application as follows:

a) Encoding: of the raw video in multiple bitrate and resolution pairs using FFmpeg[3] with the H.264 video codec;

[1] https://intel.ly/3HcXnny
[2] https://docs.docker.com/engine/install/ubuntu/
[3] https://ffmpeg.org/ffmpeg.html#Description

b) Framing: using OpenCV to produce still images from video sources for recognition of traffic road signs [4];

c) Training: a multi-class ML model using 50000 still frames[4] extracted from the input video, for classification of 42 traffic signs with an accuracy of 90% [5];

d) Inference: with the trained model for traffic sign recognition [6];

e) Package and delivery: of detected sign in a required format (in our case it is the recognized sign image);

f) Dataset storage: of the records of the road sign for inventory purposes[5];

g) Consumer: receiving a warning related to the traffic sign on the roadside. The consumer can be a car driver or a passenger inside the vehicle in this case.

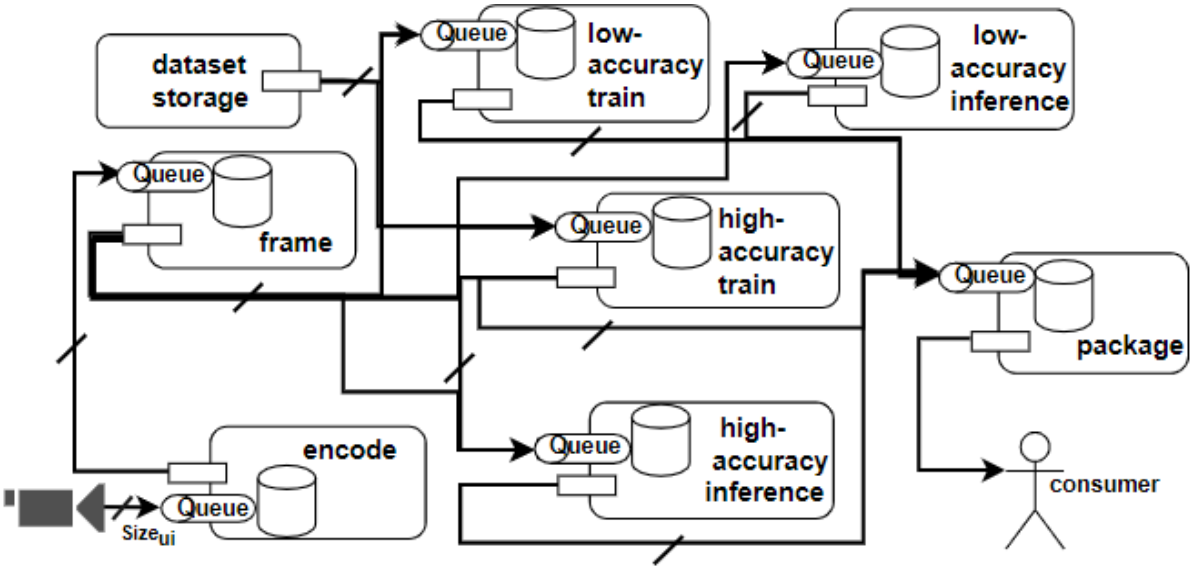

Figure 1: Traffic sign recognition application

Moreover, the application transmits the video stream or image files between the microservices through asynchronous message queue platforms such as *1)* a Kubernetes-based KubeMQ [2] for the local Edge cluster, and *2)* ZeroMQ[6]. These message queuing platforms help to temporarily store the data sent between the application components, concurrently process the data by the microservices, and increase the execution of the whole application.

[4] https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
[5] https://www.itec.aau.at/~narges/data.zip
[6] https://zeromq.org/languages/python/

3. Providing the Docker images of microservices

The Docker images of the microservices are available at public docker hub repository. The images appropriate for the machines with x86_64 architecture are as follows:

- Docker image of the encoding microservice: sina88/encode_20000:amd64
- Docker image of the framing microservice: sina88/frame_20000:amd64
- Docker image of the ML high-accuracy training microservice: sina88/hightrain:amd64
- Docker image of the ML inference microservice: sina88/inference:amd64

Regarding the ARM-based single-board devices such as raspberry pi version 4, we provided the Docker images proper for such type of architectures:

- Docker image of the encoding microservice: sina88/ubuntu-encoding:rpi4
- Docker image of the framing microservice: sina88/ubuntu-framing:rpi4
- Docker image of the ML high-accuracy training microservice: sina88/lite-training:rpi4
- Docker image of the ML inference microservice: sina88/inference:rpi4

4. Providing the public repository to the source code

The first version of this IoT application has already been published in the IEEE/ACM 21st international symposium on Cluster, Cloud, and Internet Computing (CCGrid) [7].
The new version along with its source code, can be found in our Github repository[7] for your considerations.

5. References

[1] D. Kimovski, N. Mehran, C. E. Kerth and R. Prodan, "Mobility-Aware IoT Applications Placement in the Cloud Edge Continuum," in IEEE Transactions on Services Computing, 2021, doi: 10.1109/TSC.2021.3094322.
[2] A. Balalaie, A. Heydarnoori and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," in IEEE Software, vol. 33, no. 3, pp. 42-52, May-June 2016, doi: 10.1109/MS.2016.64.
[3] N. Nikolov, et al. “Conceptualization and scalable execution of big data workflows using domain-specific languages and software containers,” Internet of Things, page 100440, 2021.
[4] Framing a video. https://gist.github.com/SiNa88/c85d8cfac641918c6de8b4f31d8cdc22 [Online; accessed March 2022].
[5] Šegvić, Siniša, et al. “Exploiting temporal and spatial constraints in traffic sign detection from a moving vehicle,” Machine vision and applications, 25(3):649-665, 2014.

[7] https://github.com/SiNa88/Traffic-sign-recognition-application

[6] P. Liu, B. Qi, and S. Banerjee. “Edgeeye: An edge service framework for real-time intelligent video analytics,” In Proceedings of the 1st international workshop on edge systems, analytics and networking, pages 1-6, 2018.

[7] N. Mehran, D. Kimovski and R. Prodan, "A Two-Sided Matching Model for Data Stream Processing in the Cloud – Fog Continuum," 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pp. 514-524, 2021.