Accepted for publication. DOI: 10.1109/IRASET52964.2022.9738158
© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

# Artificial Intelligence for Smart Cities: Comparing Latency in Edge and Cloud Computing

Ben Naets[1] , Willem Raes[2] , Rembrandt Devillé[3] ,
Catherine Middag[3] , Nobby Stevens[2] , and Ben Minnaert[1]
[1] Odisee University College of Applied Sciences, Ghent, Belgium
[2] DRAMCO Research Group, ESAT, Department of Electrical Engineering, KU Leuven, Ghent, Belgium
[3] Knowledge centre AI, Erasmus Brussels University college, Brussels, Belgium

***Abstract* — A smart city collects and uses data to streamline and improve multiple facets of city life. This approach is becoming an important strategy to keep cities a desirable place to live in while keeping them attractive and beneficial for business. The applications which constitute a smart city range from traffic to waste management, but are all dependent on data acquisition and data processing. This work focuses on the latency difference between edge and cloud computing for smart cities, which is illustrated by a real-life example: a dash cam in a car to monitor traffic in real time. Two scenarios are compared: the first one uses a single board computer on the edge to process the data and to realise inference, whereas in the second scenario, the analysis is done in the cloud, but the result is still returned to the edge. The results show the benefits and disadvantages of edge and cloud computing for a smart city environment for which latency is an important parameter, in particular when time-sensitive applications are considered such as on-board capturing of traffic situations.**

***Keywords* — *Smart cities, artificial intelligence, cloud, cloud Computing, Jetson Nano, edge computing, TensorRT***

## I. Introduction

Internet of Things sensor networks collect and analyse a lot of local data, and warn the user in time if action needs to be taken. A crucial aspect in this process is the location of the data analysis. One approach that is often used is to send the raw data in its entirety to a back-end server for analysis and processing with artificial intelligence (AI) (Fig. 1a).

In order to be able to closely monitor the process and warn the user in time, this transmission has to occur very frequently. There are three major drawbacks to this approach.

- First, the frequent remote communication results into a reduced lifespan for the often battery-powered nodes.
- Secondly, by sending raw data over long distances, additional delay (latency) is introduced.
- A third disadvantage concerns the high load on the communication network when the raw data is forwarded.

This work is supported by Flanders Innovation & Entrepreneurship VLAIO (TETRA-project HBC.2019.2638 “AISiBoCo: Artificial Intelligence on Single Board Computers”).
Corresponding author: B.N., email: ben.naets@odisee.be

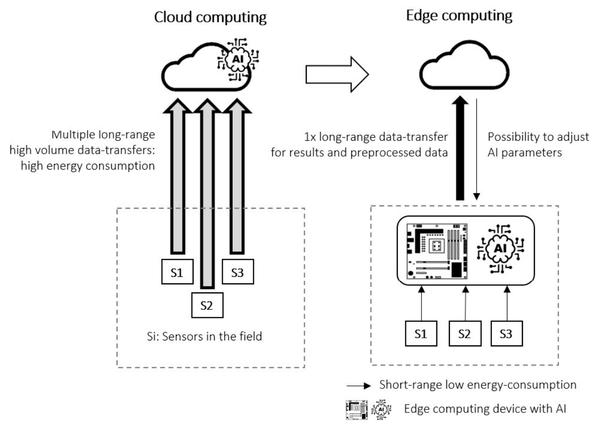
Cloud computing
Edge computing
AI
Multiple long-range high volume data-transfers: high energy consumption
1x long-range data-transfer for results and preprocessed data
Possibility to adjust AI parameters
S1
S2
S3
Si: Sensors in the field
Short-range low energy-consumption
Edge computing device with AI

Figure 1. (a) Cloud computing: sensors send all their data frequently to the cloud for processing. (b) Edge computing: the sensor data is locally aggregated and analysed by embedded algorithms; only part of the (preprocessed) data is sent to the cloud.

Because of these reasons and taken into account the increasing performance of single board computers, the analysis of data at the edge of the network (“edge computing”) using embedded AI is becoming increasingly important [1]–[3]. In this configuration, the data is processed locally, at the device itself (Fig. 1b).

This work compares the latency difference between two scenarios (cloud versus edge computing) in order to give insights in the benefits and disadvantages of edge and cloud computing which can be used when designing time critical applications such as traffic monitoring in a smart cities environment.

## II. Methodology

The experiments evaluate the performance of edge computing and cloud computing in terms of latency for an image segmentation workload. To achieve this, a camera recording of a trajectory in an urban environment is made, capturing multiple real traffic situations (Fig. 2). The recording is then used as input for the latency evaluation

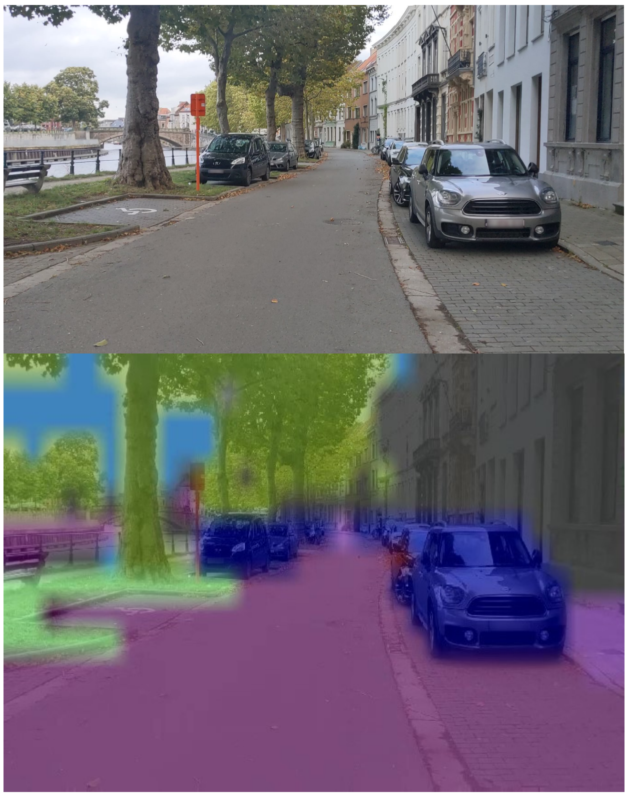
Figure 2. Raw video image and corresponding segmentation image.

experiments. In case of edge computing, the locally stored video file is used as the data source to perform real time image segmentation. In case of cloud computing, the node uses an Real Time Streaming Protocol (RTSP) video stream as input for the real time image segmentation.

## A. Segmentation model

Although the cloud environment has the ability to run more resource intensive models which achieve better results in regards to performance metrics such as intersection over union [4], this study uses the same semantic segmentation model [5] on both the edge and the cloud infrastructure since the main goal is to evaluate the latency disparity between the two configurations.

The segmentation model used for our experiments is a fully convolutional network with a ResNet18 backbone capable of per-pixel classification [6] [7]. This network is pretrained on the Cityscapes [8] dataset, which consists of pixel-level labeled images captured from video footage collected in over 50 German cities and is often used as a benchmark dataset to understand complex urban street scenes [4]. Two variants of the segmentation model are evaluated: a low and high input resolution of 512x256 and 1024x512, respectively. For both edge and cloud computing nodes, the model is optimised using the TensorRT framework for the specific target hardware. After the optimisation procedure, the model can start real time image segmentation inference.

## B. Edge computing setup

A Jetson Nano developer kit [9] is used as the edge computer and the semantic segmentation network is optimized for local computing using NVIDIA's TensorRT [10] to achieve low latency and high throughput. The TensorRT optimisation process reduces the precision of the network's weights while preserving its accuracy and optimises the model for the target GPU platform by kernel auto-tuning (Fig. 3).

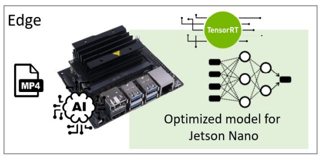

Figure 3. Overview edge computing configuration.

## C. Cloud computing setup

The cloud computing server runs on a Jetson TX2 [11] which is connected over TCP/IP with the edge device (Jetson Nano) to stream the video to the cloud. The semantic segmentation network is identical to the one running on the edge but is optimised with TensorRT for the different GPU (Fig. 4).

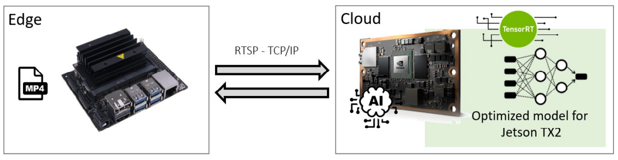

Figure 4. Overview cloud computing configuration.

## D. Network

Throughout our experiments, the best case scenario regarding network connectivity is assumed, i.e., a reliable wired connection with a download and upload speed of 90 Mbps. Although this does not correspond to a current real life dash cam application, this test configuration is applied as a benchmark to compare future tests with mobile network connectivity and to allow for a valid comparison of the latency difference between cloud and edge computing. Indeed, using a mobile network would introduce extra uncertainties depending on the coverage and network type (3G, 4G, 5G).

## E. Evaluation metric

For the edge device, the latency is defined as the time needed to load an image and compute the segmentation result using the machine learning model. For the cloud

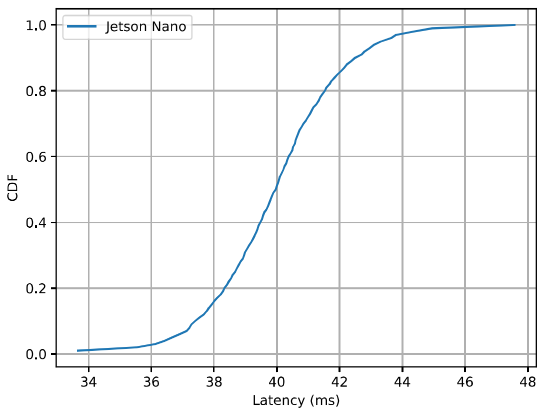

Figure 5. Cumulative Distribution Function (CDF) of the edge computing latency with a model resolution of 512x256.

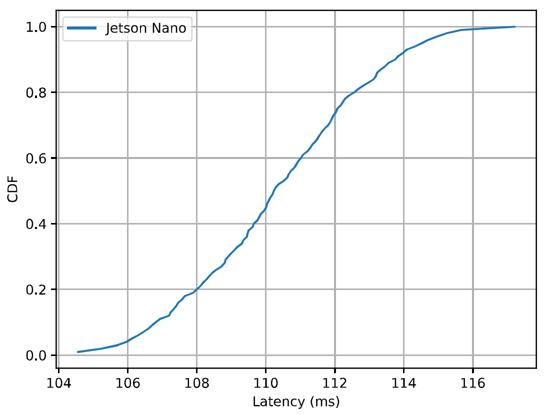

Figure 6. Cumulative Distribution Function (CDF) of the edge computing latency with a model resolution of 1024x512.

node, both the total round trip time and the time needed for computing the result are reported. Note that during the experiments, the resulting segmented images are not rendered, i.e., the rendering time is not included into the latency results.

## III. Results And Discussion

Table I and Table II summarise the results for the edge and cloud configuration, respectively.

Regarding the *edge computing* configuration, a percentile P50 latency of 40 ms and P95 latency of 43 ms per frame to process is observed for the Jetson Nano device in the case of the lower resolution model. This shows that when the data is computed locally, a very low spread on the update rate can be obtained. The latency values for the Jetson Nano increase to a P50 and P95 of 110 ms and 115 ms, respectively, when applying the larger resolution model. Due to the larger computational complexity, the latency is higher but one can again observe that the spread on update rate is very low, resulting in a deterministic inferencing pipeline. The cumulative distribution functions for the edge configuration for both resolutions are depicted in Figure 5 and Figure 6.

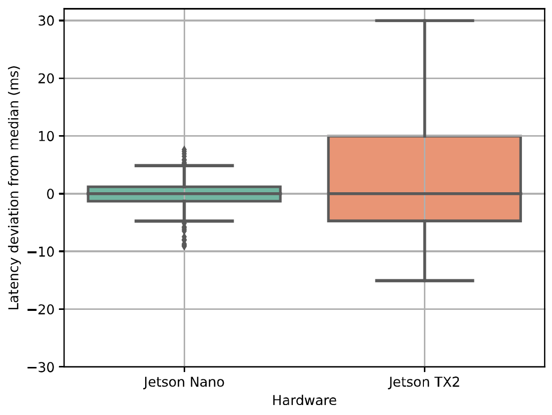

Figure 7. Comparison of the latency deviation for the 512x256 resolution between the edge (Jetson Nano) and cloud (Jetson TX2) configuration to their respective median values.

In the *cloud computing* configuration, a total P50 and P95 latency of 175 ms and 190 ms, respectively, is obtained at the Jetson TX2 node using the lower resolution input. Note that the total latency is composed of the computing and transit time. The required computing time per frame has a P50 value of 18 ms and a P95 value of 19 ms. The time in transit equals 158 ms and 173 ms for P50 and P95, respectively, and accounts for more than 90 % of the total time. From these results one can see that the transit time due to the remote data source has a significant impact on the latency and determinism of the inferencing pipeline. The median of the computing time per frame is significantly lower due to the more advanced hardware of the Jetson TX2, but the required transmission over the network and transit time introduce significantly more jitter in the update rate. This is demonstrated in more detail in Figure 7 which compares the latency deviation from both computing nodes to their respective median values.

For the higher resolution model, the P50 and P95 total latency values on the Jetson TX2 are 175 ms and 185 ms, respectively. The median required computing time increases to 50 ms due to the increased computational complexity. The P50 and P95 times in transit equal 125 ms and 134 ms, respectively. Again, the time in transit represents a significant portion (more than 70 %) of the total latency per processed frame and introduces an extensive amount of jitter in the inferencing pipeline. The cumulative distribution functions for the cloud computing configuration for both resolutions are shown in Figure 8

Table I. EDGE COMPUTING RESULTS

| Resolution | Network | Edge computing | | | | | |
|---|---|---|---|---|---|---|---|
| | | Transit | | Jetson Nano | | Total | |
| | | **P50** | **P95** | **P50** | **P95** | **P50** | **P95** |
| 512x256 | fcn-resnet18-cityscapes-512x256 | / | / | 40 ms | 43 ms | 40 ms | 43 ms |
| 1024x512 | fcn-resnet18-cityscapes-1024x512 | / | / | 110 ms | 115 ms | 110 ms | 115 ms |

Table II. CLOUD COMPUTING RESULTS

| Resolution | Network | Cloud computing | | | | | |
|---|---|---|---|---|---|---|---|
| | | Transit | | Jetson TX2 | | Total | |
| | | **P50** | **P95** | **P50** | **P95** | **P50** | **P95** |
| 512x256 | fcn-resnet18-cityscapes-512x256 | 158 ms | 173 ms | 18 ms | 20 ms | 175 ms | 190 ms |
| 1024x512 | fcn-resnet18-cityscapes-1024x512 | 125 ms | 134 ms | 50 ms | 52 ms | 175 ms | 185 ms |

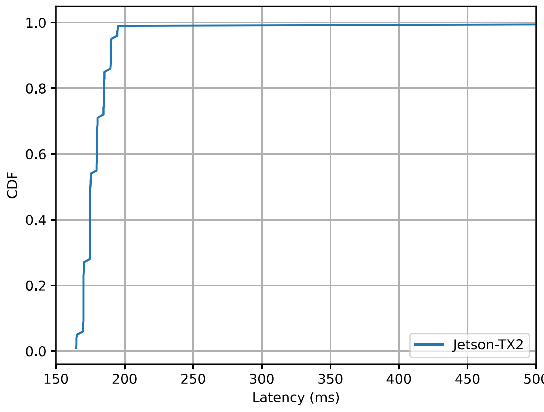

Figure 8. Cumulative Distribution Function (CDF) of the **total** cloud computing latency with a model resolution of 512x256.

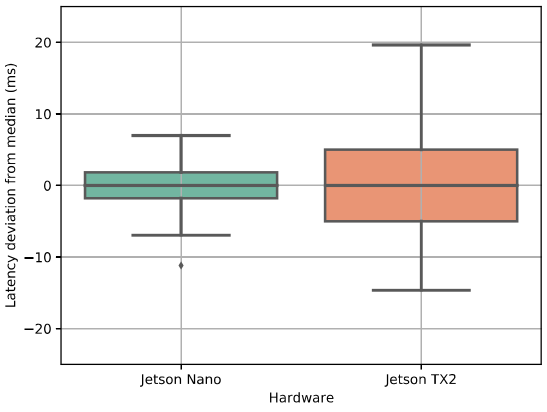

Figure 10. Comparison of the latency deviation for the 1024x512 resolution between the edge (Jetson Nano) and cloud (Jetson TX2) configuration to their respective median values.

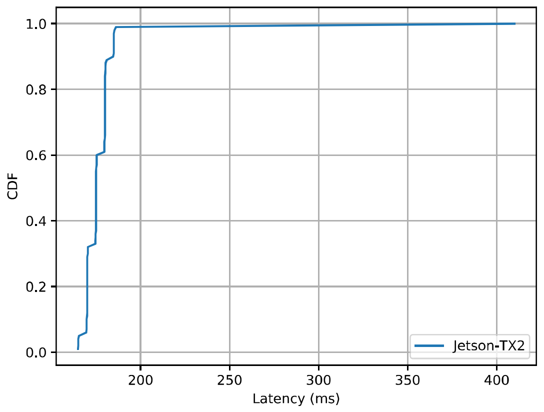

Figure 9. Cumulative Distribution Function (CDF) of the **total** cloud computing latency with a model resolution of 1024x512.

and Figure 9.

Finally, the latency deviation with respect to the median values of both hardware platforms for the larger model resolution is depicted in Figure 10.

## IV. CONCLUSION AND FUTURE WORK

An edge computing scenario was compared to a cloud computing scenario regarding latency for a smart cities application: image segmentation for a dash cam in a car to monitor real time traffic. It was experimentally demonstrated that for this practical scenario, the time for processing data is significantly lower on the edge than in the cloud configuration, due to the extra transit time. Moreover, a very low spread on the update rate can be obtained. Obviously, the compromise is the lower computing power available in the edge configuration.

The tests described in this work are intended as a benchmark to compare further experiments, in which extra uncertainties will be gradually introduced to determine the impact of each parameter. First, the wired internet connection for the cloud computing scenario will be replaced by a cellular one. Initially, the setup which streams the dash cam video to the cloud will be stationary to eliminate roaming and coverage issues of the cellular network. Afterwards the setup will be placed inside a moving car to examine the influences of network roaming and blind spots. In this way, the benefits and disadvantages of edge and cloud computing for a smart city environment can be numerically demonstrated for the on-board capturing of traffic situations.

## References

[1] L. Zhao, J. Wang, J. Liu, and N. Kato, "Optimal edge resource allocation in iot-based smart cities," *IEEE Network*, vol. 33, no. 2, pp. 30–35, 2019.

[2] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–43, 2017.

[3] L. U. Khan, I. Yaqoob, N. H. Tran, S. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 200–10 232, 2020.

[4] Cityscapes benchmark suite. [Online]. Available: https://www.cityscapes-dataset.com/benchmarks/#pixel-level-results

[5] Jetson inference: Deploying deep learning. [Online]. Available: https://github.com/dusty-nv/jetson-inference

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[9] Jetson nano developer kit. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[10] Nvidia tensorrt. [Online]. Available: https://developer.nvidia.com/tensorrt

[11] Jetson tx2. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2