Open Call 3

EdgeFlooding: Exploiting Edge computing for Real-Time Monitoring and Detection of Flash Floods

Deliverable 3: Experiment Results and Final Report

Authors	Carlo Vallati Francesca Righetti Giuseppe Anastasi Nicola Tonellotto Dpt. of Information Engineering - University of Pisa Largo Lucio Lazzarino 2 56125 Pisa
Due Date	15/04/2022
Submission Date	14/04/2022
Keywords	Smart City, Flood Monitoring, Image analysis, Edge Computing

Deliverable 3: Part I

Analysis, results, and wider impact

1 Abstract <Approx 100 words>

Flash flood monitoring systems for just-in-time notification of flooding events will be crucial to secure any city located in prone flood areas. For this reason, an experimental implementation of a flood monitoring system has been developed at the University of Maryland, Baltimore County (UMBC). The EdgeFlooding project aims at extending this system, which adopts a centralized cloud-based approach, to create a novel implementation that adopts a distributed approach based on edge/cloud computing. The extension will be carried out by the University of Pisa (UNIPI) with the support of UMBC. In order to assess the performance of such system, an extensive experimentation will be carried out by UNIPI using two Fed4Fire+ testbeds and one testbed available at UNIPI. The experimentation will be carried out in two phases: a first phase involving only the Grid'5000 testbed, a large-scale testbed for experiment-driven research in the area of parallel, distributed computing and cloud, and a second phase involving also external nodes from the Virtual Wall testbed and from the Fog testbed at UNIPI. The aim of those experiments is to assess whether a distributed edge/cloud computing approach is feasible, considering multiple image analysis algorithms

and different edge/cloud configurations. The analysis of the results will ultimately provide a set of guidelines for the implementation of future systems.

Differences with D2

The last sentence of the abstract has been modified in order to reflect the final experiments configuration, in particular the study of different image analysis algorithms.

2 Project Vision

The EdgeFlooding project aims at assessing whether a distributed computing architecture based on the edge/cloud computing paradigm can be adopted for the implementation of a flash flood monitoring system. Existing flood monitoring systems adopt a centralized cloud-based approach where monitoring stations are deployed in flood-prone areas and transmit their data (sensor data and images) to a cloud-based service for analysis. Recently, a novel computing paradigm has emerged to improve scalability and reliability of cloud systems [1]: the edge/fog computing. Edge/fog computing extends the centralized cloud computing architecture by introducing an intermediate computing layer in proximity of the physical systems. The goal of EdgeFlooding is to develop a prototype of a flood monitoring system based on the edge/cloud computing paradigm and measure its performance via a set of experiments. The aim is to highlight the advantages and the drawbacks of adopting such distributed approach to perform data analysis at the edge, in proximity of the monitoring sites.

The project will leverage an existing implementation of a cloud-based flood monitoring system developed at UMBC [1]. The system exploits the fusion of multiple data sources (e.g., social media data and smart camera images) to detect any potential flood events and assess their impact. A testbed prototype of the flooding monitoring system has been already developed and deployed in the Baltimore County area. The testbed allowed UMBC to collect a large amount of data to test and evaluate the efficacy of the prediction algorithm, which has been demonstrated to be effective.

EdgeFlooding aims at extending the current implementation to obtain a novel implementation of the system that adopts an edge/cloud approach, where a part of data analysis, namely the image analysis, is moved at the edge, in proximity of the monitoring sites. The extension of the original implementation, carried out by UNIPI with the support of UMBC, will restructure the architecture of the platform in order to organize its modules into microservices, thus allowing their deployment on different computing nodes. This modular architecture based on microservices will easily allow to carry out different experiment configurations: one edge/cloud configuration where the image analysis services are deployed on edge nodes while the social media and data aggregation services are hosted on the cloud, and one cloud configuration where all the services are deployed on the cloud as in the original implementation.

Co-funded by the Horizon 2020 Framework Programme of the European Union

This extended implementation will be exploited by UNIPI to run an extensive experimentation to evaluate its performance and assess the feasibility of adopting the edge/cloud approach in real deployments and measure its advantages/drawbacks. The experimentation will be carried out using two Fed4Fire+ testbeds and one testbed available at UNIPI and will be organized into the following two phases. A first phase in which a realistic deployment is emulated using the nodes of the Grid'5000 testbed[1], a large-scale testbed for experiment-driven research in the area of parallel, distributed computing and cloud managed by a scientific interest group (GIS) and hosted by Inria. Nodes from different locations of the testbed and with different computing capabilities will be exploited to emulate cloud computing nodes (top-range servers with powerful CPUs and GPUs) and edge computing nodes (mid-range servers with less powerful CPUs and mid-level GPUs). A second experimentation phase, instead, will aim at assessing the performance when the edge computing layer is implemented with constrained devices, e.g., a PC or an embedded system. The goal of this second phase is to assess a different configuration where the edge nodes are installed in close proximity of (or co-located with) the cameras in locations that are not suitable for the installation of servers. To this aim, nodes external to the Grid'5000 testbed will be employed, specifically the PCs available at the Virtual Wall testbed[2], a testbed hosted at and managed by imec IDLabt ilab.t in Ghent, and the embedded systems of the Fog testbed available at UNIPI.

The analysis of the results of the experiments, carried out jointly by UNIPI and UMBC, will aim at the following: (i) assess the feasibility of adopting the edge/cloud implementation in real systems and, in particular, verify the feasibility of implementing the image analysis algorithm on edge nodes; (ii) analyse different image analysis strategies; (iii) measure metrics such as analysed FPS, latency, bandwidth and CPU/RAM occupation to compare different edge/cloud configurations and highlight advantages/disadvantages of each one. Such results and conclusions are expected to be of interest not only for future flood monitoring systems but in general in the area of environmental monitoring.

Differences with D2

As in the abstract, the last period has been modified in order to reflect the final experiments configuration and the inclusion of different image analysis algorithms.

[1] Grid 5000 tested home page: https://www.grid5000.fr

[2] Virtual Wall testbed home page: https://doc.ilabt.imec.be/ilabt/virtualwall/

Co-funded by the Horizon 2020 Framework Programme of the European Union

3 Details on participants (both EU and US)

The experience and area of expertise of the US and EU teams are essentially complementary: the research team at UMBC has a proven track record of successful research on the field of artificial intelligence and machine learning for smart systems in general, while the research team at UNIPI has proven experience on distributed systems, IoT systems, edge/fog computing and, in general, on experimental evaluation exploiting real testbeds. UNIPI and UMBC have long experience in carrying out large research projects, as highlighted by the short biographies of the PIs and the team members, reported below.

Member	Role	Short Bio
Carlo Vallati (UNIPI)	He is the PI of the EdgeFlooding project and the PI of the EU team at UNIPI. He is responsible for the supervision of all the activities of the UNIPI team and for the coordination of the interaction with the UMBC team. He is in charge of all the dissemination activities carried out jointly by UNIPI and UMBC members. He is responsible for the development of the implementation plan (Task 0) and for the analysis of the experiment results (Task 3). He is involved in the adaptation of the flood monitoring system and the experimentation tasks (Tasks 1 and 2).	Dr. Vallati is currently Assistant Professor (tenured) at UNIPI's Department of Information Engineering. He received a Master's Degree (magna cum laude) and a PhD in Computer Systems Engineering in 2008 and 2012, respectively, from UNIPI. He is the director of the Cloud Computing, Big Data and Cybersecurity Crosslab funded by the Italian Ministry of Education, University and Research. He is co-author of +60 peer-reviewed papers in international journals and conferences. His research interests include IoT solutions and Cloud/Fog computing systems. He has been involved in many national and international projects and in several research projects supported by private industries. Website: http://www.iet.unipi.it/c.vallati/
Francesca Righetti (UNIPI)	She is responsible for the adaptation of the flood monitoring system for the edge/cloud computing approach (Task 1) and for the execution of the experiments	Francesca Righetti is a Postdoctoral Research Fellow at the Information Engineering Department at the University of Pisa. She received the Master's and Ph.D. degrees in Computer Engineering from the University of Pisa, Pisa, Italy, in

Co-funded by the Horizon 2020 Framework Programme of the European Union

	on the testbeds and for the collection of relevant metrics (Task 2). She is involved in the planning of the experiments and the data analysis (Task 0 and 3).	2017 and 2021, respectively. Her research interests include the Internet of Things (IoT), the Industrial Internet of Things (IIoT) and the Cloud-to-Thing Continuum (C2TC). She took part in national and international projects, including the SmarT INtelliGent RAilwaY (STINGRAY) project with the ISTI-CNR, Pisa, and the "ECOAP: Experimental assessment of congestion control strategies for the Constrained Application Protocol" project. Website: http://for.unipi.it/francesca_righetti/
Giuseppe Anastasi (UNIPI)	He is involved in the planning of the experiments (Task 0), in the supervision of the platform adaptation (Task 1) and the execution of the experiment (Task 2) and in the analysis of the results of the experiments and their interpretation (Task 3).	Giuseppe Anastasi is a Professor of computer engineering at the Department of Information Engineering (DII) of the University of Pisa, Italy. He is the director of the Industry 4.0 CrossLab, funded by the Italian Ministry of Education and Research (MIUR) in the framework of the "Departments of Excellence" program. It consists of six interdisciplinary and integrated research laboratories (CrossLabs) covering all the key areas of Industry 4.0. He has been involved in many national and international projects and in several research projects supported by private industries. Website: http://www.iet.unipi.it/g.anastasi/
Nicola Tonellotto (UNIPI)	He is involved in the extension of the flood monitoring platform (Task 1), in the execution of the experiments on the testbeds and for the collection of relevant metrics (Task 2) and in the analysis of the results of the experiments and their interpretation (Task 3).	Nicola Tonellotto is assistant professor of computer engineering at the Information Engineering Department of the University of Pisa, Italy. His research interests include cloud computing, distributed systems, and Web information retrieval. Website: https://tonellotto.github.io/
Nirmalya Roy (UMBC)	He is the PI of the US team at UMBC. He is responsible for the supervision of all the	Nirmalya Roy is currently an Associate Professor in the Information Systems department at University of Maryland

Co-funded by the Horizon 2020 Framework Programme of the European Union

	activities of the UMBC team and for the interaction with the team at UNIPI. He is involved in the extension of the flood monitoring platform (Task 1) by providing support to UNIPI. He is involved in the planning of the experiment as consultant (Task 0) and in the analysis of the results (Task 3).	Baltimore County. He directs the Mobile, Pervasive, and Sensor Computing Lab (MPSC) at the University of Maryland Baltimore County. He was a Clinical Assistant Professor in the School of Electrical Engineering and Computer Science at Washington State University from January 2012 to June 2013. Prior to that, he worked as a Research Scientist at Institute for Infocomm Research (I2R), Singapore from 2010 to 2011. He was as a postdoctoral fellow in Electrical and Computer Engineering Department at The University of Texas at Austin from 2008 to 2009. He received his Ph.D. and M.S. in Computer Science and Engineering from The University of Texas at Arlington in 2008 and 2004 respectively. He did his Bachelors in Computer Science and Engineering from Jadavpur University, India in 2001.
Aryya Gangopadhyay (UMBC)	He is involved in the analysis and interpretation of the results of the experiments carried out jointly with UNIPI (Task 3).	Aryya Gangopadhyay is a professor and the chair of UMBC's Information Systems Department. He has been a faculty member at UMBC since 1997. He has published five books and over 125 peer-reviewed research articles. Dr.Gangopadhyay's research interests are in the area of data science and machine learning, including machine learning-based solutions in areas such as cybersecurity, multi-modal data fusion for emergency response, and healthcare applications. His research has been funded by grants from NSF, NIST, US Department of Education, IBM, Maryland Department of Transportation, and other agencies. Website: https://sites.google.com/site/homearyya/

Co-funded by the Horizon 2020 Framework Programme of the European Union

Bipen Basnyat (UMBC)	As the developer of the initial flood monitoring platform, he is involved in the platform adaption (Task 1) to support UNIPI activities. He is also involved in the analysis of the results (Task 3).	Bipendra Basnyat, P.E., is a final-year Ph.D. student at the UMBC. His research is focused on the design and implementation of various elements of SmartCity components. He is the principal architect of the FloodBot System. After successfully integrating hardware designs and machine learning algorithms and deploying an end-to-end flood monitoring system, Bipen now ensures FloodBot's regular operation and management.

Differences with D2

Nicola Tonellotto role description in D2 did not include his involvement in Task 1 by mistake. In this revision of the document, the description of his involvement has been updated.

Co-funded by the Horizon 2020 Framework Programme of the European Union

4 Results

In this section the results obtained by the EdgeFlooding experiments will be presented and analysed. Before going into the details of the results, a summary of the methodology[3] to introduce the minimum set of the details required to fully grasp the results is provided in Section 4.1. In Section 4.2 a summary of the experiment scenarios is presented, while in Section 4.3 the metrics measured are outlined. In Section 4.4 the results are presented, while in Section 4.5 a final discussion and analysis is provided. The section concludes with Section 4.6 that reports some details on the technical issues experienced during the execution of the experiments that lead to the modification of the original experimentation plan and the request for one-month extension for the overall duration of the project.

4.1 Methodology

In order to run the EdgeFlooding experiments, a distributed flood monitoring platform has been implemented, exploiting an initial cloud-based prototype. The platform, released as open-source project[4], is designed using a micro-services architecture: independent modules interact each other using a message passing mechanism that allows each one to be deployed in a different host, without requiring changes in the code at runtime. This feature, in particular, allowed to seamlessly reconfigure the platform with no changes to deploy the cloud and the edge scenarios.

Shortly, the platform comprises the following components:

- *Camera module*. This module emulates a surveillance camera generating video frames at a fixed rate, 5 FPS. The frames generated by the module are extracted from the real videos to ensure the generation of realistic data.
- *Inference module*. This module implements the image analysis functionalities. The initial implementation was based on the Inception convolutional neural network version 3 [2] for the image analysis. The initial set of experiments, however, shown a high complexity of this model, thus highlighting the need to investigate also different models. To this aim, other two implementations were developed and testbed based on two other neural network models, Mobilenet [3] and YOLO version 5 [4], respectively. The inference module periodically receives video frames from one or more camera modules and analyses them exploiting one or more GPU accelerator(s) locally available on the node. At the end of the analysis the result of the inference is

[3] For a detailed introduction on the methodology adopted and the implementation of the platform the interested reader is referred to Deliverable 2: Report on Experiment Implementation and Execution

[4] EdgeFlooding platform code repository: https://github.com/EdgeFlooding/EdgeFloodingPlatform

Co-funded by the Horizon 2020 Framework Programme of the European Union

transmitted as a message to the aggregator module. The message includes the result of the inference and the list of objects detected on the frame with the corresponding inference probability. In order to support multiple camera sources submitting frames at the same time, a set of queues (one for each video source) is implemented on the inference module to store frames received from different sources. If all the GPUs are already processing frames, a frame is enqueued and waits for the completion of the current analysis. As soon as one GPU completes its analysis, a frame is removed from one queue and submitted to the GPU. A round robin policy is implemented to extract frames from the different queues to ensure fairness in the analysis of the frames from different cameras. Considering that the system aims at analyzing the frames in real time, the queue length of each camera is set to only one frame. This means that when a new frame is received, the older frame is discarded, if any, and the newer frame is enqueued. Different values of the queue size have been considered in our experiments, however, since we didn't notice a significant change in the obtained results, for the sake of brevity, we will show only the results obtained with the queue size set to one frame.

- *Aggregator module*. This module aggregates the results received by the inference modules and implements the social media monitoring functionalities. The module implements two distinct functionalities: a data aggregator that analyzes and stores the data received from the inference modules and a social media scraper that monitors Twitter for posts containing pre-defined hashtags.

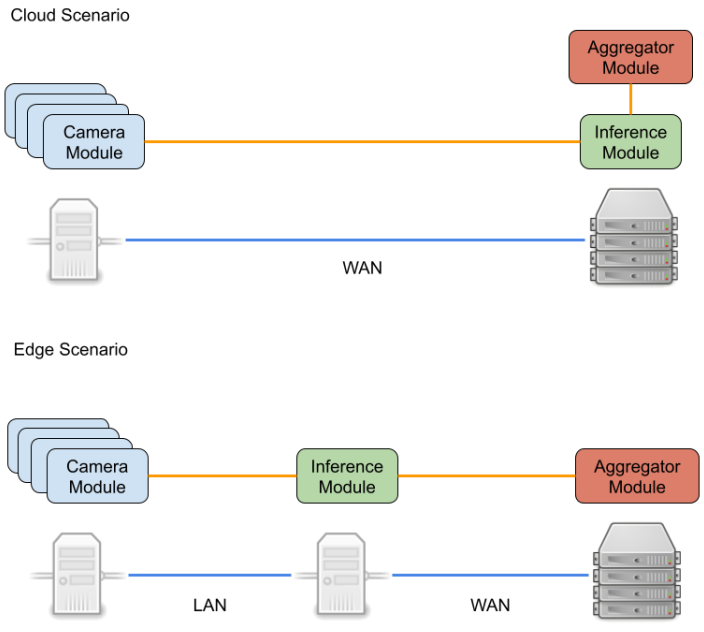

Figure 1. Platform module deployment

Co-funded by the Horizon 2020 Framework Programme of the European Union

EdgeFlooding aims at implementing two different scenarios, i.e., a cloud scenario and an edge scenario. They are implemented through two different configurations of the platform that differ each other on how the modules of the platform are deployed (see Figure 1):

- The *cloud scenario*, where image analysis, data aggregation and social media analysis are performed at the cloud layer. In this scenario the inference module and the aggregator module are deployed on the same node emulating the cloud layer.
- The *edge scenario*, where image analysis is performed at the edge layer, while data aggregation and social media analysis is performed at the cloud layer. In this case the inference module is deployed on nodes (one or more) emulating the edge layer, while the aggregator module is still deployed on the node emulating the cloud layer.

In both the scenarios, a third node is reserved to emulate the cameras. On this node, multiple instances of the camera module are instantiated to emulate different cameras generating different video streams.

4.2 Experiment scenarios

In order to assess the performance of the system under heterogeneous hardware configurations for both the cloud and edge scenarios, nodes from three different testbeds have been exploited in the experiments:

- The ***Grid'5000 testbed***, a large-scale testbed that includes multiple sites in France. The testbed is composed of powerful servers, each one equipped with one or more GPUs, both high and mid-range GPUs. This testbed is exploited to emulate the cloud infrastructure exploiting powerful servers with multiple CPU and large RAM availability with high-range GPUs. In addition, the testbed is used also to emulate an edge layer configuration, through servers with less CPU and RAM resources equipped with mid-range GPUs. This configuration emulates a resource-rich edge infrastructure.
- The ***Virtual Wall*** testbed, a single-site testbed located in Ghent, Belgium, that hosts computing nodes with less CPU and RAM capabilities than the servers of Grid'5000 and are equipped with entry-level GPUs. The testbed is employed to emulate a less powerful edge layer.
- The ***Fog*** testbed, a testbed located at the 'Cloud Computing, Big Data and Cybersecurity' laboratory at the University of Pisa, Italy, that includes embedded boards. Those boards are characterized by very limited CPU and RAM and integrates an embedded GPU with limited capabilities. Different boards are considered in our experiments, each one with different resources and cost.

In all the scenarios, the aggregator module is deployed on a powerful server on the cloud, while one or more inference modules are deployed on different nodes, according to the specific scenario considered. In each experiment, the camera module, required for the

Co-funded by the Horizon 2020 Framework Programme of the European Union

generation of the video streams, is deployed on the same testbed of the nodes where the inference modules are deployed to emulate spatial proximity.

In the following table (Table 1), the different scenarios considered in the experiments are summarized. For each scenario the node used for deploying the inference module is reported along with the resources available on each node. In each scenario, nodes have been selected mainly for the capabilities of the GPUs, which represent the bottleneck for the execution of the inference module.

Table 1. Experiment scenarios.

Scenario	Image analysis node	Resources available
Cloud – 1	Grid'5000 - Graffiti	CPU: 16 cores RAM 128GB GPU: 4 x Nvidia GeForce RTX 2080 Ti
Cloud – 2	Grid'5000 - Grouille	CPU: 64 cores RAM 128GB GPU: 2 x Nvidia A100
Near Edge	Grid'5000 - Chifflet	CPU: 28 cores RAM: 768GB GPU: 2 x Nvidia GTX 1080 Ti (only one GPU enabled)
Far Edge	Virtual Wall – Gpunode2	CPU: 6 cores RAM: 12GB GPU: 1 x Nvidia GTX 980
On-site Edge – 1	Fog – NVIDIA Jetson Xavier	CPU: 8 cores ARM RAM: 32GB GPU: 512-core Volta GPU
On-site Edge – 2	Fog – NVIDIA Jetson TX2	CPU: 4 cores ARM RAM: 8GB GPU: 256-core NVIDIA Pascal™ GPU
On-site Edge – 3	Fog – NVIDIA Jetson NANO	CPU: 4 cores ARM RAM: 4GB GPU: 128-core NVIDIA Maxwell GPU

Co-funded by the Horizon 2020 Framework Programme of the European Union

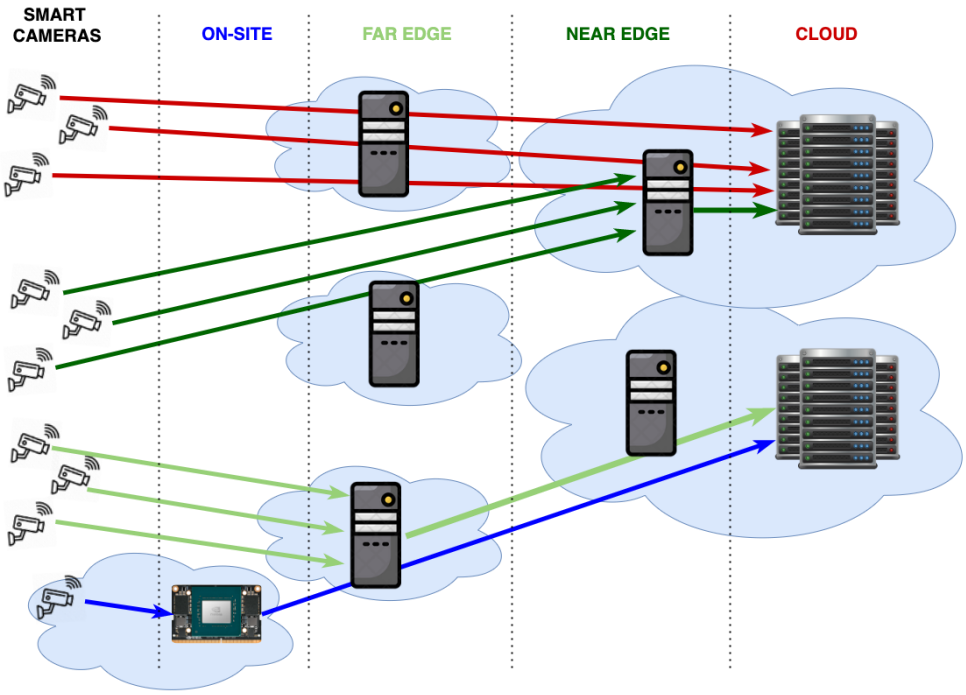

Figure 2. Experiment scenarios considered

As can be seen, four main scenarios are considered in the experiments (see Figure 2):

- A ***cloud scenario***, where the inference module is deployed on a powerful node, with multiple high-range GPUs. This scenario emulates the deployment of the module on the cloud layer and it is implemented with nodes from the Grid'5000 testbed
- A ***near edge*** scenario where the inference module is deployed on a server with a mid-range GPU. If multiple GPUs are available on the node, the module is programmed to exploit only one GPU, to further reduce the computing capability. This scenario emulates an edge layer deployed directly at the edge of the internet service provider network, e.g., through a 'datacenter in a box' deployment and it is implemented with nodes from the Grid'5000 testbed.
- A ***far edge*** scenario where the inference module is deployed on a computing node with a low-range GPU. This scenario emulates a *far edge* deployment implemented through a single server installed in a certain area in proximity of the cameras and it implemented with nodes from the Virtual Wall testbed.
- An ***on-site*** edge scenario where the inference module is deployed on an embedded system with limited resources and an integrated GPUs. This configuration emulates an on-site edge node to be installed directly in the same site of the camera and it is implemented with nodes from the Fog testbed at UNIPI.

For the cloud and on-site scenarios, different configurations have been considered, in order to assess the performance for a different number of GPUs available on the server in the cloud

Co-funded by the Horizon 2020 Framework Programme of the European Union

scenario or with embedded systems GPUs with different capabilities in the on-site edge scenarios.

For each experiment a varying number of video sources *C*, i.e., C=1, C=5, C=10, has been considered while the number of edge nodes *e* was fixed to one on the edge computing scenarios, i.e. *e*=1. A set of experiments with more than one edge node (i.e., *e*=3) to assess the scalability of the system when the number of edge nodes increases is run for a subset of the experiments.

Every scenario is repeated independently 3 times. In the following only the average results are reported, while 95th confidence intervals are removed as they are very small, due to a very low variability on the results between different replicas.

4.3 Metrics

The following table (Table 2) reports the metrics measured in the experiments and their precise definition.

Table 2. Performance metrics.

Metric Name	Unit	Definition
Analyzed FPS	Frames per Second	The average number of video frames analyzed per second by the overall system
FPS per camera	Frames per Second	The average number of video frames analyzed per second by the system per video source
Average processing time per camera	seconds	The average time between the analysis of two subsequent frames from the same video source
Data transmission overhead	Bytes	The overall amount of data transmitted from the cameras to the cloud (in the cloud scenarios) and from the cameras to the edge and from the edge to the cloud (in the edge scenarios)
GPU, CPU and RAM utilization	Percentage	The average GPU, CPU and RAM utilization percentage over a period for a node (both cloud and edge)
Data aggregation latency	milliseconds	The time between the triggering of a data aggregation process and its completion. The data aggregation process is a process that aggregates data from different video feeds and from the social media analysis process.
Frame analysis delay	milliseconds	The average time required for the analysis of a single frame on the inference module. The delay includes: *Inference time*, the time required for the analysis of the frame, and the

Co-funded by the Horizon 2020 Framework Programme of the European Union

		TX and Queue time, the time required for the transmission of the frame from the camera to the inference module and the queuing time before the analysis.

4.4 Experiment results

In this section, the results of the EdgeFlooding experiments are presented. As highlighted in Section 4.2, the initial implementation of the inference module was based on a cloud-based prototype provided by UMBC. This implementation exploited the Inception Network v3 model for image analysis. The experiments carried out with this initial implementation are presented in Section 4.4.1. Since this first set of results, partially introduced also in D2, confirmed the complexity of this model, a different lightweight implementation with different models was produced and assessed. The results obtained with this revised version are presented in Section 4.4.2.

4.4.1 Initial implementation analysis

In this section the overall performance of the system is analysed in its initial implementation where image analysis is performed on the inference module using Inception net, the model initially considered for the implementation of the system. The following table (Table 3) summarizes the scenarios for which the deployment of our system was successful:

Table 3. Initial implementation -Inception net deployment results: • successful, x failed

Cloud 1	Cloud 2	Near Edge	Far Edge	On Site Edge 1	On Site Edge 2	On Site Edge 3
•	•	•	•	x	x	x

As can be seen, the deployment was successful on all the scenarios except for the three on-site edge configurations. In those configurations, the deployment failed due to the reduced amount of RAM available on the embedded systems considered in our experiments, which was not sufficient for the execution of the initial implementation of the inference module.

4.4.1.1 Single edge scenario

Initially, the cloud scenarios versus the near edge (that exploit nodes from the Grid'5000 testbed) and the far edge scenarios (that exploits nodes from the Virtual Wall testbed) with only one edge node, i.e., e=1, are considered. Scenarios with more than one edge nodes will be presented afterwards.

Co-funded by the Horizon 2020 Framework Programme of the European Union

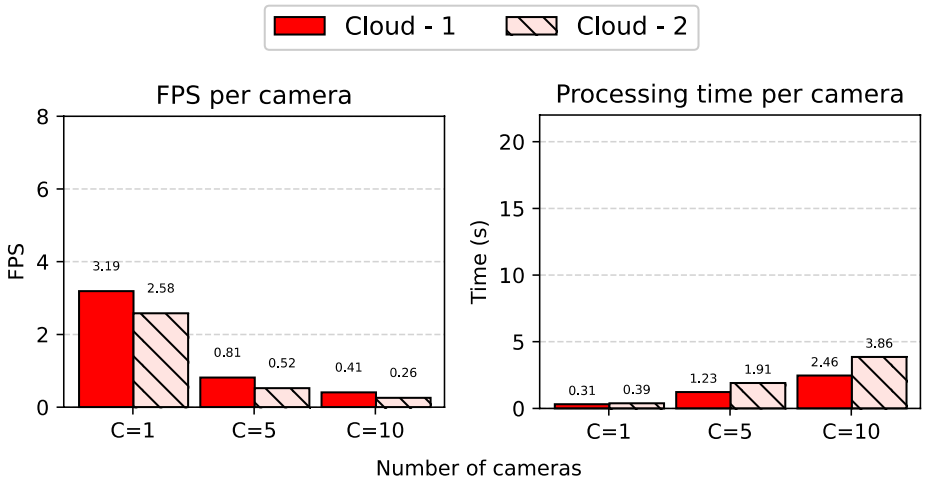

Figure 3. Image analysis performance per camera Cloud scenarios.

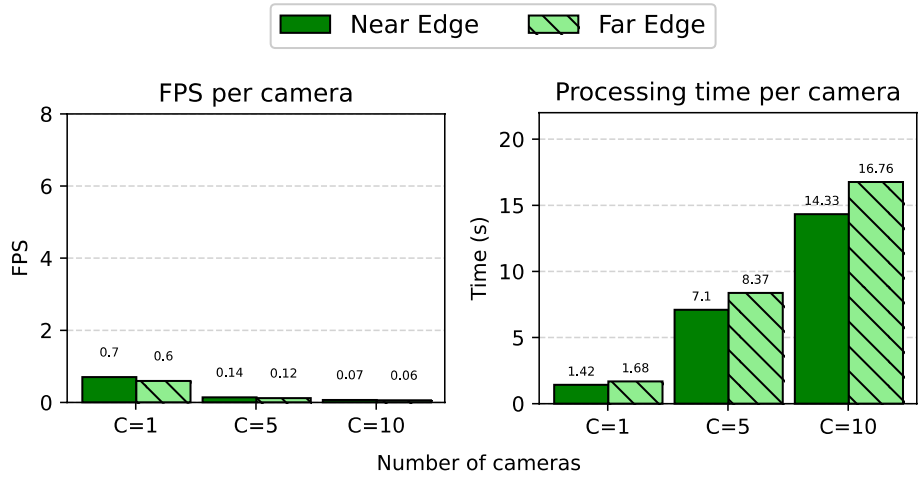

Figure 4. Image analysis performance per camera Edge scenarios.

Figure 3 and Figure 4 show the performance of the image analysis performed by the system. Specifically, the graph on the left reports the average FPS analysed per camera, while the graph on the right reports the average time required for the analysis of two images from the same camera. It is worth to highlight that in the scenarios with multiple cameras the round robin policy applied to extract enqueued frames ensures fairness among different sources thus ensuring a comparable image analysis delay among them.

As expected, the resulting framerate for each source is significantly influenced by two factors: the configuration of the system, i.e., cloud or edge, and the number of video sources. If the average FPS obtained in the cloud scenario is compared with the one obtained in the edge one, it can be noticed that, in the former, the system is capable of analysing between 4 and 1 FPS, while, in the latter, the system is not capable of analysing more than one FPS. This can be explained with the fact that the inference module exploits a different set of GPUs available on the node where it runs, i.e., more than one high-level GPUs in the cloud scenarios (Grid'5000

Co-funded by the Horizon 2020 Framework Programme of the European Union

testbed) and only one mid-level accelerator in the edge scenarios (Grid'5000 for the near edge scenario and Virtual Wall testbed for the far edge scenario). The cloud and edge accelerators offer different performance and results in terms of frames that can be analysed per second. If the results obtained with an increasing number of video sources are compared, it can be noticed that in both the cloud and edge scenarios the number of FPS is reduced significantly when the number of sources *C* is increased. This is noticeable in particular in the edge scenarios where the FPS drops significantly, reaching a value lower than 0.1 FPS with C=10. This can be explained by considering the round robin policy adopted by the inference module to select the frames, which divides the limited capacity of the GPU accelerators among different video sources. If the two different cloud configurations are compared, it can be noticed that cloud 1 configuration results in a number of FPS analysed per source slightly higher than cloud 2. This can be explained with the fact that the cloud 1 server is equipped with 4 GPUs instead of the 2 GPUs available in the cloud 2 server: even though cloud 2 exploits more powerful GPUs, i.e., Nvidia A100, rather than Nvidia GeForce RTX 2080, the more GPUs available in cloud 1, i.e., 4 vs 2, provides a slight advantage. The higher computing capabilities offered by Nvidia A100 over Nvidia GeForce RTX 2080 do not compensate the lower level of parallelism, which allows cloud 1 to analyse 4 frames in parallel. If the performance of the two edge configurations is compared, it can be noticed that they result in a very small difference. This can be explained with a small difference in the performance for the Near and Far edge configurations, the former equipped with one Nvidia GTX 1080 Ti, the latter with one Nvidia GTX 980. Such small difference in the performance in terms of FPS confirms again that the performance in the image analysis mainly depends on the performance of the GPU, since the large set of resources available to the Near edge node compared with the Far edge node (28 cores vs 6 cores / 768GB vs 12GB) does not provide a significant advantage.

The processing time for the analysis of two consecutive frames from the same source is dual with the average FPS analysed per camera. As the number of video sources increases, the processing time between two frames increases as well. This is due to the fact that the capacity of the system is shared across different video sources, which reduces the capacity allocated to each camera. If the results obtained in the cloud and edge configurations are compared, it can be noticed that there is a significant difference between the two configurations. This confirms that the edge scenario does not scale well with the number of video sources, due to the availability of only one accelerator with less computing capabilities. If the results are analysed in absolute terms, it can be noticed that, in general, the average time for a frame to be analysed is in the order of a few seconds for the cloud configuration, however, it increases to more than 10 seconds in the edge configuration with C=10. Considering that the generation rate at the video source is 5 FPS, i.e., equals to one frame generated every 200ms, it follows that the system is not capable of keeping the pace with the data generated by the cameras in any of the considered scenarios, even in the scenario with C=1. This results in a large number of frames that are discarded and are not analysed by the system.

Co-funded by the Horizon 2020 Framework Programme of the European Union

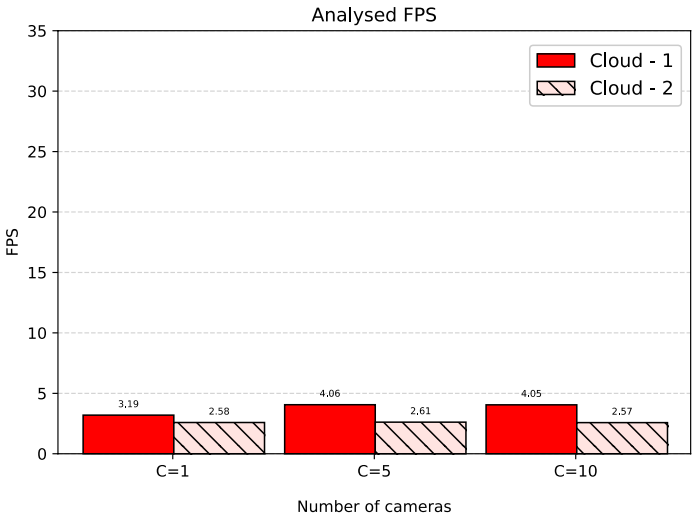

Figure 5. Overall Analysed FPS in cloud scenarios

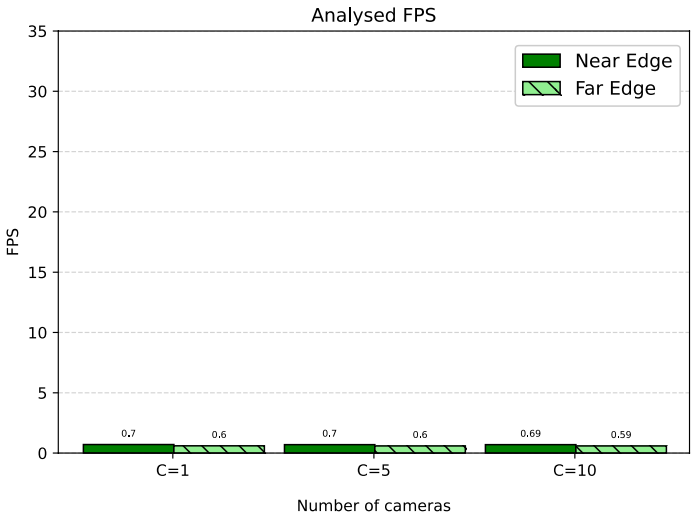

Figure 6. Overall Analysed FPS in the edge scenarios

In order to estimate the amount of frames that are discarded by the system, in Figure 5 and Figure 6 the overall Analysed FPS is shown. If the Analysed FPS is compared with the nominal input rate, i.e. 5 FPS in the C=1 scenario, 25 FPS in the C=5 scenario and 50 FPS in the C=10 scenario, it can be seen that a large number of frames are dropped by the system in both the cloud and edge configurations, thus confirming that the overall system is not capable of handling the large amount of frames injected by the video sources By considering the overall Analysed FPS in absolute terms, it is evident that in both the cloud and edge configurations, when an increasing number of video sources is considered, only a slight increase in the overall FPS analysed can be seen on the system, thus confirming that it is rarely idle, even with one video source.

Co-funded by the Horizon 2020
Framework Programme of the European Union

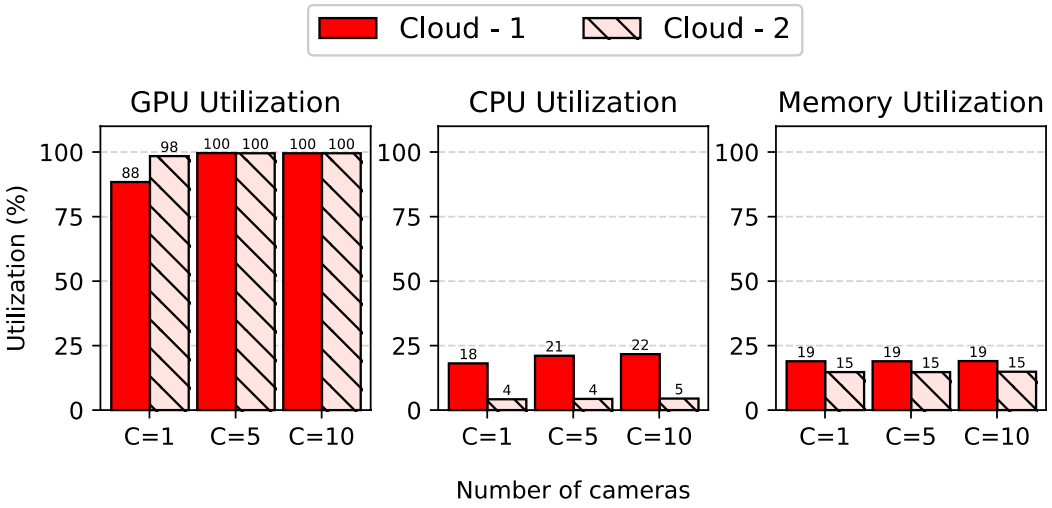

Figure 7. GPU, CPU and RAM utilization in the cloud scenarios

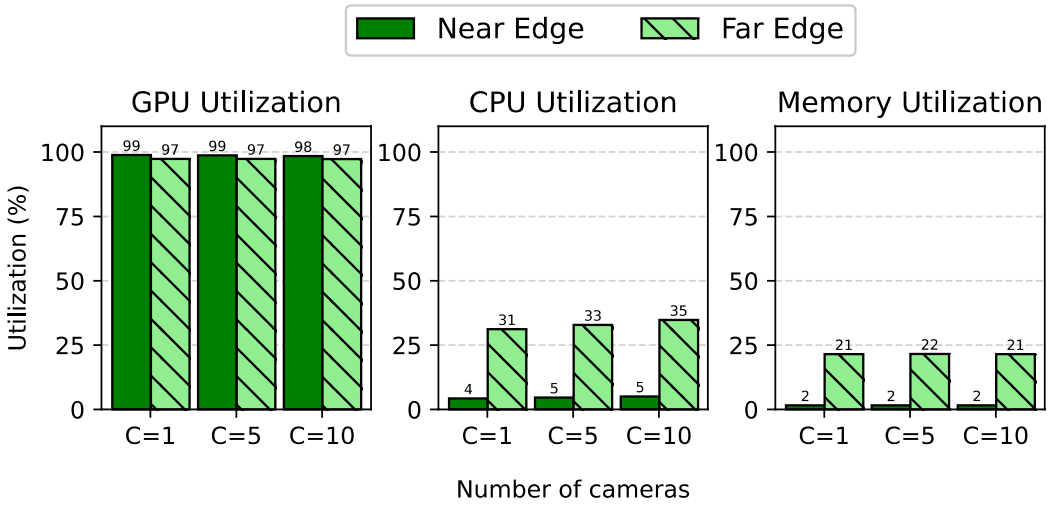

Figure 8. GPU, CPU and RAM utilization in the edge scenarios

In order to confirm that the GPU accelerator is the bottleneck of the system, and that it limits the overall image analysis performance, the percentage of GPU, CPU and RAM utilization is measured on the nodes where the inference module is deployed. In Figure 7 and Figure 8 the GPU, CPU and RAM utilization for the cloud and edge scenarios, respectively, are reported. As can be seen, the GPU accelerators are fully utilized in all the scenarios except for the cloud 1 scenario with C=1, which results in a utilization that is still quite high, i.e., slightly below 90%. If the CPU and RAM utilization are considered, instead, it can be seen that they are low in all the considered scenarios, and hence do not represent a bottleneck for the performance of the system, as all the values are below 40%. If the CPU utilization between the two cloud configurations (both obtained with nodes from the Grid'5000 testbed) is compared, it can be noticed that one configuration, i.e., cloud 2, results in a significantly lower utilization. This can be explained considering the number of cores (64 cores) of the cloud 1 server, which is

Co-funded by the Horizon 2020 Framework Programme of the European Union

significantly higher than the number of cores available in the cloud 2 server (12 cores). The same trend can be noticed if the CPU utilization obtained with near and far edge (28 cores in near edge - Grid'5000 testbed nodes - vs 6 cores in far edge – Virtual Wall testbed nodes) are compared. It is worth to highlight that in some cases, i.e., the near edge, the CPU utilization is even lower than the CPU utilization in the cloud. This is due to the fact that in the cloud scenarios, other processes run on the cloud node, i.e., the aggregation and the social media analysis processes. From the analysis of the results of the RAM utilization, it emerges that the near edge scenario results in a very low RAM utilization, if compared with the others. This can be explained with the very large RAM capacity that characterizes the far edge node, which offers up to 768GB.

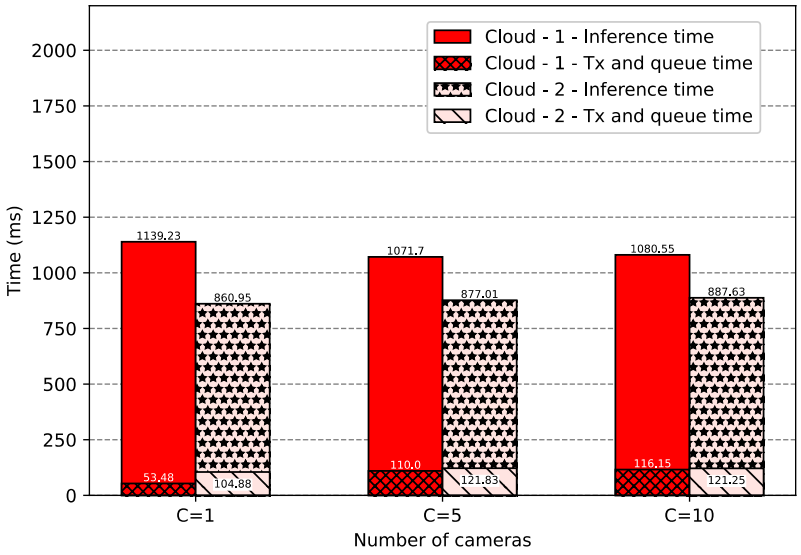

Figure 9. Frame analysis delay in cloud scenarios

Co-funded by the Horizon 2020
Framework Programme of the European Union

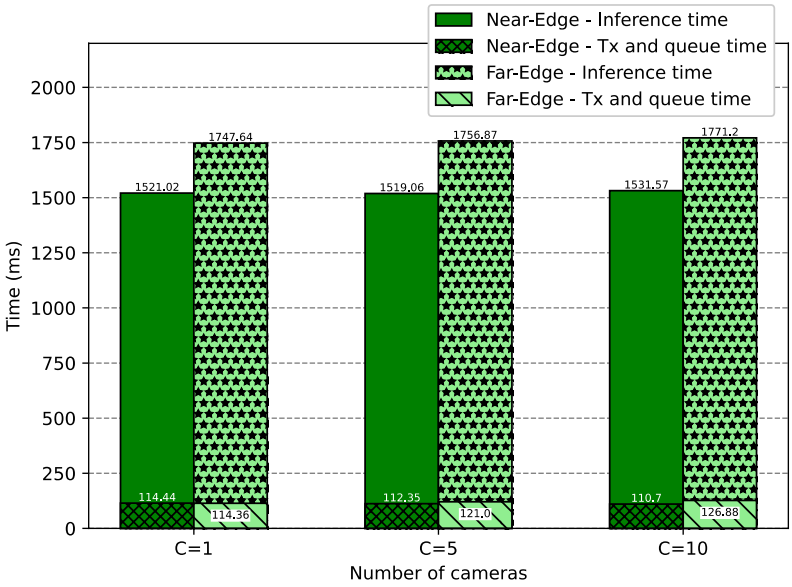

Figure 10. Frame analysis delay in edge scenarios

Such results confirm that the bottleneck of the system is the GPU. In order to confirm that such bottleneck is caused by the complexity of the inference operations in the current implementation of the system, in Figure 9 and Figure 10 the frame analysis delay is shown. Those data report the analysis delay of the frames that are analysed and not discarded by the inference module, in the cloud and edge configurations, respectively. As can be seen the majority of the delay is caused by the inference time, the time required for the inference module to analyse the frame. The transmission and queue time, instead, is very low, due to the fact that frames are continuously replaced in the queue of the inference module, which is configured to store only one image[5], the most recent one. As can be noticed by comparing the inference time in the cloud and in the edge scenarios, it emerges that the inference time strictly depends on the type of the accelerator: the more powerful the accelerator is, the shorter the inference delay is. If the inference time is analysed in absolute terms, its values are between 1s and 2s, thus significantly higher than the time between the generation of two frames on a single camera, confirming that the system is not capable of analysing all the images, even in a simple scenario with a single camera (C=1), due to the complexity of the inference algorithm. If the inference time obtained with the different configurations is compared, it can be noticed that there is a significant difference between high-range GPUs in the cloud and mid-range GPUs in the edge, i.e., around 1s in the cloud and up to 1.6s in the edge. If the results obtained in the two cloud configurations are compared, however, no substantial difference can be noticed. This explains the advantage that the cloud 1

[5] Scenarios with larger queues have been analyzed and run, however, the difference in the obtained results were negligible, therefore their presentation is omitted for the sake of brevity.

Co-funded by the Horizon 2020 Framework Programme of the European Union

configuration provides over the cloud 2 in terms of overall FPS analysed that was shown in Figure 5: more parallelism provided by more GPUs, 4 in cloud 1 and 2 in cloud 2, is more convenient than less GPUs with more computing power.

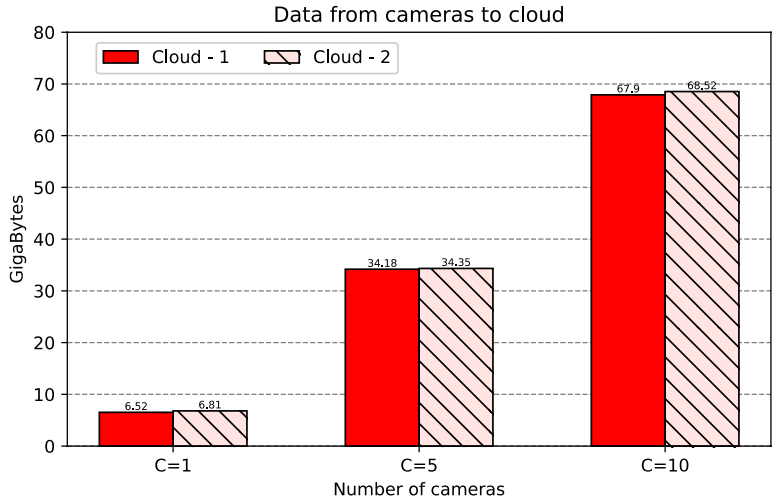

Figure 11. Communication overhead in the cloud scenarios

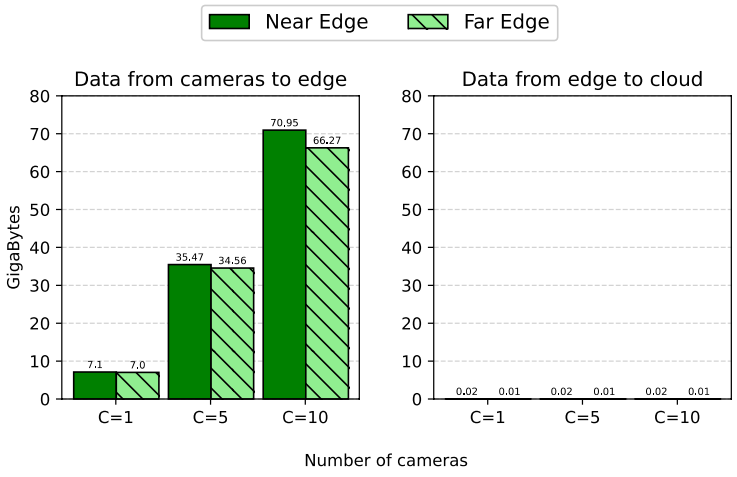

Figure 12. Communication overhead in the edge scenarios

In order to compare the communication overhead between the cloud (Grid'5000 testbed) and the edge (Grid'5000 for the near edge scenario and Virtual Wall testbed for the far edge scenario) configurations, in Figure 11 and Figure 12 the results of the data transmitted between the different modules of the system are reported in the cloud and edge configurations, respectively. Specifically, Figure 11 reports the overall data transmitted from the cameras to the inference module running on the cloud for the cloud scenario, while Figure 12 reports the overall data transmitted from the cameras to the inference module running on

Co-funded by the Horizon 2020 Framework Programme of the European Union

the edge (graph on the left) and the data transmitted from the inference module to the aggregator running on the cloud (graph on the right) for the edge scenario.

As can be seen, the cloud configuration is characterized by a significant communication overhead, in the order of gigabytes, that increases with the number of cameras. This is due to the fact that each camera has to transmit the video frames to the inference module on the cloud. The same amount of data is transmitted also in the edge configuration, however, it occurs between the cameras and the edge layer, where the inference module runs, while the communication between the edge layer and the cloud is limited, i.e., in the order of tens of megabytes. The latter is due to the fact that between the edge and the cloud only the results of the inference are transmitted, which are limited in size if compared with the frames of the video feeds.

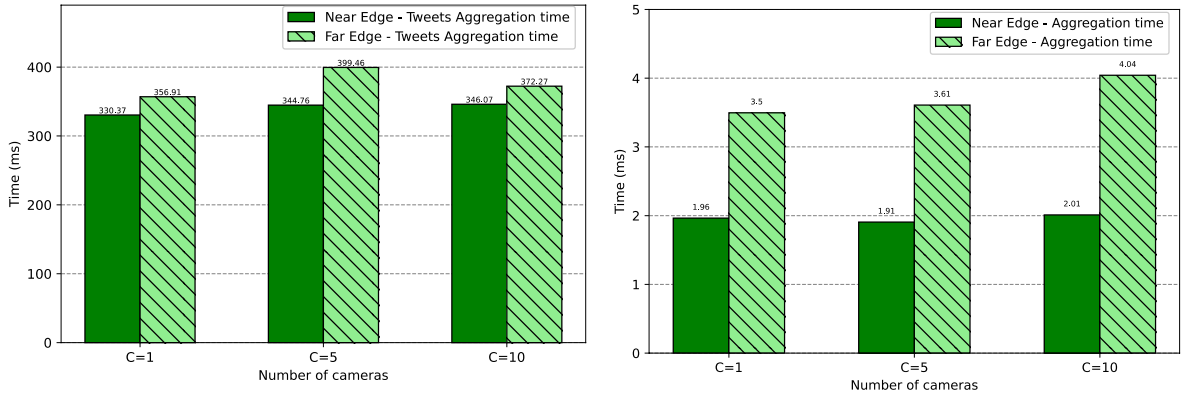

Figure 13. Cloud aggregation performance

Finally, in Figure 13 the performance of the aggregation module that is always deployed on the cloud node are reported, in order to verify that the aggregation process is performed without scalability issues. The results obtained with the cloud scenarios are omitted for the sake of brevity since they results in similar performance. The performance is measured with two metrics: (i) the aggregation latency, defined as the time required for the aggregator thread to complete the aggregation of all the data received from the inference module (right graph) and (ii) the time required by the social media thread to retrieve and analyse the tweets (left graph). As it can be seen, the aggregation latency is very low, between 2 ms and 4 ms, thus confirming that the aggregation process is performed without issues in a very short amount of time. The latency of the social media thread to complete, instead, is higher if compared with the aggregation latency, i.e., in the order of 350 ms. This is due to the fact that the retrieval process of the latest tweets requires multiple Twitter APIs invocations, which can take hundreds of milliseconds, thus influencing the overall latency.

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.4.1.2 Multiple edge scenarios

In this section the results obtained with more than one edge node, i.e., e=2 ande=3, are presented. For these experiments, only the near edge configuration is considered, since it is the one implemented using the Grid'5000 testbed, which can provide up to 3 nodes with the same configuration. For each scenario, the same values of C are considered, i.e., C=1, C=5 and C=10. In this case, however, C represents the number of video sources assigned to each edge node. The goal of those scenarios is to show that the system implementation can scale even when multiple edge nodes are employed. For the sake of brevity in the following only a few of the metrics evaluated are reported, as the majority of them confirmed the results that we obtained for the e=1 scenario.

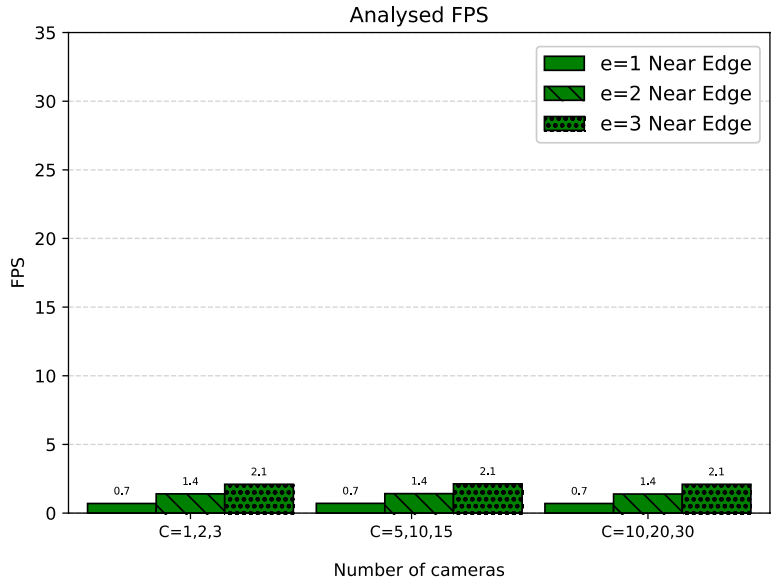

Figure 14. Overall Analysed FPS in the near edge with multiple nodes

In Figure 14 the analysed FPS by the overall system is reported, in the scenario with e=1, e=2 and e=3. As can be seen, the overall analysed FPS increases with the number of edge nodes, as each edge node analyses independently the frames received from its group of video sources. Even though in this case the parallel analysis of frames occurs, it can be noticed that, by comparing the nominal input rate with the analysed FPS, the system is still not capable of handling the large quantity of frames injected by the cameras. If the results obtained with e=3 and with e=1 are compared, it can be noticed that the overall analysed FPS obtained with e=3 is exactly three times the rate obtained with e=1, thus confirming that each node does not influence the others in terms of scalability. This was largely expected since each edge node analyses its frames in an independent manner from the others.

Co-funded by the Horizon 2020 Framework Programme of the European Union

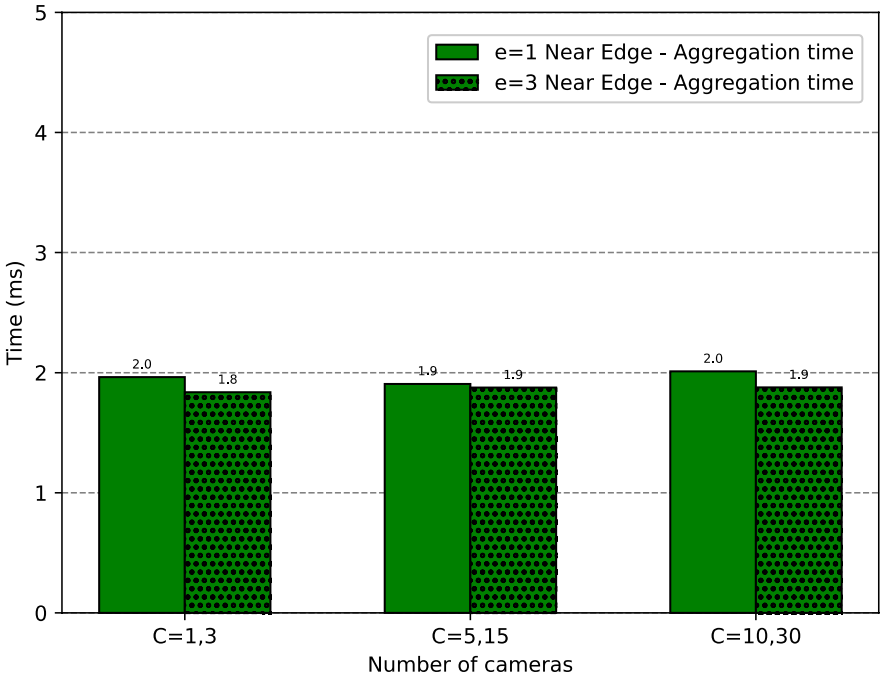

Figure 15. Cloud aggregation performance with multiple edge nodes

The only potential bottleneck from the system can be the aggregation process implemented in the cloud, which receives in this configuration data from multiple edge nodes. In Figure 15 the time required to complete the aggregation is shown. As it can be seen the aggregation time is still very low and the aggregation time is comparable in both e=1 and e=3 configurations.

4.4.1.3 Conclusions

The results presented in Sections 4.4.1.1 and 4.4.1.2 provide an analysis of the performance of the system with different cloud and edge configurations. They offer an assessment of the system scalability with a varying number of video sources (up to 10) and multiple edge nodes (up to 3).

By analysing the performance of the overall system in terms of FPS analysed (overall or per video source) it can be observed that the system is not capable of handling in full the analysis of the images generated by the video sources. This is clear even in the case with only one video source, as the system results in a number of frames analysed per second lower than the number of frames generated by the camera. This occurs in both the cloud and edge configurations, although in the cloud scenario it is mitigated by a more powerful hardware accelerator used to analyse the images and a larger set of GPUs available on the cloud node. Such results identify the inference module as the bottleneck of the system. The reason for such low performance is due to the Inception Network v3 model used to analyse the images. The Inception v3 convolutional neural network v3 [2] model is known in literature to be one

Co-funded by the Horizon 2020 Framework Programme of the European Union

of the most complex models developed to analyse the images on cloud environments, and the obtained numbers are consistent with other performance obtained from other studies in literature [5]. The model was selected by UMBC for the implementation of the original cloud-based system [1] as it resulted in a good level of accuracy[6], however, this first implementation was not originally designed to be deployed on constrained environments, e.g., near edge, far edge or on-site.

Based on the analysis of this set of results, the opportunity of assessing the performance of different neural network models has been considered and analysed. To this aim, two more lightweight models were identified:

- MobileNet [3], a convolutional neural network for image analysis specifically designed with a lightweight architecture to run on mobile systems.
- YOLOv5 – You Look Only Once [4], an object detection algorithm specifically designed for embedded and constrained systems in general.

Both the models were integrated in our prototype in order to obtain a lightweight implementation of the prototype. Such lightweight implementation was assessed by replicating the same scenarios considered for the evaluation of the initial implementation of the platform. The results are presented in the following section.

4.4.2 Lightweight implementation analysis

In this section the overall performance of the lightweight implementation of the system with both Mobilenet and YOLO is analysed. The results presented in Section 4.4.1 are also reported here for comparison. For the sake of brevity, for the cloud scenario only the cloud 1 scenario is reported as it is the one that provided the better performance thank to the parallelization offered by 4 GPUs.

The following table (Table 4) summarizes for which scenarios the deployment of our system was successful.

[6] In our experiments we did not measure the accuracy of the systems, as the EdgeFlooding experiments aim at assessing the performance of the whole system with different hardware configurations. The selection of the proper model has been carried out in a preliminary set of experiments by UMBC. It is worth to highlight that the EdgeFlooding experiments exploits the same videos used by UMBC to select the models, therefore an analysis of the accuracy on the EdgeFlooding experiments would have been redundant.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Table 4. Lightweight implementation deployment result: • successful, x failed

Inference Algorithm	Cloud 1	Near Edge	Far Edge	On Site Edge 1	On Site Edge 2	On Site Edge 3
Inception	•	•	•	x	x	x
Mobilenet	•	•	•	•	•	x
YOLO	•	•	•	•	•	•

As can be seen, with YOLO the deployment of our system was successful on all the considered configurations, while with Mobilenet the deployment was successful on all the configurations, except the on-site edge 3. In this case, the deployment failed due to the reduced amount of RAM available that is only 4GB on the NVIDIA Nano embedded system.

In the following the results of the cloud, near edge and far edge configurations are first presented, then the results of the on-site scenarios are introduced. The latter scenarios, the on-site ones, are presented in a separate subsection, considered their peculiarity.

4.4.2.1 Cloud, near edge and far edge scenarios

In this section the results obtained with the cloud, near edge (Grid'5000 testbed) and far edge (Virtual Wall testbed) scenarios are introduced and analysed.

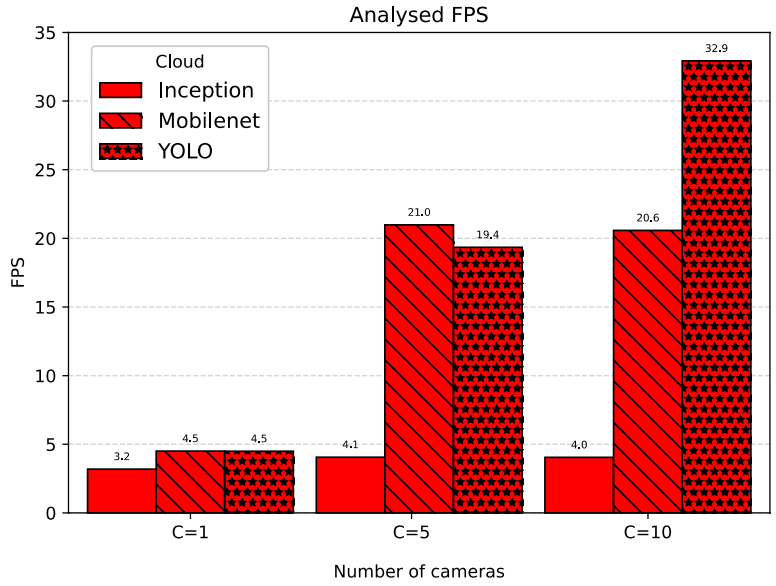

Figure 16. Overall image analysis performance Cloud scenarios with Mobilenet and YOLO

Co-funded by the Horizon 2020 Framework Programme of the European Union

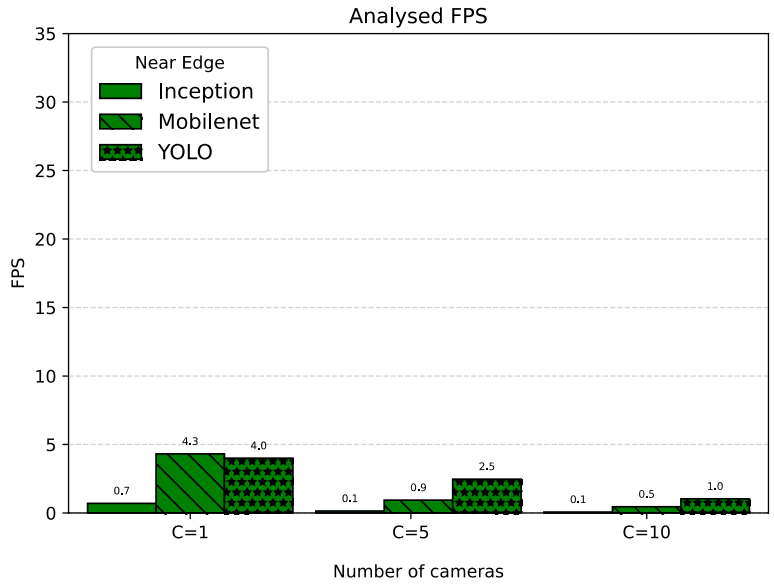

Figure 17. Overall image analysis performance near edge scenarios with Mobilenet and YOLO

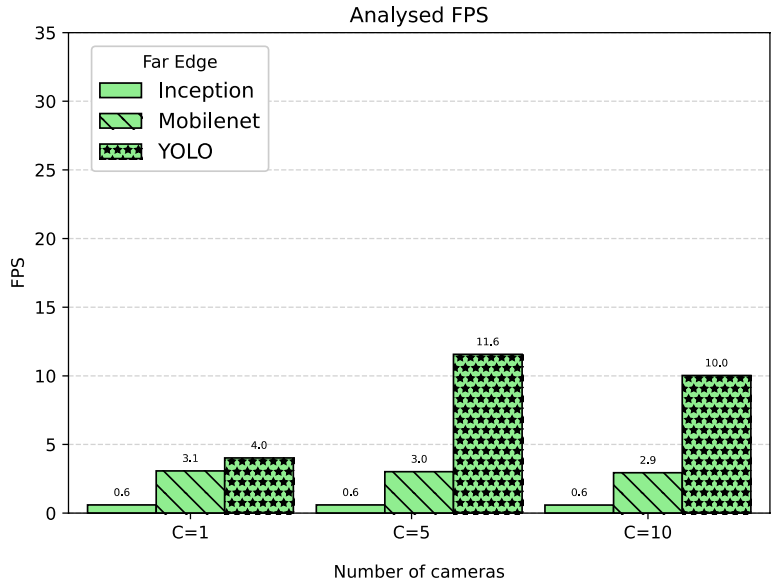

Figure 18. Overall image analysis performance far edge scenarios with Mobilenet and YOLO

Figure 16, Figure 17 and Figure 18 show the performance of the overall image analysis performed by the system in the cloud, near edge and far edge scenarios respectively. As in Section 4.4.1.1, the graph reports the overall analysed FPS on the system. As expected, the two more lightweight inference models help in significantly increasing the overall performance of the system. If the cloud scenario is considered (Figure 16), it can be noticed that both Mobilenet and YOLO result in an analysed FPS that is close to the nominal input rate with both C=1 (nominal input rate 5 FPS) and C=5 (nominal input rate 25 FPS), while with C=10

Co-funded by the Horizon 2020 Framework Programme of the European Union

(nominal input rate 50 FPS) it results in very high analysed FPS, although lower than the rate of frames injected in the system, thus confirming that the capacity is reached. At C=10 the decrease in the overall analysed FPS is more highlighted with Mobilenet that results in half the frames analysed, thus confirming that YOLO is the most lightweight option. If the near and far edge scenarios are analysed (Figure 17 and Figure 18, respectively), it can be seen that the adoption of a lightweight inference model helps in increasing significantly the overall performance of the system, w.r.t. Inception net. The overall analysed FPS, however, is still significantly lower than the nominal input rate in the scenarios with C=5 and C=10, thus resulting in the drop of many frames.

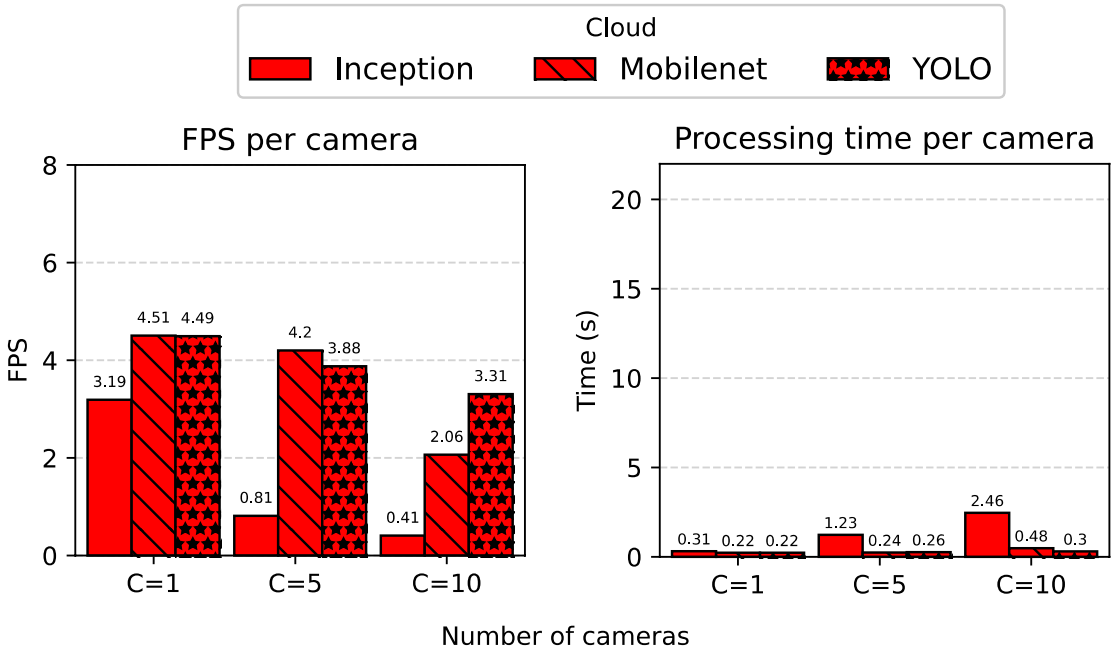

Figure 19. Image analysis performance per camera Cloud scenarios with Mobilenet and YOLO

Co-funded by the Horizon 2020 Framework Programme of the European Union

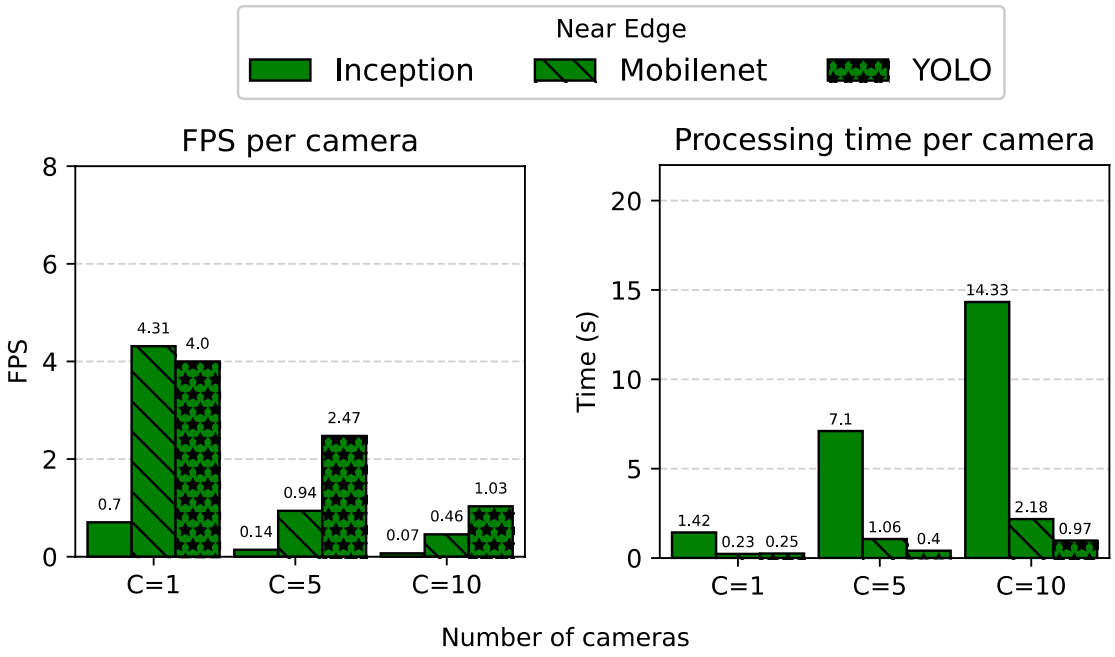

Figure 20. Image analysis performance per camera near edge scenarios with Mobilenet and YOLO

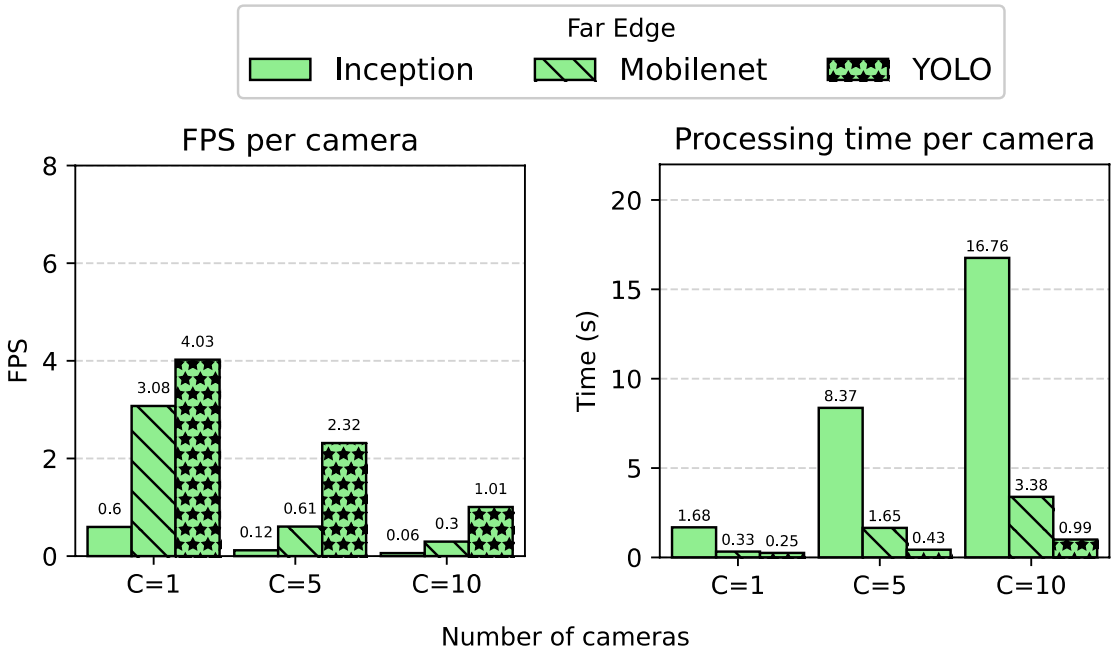

Figure 21. Image analysis performance per camera far edge scenarios with Mobilenet and YOLO

The improvement in the overall performance of the system is highlighted also by the metrics per video source, which are reported in Figure 19, Figure 20 and Figure 21 that show the analysed FPS per camera and the frame processing time per camera in the cloud scenario,

Co-funded by the Horizon 2020 Framework Programme of the European Union

near edge scenario and on site scenarios, respectively. As expected, the overall FPS per camera increases significantly with both Mobilenet and YOLO in all the scenarios. The improvement is especially noticeable with YOLO, as in both the near and far edge scenarios it can guarantee up to 1 FPS per source even in the case with C=10. Mobilenet results in lower performance than YOLO, however, also in this case the improvement is significant.

When considering the processing time per camera, this improvement for the edge configurations can be translated in a metric that measures the impact for the application: the usage of a more lightweight inference model significantly reduces the time between the analysis of two subsequent frames from the same source, which is reduced from one frame every approximately 10 seconds with Inception net to one frame every 2/3 seconds with Mobilenet and 1 frame every second with YOLO.

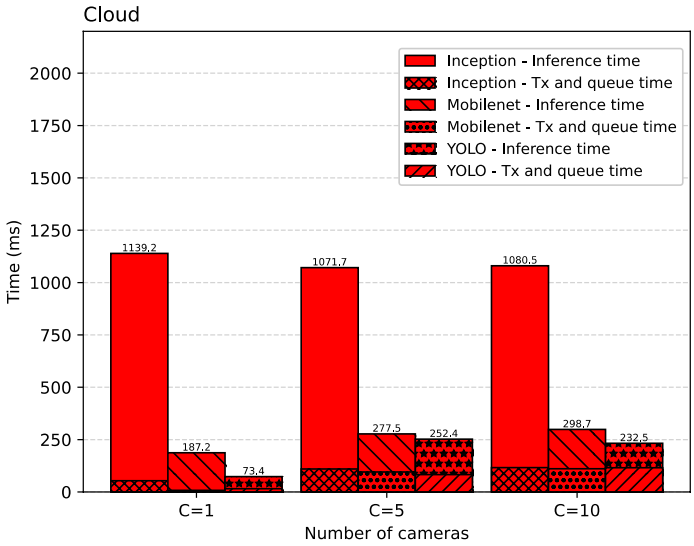

Figure 22. Frame analysis delay in cloud scenarios with Mobilenet and YOLO

Co-funded by the Horizon 2020
Framework Programme of the European Union

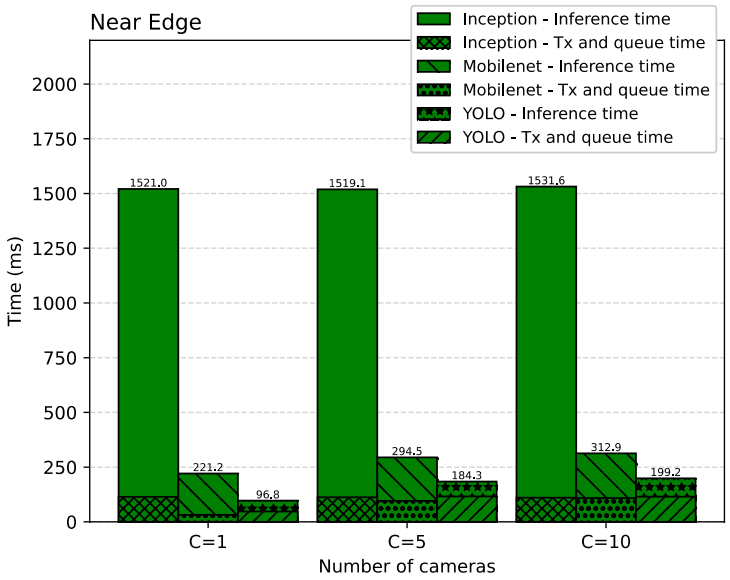

Figure 23. Frame analysis delay in edge scenarios with Mobilenet and YOLO

The overall improvement of the performance of the system is the result of a more lightweight model for inference, which reduces significantly inference time. In order to directly measure this improvement, in Figure 22 and Figure 23 the frame analysis delay for the cloud and near edge scenarios is reported. For the sake of brevity, the results for far edge are omitted, as they lead to the same conclusions. As it can be seen, Mobilenet and YOLO result in an inference time that is significantly lower than Inception net, thus helping the whole system to achieve a higher analysis rate.

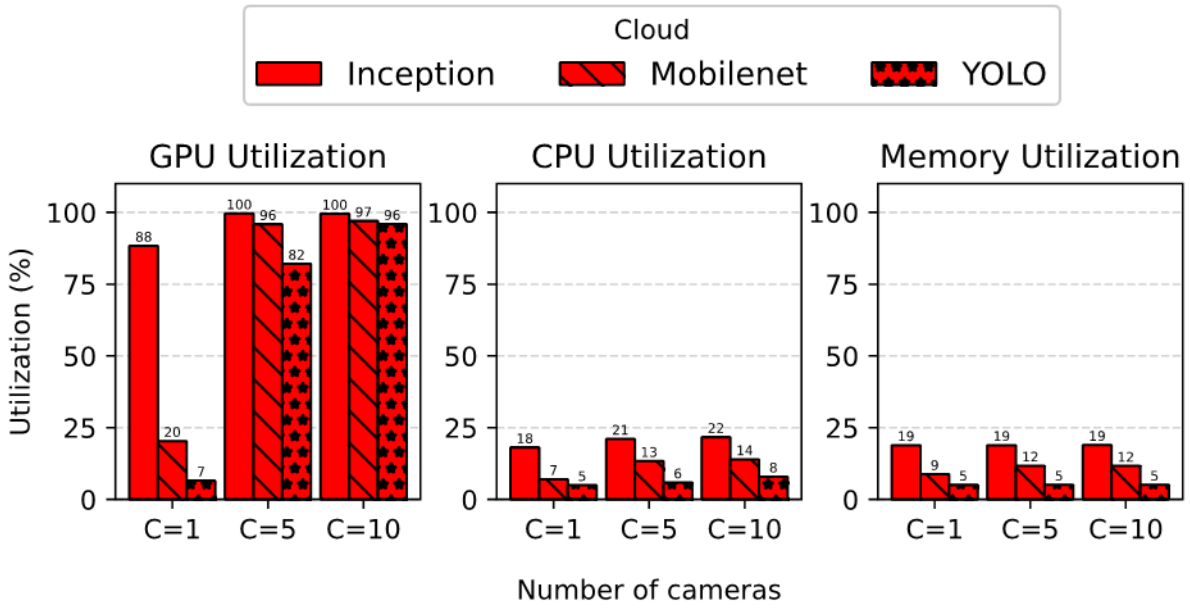

Figure 24. Resource utilization in the cloud scenario with Mobilenet and YOLO

Co-funded by the Horizon 2020 Framework Programme of the European Union

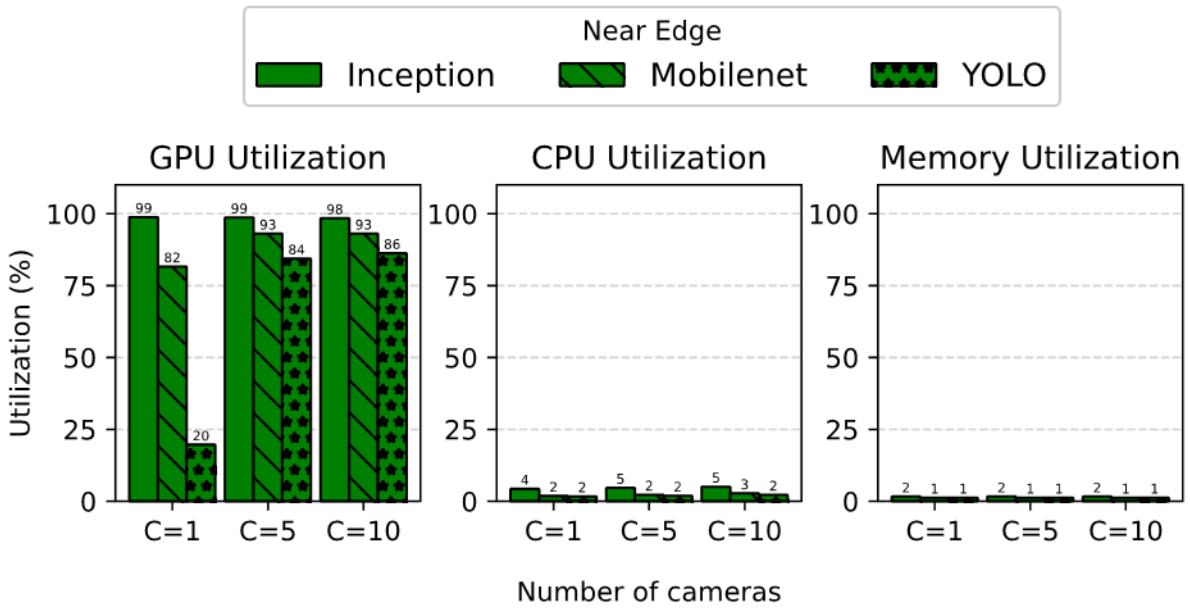

Figure 25. Resource utilization in the near edge scenario with Mobilenet and YOLO

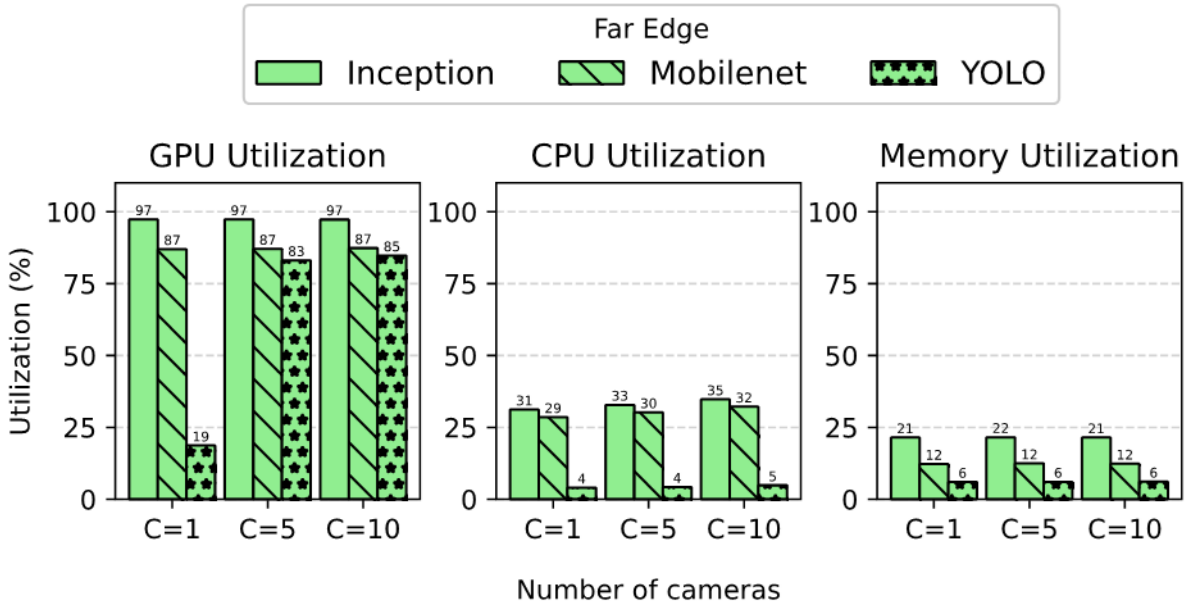

Figure 26. Resource utilization in the far edge scenario with Mobilenet and YOLO

To conclude, in Figure 24, Figure 25 and Figure 26 the resource utilization is analysed for the cloud, near edge and far edge scenarios, respectively. As it can be seen, compared with Inception net, Mobilenet and YOLO both results in a lower GPU utilization, significantly lower in some configurations. If we consider the scenario with only one video source (C=1), we can notice that YOLO results in a very low GPU utilization, i.e., lower than 20% in all the scenarios, thus confirming that all the configurations are capable of handling a single video source with YOLO. Mobilenet, instead, is confirmed to be more complex, as it results in a higher GPU

Co-funded by the Horizon 2020 Framework Programme of the European Union

utilization for both near and far edge scenarios, i.e., between 80% and 90%, while it results in a low utilization in the cloud scenario, i.e. around 20%, due to the multiple and more powerful GPUs available. If the scenarios with multiple video sources, i.e., C=5 and C=10, are considered, higher GPU utilization is displayed, confirming that the system is – or it is about to be – saturated. The other metrics, i.e., CPU and RAM utilization, confirm that also the lightweight implementation does not have a bottleneck in the performance in terms of CPU and RAM, as they both results in low utilization.

4.4.2.2 On-site edge scenarios

In this section the results obtained with the on-site configurations (all obtained with the Fog testbed) are introduced and analysed.

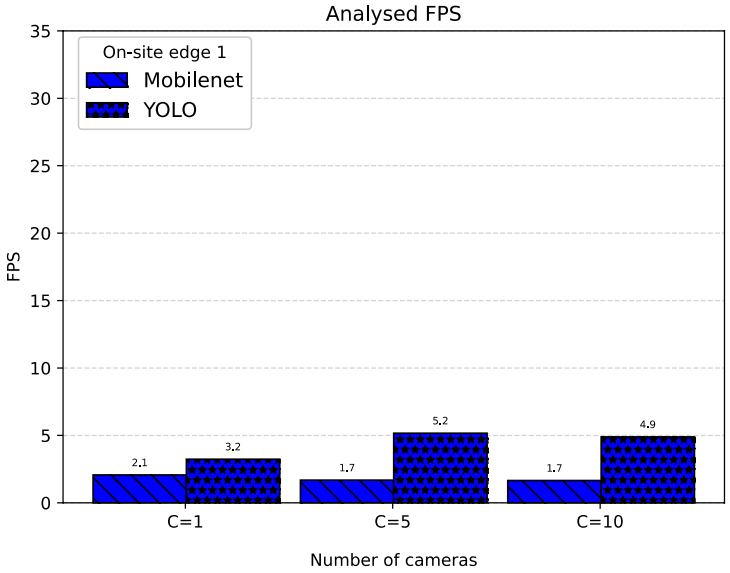

Figure 27. Overall image analysis performance on-site edge 1 scenario with Mobilenet and YOLO

Co-funded by the Horizon 2020 Framework Programme of the European Union

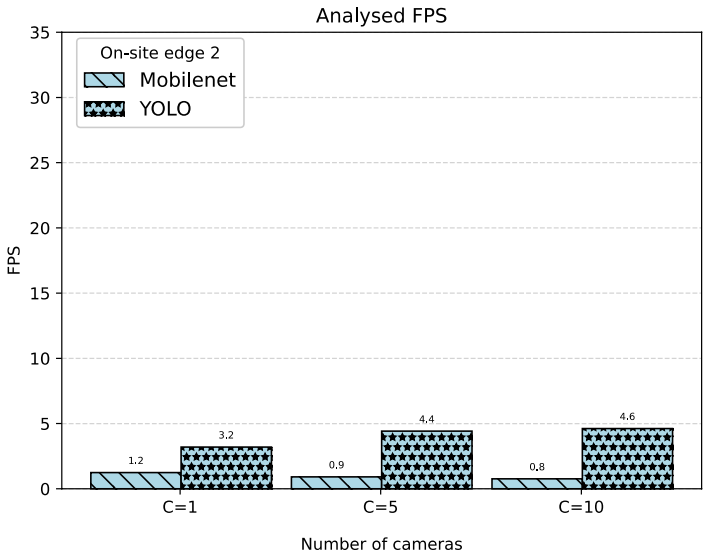

Figure 28. Overall image analysis performance on-site edge 2 scenario with Mobilenet and YOLO

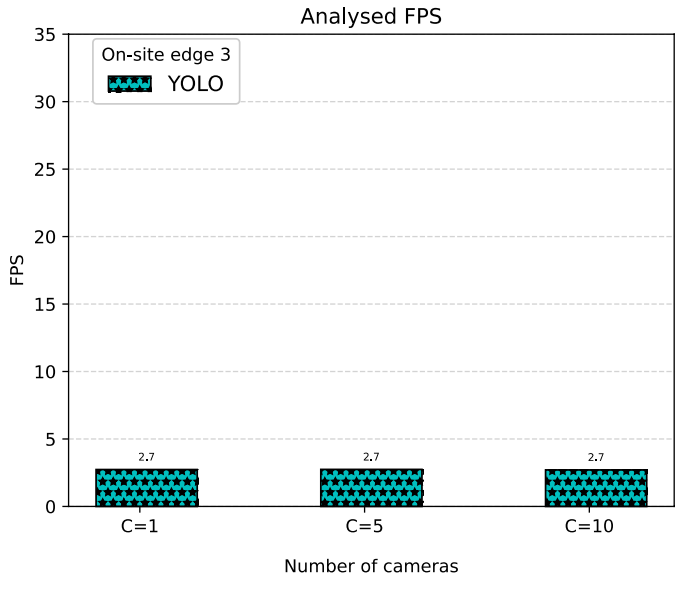

Figure 29. Overall image analysis performance on-site edge 3 scenario with YOLO

Figure 27, Figure 28 and Figure 29 report the performance of the image analysis with three different embedded systems with Mobilenet and YOLO, except for the configuration on-site 3 that reports only YOLO, since Mobilenet could not fit the limited RAM of the NVIDIA Jetson Nano. The analysed FPS shows that the overall analysis rate is significantly lower than the one obtained with the other cloud and edge configurations. This was expected, considering the low capabilities of the accelerator available on the embedded systems. As it can be seen, when the number of sources C is increased, the overall analysed FPS only slightly increases, thus confirming that the system is already saturated with only one video source, the configuration for which the on-site configurations are more likely to be adopted. If the results with

Co-funded by the Horizon 2020 Framework Programme of the European Union

Mobilenet and YOLO are compared, it can be noticed that, as expected, YOLO results in a higher analysed FPS than Mobilenet, due to its more lightweight structure designed for constrained systems.

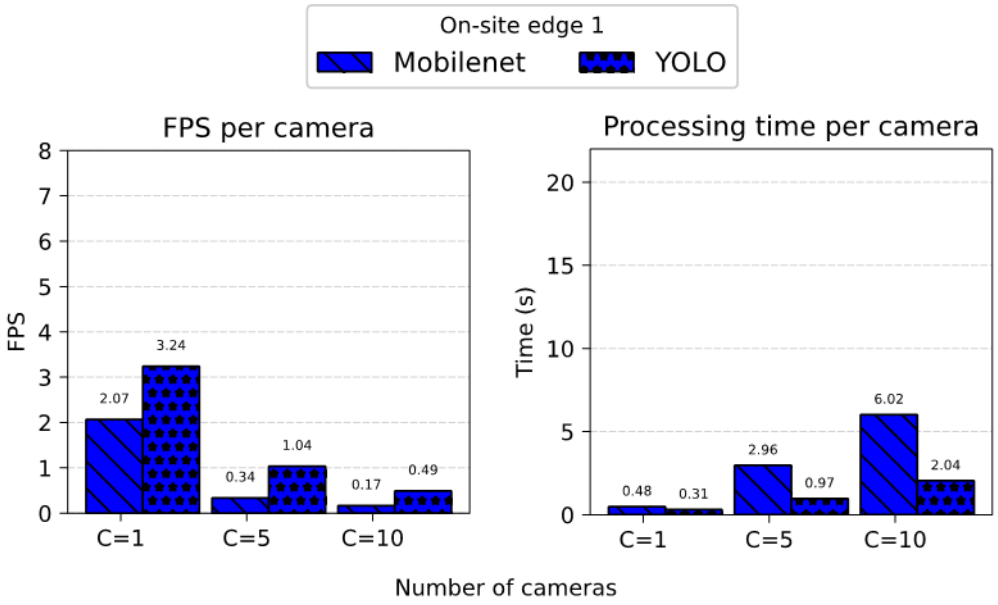

Figure 30. Image analysis performance per camera on-site 1 edge scenario with Mobilenet and YOLO

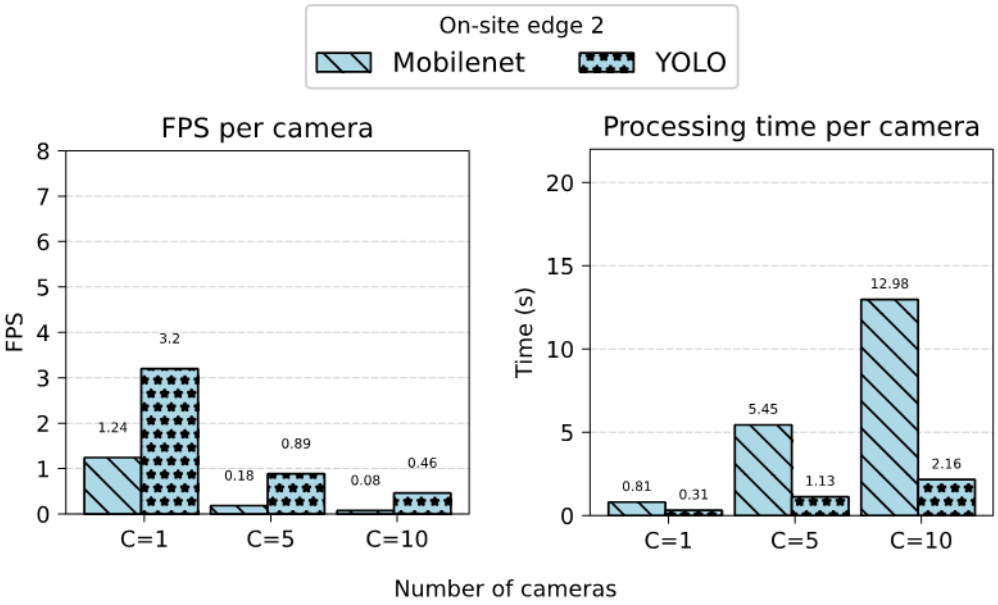

Figure 31. Image analysis performance per camera on-site 2 edge scenario with Mobilenet and YOLO

Co-funded by the Horizon 2020 Framework Programme of the European Union

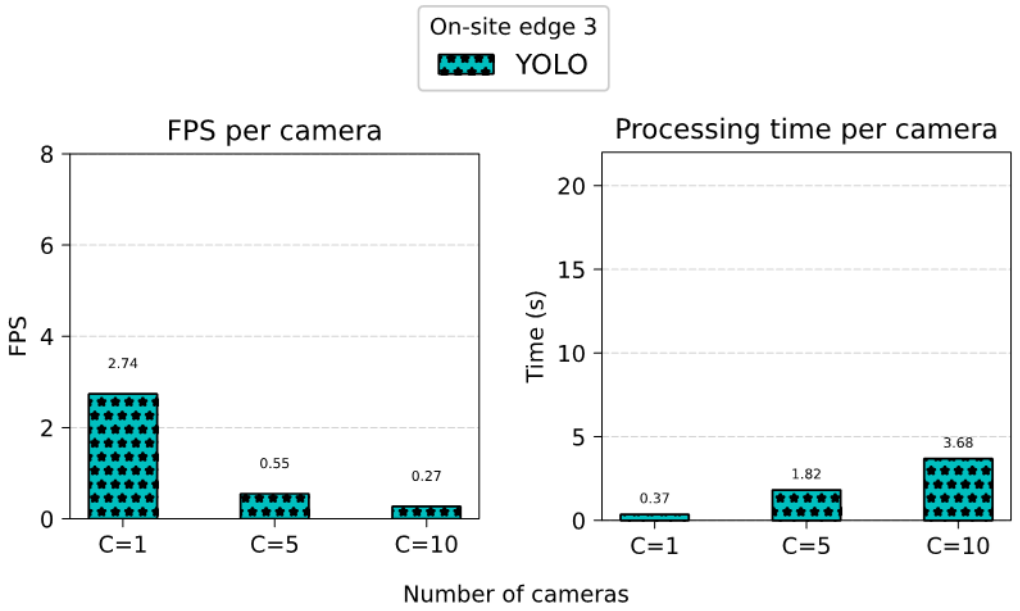

Figure 32. Image analysis performance per camera on-site 3 edge scenario with YOLO

Figure 30, Figure 31 and Figure 32 report the analysed FPS per camera and the frame processing time per camera in the three on site scenarios with different embedded systems. If the analysed FPS per camera is considered, it can be noticed that as the number of video sources increases, the analysed framerate significantly decreases, thus confirming that this configuration is not well suited to handle data from multiple video sources. However, if the framerate obtained with Mobilenet and YOLO are compared with the framerates per camera obtained with Inception net in the edge configurations (see Section 4.4.1.1), it can be noticed that the analysed FPS per camera values are in the same order of magnitude, thus confirming the significant difference in terms of complexity with Inception net. If the results obtained with the three embedded systems are compared, it can be noticed that only a slight difference can be noticed between the on-site 1 and 2 configurations, while the difference is more highlighted with the on-site 3 configuration, i.e., with the NVIDIA Jetson NANO that is a very constrained device. The difference in the performance is very noticeable also if the processing time per camera is considered. If the obtained values are considered in absolute terms, however, it can be noticed that low processing times are obtained in the scenario with only one source, i.e., less than 1s in all the configurations, thus confirming the feasibility of adopting the on-site configurations to analyse the images for at least one camera. If multiple cameras are considered, longer processing times are obtained, they are, however, in the same order of magnitude than the ones obtained with Inception net in the edge configurations.

Co-funded by the Horizon 2020
Framework Programme of the European Union

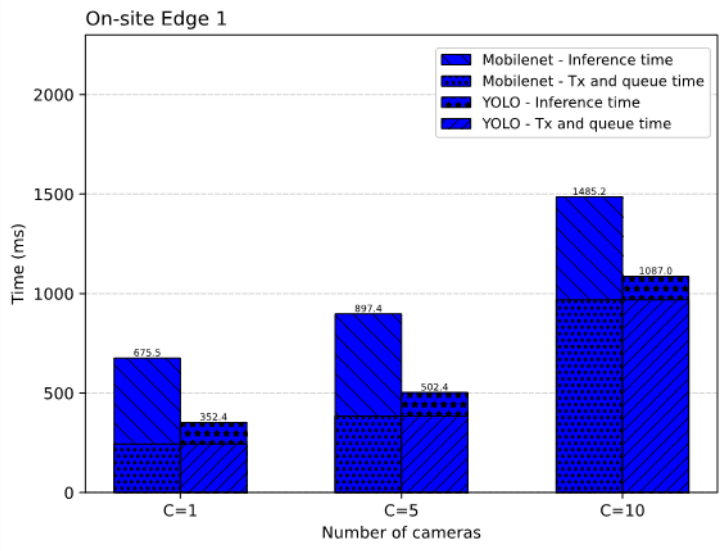

Figure 33. Frame analysis delay in on-site 1 edge scenario with Mobilenet and YOLO

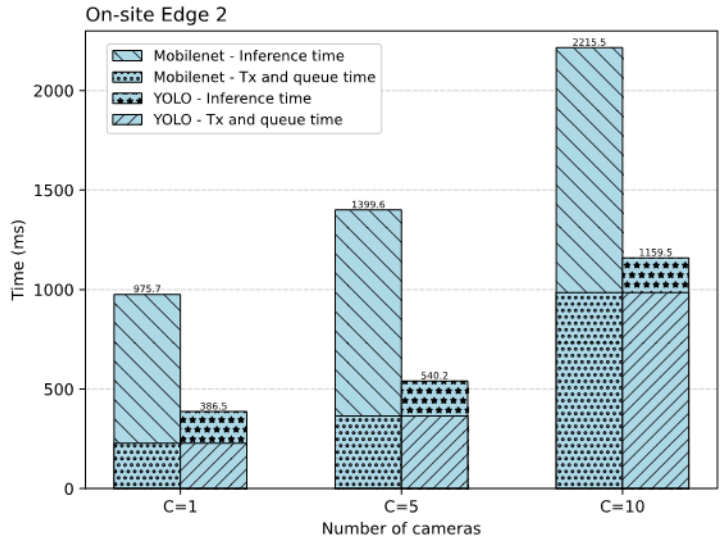

Figure 34. Frame analysis delay in on-site 2 edge scenario with Mobilenet and YOLO

Co-funded by the Horizon 2020
Framework Programme of the European Union

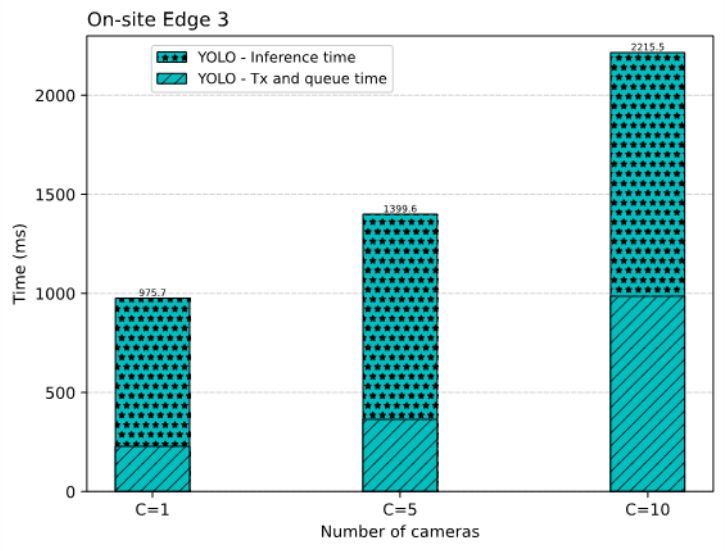

Figure 35. Frame analysis delay in on-site 3 edge scenario with YOLO

The analysis of the frame delay is reported in Figure 33, Figure 34 and Figure 35 for the on-site edge 1, 2 and 3 scenarios, respectively. If we consider the scenarios with C=1, it can be noticed that the on-site 1 configuration results in the lowest inference time, thus confirming that it is the embedded system with the more powerful embedded GPU. This difference is more highlighted with Mobilenet, while with YOLO only a slight difference can be noticed. The on-site 3 configuration is confirmed to be very constrained, as it results in the highest inference time with YOLO, the only model that fits in this hardware. If we consider the scenarios with more than one video source, i.e., C=5 and C=10, we can notice that the TX and queue time increase. This can be explained with the longer time required by the system to handle the analysis of frames from different sources.

Co-funded by the Horizon 2020
Framework Programme of the European Union

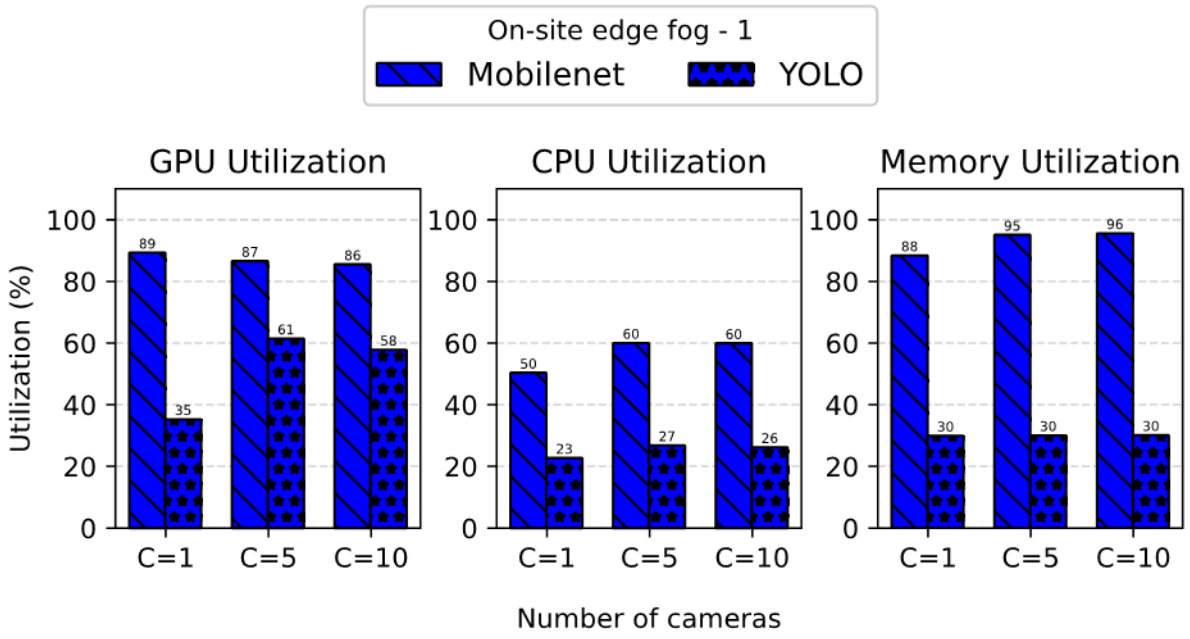

Figure 36. Resource utilization in the on-site 1 edge scenarios

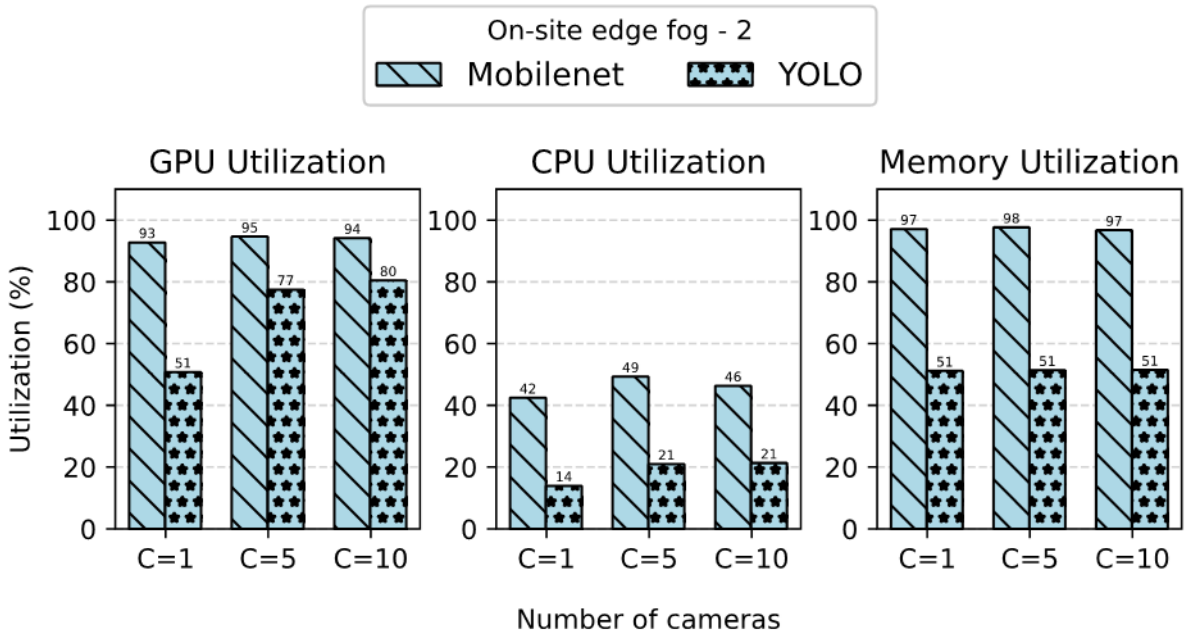

Figure 37. Resource utilization in the on-site 2 edge scenarios

Co-funded by the Horizon 2020
Framework Programme of the European Union

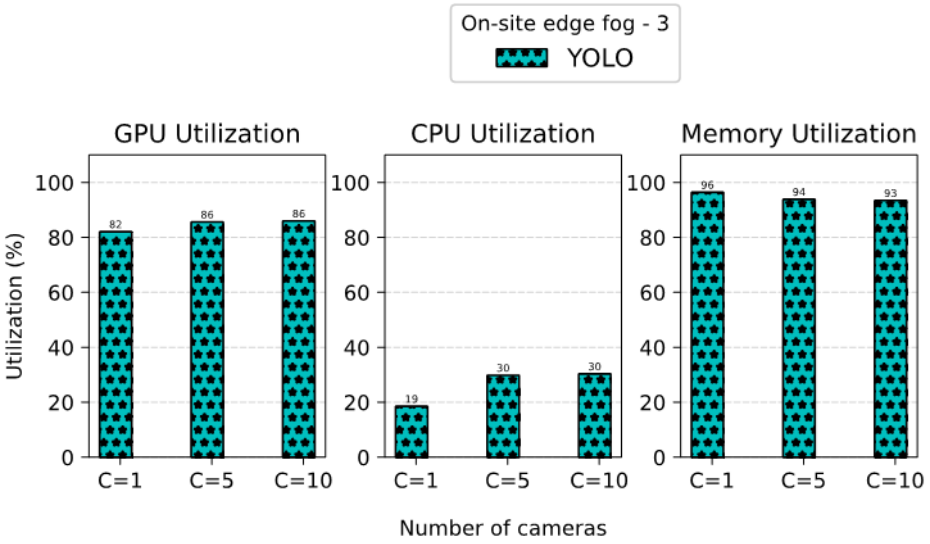

Figure 38. Resource utilization in the on-site 3 edge scenarios

To conclude the analysis of the on-site configurations in Figure 36, Figure 37 and Figure 38 the resource utilization analysis is reported. As it can be seen, Mobilenet results in a very high resource utilization, in terms of GPU, CPU and RAM. RAM, in particular, is at the limit of its availability, i.e., higher than 90% for on-site 2. If YOLO is considered, instead, we can notice a different resource utilization profile. Specifically, the RAM utilization is significantly lower and also the CPU utilization, for both on-site 1 and 2, while they are very high for the scenario on-site 3, which is at the limit of its resource capacity, especially in terms of memory. The lower CPU and RAM utilization for YOLO experienced in the scenarios on-site 1 and 2 is due to the reduced complexity of the model. If the GPU utilization obtained with YOLO is analysed a lower utilization can be noticed as well. This lower GPU utilization, compared with Mobilenet, is caused by the highest complexity for the execution of the operations required to manage the images, e.g., queue extraction, copy to GPU, etc, that require more time in an embedded system, thus resulting in an overhead per image analysed. This overhead is negligible with Mobilenet with YOLO, however, a higher analysed frame rate is reached, and more frames are analysed, i.e. approximately 4 times the number of frames analysed with Mobilenet. This higher number of frames analysed result in a higher overhead that results in a bottleneck for YOLO when executed in on site 1 and 2, thus refraining the system from reaching a higher framerate.

4.5 Discussion and Analysis on Results

A detailed discussion and analysis of the single results were already offered in Section 4.4, together with the presentation of the experiment results. In this section, an overview of the key insights provided by the results is offered, while a general overview on the conclusions that can be drawn from the experiments will be presented in Section 8.

Co-funded by the Horizon 2020 Framework Programme of the European Union

As already highlighted in Section 4.4.1.3, the initial implementation of the system, based on the Inception net inference model for image analysis, resulted in low performance in terms of overall analysed FPS, due to the complexity of the inference model. This is highlighted, in particular, in the scenario with the lowest load, i.e., the scenario with only one video source: even though the load is low, the system is not capable of analysing all the images that are received, consequently many images are dropped and the time to analyse an image from the same source increases. This issue is particularly clear in the edge scenarios where the time to analyse an image from the same source increases up to the order of tens of seconds, while it is mitigated in the cloud scenarios, where the time is in the order of seconds, thanks to the higher capability and the higher number of GPUs available. The low performance caused by the complexity of the model is confirmed by the analysis of the image inference delay and the analysis of the resource utilization. The latter, in particular, displayed a very high GPU utilization, while the CPU and RAM utilization was lower, thus confirming the GPU as the bottleneck of the system. The implementation of the other functionalities of the system, instead, i.e., social media monitoring and data aggregation, did not show criticalities, as it was also highlighted in the multi-edge scenario where multiple edge nodes were considered in the experiments.

This first set of results (Section 4.4.1) highlighted that the overall system can be considered suitable for the specific use-case, i.e., flash-flood monitoring, in both cloud and edge computing configurations. As expected, the cloud configuration ensures better scalability over the edge configuration, however, both of them can be considered acceptable for a real implementation as they ensure performance that are adequate for a real deployment in the considered use case. A more lightweight implementation that could guarantee a higher analysed framerate, however, could be considered in order to support a fine-grained analysis of the frames.

To this aim, two additional inference models have been considered to develop a lightweight implementation, namely Mobilenet and YOLO. The results presented in Section 4.4.2 highlighted that the two inference models result in a higher analysed FPS if compared with Inception net, thus ensuring on one side scalability w.r.t. the number of video sources, on the other, a lower time between the analysis of two subsequent frames from the same source. The latter, in particular, will allow a more fine-grained analysis of the images from the same source. The results presented in Section 4.4.2.2 also highlighted that the two lightweight models, YOLO in particular, are suitable for the implementation on embedded systems that could be installed directly on the camera site for the analysis of the images. Experimental results showed that such solution is suitable for at least one video source per edge node with a resulting analysed FPS that is comparable with the one obtained with Inception net deployed in a cloud environment.

To conclude, our results confirmed the possibility to adopt different edge computing configurations (near edge, far edge or on-site edge) for the implementation of our edge

Co-funded by the Horizon 2020 Framework Programme of the European Union

flooding monitoring system. The adoption of different inference models with different complexity resulted in different performances, especially in terms of time between the analysis of two subsequent frames from the same video source. Overall, the results showed that a complex model, like Inception net, results in a low overall analysed framerate, thus ensuring a coarse-grained analysis of the video feeds, while other more lightweight models, i.e., Mobilenet and YOLO, can ensure a higher framerate, thus enabling a fine-grained analysis. In addition to this, the experiments also highlighted the possibility to analyse the images directly on-site via the installation of embedded systems with GPU support. Such hardware is suitable for the implementation of both the Mobilenet and YOLO models. Our results showed that such they can fairly support the analysis of images from one video source.

4.6 Technical issues and modifications to the original project plan

During the implementation of EdgeFlooding and the execution of the experiments two major issues were experienced by UNIPI and UMBC. These two issues lead to the modification of the original project timeline, which however did not impact on the overall execution of the project, neither had impact on the final objectives and goals.

In the following we report a short summary of those issues, as motivation to the modifications applied to the execution plan:

- *Reduced availability on the Virtual Wall testbed*. As already highlighted in D2, a reduced availability of the nodes of the Virtual Wall testbed was experienced between M3 and M4. After discussing the issue with the NGIatlantic.eu team, a revision of the original plan was performed to run initially the experiments only with the resources that were available and to implement a second phase of experiments with also the nodes of Virtual Wall. The availability issue was resolved at M5, thus allowing a successful execution of all the experiments originally planned.
- *Grid'5000 instantiation issue.* At the beginning of M5, a bug in the reservation system of the Grid'5000 testbed (the system did not allow to instantiate a specific node with an Ubuntu image) delayed a bit the execution of some of the planned scenarios. Initially we contacted the Grid'5000 support to open a bug report on the issue. The bug was confirmed by the technical team that worked on a solution. In order to not delay further the experiments, we selected a different node that was not affected by the bug with similar features for the reminder of the experiments. Such issue, however, slightly delayed the timeline. For this reason, a one-month extension was asked and granted in order to complete both experimentation and the analysis of the results.

Co-funded by the Horizon 2020 Framework Programme of the European Union

5 Present and Foreseen TRL

The project EdgeFlooding produced two main outcomes: (1) a prototype of a cloud/edge computing platform for the detection of flash floods; (2) a set of guidelines on the performance of different image analysis algorithms in cloud/edge environments.

In the following we overview the readiness level of those two outcomes:

- The prototype of a cloud/edge computing platform for the detection of flash floods via image analysis is written in python and it is freely available on the Github repository of the project. The platform that started from a TRL 3, as it was an initial experimental prototype provided by UMBC, reached at the end of the EdgeFlooding project TRL 4, as it was validated on a realistic environment created via two Fed4Fire testbeds. The development and further studies on the platform are the focus of the collaboration between UNIPI and UMBC that will go beyond the end of this project. UNIPI and UMBC are planning to create a demonstrator that will exploit real video feeds from real cameras deployed on a real location to test the performance of the system on the field. This will help to improve further the TRL towards level 5.
- The results of the EdgeFlooding experiments, as it will be also highlighted in Section 8, provided a set of guidelines for systems using inference algorithms for image analysis at the edge. Such algorithms are often used in many other contexts to implement object detection and recognition functionalities. For this reason, the performance of such inference algorithms tested on different edge/cloud configurations can be used for the design of other systems in other areas in the future. Although the maturity of performance evaluation results and the resulting guidelines cannot be assessed via TRL, it is worth to highlight that the results of EdgeFlooding are valuable for the scientific community as they represent an exhaustive effort in analysing different inference models in a wide range of cloud/edge realistic configurations, which is, at the best of our knowledge, the first effort in literature.

Co-funded by the Horizon 2020 Framework Programme of the European Union

6 Exploitation, Dissemination and Communication Status

UMBC and UNIPI already submitted a workshop paper to IEEE SSC 2022, the 8th IEEE International Workshop on Sensors and Smart Cities, co-located with IEEE SMARTCOMP 2022, the 8th edition of the IEEE conference on Smart Computing. The paper currently under peer-review included an overview of the EdgeFlooding project and an initial analysis of the results obtained in this first phase of the experimentation.

In addition, UMBC and UNIPI are working on a joint journal publication to present the complete results of our experiments and the analysis presented in this deliverable. Currently this journal paper is planned to be submitted to IEEE Internet of Things journal.

In addition to this paper under preparation, we also plan to submit a demo proposal to IEEE SMARTCOMP 2022 to showcase the EdgeFlooding platform.

In order to present the outcomes of the project to the research community, UNIPI created a public web page: https://edgeflooding.github.io. The page currently contains information about the EdgeFlooding project and its objective and it will be updated with more details about the results of the experiments, as soon as the above mentioned dissemination activities are completed. Moreover, a public repository with the code of the distributed platform has been made publicly available: https://github.com/EdgeFlooding.

In addition to this, the EdgeFlooding project has been included to the list of the active projects of the Department of Information Engineering at UNIPI (https://www.dii.unipi.it/) and within the Cloud Computing, Big Data and Cybersecurity laboratory (https://crosslab.dii.unipi.it/lab-cloud-computing-big-data-cybersecurity). As part of the list of active projects, a short presentation of EdgeFlooding has been included on the presentations of the department and of the laboratory. Such presentations have been already delivered in the past weeks to multiple institutions and industries. Some industrial partners already demonstrated to be interested in potential collaborations with the department and the laboratory, as they showed interest in the activities of the project and in particular on the know-how related with image analysis techniques implemented at the Edge. Further collaboration opportunities will be investigated in the following months.

The team at UNIPI also created a M.S.c thesis opportunity to work on the project activities. One student of the M.S.c degree in Computer Engineering worked on the project and was involved in both the implementation of the platform and the execution of the experiments. The student will graduate at the end of April.

Co-funded by the Horizon 2020 Framework Programme of the European Union

7 Impacts

Impact 1: Enhanced EU – US cooperation in Next Generation Internet, including policy cooperation.

The EdgeFlooding project helped to establish a tight collaboration between UNIPI and UMBC. As part of the project activities, plenary videoconferences have been scheduled periodically (at least one per month) in order to exchange updates on project's activities and discuss future directions with all the project participants. During the first phase of the project, more frequent videoconferences have been performed in order to draw the experimentation plan (Task 0) and to get support from UMBC for the implementation work carried out by UNIPI (Task 1). Such periodic interactions and conference calls helped to establish a tight link between the two research groups, which already planned further activities, after the completion of the project and other collaboration opportunities on other fields.

Impact 2: Reinforced collaboration and increased synergies between the Next Generation Internet and the NSF program.

As part of the activities carried out at UNIPI, a M.S.c. thesis was kicked off from the project. One student was involved in all the project activities. Unfortunately, the pandemic situation and the extension of the state of emergency in Italy until March 31st, which significantly restricted travels for UNIPI personnel and the opportunity to host extra-EU visitors, did not allow the opportunity to implement visits or the exchange of researchers/students during the execution of the project. Both UNIPI and UMBC, however, are committed towards a long-term collaboration, which also involves exchange of students and researchers.

The work carried out by UMBC was funded with a previous grant from NSF obtained under the "Smart and Connected City" programme. UMBC will seek the opportunity to submit future project proposals to NSF calls to continue the research collaboration with UNIPI on the same or similar topics.

Impact 3: Developing interoperable solutions and joint demonstrators, contributions to standards.

As result of EdgeFlooding, a prototype of the flood monitoring platform has been finalized. The prototype also includes a lightweight version, optimized for the execution on constrained devices. As part of the project's dissemination activities, the platform code has been made publicly available on the github repository of the project. This open-source code is expected to be useful to other researchers working on the same area of distributed systems for monitoring based on image analysis and it could be used as starting point for the implementation of other similar platforms, not only for flash flood monitoring but also in other areas.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Impact 4: An EU - US ecosystem of top researchers, hi-tech start-ups / SMEs and Internet-related communities collaborating on the evolution of the Internet

The results of the EdgeFlooding experiments provided meaningful insights to assess the feasibility of adopting the edge computing approach for the implementation of real systems. The results of the performance evaluation carried out in this project have the potential to provide guidelines on the distributed edge computing approach not only applied to the area of flash flooding monitoring. Such guidelines could be extended to other environmental monitoring systems that exploit image analysis, considering in particular the results obtained with different inference algorithms characterized by different levels of complexity.

Co-funded by the Horizon 2020
Framework Programme of the European Union

8 Conclusion and Future Work

The EdgeFlooding experiments provided an extensive evaluation of a flash-flooding detection platform prototype based on image analysis considering different platform configurations and different cloud/edge deployment scenarios. The results of the experiments are valuable not only for the specific use-case considered, but they are also useful for the design and development of systems that adopt image analysis in cloud/edge platform. The experiments overall showed the feasibility of adopting different edge computing configurations for the implementation of the platform and hosting the image analysis process. The results, however, highlighted a noticeable difference with respect to the inference algorithm used and the cloud/edge configuration adopted in terms of analysed framerate and the time between the analysis of two different images from the same camera. In particular the following general guidelines, beyond the specific use case considered in EdgeFlooding, can be drawn:

- The Inception net inference model can be adopted even in near and far edge configurations, while its implementation is unfeasible on embedded systems. The overall resulting analysed framerate in edge configurations, however, is low and consequently the time between the analysis of two images from the same source is in the order of tens of seconds. This can be considered feasible for applications that only require a coarse-grained analysis of images and can tolerate longer times between the analysis of two subsequent frames.
- The Mobilenet inference model is more lightweight, and it can be implemented in all the configurations considered except only for the embedded system with the lowest RAM capacity, namely the NVIDIA Jetson Nano. This inference model guarantees higher framerates also in both near and far edge configurations with a time for the analysis of two consecutive frames from the same source in the order of seconds. This option can enable fine-grained applications that require continuous analysis of frames without significant delays.
- The YOLO inference model is the most lightweight and it can be implemented in all the considered scenarios. It is the one that results in the highest framerate, and it is suitable for the implementation also on embedded systems, the on-site scenario. The adoption of the on-site scenario, however, should be limited to the analysis of frames from only one video source, considering the limited capabilities of the embedded systems considered in the experiments.

To conclude, the experiments highlighted that the selection of the inference model and the configuration of the deployment should be performed based on the requirements of the application in terms of delay and image analysis granularity. The selection should be also performed in terms of the required accuracy provided by the inference model, aspect that, however, is strictly dependent on the specific use-case and it was considered out of scope for these experiments.

Co-funded by the Horizon 2020 Framework Programme of the European Union

As future work UNIPI and UMBC plan to carry out the following activities in the short-term future:

- Derive an analytical model using the queuing theory to model the delay and the framerate of the image analysis system deployed in a specific cloud/edge configuration. The model will be validated though the results obtained via the EdgeFlooding experiments. The objective is to create a mathematical framework to be used to dimension an image analysis system and select the most appropriate configuration.
- Create a demonstrator where the platform prototype developed during the EdgeFlooding activities is connected to real cameras deployed on a real flash-flood prone environment. The objective is to build a permanent testbed to showcase the outcome of the experiments via a demonstrator deployed in a real environment.

In the framework of a long-term collaboration between UNIPI and UMBC, the two research groups plan also to tackle additional research topics, still in the area of edge computing and image analysis for the creation of smart systems, thus leveraging the experience gathered during the execution of the EdgeFlooding project.

Co-funded by the Horizon 2020 Framework Programme of the European Union

9 References

[1] N. Singh, B. Basnyat, N. Roy and A. Gangopadhyay, "Flood Detection Framework Fusing The Physical Sensing & Social Sensing," 2020 IEEE International Conference on Smart Computing (SMARTCOMP), Bologna, Italy, 2020, pp. 374-379

[2] Inception convolutional neural network Wikipedia page: https://en.wikipedia.org/wiki/Inceptionv3

[3] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[4] YOLO – You Only Look Once version 5: https://github.com/ultralytics/yolov5

[5] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7310-7311).

10 Glossary

UNIPI	University of Pisa
UMBC	University of Maryland Baltimore County
WIT	Waterford Institute of Technology (Coordinating Partner)
KPI	Key Performance Indicator
GPU	Graphical Processing Unit
ISP	Internet Service Provider
FPS	Frames Per Second

Co-funded by the Horizon 2020 Framework Programme of the European Union

Deliverable 3: Part II

Financial and cost information

This part is to be treated as a consortium confidential deliverable, and access is restricted to consortium partners and EU commission operatives.

Co-funded by the Horizon 2020
Framework Programme of the European Union