Performance characterization of video analytics workloads in heterogeneous edge infrastructures

Daniel Rivas\textsuperscript{1,2} | Francesc Guim\textsuperscript{3} | Jordà Polo\textsuperscript{2} | David Carrera\textsuperscript{1,2}

\textsuperscript{1}Department of Computer Architecture, Universitat Politècnica de Catalunya, Barcelona, Spain
\textsuperscript{2}Data Centric Group, Barcelona Supercomputing Center, Barcelona, Spain
\textsuperscript{3}Data Center Group, Intel Corporation, Barcelona, Spain

Correspondence
Daniel Rivas, c Jordi Girona 1, 08034 Barcelona, Spain.
Email: daniel.rivas@bsc.es

Summary
Powered by deep learning, video analytic applications process millions of camera feeds in real-time to extract meaningful information from their surroundings. And this number grows by the minute. To avoid saturating the backhaul network and provide lower latencies, a distributed and heterogeneous edge cloud is postulated as a key enabler for widespread video analytics. This article provides a complete characterization of end-to-end video analytics across a set of hardware platforms and different neural network architectures. Each platform is selected to fill a different gap in a distributed, shared, and heterogeneous infrastructure. Moreover, we analyze how performance scales on each of these platforms with respect to the amount of resources dedicated to video analytics. Finally, we extract the key conclusions of the characterization to build an experimental model to estimate performance and cost of end-to-end video analytics in different edge scenarios. Our experiments show that managing video analytics workloads efficiently requires awareness of both, the platforms in which these are executed, and the full end-to-end pipeline. To the best of our knowledge, this is the first work that provides a complete characterization of end-to-end video analytics in heterogeneous edge platforms.

KEYWORDS
DNN, edge cloud, end-to-end video analytics, inference, video analytics, video decoding

1 | INTRODUCTION

Computer vision, thanks in part to the rapid advancements in deep learning, has boosted the use of cameras to automate the analysis and understanding of our surroundings. Video analytics is becoming increasingly popular, but its widespread adoption presents two major challenges: 1) the amount of data it generates, and 2) its computational complexity.

First, the amount of data that video analytic generates. An IP camera streaming a 1080p video generates around 0.5 MiB in 1 s; the same camera will generate 1.4 TiB of data over the span of 1 month. With the current state of the internet, where 70% of the network traffic corresponds to video,\textsuperscript{1} adding millions of cameras upstreaming content to data centers threatens to saturate backhaul networks. Therefore, the only foreseeable solution is for data to be processed close to where it is generated by means of deploying compute resources at the edge of the network. In response to this problem, the edge cloud has already been appointed as one of the main accelerators for video analytics.\textsuperscript{2} The edge cloud promises to alleviate pressure (and costs) of the infrastructure while, at the same time, offer improved latency and overall QoS.\textsuperscript{3} The edge cloud is defined as a decentralized and heterogeneous cloud where compute and storage is placed toward the edge of the network. Moreover, the further we move resources toward the edge of the network, the area of coverage becomes smaller and the requirements more specific. Therefore, the mix of hardware platforms to deploy can be tailored for each specific location.

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

© 2021 The Authors. Concurrency and Computation: Practice and Experience published by John Wiley & Sons Ltd.
Second, video analytics is computationally expensive. A common video analytics pipeline is composed of two main phases: video decoding and neural network (NN) inference. Traditionally, hardware platforms have been evaluated based exclusively on their performance for NN inference,\textsuperscript{4,5} while overlooking the impact of feeding the networks. It used to be a reasonable approach as only a few years ago, state-of-the-art accelerators were not able to execute state-of-the-art NNs at more than a few frames per second.\textsuperscript{6} The complexity and size of NNs have followed an upward tendency. However, NNs are not something as new or esoteric as they used to be and the tendency of improving accuracy by adding more layers is long gone. On the contrary, we now have a plethora of different NN architectures and optimization techniques at our disposal that widely expand the throughput-accuracy trade-off. For example, model distillation\textsuperscript{7} proposes to use large and accurate NN, such as ResNet\textsuperscript{8} or VGG,\textsuperscript{9} to teach much smaller NN models how to generalize. Moreover, model specialization trains small models that are only suited for specific tasks that can be assisted by large models in cases of uncertainty.\textsuperscript{4,10} Such techniques reduce, or even fully remove, the need to have complex generalist models in many scenarios, particularly in resource constrained environments, such as the edge cloud.

As NNs take over data centers, faster and more efficient hardware has been developed to accelerate NN execution. NVIDIA included tensor cores\textsuperscript{11} in its GPUs, and even launched an inference-optimized GPU; Intel extended its AVX-512 units with vector neural network instructions (VNNI) to accelerate common operations on convolutional neural network algorithm;\textsuperscript{12} and other low-power accelerators have been released, such as the Intel MyriadX\textsuperscript{13} or the Google Coral TPU. The tendency to speed up NN execution is clear.

Overall, the improvements in hardware plus newer and faster neural networks are shifting the bottleneck from the inference to the rest of the pipeline. This is specially true for video analytics, since these improvements have not been correlated with improvements in video decoding—a task that is also computationally expensive.\textsuperscript{14} This problem has been already highlighted in the literature,\textsuperscript{15} putting more resources into optimizing frame decoding to reduce overall time to solution of video queries. However, video queries assume bulk processing of videos, rather than frame by frame, and these methods rely on an initial exploration of the solution space to select the set of optimizations to apply depending on the time requirements of each query, which can take up to hours. Therefore, such methods have a different set of requirements and constraints than real-time video analytics in the edge, where latency deadlines are in the order of a few hundred milliseconds and the low level of user aggregation does not allow to consider any batch processing optimizations.

Processing video analytics on data centers in real-time can become cost prohibitive due to the amount of data it constantly generates and what that implies for the whole backhaul infrastructure. Therefore, the edge cloud must assist data centers if not completely overtake the task of processing. The heterogeneity of locations and requirements in edge deployments leads to one significant challenge: unlike traditional data centers, edge deployments are susceptible to be constrained due to nontechnical factors—other than budget—that are often linked to the geographical location itself. In some cases, edge nodes will be constrained due to limited connectivity—for example, only broadband or satellite connections available, limited power budget—for example, solar powered nodes, or even limited physical space—for example, a small fog node installed on a street light. In other cases, edge nodes are installed for one reason—that is, IoT gate—but can be re-purposed to accommodate video analytics—for example, an IoT gate with a low-power USB accelerator. The edge of the network is a broad term and in such heterogeneous landscape, we have to fully understand how each platform can best serve in each potential scenario. Consequently, the evaluation of edge platforms should not only consider quantitative metrics, such as performance or cost, but should also be considered within the context in which these platforms will potentially be deployed.

To address this gap, this article makes the following contributions: first, we propose a classification of hardware platforms based on their compute capabilities and the locations they are more likely to be deployed considering also power and size requirements. This classification is complemented with a classification of neural networks based on their size and expected use cases to solve. Second, we characterize six hardware platforms—including media and neural network accelerators—, each falling into a different edge use case. Third, we extract the main conclusions of the characterization and introduce them into an experimental model. We use the resulting model to estimate performance and cost while demonstrating two things: considering video decoding and inference together have a considerable impact on the performance projections, and how that inevitably impacts the design of the optimal infrastructure for a given deployment. Hence, the design of edge infrastructures requires more accurate performance models that consider heterogeneity of platforms to its full extend.

The article is organized as follows: in Section 2, we explore the state-of-the-art, in which we find motivation and justification to classify heterogeneous platforms and neural networks for video analytics. In Section 3, we describe the categorization of the edge cloud that we use to select the evaluation platforms. Similarly, in Section 4, we describe the categorization of the neural network architectures that we consider in our characterization. In Section 5, we describe the methodology followed throughout this article. In Section 6, we present and analyze the characterization of video analytics, whose conclusions are summarized and used to build an experimental model to explore performance and cost of different potential use cases. We revise the related work in Sections 7 and 8 concludes the article.

2 | BACKGROUND AND MOTIVATION

2.1 | Edge cloud

Edge computing is a distributed computing paradigm in which compute and storage resources are moved to the edge of the network, that is, close to the end-user. The definition of edge varies depending on the agent deploying the infrastructure. Telecommunication service providers (telecoms)
consider the edge to be a point close to the end-user but still under the telecoms’ infrastructure and control, while service providers refer to the edge as the different locations from which their services are served. Sometimes, certain IoT devices or other end user devices with limited compute capabilities are considered to be the edge of the network, but these have different design challenges. In this article we only consider devices that are part of the infrastructure of any telecom or enterprise, but not the end-user.

The edge cloud has been appointed as a key enabler of video analytics. In some cases, the edge is considered the only realistic approach to meet the latency requirements needed for real-time video analytics. Nevertheless, the edge cloud presents new challenges that will have to be addressed on new edge deployments. Specifically, we focus on the many possible geo-locations that make edge deployments subject to a series of limitations or constraints not shared with traditional centralized data centers. Moreover, different locations with different constraints and also requirements push the heterogeneity of the edge infrastructure. Street cabinets, remote locations powered by solar panels, or rural areas with only satellite connectivity are all constrained by different factors—that is, physical space, power budget, or thermals—but, at the same time, will also have a different level of user aggregation. Therefore, the orchestration layer must relate each location to its compute capacity on a specific workload in order to optimally allocate resources while meeting latency constraints. Moreover, in this article we assume a hierarchical edge architecture divided into multiple tiers where each tier increases capacity and aggregates services from the tiers below. Such architecture has proven to increase overall system’s efficiency while reducing the average response time.

### 2.2 Real-time video analytics

Real-time video analytics differs from other types of video analytics in that the results of the analysis are used for real-time decisions. This has two implications on how users and services have to be considered from the system orchestrator’s point of view. First, videos generated from cameras must be constantly analyzed, even if most of the contents are irrelevant. Second, latency and throughput are usually strict constraints attached to every use case, such as traffic control, security, or digital assistants.

However, other types of video analytics that continuously analyze images have been proposed as means of pre-indexing the contents of frames to speed-up future potential queries over certain footage. Although very similar, this type of analysis is more tolerant to latency and therefore give orchestrators more freedom to dynamically allocate resources depending on whatever is available at a certain moment.

#### 2.2.1 Convolutional neural networks

More specifically, convolutional neural networks (CNNs) are the de facto standard for image classification or object recognition tasks. This type of neural networks works by applying a series of transformations (called layers) to the input matrix—that is, usually a RGB matrix containing the image—and, as a result, it returns a vector of predictions, where each value is the probability that the image contains an object of a certain class. CNNs involve different types of layers but the most used ones are: convolutions (nonlinear transformations), activations (e.g., ReLU or Rectified Linear Uniform), MaxPooling (to reduce dimensionality), and a final Fully Connected layer that generates the vector of probabilities. Each layer might be repeated several times throughout the network. In general, the deeper (more convolutional layers) the neural network, the better at learning robust features that generalize well for new data. However, if the number of layers is not balanced with respect to the size and variety of the training dataset, there is a point at which the network will start overfitting, that is, learning features that work specifically well for the training data but not for new data.

State-of-the-art image classification models are often evaluated by their accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) or other similar challenges. In the case of ImageNet, models are trained using 1.2 million images belonging to 1000 classes. These models have to correctly identify objects as diverse cars, insects, or balls while still being able to differentiate between a green snake and a vine snake, for example. Nevertheless, this level of generalization is rarely required in edge deployments. A static traffic camera is probably never going to spot a vine snake and even if it does, it is most probably irrelevant to the use case. On the other hand, a static traffic camera is probably going to spot cars from just a few different perspectives. This observation allows us to exploit an NN optimization technique called model specialization, in which models are specialized on one or a small set of tasks instead of focusing on generic image classification. This way, we can safely exploit, among other techniques to reduce complexity, model compression without the fear of overfitting.

#### 2.2.2 Video transcoding

Video transcoding is the process of converting a video from one compression format into a different one. Video compression formats make use of spatiotemporal information to try to predict current frames based on previous frames. Video compression is assisted by several existing compression techniques, such as the discrete cosine transform (DCT) and motion compensation (MC). The common denominator of these techniques
is being computationally expensive tasks. However, any form of video distribution would be prohibitive without compression. A single uncompressed RGB frame at a resolution of 1920 × 1080 requires 5.93 MiB. The same space can fit around 30 s of video at 24 frames per second and same resolution.

In some cases, the result of video analytics generates a new stream. For example, an application that does object recognition may overlay bounding boxes around the detected objects, along the category of the identified object. This requires the newly generated images to be re-encoded in the form of a compressed video. However, in this article, we focus solely on the decoding for two reasons. First, the contents of a video cannot be analyzed without decoding while encoding is not always required. Secondly, by focusing on frame decoding, the resulting analysis is simplified and clearer.

3 | EDGE CLOUD ARCHITECTURE

Throughout this article, we consider three tiers of edge infrastructures whose characteristics are primarily determined by their proximity to the end-user: Data Center, Near Edge, and Far Edge. We also consider that their proximity to the end-user will determine, or influence at the very least, the physical locations where said infrastructures can be deployed. The physical location, however, determines a series of aspects that narrow down the number of possible hardware configurations. These categories, depicted in Figure 1, are defined as follows:

**Data Center**
- Centralized. Regularly, data centers are only limited by the budget and are usually sized to aggregate the maximum number of clients. These deployments may include any kind of accelerators (GPU, FPGAs, etc.).

**Near Edge**
- Deployed in street cabinets. Small form factor to 1U server nodes. Power might be limited to what a regular outlet can provide (1800 to 2400 W), while size is limited to what a street cabinet can fit, around one to five nodes. Air-cooled 10 gigabit Ethernet is common for these type of nodes. These deployments may include up to one low-profile PCIe accelerator.

**Far Edge**
- From IoT gates to a single desktop processor in a small form factor. Total power is limited to 50–100 W due to cooling and thermal limitations. These nodes are designed to be in the open and subject to solar radiation while working with no more than a small fan or even passive cooling in some cases. These nodes are often used as WiFi or Broadband access points. Therefore, their connectivity is limited, ranging from a few hundred megabytes wireless up to gigabit Ethernet. The range of accelerators that can be installed in these nodes is also limited to small low-power USB or M.2 accelerators. On the other hand, media acceleration on these platforms is common.

More precisely, these categories are defined by their proximity to the end-user along with the difficulties associated with such proximity. In other words, a room full of servers in the middle of a metropolis does not fit under the Far Edge definition only because it is close to the users nearby because it has unlimited compute and infrastructure resources.

Table 1 summarizes the technical considerations of the previous categorization. And based on these categories, Table 2 lists the particular set of hardware platforms evaluated in this article and its specification, while Table 3 lists the accelerators.

![Figure 1](image_url) Example of a 3-Tier edge architecture. From left to right, the Far Edge is composed by multiple and geographically distributed small nodes that are directly connected to where data is produced. Near Edge locations aggregate multiple Far Edge nodes and are only a few hops from them. In urban areas, Near Edge nodes can deployed at the edge of the backhaul network, which gives enough aggregation, space, and isolation to place a server infrastructure with multiple nodes. Finally, a traditional centralized Data Center with no theoretical upper limit on size or power but consequently limited to handful of locations per country or even continent, with the corresponding cost in latency.
TABLE 1  Technical specifications and restrictions for each infrastructure category

| Infrastructure | Form factor | Max. nodes | Power budget | Accelerator | Cooling |
|----------------|-------------|------------|--------------|-------------|---------|
| Data center    | Blade server | 10–100     | Any          | Any         | Any     |
| Near edge      | SFF/blade   | 5–10       | <1 KW        | Low profile PCIe | Air     |
| Far edge       | IoT gate    | 1–5        | <50 W        | SoC/USB/M.2  | Passive |

TABLE 2  Hardware platforms selected

| Processor                  | Cores | Frequency | Memory | Peak bandwidth | TDP   |
|---------------------------|-------|-----------|--------|----------------|-------|
| Data center               | Intel Xeon Gold 6248 (×2) | 20 (×2) | 2.5–3.9 GHz | 384 GB | 128 GB/s | 150 W (×2) |
| Data center (GPU)         | Intel Xeon Silver 4114    | 10 (×2) | 2.2–3 GHz   | 64 GB  | 96 GB/s  | 85 W (×2)  |
| Near edge                 | Intel Xeon D-2183IT       | 16      | 2.2–3 GHz   | 128 GB | 96 GB/s  | 100 W      |
| Far edge                  | Intel i7-8700T            | 6       | 2.4-4 GHz   | 16 GB  | 25 GB/s  | 35 W       |
| Fog IoT                   | Intel i5-8265U           | 2       | 1.6 GHz     | 8 GB   | 25 GB/s  | 10 W       |

TABLE 3  Media and DNN accelerators selected

| Device                  | Acceleration | TDP  | Interface |
|-------------------------|--------------|------|-----------|
| Intel HD 620            | Media        | 15 W | SoC       |
| NVIDIA GeForce RTX2080 | Media        | 215 W| PCIe      |
| Intel Myriad X          | DNN          | 4 W  | USB       |

4  HETEROGENEOUS MODELS FOR AN HETEROGENEOUS EDGE

For the analysis we select different state-of-the-art image classification CNN architectures that range in terms of complexity and size. These models can be divided in three different categories:

- **Reference models** — Big and complex models that achieve state-of-the-art accuracy. These models are good at generalizing.
- **Edge models** — Smaller models with lower accuracy initially oriented to run at real-time speed (i.e., 15–30 fps) on mobile devices with low compute power. These models are still relatively good generalizing while keeping their runtime low.
- **Specialized models** — When trained for a highly specific problem, compressed models have proven to achieve near reference accuracy while running orders of magnitude faster. These models are specialized to the objects and objects’ perspective of each deployment and do not generalize well to new scenarios.

Each category provides a different trade-off in terms of accuracy, runtime, and generalization. The justification for considering all three categories in our study is duly justified in previous work that shows how a combination of these categories can help to dramatically reduce runtime and costs while minimizing the accuracy loss.\(^4\text{,}^7\text{,}^10\text{,}^21\) In summary, previous work in the field of video analytics has focused on maximizing the accuracy of Edge and Specialized models while minimizing the number of times the reference models (also known as ground-truth models) are executed. Only in cases of uncertainty, the reference model is invoked. In the context of an heterogeneous edge, considering some or all of these categories is even more justified as each one may adapt better to a different platform within the edge infrastructure.

Table 4 provides more details about the models selected for evaluation on each of the three aforementioned categories. The selected models are a set of representative state-of-the-art models from the PyTorch Model Zoo.\(^22\) The models vary in complexity and size. As previously mentioned, several previous works have focused their efforts on reducing the number of times an expensive, generalist model is required. Some works focus on creating a classification cascade where only in cases of uncertainty the reference model is needed.\(^4\text{,}^21\) These works make use of highly inexpensive but also specific CNN. Other approaches propose to divide labeling in two steps: ingest time and query time, that is, the moment in which live video is captured versus the moment at which a certain footage is queried looking for certain objects in scene.\(^10\) This method executes a cheap NN during ingest time while more expensive NNs are reserved for when high accuracy is needed at query time. They compare the relative recall of ResNet18 and two compressed and specialized variants (with four and six layers removed and scaled to 112 and 56 pixels, respectively). The results of that work demonstrate how by considering the top-K instead of only top-1, they reach 99% relative recall with respect to their reference model (YOLOv2). For the specialized models, we consider ResNet16 and ResNet14, that is, a ResNet18 with two and four layers removed.
### TABLE 4 Models evaluated on each category

| Category | Model       | Complexity (GFLOPs) | Size (MParams) | Input    | Top-1      | Top-5      |
|----------|-------------|---------------------|----------------|----------|------------|------------|
| Reference| Faster R-CNN| 57.203              | 29.162         | 600 × 1024| 78.51%     | 94.45%     |
|          | ResNet152   | 22.709              | 66.746         | 224 × 224|            |            |
| Edge     | MobileNetv2  | 0.615               | 3.489          | 224 × 224| 71.22%     | 90.18%     |
|          | SqueezeNet1.1| 0.785               | 1.236          | 224 × 224| 58.3%      | 81%        |
| Specialized| ResNet16   | 0.702               | 112 × 112      | –        | –          | –          |
|          | ResNet14    | 0.124               | 56 × 56        | –        | –          | –          |

## 5 METHODOLOGY

### 5.1 End-to-end characterization

In this article, we aim to evaluate the different trade-offs present in continuous video analytics. As discussed in Section 2, the typical end-to-end video analytics pipeline consists of three distinct phases: frame decoding, frame preprocessing, and inference. We characterize each phase individually and then we show how considering the end-to-end pipeline impacts the selection of platforms in different environments.

#### 5.1.1 Video decoding

To evaluate decoding, there are multiple multimedia frameworks that are widely used and equally valid. Throughout this analysis, all decoding results were obtained using the GStreamer framework. GStreamer is a pipeline-based multimedia framework that allows to link a wide variety of media processing systems and build a highly customized workflow. OpenCV’s VideoCapture class allows us to select GStreamer as the media backend and then specify the pipeline according to the format and the accelerator used.

Regarding the video formats, we considered three of the most widely used: H.264, H.265, and VP9. VP9 is rarely used in live-streaming applications but it is widely used in different Video-On-Demand platforms, such as YouTube and is open-source and royalty-free. On the other hand, H.264 and its more efficient evolution H.265 are proprietary and their use requires a royalty. However, these two are almost the de facto standard when it comes to video live streaming and most devices worldwide (including IPTV cameras) include hardware support for H.264 and H.265.

Comparing video formats is not straightforward. There are many criterion for which video formats may be compared: image quality, compression ratio, speed, hardware support (or lack of). Moreover, encoders are configurable with multiple options to tweak all the aforementioned criterion. For this study, we wanted to focus the comparison on the performance for equivalent qualities, regardless of other encoding settings. Therefore, we have used a video downloaded from a video-on-demand platform and is preencoded in several formats and resolutions to match a wide range of user devices and are all expected to have, at least, similar perceived quality.

#### 5.1.2 Neural network inference

The inference of neural networks has been evaluated using the OpenVINO toolkit for the CPU and MyriadX executions and PyTorch for the NVIDIA GPU. There are multiple frameworks available that are able to do training and execution and each one provides a different set of functionalities and optimizations. OpenVINO only supports the execution and not training but it is the only framework that supports the Intel MyriadX accelerator. Nevertheless, in our evaluations we do not consider the training of the models and focus only on the inference. All the pretrained models used in this article are publicly available in the PyTorch Model Zoo and the OpenVINO Model Zoo repositories. We used single precision models for the CPU and GPU evaluations, and half precision for the Intel MyriadX.

### 5.2 Evaluation metrics

Each of the phases of video analytics are evaluated under the following metrics:

- Throughput, that is, frames per second.
Latency, that is, time to process a single frame or request. We always consider the end-to-end latency and not the average latency, that is, if a NN takes 2 ms to process a batch of two images, we consider latency to be 2 ms for each image and not the 1 ms average.

Power efficiency, that is, frames per second per unit of energy (Joules). For power consumption, we measure the average nominal power consumed during the execution interval. Then, we subtract the portion of the idle power that we believe should not count as part of the execution’s power. For example, in a system with an Intel Xeon 4114 with a RTX2080, the idle power drawn is 25 W for the CPU and DRAM, and 16 W for the GPU. If we are measuring decoding on a single CPU core, we fully subtract the idle power from the GPU plus 22.5 W from the CPU, as each one of the 10 cores adds 2.5 W. If we are measuring inference on the GPU while using a single core, 22.5 W are deducted from the CPU idle power while nothing from the GPU, as it is being fully utilized by the inference execution alone.

However, we do not consider the accuracy of the models nor the task for which they were trained for two reasons. First, the accuracy of a model highly depends on the training process and is extremely domain-dependent. Hence, we assume a model is selected upon its known accuracy and ability to generalize and we focus on the trade-offs that such model may expose in different hardware platforms. Second, the execution graph of CNNs is usually fixed. Thus, regardless of a model’s accuracy on a specific task, its throughput is determined by the type and number of layers (complexity) and the number of trained weights (size). Therefore, the conclusions of this study can be safely extended to equivalent CNN architectures trained on other tasks.

5.2.1 Environment, tools, and platform telemetry

All the platforms used for evaluation share the same setup: CentOS 7.8 (kernel version 5.4.15), with Hyper-Threading disabled and the intel_pstate CPU scaling governor, running Docker version 19.03.11. The platform with the NVIDIA GPU uses CUDA version 11.1 and driver version 455.23.05.

The telemetry of the hardware platforms was obtained using a custom Python package developed for this project. Through different interfaces, the package collects multiple metrics to provide a global view of the system: system utilization for CPU, memory, disks, network, and sensors; memory bandwidth and cache metrics through performance monitoring counters; package, core, and DRAM power consumption through running average power limit (RAPL) interface for desktop and mobile architectures and through IPMI and performance counters for server architectures; GPUs are monitored through queries to the nvidia-smi command-line tool, or RAPL for the integrated GPUs. The components have some overlap regarding the statistics they provide, as not all platforms support all interfaces. Therefore, for each platform we used a combination of components to ensure a complete analysis.

5.3 Reproducibility of the results

One of the outcomes of this study is that the optimal platform will highly depend on numerous factors. Among these factors, some will be subject to change while some others will be predetermined and fixed by the specifics of the deployment. Therefore, we wanted to make possible for anyone to reproduce this kind of study under different configurations.

In the public repository of this project, we provide all that is required to ensure the reproducibility of our experiments. The repository contains the scripts (for setup and execution) and the Dockerfiles to build the images we used for this article.

6 RESULTS

In this section, we present the characterization of video analytics and use the cornerstones of its results to build the experimental model that allows us to explore performance and cost trade-offs of different hardware configurations. The experiments are divided as follows:

1. Video decoding: characterization of the decoding phase. The analysis includes performance scalability, latency impact, and energy efficiency with respect to the resources allocated to the task, the encoding, and the resolution of the input video. Trade-offs of hardware media decoders are also explored.

2. Neural network execution: characterization of the inference phase. The analysis also includes performance scalability, latency impact, and energy efficiency with respect to the resources allocated to inference. Trade-offs of low-power (MyriadX) and high-end (GPU) neural network accelerators are also explored.

3. End-to-end performance projection. We build an experimental model from the findings of the previous characterizations to estimate throughput, latency, and power within different constrained scenarios. The experiments explore the trade-offs of the different hardware platforms and highlight how each one fits into a different context that the model can help identify.
6.1 | Frame decoding

When evaluating video analytics, neural network execution tends to get all the attention, and frame decoding is often overlooked. However, it is important that the execution of these two phases is balanced. In this section we start looking at frame decoding in isolation.

Figure 2 shows a general overview of the decoding performance and the power efficiency of the analyzed hardware platforms. Far Edge, Fog IoT, and Data Center with GPU platforms use the hardware decoder, while the Near Edge and Data Center platforms use only the CPU. From these results, we can extract two conclusions. First, the performance of each codec seems to highly depend on the actual implementation of the decoder. While CPU implementation of the H.264 decoder is 1.5× to 2× faster than its modern version H.265, this difference reverses on the NVIDIA GPU and vanishes on the Far Edge and Fog IoT, also using hardware decoder. On the other hand, VP9 seems to be consistently in between along platforms.

Second, we can observe how the more resource-constrained Fog IoT platform achieves the lowest performance but, at the same time, it is the most power efficient platform for decoding. This highlights one main differences among these platforms: user consolidation versus energy efficiency. Similarly, user aggregation is further emphasized in the server platforms. Near Edge and Data Center platforms are close in terms of both performance and efficiency; specially after adjusting to core count. However, it is important to notice that in order to keep the figure legibly and not distort the Y-Axis, these results correspond to a single socket of the Data Center platform, that is, only showing half of its peak performance, although efficiency should not vary. Moreover, we observe how the performance of the discrete GPU is on par to what the other platforms achieve. This parity will prove to be particularly important when jointly considering decoding and neural network inference.

6.1.1 | Scalability

Figure 3 shows how decoding performance scales with respect to the number of cores used for different input resolutions. The results correspond to single process executions in which a single input video is decoded as fast as possible. Results show a clear impact of the input resolution on the scalability. The higher the resolution, the higher the number of cores needed to achieve peak performance. While a single core suffices to decode a 144p video at full speed, 6 cores are needed for a 1080p video and 10 for a 4K video. After reaching peak performance, increasing the number of cores degrades performance. We have observed this to be consistent among the codecs we considered.

Figure 4 shows the decoding performance and the average latency per frame as the number of cores increases. In many cases, decoding performance will not become the major bottleneck. Video streams at 1080p and 30 or less frames per second are common in video analytics and the performance for a single core is more than capable of such frame rate. However, the number of cores used for decoding also impacts the latency per...
As the figure shows, average latency drops around 40% for both H.264 and H.265 when moving from one to two cores. Using eight cores for the decoding reduces the latency by an 80% with respect to the single-core execution. In some latency-critic applications, this may have an inevitably adverse effect on the end-to-end latency.

These results show that the decoding scalability benefits from more data. This is the classic problem of weak versus strong scaling. This particular implementation provides good scalability as the problem grows. However, for small data sets, that is, images with small resolution, the application is not capable to scale with the number of cores. Therefore, we repeated the same experiment but increasing the number of processes while the number of cores remains the same to see how scaling out affects the decoding performance. Figure 5 shows the results of this experiment. The figure shows the average throughput per process and the total throughput accumulated by all the processes decoding simultaneously. The figure also shows the average latency per process. We can observe how the total throughput of the platform scales linearly, in contrast to the scale-up results. By using the 20 cores in the machine divided in five processes, that is, four cores per process, instead of a single one, we achieve a 4x speedup. The average performance per process drops by a 15% and the average latency increases by the same number. However, it is important to recall that the maximum frequency when all cores are working is reduced by a 20%—and a 15% with respect to four cores.

6.1.2 Software versus hardware decoding

The following experiment compares performance, latency, and energy efficiency of the software decoder and the hardware decoder (i.e., present within the integrated graphics card). Hardware accelerators, as previously mentioned, usually provide higher efficiency and/or higher throughput but tend to require full utilization to achieve them. In this experiment, we wanted to test both scenarios: low and high load.

Figure 6 shows the efficiency—in terms of average millijoules per frame—and CPU utilization while increasing the number of input streams—each stream is H.264 1080 × 1920, @25 fps—decoded simultaneously and synchronized with the input frame rate. The load is increased from one to ten streams and, then, one execution with a single stream with asynchronous decoding, that is, no limit on frame rate, to compare their peak performance. The hardware decoder yields up to 3.7x higher efficiency than the CPU at the same throughput and 4.2x higher efficiency when the accelerator is fully utilized. Moreover, the hardware decoder frees the CPU for other uses (e.g., DNN execution) and, in the experiments, does not go beyond 8% CPU utilization. We can conclude that the hardware decoder, contrary to other hardware accelerators, is more efficient than its software counterpart regardless of the amount of work. However, the main drawback of the hardware decoder is its flexibility, as the formats and format options it supports are fixed and limited, which may make it unfeasible in some scenarios.
Figure 6 shows the throughput (in frames per second), the average latency per inference (in milliseconds), and the efficiency of the platform (in frames per watt). We divide the results into the three categories of models we described in Section 4. The first thing we notice is the clear difference among the three categories for all platforms and metrics. There is roughly one order of magnitude difference in performance for each category. With a closer look to the reference models, we observe that Faster R-CNN is itself one order of magnitude slower than ResNet152, achieving a maximum 15 fps on the GPU. On the other end, we observe how the GPU achieves more than 25,000 fps on ResNet14, the smallest of the evaluated models. This makes the GPU 4x faster than the rest of the platforms. However, this is a number that will be revised again in Section 6.3.

Regarding the CPU executions, the Intel Xeon 6248 on the Data Center platform is well above the rest on six five models, even after averaging per core. The Intel Xeon D-2183IT on the Near Edge platform comes behind, as expected, with lower memory bandwidth—4x versus 6x memory channels—and lower boost frequency—3 versus 2.2 GHz when utilizing all cores. On the other hand, the Intel i7-8700T on the Far Edge platform is relatively close to the Near Edge and this is due to one of the many trade-offs in edge deployments: user aggregation. The Far Edge’s higher frequency—up to 4 GHz—outweighs the bigger caches and higher memory bandwidth on the Near Edge platform server, as the performance of NN execution is tightly coupled to the core’s frequency. However, the 16 cores of the Xeon ensure greater scalability to aggregate more users, along with an on-par 10 Gbps Ethernet that desktop platforms lack. Finally, the so-called Fog IoT and the MyriadX accelerator—both considered low-power solutions—are left behind in terms of performance but on par between them, with the accelerator yielding slightly better throughput and efficiency. Nevertheless, the MyriadX accelerator has the advantage of freeing the CPU for other tasks.
6.2.1 | Scalability

Similar to other parallel tasks, the amount of resources put into neural network execution may scale in two directions: up—that is, more resources to process the same amount of work—or out—that is, more resources to process more work in parallel. In practice, scale-up translates to requests processed faster (lower latency) and scale-out translates to higher throughput. Figure 8 shows the speedup of the different neural networks run on each of the analyzed platforms with respect to the compute resources scaled in scale-out mode. The baseline corresponds to an execution using a single core. Solid lines represent the speedup as it was measured by the application. We can observe how the data center and near edge platforms start diverging from their ideal speedup as the number of cores increase. While this may seem like a different bottleneck is hit (e.g., memory), it is important to notice that data center platforms’ design is not driven by single-thread performance but by user aggregation and reliability. Therefore, due to thermal and power limitations, the maximum frequency at which a core is able to run is capped as more cores are used. For example, up to two cores in the data center platform are able to run at 3.9 GHz simultaneously but the maximum frequency is gradually reduced down to 3.2 GHz when all 20 physical cores are being used. The dotted lines in Figure 8 represent the speedup adjusted to the maximum frequency, that is, frequency with a single core. It can be observed how this adjustment brings speedup back to near-ideal speedup.

On the other hand, with scale-up mode we have the option to increase the number of cores used for each individual request. Figure 9 shows the speedup of the different models on each platform as the number of cores increases. Notice that in scale-up mode, the latency of each request is equal to the inverse of the throughput, as the benchmark serializes the work. We can observe how the throughput scales very differently for each model. By taking a new look at the characteristics of each model, we can see how the bigger a model is (in terms of complexity and number of parameters), the more it benefits from more parallelism.

6.2.2 | Accuracy

Depending on the use case, the accuracy of the model to deploy may be even more important than the number of inferences per second it can achieve on a specific hardware. As already mentioned, we do not consider accuracy as another variable on our search space as we assume it should be decided when deciding what and how. However, we would like to understand how it could impact throughput and latency. The experiment depicted in Figure 10 considers four models with the same backbone model (MobileNetV2) in four different flavors with different accuracies. The figure shows the number of inferences per second, the inference latency, and the power consumption of face detection executed on a Near Edge platform. The different data points correspond to executions with different number of threads per inference and different number of simultaneously executed inferences. The datapoints closer to the origin correspond to scale-up executions and the others correspond to scale-out executions. We can observe how each model has its clear distinct space in the grid. Accuracy—working as a proxy for complexity in this scenario—determines the scalability of the model. Moreover, accuracy expands or shrinks the area in which resources have to be allocated to meet constrained scenarios. The regression lines of each model further emphasize what the scalability experiments showed: scale-out increases performance linearly while scale-up reduces latency logistically.

---

**FIGURE 8** Throughput scalability with respect to the number of cores used for DNN execution on the different platforms
Throughput scalability with respect to the number of cores used for the execution of neural networks in scale-up mode, that is, no parallelization of requests.

![Throughput Scalability Graph](image)

Inferences per second (Y-Axis), inference latency (X-Axis) and power consumption (bubble size) of face detection model executed on an Intel Xeon D-2183IT. All four models are based on MobileNetV2 as their backbone. Models 1 to 3 use 2× SSD (Single Shot Detector) heads as detectors. The higher accuracy is achieved by increasing the input size: 3×256×256, 3×384×384, and 3×448×448 for 1, 2, and 3, respectively. Model 4 achieves higher accuracy by using a fully connected one-shot (FCOS) head, a more complex but more accurate detector. Model 4’s input size is 3×416×416.

![Inference Performance Graph](image)

| Model                  | AVX-512 (+VNNI) | AVX-512 | AVX2 |
|------------------------|-----------------|---------|------|
| YOLOv2                 | 235.8%          | 84.9%   | 66.6%|
| Faster R-CNN           | 178.9%          | 82.4%   | 55.0%|
| MobileNetv2 + SSD     | 149.9%          | 47.3%   | 9.2% |
| Average                | 188.2%          | 71.5%   | 43.6%|

Apart from the input size, quantization is another technique that aims to improve inference throughput. Quantization reduces the size of each weight in the model. Therefore, improving the model’s density (weights per bit), specially important with accelerators and vector units. If done properly, the drop in accuracy caused by the lower precision is minimal. For the following experiment, we used a different set of models available in the OpenVINO Model Zoo already quantized in INT8 precision. To quantify the speedup brought by the processor’s vector unit, and more interestingly, the new VNNI instruction set, we compare the performance gains on three of the platforms. It is important to recall that the VNNI instruction set only works with 8-bit integer operations. We use the performance of the same model with single precision (FP32) as baseline for each of the three CPU platforms and then compare the performance gains by moving to 8-bit weights. Table 5 summarizes the results. The Intel Xeon Gold 6248 on the Data Center platform (with support for VNNI) achieves 2–3× higher speedup than an Intel Xeon D-2183IT, both with AVX-512.

![Quantization Performance Table](image)
6.3 End-to-end performance projection

Previous results have shown some of the key differences between CPUs, GPUs, and the MyriadX accelerator, as well as how decoding and inference scale with respect to the available resources.

Decoding There are three factors to take into account when evaluating decoding: video resolution, video codec, and whether hardware acceleration is available. The first two factors are rarely decided at will and are commonly determined by the kind of cameras that are installed. The third one can, or should, be influenced in cases in which decoding may become a bottleneck. Results have shown how the scalability of the software decoder increases as the resolution increases. However, there is a point at which more cores only increase the overhead and the decoding speed goes down. This is more important than it may seem because, in many applications, the decoding and the inference phases are not decoupled and use the same cpus. A common pitfall is to increase number of cores hoping to increase inference speed while, in reality, the decoding is slowed down. This is specially important in a synchronous workflow, where the latency of each independent phase cannot be hidden by the previous phase and the end-to-end latency increases proportionally.

Inference CPUs are able to operate in two different modes: latency mode or scale-up (increasing the number of threads per inference) and throughput mode or scale-out (increasing the number of inferences executed simultaneously). In throughput mode, performance (inferences per second) increases linearly with respect to the number of cores at the expenses of higher latency. Higher latency is caused by frequency capping, which increases as more cores are used. However, performance scales linearly with respect to the number of cores for all reviewed models. After removing the effects of the reduced frequency, scaling is close to ideal in most cases. On the other hand, latency mode serializes requests to reduce the latency. However, the degree at which latency can be reduced will depend on the size of the model, as smaller models will see no benefit from latency mode. At the same time, the size of the model has an inverse effect running in throughput mode and bigger models will see an increment in latency as more requests are executed in parallel. Moreover, inference performance does not depend on the contents of the frame, and so when a neural network has been characterized in a platform, the performance is known (considering caches are already warm), and the latency of a given request is likewise invariant. Hence performance and latency are affected by a set of deterministic factors, such as core frequency or number of cores available.

Finally, we combine both video decoding and execution of neural networks and consider the end-to-end pipeline to see how performance and cost per frame are impacted. Figure 11 shows an estimation of throughput and cost for the different platforms evaluated in this article considering a FullHD video stream encoded in H.264. We focus on a single model for each of the three model categories used during the characterization. ResNet152 as the reference model, MobileNetV2 as the edge model, and a ResNet14 (ResNet18 with four layers removed) with its input size downscaled to $56 \times 56$. Then, we limit the maximum latency allowed for each configuration to be considered. The throughput (frames per second) corresponds to configurations that minimize the cost (watts per frame) and the cost corresponds to configurations that maximize throughput. As we relax the latency constraint, performance increases and cost reduces, especially for the ResNet152. We observe two major points. On the one hand, the execution of large models does require the use of a GPU, unless latency constraints are largely relaxed. On the other hand, we observe how in the case of small models, the GPU is relatively close to the CPU platforms, as decoding becomes the major bottleneck. Moreover, when considering the cost (in this case, the inverse of efficiency), resource constrained platforms even with low-power accelerators such as the MyriadX, are close to what throughput-oriented accelerators provide.
7 | RELATED WORK

Video analytic systems, while not specific to the edge, are key to achieve an efficient and cost-effective widespread adoption. In this article, we base most of our assumptions and considerations on top of what some of these systems have previously proposed, as described in Section 2. However, these systems focus on speeding up video querying and not on real-time video analytics.

With respect to the characterization and evaluation of video analytics, most of the work is focused on the evaluation of neural networks, while other phases of video analytics have been mostly overlooked. Authors in Reference 26 provide an extensive analysis of the speed/accuracy trade-offs for several modern convolutional object detectors, whose goal is to serve as a guide for selecting a detection architecture depending on whether accuracy or speed is favored over the other. A similar but more experimental analysis is presented in Reference 5, where authors provide an extensive benchmark of the most representative deep neural network architectures for image classification and highlight the relationship between complexity and the size of a model with its speed. Execution of neural network, including CNNs, has been extensively characterized on both ends of the network. On datacenters, Researchers at Google27 characterized three different datacenter platforms—Google’s tensor processing unit (TPU), a NVIDIA K80 and a Haswell server-CPU—executing a mix of representative neural networks and showing how a specialized architecture can deliver better performance at lower cost. On the other end, resource usage on edge devices, including low-power accelerators such as Intel Myriad X, Google Coral, has been characterized by authors in Reference 28. However, to the best of our knowledge, none of these works have aimed to provide a characterization of end-to-end video analytics workloads for heterogeneous platforms.

With respect to previous evaluations, we add two ideas: (1) we evaluate a range of CNNs, not only those with state-of-the-art accuracy and (2) we analyze how these CNNs perform under different hardware, ranging from traditional data center platforms with power-hungry graphic accelerators all the way down to fanless CPU nodes with USB low-power NN acceleration. By combining these two into a projection model, we can relate each platform to the type of model for which it is best suited.

8 | CONCLUSIONS

We have characterized end-to-end real-time video analytics and analyzed video decoding and the execution of neural networks separately. Results have shown that decoding and inference are subject to different considerations and demonstrated the implications of evaluating platforms solely by their inference performance, since both—decoding and inference—can become the bottleneck, depending on the platform and the use case. We achieved this by evaluating an heterogeneous mix of hardware platforms, including two types of accelerators (a throughput-oriented NVIDIA GTX2080 and low-power Intel MyriadX) and showing how each one these platforms is best suited for a different deployment. Moreover, the evaluation of neural networks was divided into three categories, namely reference, edge, and specialized. This allowed us to highlight the strengths and weaknesses of each platform for different types of deployments and show where each platform can be best deployed. Finally, we have introduced the key outcomes of the characterization into a projection model to estimate end-to-end performance, latency, and cost for video analytics. Thanks to the model, we were able to quantify the impact of considering the full end-to-end pipeline of video analytics and analyze platforms in terms of throughput and cost on different deployment scenarios. This model is the first step toward an analytical model to help us design and optimize heterogeneous edge cloud infrastructures.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Economy of Spain under contract TIN2015-65316-P and the Ministry of Science under contract PID2019-107255GB-C21/AEI/10.13039/501100011033; the Generalitat de Catalunya under contract 2014SGR1051; the ICREA Academia program; and Intel Corporation.

ORCID

Daniel Rivas https://orcid.org/0000-0003-0163-7890
Francesc Guim https://orcid.org/0000-0003-4738-5473
Jordà Polo https://orcid.org/0000-0001-5422-7890
David Carrera https://orcid.org/0000-0003-4898-3424

REFERENCES

1. Cisco, Cisco visual networking index: forecast and trends, 2017-2022, 2017. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html. Accessed October 22, 2019.
2. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. IEEE IoT J. 2016;3(5):637-646.
3. Bilal K, Erbad A. Edge computing for interactive media and video streaming, 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC). Valencia, Spain; 2017:68-73. https://doi.org/10.1109/FMEC.2017.7946410.
4. Kang D, Emmons J, Abuzaid F, Bailis P, Zaharia M. Noscope: optimizing neural network queries over video at scale; 2017; arXiv:1703.02529.
5. Bianco S, Cadene R, Celona L, Napoletano P. Benchmark analysis of representative deep neural network architectures. IEEE Access. 2018;6:64270-64277.

6. Redmon J, Farhadi A. YOLO9000: Better, faster, stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI; 2017:6517-6525. https://doi.org/10.1109/CVPR.2017.690.

7. Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network; 2015. arXiv:1503.02531.

8. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV; 2016:770-778. https://doi.org/10.1109/CVPR.2016.90.

9. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. Paper presented at the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA; 2015. arXiv:1409.1556:7-9.

10. Hsieh K, Ananthanarayanan G, Bodik P, et al. Focus: querying large video datasets with low latency and low cost. Paper presented at: Proceedings of the 13th (USENIX) Symposium on Operating Systems Design and Implementation (OSDI) 18. Carlsbad, CA; 2018:269-286.

11. NVIDIA. NVIDIA tesla V100 GPU architecture; August 2017. https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf. Accessed August 07, 2020.

12. Intel. Intel deep learning boost on second generation Intel Xeon scalable processors; November 2019. https://software.intel.com/content/www/us/en/develop/articles/introduction-to-intel-deep-learning-boost-on-second-generation-intel-xeon-scalable.html. Accessed August 07, 2020.

13. Intel. Intel movidius myriad X vision processing unit. https://www.intel.com/content/www/us/en/products/processors/movidius-vpu/movidius-myriad-x.html. Accessed August 07, 2020.

14. Vetro A, Christopoulos C, Sun H. Video transcoding architectures and techniques: an overview. IEEE Signal Process Mag. 2003;20(2):18-29.

15. Kang D, Mathur A, Veeramacheni T, Ballis P, Zaharia M. Jointly optimizing preprocessing and inference for DNN-based visual analytics; 2020. arXiv:2007.13005.

16. Ananthanarayanan G, Bahl P, Bodik P, et al. Real-time video analytics: the killer app for edge computing. Computer. 2017;50(10):58-67.

17. Kang D, Ballis P, Zaharia M. Challenges and opportunities in DNN-based video analytics: a demonstration of the Blazelet video query engine. Innovative Data Systems Research (CIDR). Asilomar, California, January 13-16, 2019.

18. Tao X, Ota K, Dong M, Qi H, Li K. Performance guaranteed computation offloading for mobile-edge cloud computing. IEEE Wirel Commun Lett. 2017;6(6):774-777.

19. Tong L, Li Y, Gao W. A hierarchical edge cloud architecture for mobile computing. IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications. San Francisco, CA; 2016:1-9. https://doi.org/10.1109/INFOCOM.2016.7524340.

20. Russakovsky O, Deng J, Hao S, et al. Imagenet large scale visual recognition challenge. Int J Comput Vis. 2015;115(3):211-252.

21. Anderson MR, Cafarella M, Ros G, Wenisch TF. Physical Representation-Based Predicate Optimization for a Visual Analytics Database, 2019 IEEE 35th International Conference on Data Engineering (ICDE). Macao, China; 2019:1466-1477. https://doi.org/10.1109/ICDE.2019.00132.

22. PyTorch. PyTorch model zoo; https://pytorch.org/docs/stable/torchvision/models.html. Accessed October 15, 2020.

23. YouTube. Costa Rica in 4K 60fps HDR (Ultra HD). https://www.youtube.com/watch?v=LXb3EKWsInQ. Accessed August 19, 2020.

24. OpenVINO. Open model zoo repository; https://github.com/openvino-toolkit/open_model_zoo. Accessed August 15, 2020.

25. Tools for platform characterization. https://github.com/HiEST/py-sysmon. Accessed September 25, 2020.

26. Huang J, Rathod V, Sun C, et al. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI; 2017:3296-3297. https://doi.org/10.1109/CVPR.2017.351.

27. Jouppi NP, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit. Paper presented at: Proceedings of the 44th Annual International Symposium on Computer Architecture. Toronto, ON, Canada; 2017:1-12.

28. Antonini M, Vu TH, Min C, Montanari A, Mathur A, Kawsar F. Resource characterisation of personal-scale sensing models on edge accelerators. Paper presented at: Proceedings of the 1st International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things. New York, NY; 2019:49-55.

How to cite this article: Rivas D, Guim F, Polo J, Carrera D. Performance characterization of video analytics workloads in heterogeneous edge infrastructures. Concurrency Computat Pract Exper. 2023;35(14):e6317. https://doi.org/10.1002/cpe.6317