# EFFICIENT RECONFIGURABLE CPS FOR MONITORING THE ELDERLY AT HOME VIA DEEP LEARNING

**Daniel Deniz**
Computer Architecture and Technology
CITIC, University of Granada
Granada, Spain
danideniz@ugr.es

**Juan Isern**
Computer Architecture and Technology
CITIC, University of Granada
Granada, Spain
jisern@ugr.es

**Jan Solanti**
Tampere University
Tampere, Finland
jan.solanti@tuni.fi

**Pekka Jääskeläinen**
Tampere University
Tampere, Finland
pekka.jaaskelainen@tuni.fi

**Petr Hnětynka**
Charles University
Prague, Czech Republic
petr.hnetynka@d3s.mff.cuni.cz

**Lubomír Bulej**
Charles University
Prague, Czech Republic
lubomir.bulej@d3s.mff.cuni.cz

**Eduardo Ros**
Computer Architecture and Technology
CITIC, University of Granada
Granada, Spain
eros@ugr.es

**Francisco Barranco**
Computer Architecture and Technology
CITIC, University of Granada
Granada, Spain
fbarranco@ugr.es

## ABSTRACT

The rapid growth of the aging population poses serious challenges for our society e.g. the increase of the health care costs. Fueled by the information and communication technologies (ICT), Smart-Health Cyber-Physical Systems (CPS) offer innovative solutions to ease this burden. This work proposes a general framework for CPS-based solutions that adapt in run-time to optimize the overall performance of the system, continuously monitoring system working qualities such as bandwidth utilization, response time, accuracy, or energy consumption. Adaptation is done through automatic reconfiguration of processing Deep Learning models deployed on local edges. Local processing is performed in embedded devices that provide short latency and real-time processing despite their limited computation capacity compared to high-end cloud servers. Furthermore, a virtualization platform allows the system to offload any occasional intensive computation to such shared high-end servers. This paper demonstrates this reconfigurable CPS in a challenging scenario: indoor ambient assisted living for the elderly. Our system collects lifestyle user data in a non-invasive manner to promote healthy habits and triggers alarms in case of emergency to foster autonomy. Local edge video processing nodes identify indoor activities powered by state-of-the-art deep-learning action recognition models. The optimized embedded nodes allow the system to locally reduce cost and power consumption but the systems needs to maximize the overall performance in a changing environment. To that end, our solution enables run-time reconfiguration to adapt in terms of functionality or resource availability. The experimental section shows a real-world setup performing run-time adaptation with different reconfiguration policies. For that example, run-time adaptation extends the working time in more than 60% or increases critical action recognition accuracy up to 3x.

***K*eywords** Cyberphysical systems · Indoor monitoring · Machine Learning · Run-time adaptation

## 1 Introduction

Life expectancy has significantly increased along the last century causing a dramatic world population ageing. By 2050, one in six people are expected to be over 65 [1]. This fact poses major social and economic challenges due to the resources needed to assist the elderly (or dependent people) that live in nursing homes or alone in their own homes [2]. Thus, one of the direct effects of ageing is the increase in demand for systems able to monitor them [3] and improve their quality of life.

Smart health (S-health) systems bring the information and communication technologies (ICT) to the medical ecosystem, including data management, wearable devices, or the Internet of the Things (IoT). S-health alleviates health-care costs and provides new convenient solutions [4, 5]. For example, systems that enable remote real-time health monitoring for early detection of clinical deterioration [6], systems that give body posture recommendations while sitting on a chair for long periods of time [7], or as in our case, solutions for supporting the elderly independence offering safety guarantees with remote monitoring through Ambient Assisted Living (AAL) [8, 9]. Within AAL, of particular interest are systems to monitor lifestyle [10], to detect and prevent accidents such as falls [11] or to provide rapid emergency response [12]. A gap analysis of current monitoring systems leads us to identify the most relevant issues: 1) data privacy protection, essential in the Health field; 2) high energy consumption due to continuous operation that require very frequent battery replacements in stand-alone systems; 3) triggering of false alarms, reducing the trust on the system itself and increasing economic cost. In this work, these issues are addressed via edge local processing and automatic system reconfiguration for the edge-cloud system. First, local processing avoids transmitting data out the network, inherently protecting sensitive information. The automatic reconfiguration targets the optimization of resource usage or energy consumption, and the reduction of false alarms specially for critical actions.

The rapid evolution of networks, Artificial Intelligence (AI), and Systems-on-a-Chip embedded devices has greatly contributed to the development of s-health Cyber-Physical Systems (CPS) [13, 7]. CPS facilitate the integration of the physical world with software and network components that interact with humans [14, 15]. The processing nodes are autonomous embedded devices that are tightly connected with physical systems and their environments [16]. In the s-health domain, CPS provide the following advantages: scalability, supporting the deployment of systems with dozens or hundreds of nodes; adaptation to the environment, resource availability, or user roles; optimal performance through adaptation, offering an intelligent response to the changes in the real-world environments [17].

Processing nodes are stand-alone embedded components that independently implement design-time methods. The collaboration of the different processing nodes in distributed CPS optimize the overall system performance e.g. accuracy or energy consumption. However, real-world environments are continuously changing, especially when humans are in the loop. At run time, the system architecture must adapt to these changes to maintain or improve the overall system performance and efficiency [18, 19]. Conceptually, run-time adaptation is an instance of a MAPE-k loop — a generalized feedback control loop for adaptive systems [20], which consists of several phases (Monitoring, Analysis, Planning and Execution) operating with shared Knowledge. Adaptation is typically achieved by reconfiguration of the system in the Execution phase in response to information collected in the Monitoring phase. While conceptually simple, making a system adaptive has non-trivial impact on system architecture and requires a system-specific implementation of each of the phases.

Quality and Resource Management (QRM) plays a crucial role, because system configuration has many degrees of freedom and finding an optimal configuration for a given task and circumstances becomes a non-trivial multi-objective optimization problem [21]. QRM tools therefore provide solutions in form of (optimal) configurations for different situations which require making trade-offs between bandwidth utilization, response time, accuracy, and energy consumption [22]. Run-time adaptation is then addressed through reconfiguration policies [23, 24], establishing operating modes designed for achieving specific performance levels and resource utilization [25].

An important aspect of adaptive CPS, and especially those with humans in the loop, is to make their operation observable to ensure that they operate (and make decisions) as intended. During operation, these systems produce huge amounts of data which need to be presented properly to make it suitable for human consumption. To handle reconfiguration and data visualization, our CPS uses FIVIS, a system based on the IVIS integrated framework for storage, analysis and visualization of system monitoring data [26].

Video processing uses state-of-the-art Deep Learning (DL) architectures [27] that are optimized for deployment on our local edge nodes: Jetson Nano [28] devices. These embedded devices automatically perform action recognition and are able to autonomously reconfigure themselves. Our optimized solutions for these battery-powered stand-alone edge nodes achieve real-time performance within low energy budgets. Despite using state-of-the-art technology, these

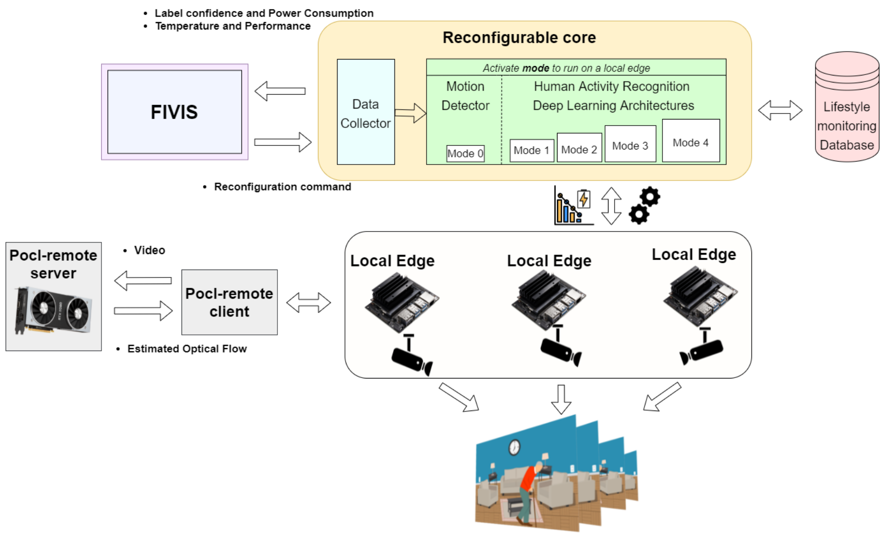

• Label confidence and Power Consumption
• Temperature and Performance
Reconfigurable core
FIVIS
Data Collector
Activate mode to run on a local edge
Motion Detector
Human Activity Recognition Deep Learning Architectures
Mode 0
Mode 1
Mode 2
Mode 3
Mode 4
Lifestyle monitoring Database
• Reconfiguration command
Pocl-remote server
• Video
Pocl-remote client
• Estimated Optical Flow
Local Edge
Local Edge
Local Edge

Figure 1: Reconfigurable CPS architecture: Local edge nodes perform indoor action monitoring from video batches of 64 frames at 25 fps (frames per second). The reconfigurable core retrieves action labels and quality data from the embedded devices. Quality data are remotely monitored by the QRM tool FIVIS. In our work, FIVIS analyzes the data and makes decisions about run-time reconfiguration to optimize the overall system working time (bearing in mind that edge nodes are battery-powered) and the action recognition accuracy. Finally, PoCL-R is used to seamlessly offload execution to a high-end server to e.g. increase the overall accuracy when identifying critical actions while avoiding additional energy consumption.

processing (camera) nodes have limited resources. This makes them impractically slow for some of the more demanding analysis tasks. Therefore, these tasks need to be performed only occasionally and their execution can be offloaded to a shared on-site high-end compute server equipped with a powerful GPU (see Figure 1). Our CPS uses the **PoCL-Remote** (*PoCL-R*) framework [29] for this: *PoCL-R* is a distributed execution driver for heterogeneous architectures implemented in OpenCL.

This work presents an efficient reconfigurable CPS for lifestyle monitoring system on low-end devices, integrating a remote monitoring and QRM tool, as well as a remote resource virtualization platform. Next, the main contributions are detailed: 1) the design and development of the HAR module based on DL models for indoor lifestyle monitoring; 2) the optimization and integration of action recognition on embedded devices, achieving real-time performance and enabling low-energy processing, while inherently preserving privacy by processing images at the edge; 3) the integration with an online monitoring and QRM tool (FIVIS) that enables system reconfiguration, by adapting the system at run time (extending the working time while preserving recognition accuracy); 4) the use of PoCL-R in the local nodes, a virtualization tool that helps the system offload demanding computation to a high-end cloud server; 5) the release of the Combined Indoor Action Video (CIAV) dataset, an indoor video dataset for performing lifestyle monitoring (publicly available).

## 2 Related Methods

In this section, we present the state-of-the-art of the main components that are part of our system: our video-based HAR to classify actions; remote virtualization of resources; quality and resource monitoring and management.

### 2.1 Human Activity Recognition

Human Activity Recognition (HAR) systems interpret and recognize actions performed by subjects [30, 31]. Interesting examples of these actions for our case are e.g. walking, personal hygiene, having lunch, or watching TV [32]. Actions recognition is done in the literature using: a) wearable-sensor-based HAR [33] and b) vision-based HAR [34]. Wearable sensors facilitate HAR mainly through collection and analysis of data from inertial sensors. In contrast, video-based methods only use RGB or RGB-Depth cameras. The main advantage of the video-based methods is that they are less intrusive and easier to adopt, since the elderly are sometimes reluctant to wear any smart devices.

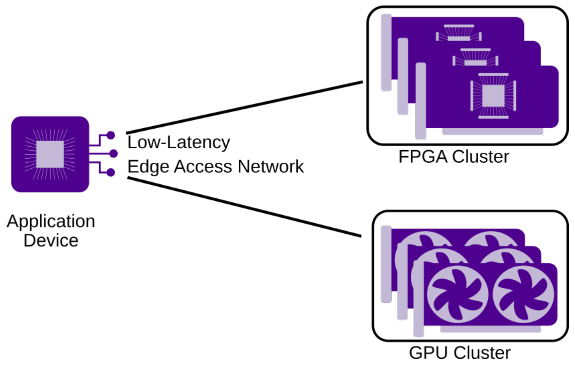


Figure 2: High level overview of PoCL-R

Regarding video-based methods, smart AI-powered solutions (specifically Deep Learning - DL) for action recognition offer state-of-the-art performance compared to solutions that use hand-crafted features [35]. In recent years, Convolutional Neural Networks (CNNs) have boosted action recognition performance [27]. For example, [36] introduced the *3D Conv-Net* that uses a deep neural network with 3D convolutions to extract spatio-temporal features from videos. While this network outperforms the alternatives using 2D convolutions, it requires large training datasets due to its complexity. In [37], authors use a *Recurrent Convolutional Network (RCN)*, using 2D-convolution layers to extract the spatial information from a sequence of frames while the RNN units learn temporal patterns. In addition, [27] introduced the *Two Stream I3D* that combines streams with RGB and motion cues. Motion estimation uses Dense Optical Flow TV-L1 [38], a computationally demanding method. Finally, the outputs of both streams are combined by adding their *logits* (raw un-normalized predictions) values and applying a *softmax* function to obtain the final action confidence. This HAR method achieves the best results in terms of accuracy at the expense of significantly more resources.

Another key aspect is the preservation of the subject's privacy, usually ensured by adding encryption layers [39]. However, in our case, edge processing inherently provides a way of protecting sensitive data by processing video frames locally, without transmitting any raw data out to the network.

### 2.2 Distributed Heterogeneous Execution Offloading

To keep the cost of the system low, it is useful to offload heavy compute workloads to a shared on-site server while only performing lighter computation on the edge nodes. Within this work, the virtualization platform for offloading is provided by *PoCL-R* [29], which is a distributed execution driver built within *PoCL* [40], an open-source framework implementing the *OpenCL* API [41].

A high level overview of this setup is shown in Fig 2: Remote servers that contain powerful GPUs or other OpenCL-supported compute devices run a special daemon process that the *PoCL-R* client-side driver connects to. The daemon reports available compute devices to the remote driver which in turn exposes them to the application in the same way as local devices on the machine running the application, making it easy to switch between local and remote devices in the same OpenCL platform.

### 2.3 Quality and Resource Management

QRM plays an extremely important role in CPS because system configuration has many degrees of freedom and finding an optimal one is a difficult problem. This is a live area of research, with many different approaches to dynamic reconfiguration and self-adaptation in CPS, but there is a lot of fragmentation and no comprehensive tool-set [42].

On the pragmatic side, solutions used in production are mostly focused on monitoring and visualization, with automation limited to rule-based actions. An example of such a solution is Grafana [43], an open-source framework for visualization of monitoring data from CPS and IoT systems. Grafana provides a multitude of attractive visualizations to make it easier for people to understand the data. Similar projects include Kibana [44], a visualization dashboard for ElasticSearch, or Chronograph [45], a visualization framework for the InfluxDB time-series database. A more complete list of such tools can be found in [46].

These tools are mostly focused on rapid construction of simple monitoring dashboards using predefined visualization widgets and common charts such as histograms, line graphs, pie charts, etc. They make it extremely easy for non-experts to visualize raw data in static dashboards. At the same time, though, their focus on simplicity and user-friendly UI makes them less suitable for creating customized, expert-grade visualizations that need to react on the data (e.g. creating charts with dynamic limits, dashboards with conditional layouts), or actual management of the monitored system.

For this reason, our QRM tool of choice is FIVIS, which provides the flexibility needed for building a customized QRM solution. FIVIS is developed using the IVIS framework [26], an extensible platform for storage, visualization, and analysis. As shown in Figure 3, the architecture of FIVIS is split between the back-end (the upper part of the figure) running on a server and the front-end (the lower part of the figure) running in a web browser.

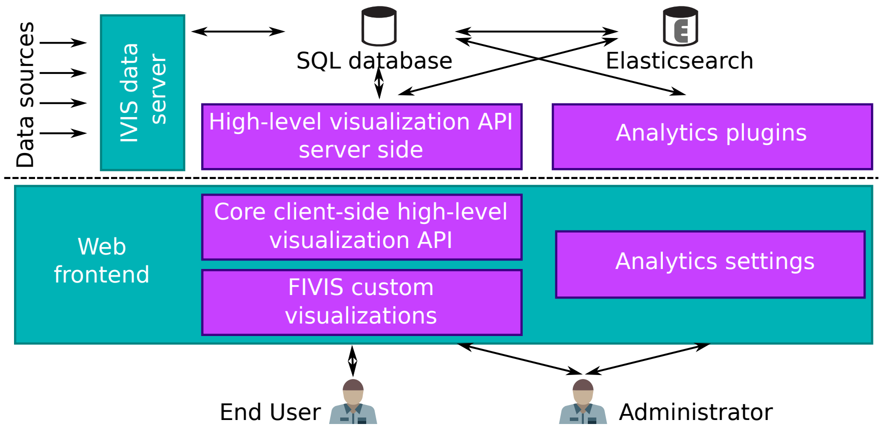


Figure 3: Architecture of the FIVIS data visualization and analysis system: back-end running on the server and the web front-end.

The back-end manages all the data (an SQL database is used as primary data storage, while the ElasticSearch engine [47] allows for fast searches and on-the-fly-data aggregations), provides a visualization API to the front-end, and schedules execution of custom analysis jobs which can process and act upon incoming data. To ensure timely processing of incoming data, the system allows scheduling analysis tasks within a latency-managed edge-cloud compute infrastructure [48].

The front-end provides user interface, both for end-users and administrators. The end-users are primarily intended to consume data in visual form, and will primarily interact with dashboards containing visualizations of data produced by the monitored system or the analysis jobs. The administrators are responsible for setting up the dashboards and analysis jobs, and will primarily interact with the management part of the user interface. Because FIVIS focuses on flexibility, both visualizations and analysis jobs can be defined and customized using an embedded code editor, which automatically builds and deploys changes without service interruption.

## 3 Our Contribution

Our work is split into four main components: 1) the CPS Core; 2) the HAR module, considering several DL alternatives, providing different accuracy vs. power consumption trade-offs; 3) the heterogeneous execution platform with virtualization support based on *PoCL-R*; and 4) the QRM tool (FIVIS) which collects, analyzes, and visualizes operational data from the system and triggers reconfiguration actions based on the analysis of results.

### 3.1 CPS Core

The CPS Core component runs in the same network as the local edge nodes and serves as a reconfiguration coordinator, primary data aggregator, and a gateway between the local system and the QRM tool (FIVIS). For each local edge node, the CPS Core gathers operational data and sends it over a secure network connection to FIVIS. The data includes: video analysis results, such as the confidence of recognized actions; application-level qualities, such as performance in terms of frames per second; and system-level metrics, such as power consumption and operating frequencies of hardware components. The data is continuously analyzed by a task deployed in FIVIS, which sends back high-level reconfiguration requests to the CPS Core component, which then triggers specific run-time adaptation actions on the local edge nodes.

## 3.2 HAR module

HAR instances are responsible for video action recognition and run on local edges. Several DL-based models for action recognition are implemented, optimised, and deployed at the distributed embedded devices. The set of alternatives offers different accuracy vs time performance and power consumption trade-offs for runtime adaptation. When the CPS core receives a reconfiguration command from FIVIS, it automatically chooses the most appropriate alternative to be deployed on the local edge nodes.

### 3.2.1 Deep Learning Architectures

Design-time decisions and the selection of the most suitable DL model are driven by trade-offs considering: the limited hardware resources available in local nodes in terms of GPU cores and memory; the need to perform the final inference in real-time to accelerate decision making; the complexity of the architecture that determines the power and thus the total working time of our battery-powered devices; the accuracy of the solutions, specially when recognizing critical actions.

The mentioned *Two Stream 3D-ConvNet* (*Two-Stream I3D*) network [27] is selected as the basis for the HAR module. It achieves the best results in terms of accuracy on the large-scale action dataset *Kinetics* [49], and the pre-trained network parameters are already available. The network model is refined applying Transfer Learning [50] and Fine Tuning [51]. In this way, faster convergence is obtained on new datasets while preventing overfitting [27]. Note that this network model is computationally intensive with a large number of parameters, mostly due to resource-intensive 3D-convolution layers.

One of the streams, the *RGBI3D* network was originally designed and pretrained to perform the analysis over a 64-frame video recorded at 25 fps with a $224 \times 224$ resolution. This amount of data makes it difficult for the embedded devices to achieve real-time performance, requiring a large amount of computational resources and therefore, quickly draining the batteries. Depending on the input size, taking into account the number of frames required and the resolution of the images, the number of operations needed to perform the inference by the DL model varies. For example, if one reduces the number of frames and resolution of the input frames, reduced processing time is expected, also reducing the amount of required memory. However, accuracy might also be impacted.

### 3.2.2 Enhancement of critical action recognition

On the other hand, we are interested in improving the system overall accuracy, specially for the recognition of critical actions such as falls. According to [27], accuracy is significantly improved when using the *Two Stream I3D* alternative: adding the *Optical Flow I3D* stream to learn from motion cues. The motion pattern of a person *falling down* is very informative and helps improving accuracy rates. However, as mentioned before, optical flow estimation is a compute-intensive task that comes at a high cost in terms of power, resources, and hurting processing time. Since the action recognition task is done locally, the higher cost associated to the *Two Stream I3D* architecture is even more concerning given the limited resources of our embedded devices. The virtualization mechanism explained in Subsection 3.3 enables the offloading of computationally-intensive tasks. Also, the consequence is the reduction of the resource utilization and energy on the local edge nodes, while simultaneously extending the working time on these battery-powered nodes. Processing time is also reduced provided the parallelization of data flows: optical flow estimation on the remote server and *RGBI3D* stream and *Optical Flow I3D* network (Two Stream I3D) inference on the edge nodes.

## 3.3 Remote GPU virtualization with PoCL-R

Offloading to remote GPUs is done by virtualizing them to appear as if they were local *OpenCL* devices on the SoC running the main application via *PoCL* and its *PoCL-R* backend driver. This driver exposes remote GPUs to the application in a similar manner to local GPUs. Applications can be forced to use remote GPUs with no code changes either by linking against the PoCL library directly to obtain OpenCL symbols or by commanding the standard Installable Client Driver (ICD) loader to load PoCL instead of the system native OpenCL implementation.

In an OpenCL platform including *PoCL-R* devices, the remote proxied devices are indicated by an extra ”pocl-remote” string in their name. The application queries the device names from the *OpenCL* API to determine which tasks to submit to which device in order to optimize energy use and latency based on the proximity of the device. *PoCL-R* attempts to minimize the execution latency of OpenCL commands sent to remote devices. To that end, it makes use of a minimal custom communication protocol and relies heavily on event dependencies between commands to defer command scheduling to the remote nodes and avoid host application involvement. Event completion signaling and data transfer between remote servers is also done in a peer-to-peer fashion whenever possible as seen in Fig 4. This is done to avoid congesting the link to the host application and to improve performance scaling when adding more remote

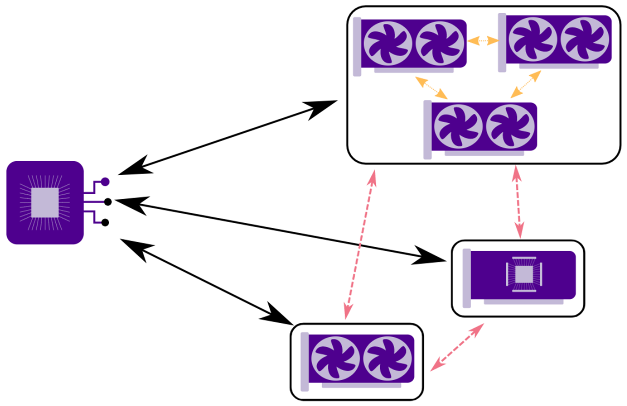


Figure 4: *PoCL-R* tries use the shortest available path to transfer data between devices: either directly between devices on the same remote server *(represented by a black rectangle)* or in a peer-to-peer fashion between remote servers. In typical cases the host device is only involved in submitting new commands, uploading completely new data or fetching the computed results.

servers. Assigning each compute task to a specific device is left to the application, however, the application has to be specifically written with scaling in mind if that is desired.

The remote servers run a special *PoCL-R* daemon that the applications connect to. The daemon uses the server native OpenCL implementation to access its local devices and expose them to *PoCL-R* clients. In order to facilitate peer-to-peer signaling and data transfer the daemons on all servers configured in the host application additionally connect directly to each other after the initial handshake from the host application.

### 3.4 Quality and Resource Management

The HAR module provides several DL network models with different properties, and the edge hardware can be operated at various performance levels, resulting in a system with many possible configurations. The goal of QRM is to adapt the run-time configuration of the system to accomplish higher-level objectives, i.e., reduce power consumption so that the system operates longer without compromising the action detection outcomes.

As mentioned in Section 1, run-time adaptation is implemented as a MAPE-k loop, with different architectural components responsible for different activities. The loop itself is built around a central component, knowledge, which is provided by FIVIS, and which collects information necessary for making adaptation decisions.

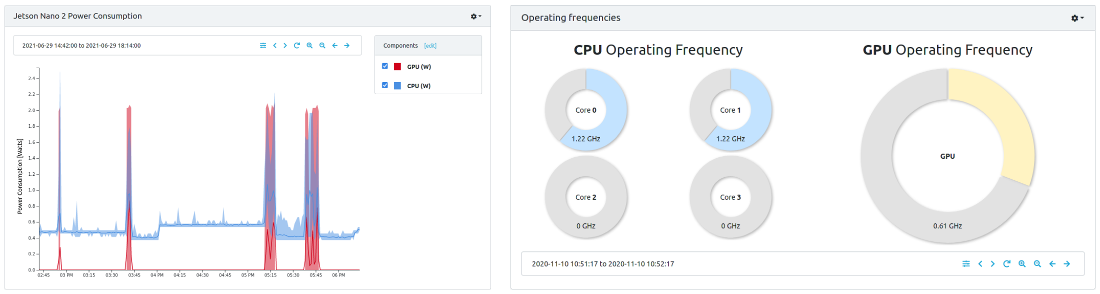
#### 3.4.1 Monitoring

Monitoring is the basis of any feedback loop. In our system, the local edge nodes report system information to the reconfigurable core component, which aggregates data from multiple nodes and sends it to FIVIS. In addition to making the monitoring data available to other consumers (and other components of the adaptation loop), FIVIS also makes it possible to visualize the collected data to enable humans to observe and supervise the system operation.

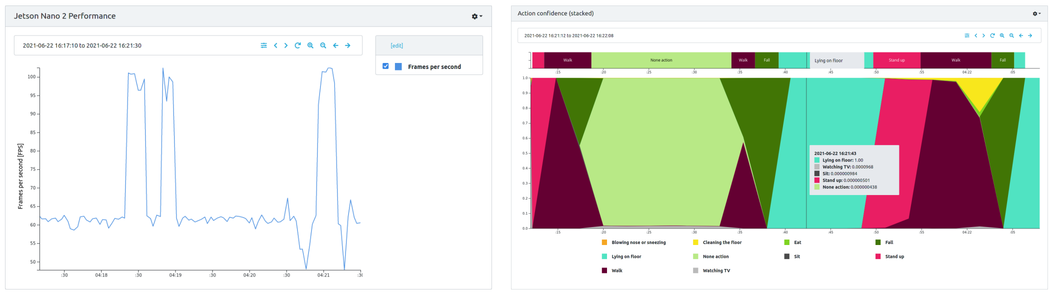
In this particular case, our system collects both system-level and application-level metrics. On the system level, we monitor utilization, temperatures, and operating frequencies of various hardware components (CPU, GPU, DRAM) as well as overall power consumption. This is illustrated in Figure 5a, which shows selected visualizations of system-level metrics. On the application level, we monitor the action recognition performance and action recognition confidence for the active DL models, as well as the active system operating mode. This is illustrated in Figure 5b, which shows selected visualizations of application-level metrics.

#### 3.4.2 Analysis and Planning

The analysis and planning phases of the loop are carried out by an analysis task defined and deployed in FIVIS. The task analyses data from two monitored qualities to determine the desired operating mode of the edge system. It uses the system power consumption (when running on battery) and action recognition confidence to maximize operating time and to minimize false alarms when identifying critical actions. FIVIS executes the analysis task whenever new data is received for the relevant quantities. Because the analysis tasks can be quite resource demanding, FIVIS supports running the analysis tasks in a distributed fashion on several machines and balances the load.



(a) Power consumption and operating frequencies of system components.



(b) Action recognition performance in frames per second and a stacked view of recognized action confidence.

Figure 5: Selected visualizations of monitoring data collected within FIVIS.

The reconfiguration policy implemented in the analysis task determines the desired system operating mode based on the following monitored qualities:

- *Battery power consumption*. Battery levels are monitored when running a DL model on the embedded devices. When the remaining battery level reaches a certain threshold, the system considers deploying a less computationally intensive (and thus, a less power demanding) DL network model while ensuring a high recognition accuracy (especially for critical actions).
- *Action recognition confidence*. The DL models recognize many different actions, but to make smart reconfiguration decisions, we need to evaluate the risk level and duration of the recognized actions. With respect to action risk level, we differentiate between critical and non-critical actions. Critical actions correspond to risky or life-threatening situations, such as *falling* or *lying on the floor*, and eventually trigger alarms that will be attended to by the emergency system. With respect to action duration, we differentiate between brief and long actions [52]. Long actions occur (often repeatedly) over a longer period of time, and include actions such as *eating* or *watching tv*, as opposed to brief actions such as *sitting down*, *standing up*, or *using inhaler*. The information on action recognition confidence—together with its risk level and duration evaluation—allows the system to choose the right DL network model for the situation. When long non-critical actions are occurring, the system uses a computationally less demanding DL model, resulting in reduced power consumption while maintaining sufficient recognition accuracy. When the system detects a critical action, it switches to a more accurate (but more resource- and power-demanding) DL model to increase recognition accuracy and confirm whether the suspected critical action really occurred, avoiding false alarms.

The analysis task produces two kinds of outputs. One of the outputs provides information on the current state (alarm/no-alarm) to the central monitoring console, with additional details whenever the system enters an alarm state, e.g. the action that triggered the alarm, or the range of timestamps in the video feed. This allows a human supervisor to verify the alarm and trigger real-world emergency action. The second kind of output provides information on the desired operating mode to the reconfigurable core component.

### 3.4.3 Execution

The execution phase of the loop (and the actual reconfiguration) is carried out by the reconfigurable core component, which receives triggers from the analysis task in form of callbacks on a REST endpoint. When the operating mode of local nodes differs from the desired operating mode, the reconfigurable core component adapts the runtime configuration of the local edge nodes to comply with the desired operating mode.

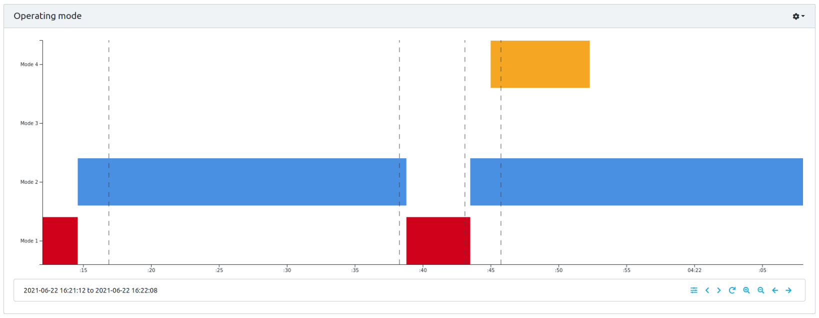


Figure 6: Chart indicating active system operating modes and reconfiguration events.

The active system operating mode is one of the qualities reported by the reconfigurable core to FIVIS for monitoring purposes, to provide developers and system operators with additional insight on system activity. Just like other metrics, FIVIS makes it possible to visualize the information in a custom chart, as shown in Figure 6. The horizontal blocks shown in the chart indicate the time interval in which a particular system configuration was active, while the vertical lines mark reconfiguration triggers issued by the analysis task.

## 4 Results and Discussion

In this section, first we present our indoor action dataset—a collection compiled from different available datasets to specifically target indoor lifestyle actions. Next, the DL network models are evaluated and compared in terms of inference time, accuracy, complexity (GFlops), and energy consumption. In addition, we assess the impact of the virtualization tool PoCL-R when offloading the optical flow estimation for the *Two Stream I3D* model in order to improve the overall system accuracy.

Finally, we introduce a reconfiguration policy with two objectives: 1) extending the system working time by reducing its energy consumption when on battery power; 2) optimizing the overall system accuracy when identifying critical actions. Finally, we evaluate the benefits of the reconfiguration policy in two real-world scenarios, and show qualitative results of daily life activities.

Network models are implemented using Keras [53] with the Adam optimizer for the training. After that, we perform an optimization through TensorRT [54] to improve the performance of these alternatives on the SoC devices (Jetson Nano). This enables us to reduce the memory usage of the DL models on the edge devices, and it also reduces the time needed to load the model parameters on the GPU.

Regarding the local edge nodes, HAR is implemented using the NVidia Jetson Nano [28] embedded devices. These power efficient SoCs have 4 CPU cores and 1 GPU with 128 CUDA cores that share 4GB of memory. It is a powerful device but it is limited when working on machine learning applications compared to the usual cluster or high-performance architectures used for state-of-the-art works. This is precisely the reason why reconfiguration plays a crucial role, enabling optimal use of the devices when using low-cost low-power edge nodes. An additional advantage of the Jetson Nano platform is that it is fully configurable, meaning that both CPU and GPU frequencies can be changed. To further reduce power consumption, our application configures the node to: 1) use only 2 CPU cores at 1.2 Ghz; 2) reduce the operating frequency of the GPU to 614 Mhz.

### 4.1 Indoor Action Dataset

The DL model alternatives are trained and evaluated using the Combined Indoor Action Video (CIAV) dataset[1] presented in [55]. Actions and samples were obtained from six well-known datasets: *HMDB51* [56], *UCF-101* [57], *Fall Detection* [58], *Charades* [59], *STAIR* [60], and *Kinetics* [49]. The dataset includes all relevant actions for indoor lifestyle habits. Specializing our dataset is crucial because there are currently no datasets that cover a reasonable amount of actions useful for performing indoor lifestyle monitoring. Furthermore, mixing clips from different sources allows us to create a heterogeneous dataset that represents classes with different actors and scenes, helping us to better generalize.

The CIAV dataset includes 18 actions describing indoor daily life actions: *bandaging, blowing nose or sneezing, cleaning floor, cooking, eating, falling down, hitting, lying on bed or sofa, lying on floor, running, sitting down, sleeping, smoking, standing up, using inhaler, walking, watching TV*. Additionally, we include a *no-action* category to help the

[1]`https://github.com/DaniDeniz/CIAV-dataset`

Table 1: DL Architectures performance and energy consumption

| Base architecture | #Frames | Spatial Resolution ( channels) | F1-Score | Inference (ms) |
|---|---|---|---|---|
| RGBI3D | 64 | $224 \times 224( \times 3)$ | **82.81** | 5551.58 |
| RGBI3D | 64 | $112 \times 112( \times 3)$ | 80.50 | 1595.42 |
| RGBI3D | 32 | $224 \times 224( \times 3)$ | 80.35 | 2878.06 |
| RGBI3D | 32 | $112 \times 112( \times 3)$ | 79.70 | 829.78 |
| RGBI3D | 16 | $224 \times 224( \times 3)$ | 78.35 | 1448.67 |
| RGBI3D | 16 | $112 \times 112( \times 3)$ | 78.00 | **426.50** |
| Optical Flow I3D | 64 | $224 \times 224( \times 2)$ | 78.21 | 5072.94 |
| Optical Flow I3D | 64 | $112 \times 112( \times 2)$ | **79.17** | **1473.00** |

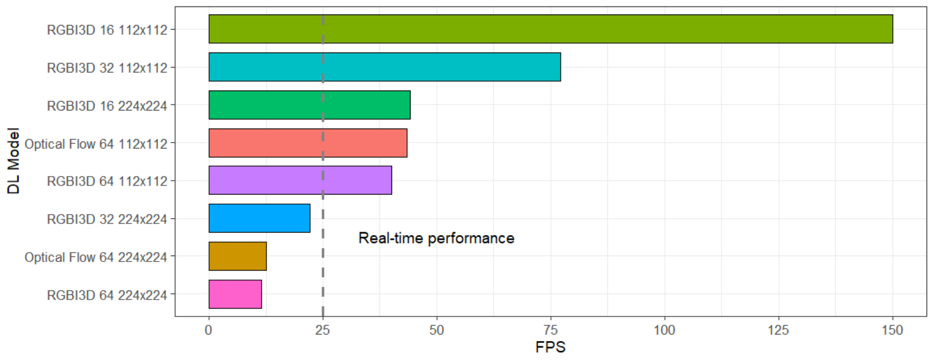


Figure 7: Inference time performance (fps) of the proposed DL netwok models. Real-time performance is reached when the model inference time is less than or equal to the time needed to receive a new batch of 64 frames recorded at 25 fps (dashed line).

DL model associate the presence of a human with the actions, thus preventing it from making decisions based only on the scenario. Note that the clips for this category only include indoor scenes without humans. The total is 21399 video examples with an average length of 10 s, that are split up into training (80%), validation (8%), and test sets (12%). Given the different nature of the original datasets, our CIAV dataset is highly imbalanced, with some classes containing significantly more clips than others. For instance, the dataset includes more than 1800 examples of *cooking* and less than 600 of *watching tv*.

## 4.2 Evaluation of Deep-Learning models

The original model uses $224 \times 224$ 64-frame videos for the *RGBI3D* network, resulting in a huge amount of data to be processed by our edge devices. In order to reduce complexity (also reach real-time performance and reduce power consumption), six alternatives that use the *RGBI3D* backbone and its pretrained weights are presented (see Table 1). Regarding resolution, we consider a down-sampled input of $112 \times 112$ pixels, and sampling temporal resolution by selecting equally distanced frames of the original 64 frames recorded at 25 fps, keeping 32 and 16 frames (1 out of 2, and 1 out of 4 respectively) without interpolation.

Alternatives are trained similarly, using the *RGBI3D* backbone, with the pretrained weights [27]. First, their architecture is adapted according to the input size. Next, they are trained applying Transfer Learning [50], meaning that only the last layers of each network model are trained using the CIAV dataset, while keeping the weights of the remaining layers frozen. Note that the objective of Transfer Learning is to adapt the DL architecture to our CIAV dataset, retaining the knowledge obtained from the *Kinetics* dataset, favouring generalization. After Transfer Learning, Fine Tuning [51] is carried out, training the whole architecture again. Fine Tuning optimizes the DL model and specializes it for indoor action recognition. In this step, weights for the adaptation to the different input configuration designs are learned. Also, output layers are adjusted to consider each action equally, given the imbalanced nature of the dataset. The combination of Transfer Learning and Fine Tuning significantly boosts the overall recognition performance.

Table 1 shows performance and resource utilization for the proposed architectures attending to macro F1-score, inference time, and power consumption on the embedded device. The macro F1-score metric takes into account the effect of imbalanced datasets by performing a harmonic mean of the precision $\frac{TP}{TP+FP}$ and recall $\frac{TP}{TP+FN}$ values, equally weighing the contribution from all classes regardless of the number of examples per class. For the macro F1-score, the

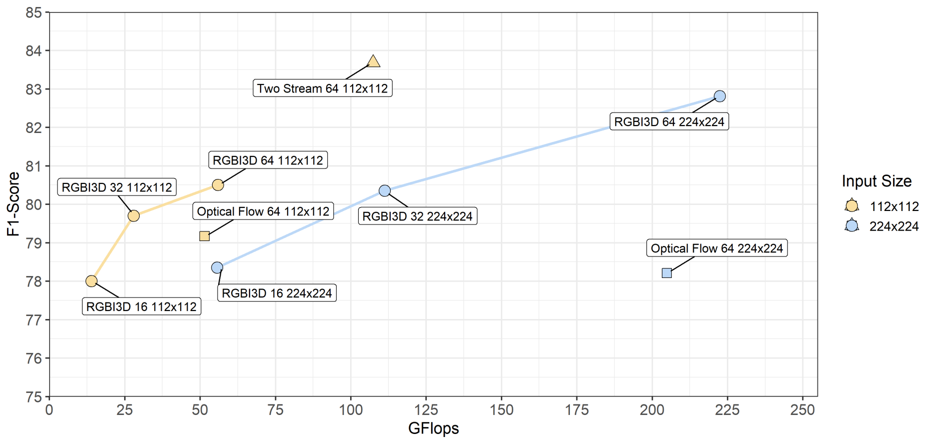


Figure 8: F1 score vs GFlops. GFlops refer to the number of multiplications and additions operations performed for the DL inference. Therefore, GFlops are directly proportional to the time performance, computational resources, and power consumption. Note how the alternatives with lower spatial resolution ($112 \times 112$) reach a reasonable F1 score using less computational resources (operations) compared to the DL network models with higher spatial resolution.

*RGBI3D 64* - $224 \times 224$ alternative obtains the best performance, although as mentioned earlier, this comes at a very high cost in terms of processing time (inference) and power or resources. When comparing to *RGBI3D 16* - $112 \times 112$, the fastest alternative, accuracy is reduced only about 5% but inference time is about 13x less than the inference time for the *RGBI3D 64* - $224 \times 224$ alternative, and power consumption is reduced to half.

Real-time processing is ensured with inference times below 2560 ms (64 frames at 25 fps), reached by most alternatives as shown in the last column in Table 1. Bold values show the lowest inference times for the *RGBI3D* and the *Optical Flow I3D* streams. Figure 7 shows inference times in fps, for the developed network models. The *RGBI3D 16* - $112 \times 112$ network model reaches up to 150 fps, 6x faster than the real-time threshold mentioned before. All DL network models with $112 \times 112$ input spatial resolution reach real-time performance, including the *RGBI3D 16* - $224 \times 224$ alternative that samples 1 out of 4 frames. This also highlights the importance of optimizing the number of operations and thus, the complexity of network models, in order to achieve higher performance architectures.

Floating point operations represents the number of multiplications and additions computed for each DL network model inference and thus, it is a measure of complexity of machine learning models. Thus, the number of floating-point operations directly impacts the time performance and power consumption of these models. In Figure 8, accuracy measured with F1-score is compared to the Giga Floating points operations (GFlops) for the proposed alternatives. The most important result of this graph is that when the spatial resolution is subsampled, a high decrease in terms of GFlops is achieved, maintaining a limited loss of accuracy. For example, the *RGBI3D 64* - $112 \times 112$ model computes approximately 75% less GFlops compared to the *RGBI3D 64* - $224 \times 224$ alternative. At the same time, the F1 score is only reduced by less than 3%. Additionally, temporal resolution seems to have a great impact on the F1 score: the *RGBI3D 64* - $112 \times 112$ and the *RGBI3D 16* - $224 \times 224$ network models require a similar number of GFlops but the first one obtains 2.7% higher F1 score. Note also that the *Optical Flow* architectures compute in average 8% less operations than the *RGBI3D* architecture with similar input resolutions. RGB data have 3 channels and optical flow only 2, motion in the x and y directions. Finally, we reach the highest recognition performance with the *Two Stream I3D 64* - $112 \times 112$ network, achieving 1% higher F1 score using less than half the resources of the *RGBI3D 64* - $224 \times 224$ model.

The *Two Stream I3D* network inference is done by combining the *RGBI3D 64* - $112 \times 112$ and the *Optical Flow 64* - $112 \times 112$ results. This DL model obtains better results for the recognition of critical actions. The optical flow estimation can be done in the local CPU, the local GPU, or the remote server via PoCL-R. Using the local CPU is not feasible, taking more than 30 s to estimate the optical flow of a $112 \times 112$ video stream of 64 frames. Using the local GPU (CUDA) is faster taking less than $2.645 \pm 0.016$ s. However, using PoCL-R to access the remote GPU through a 1000BASE-T Ethernet connection (1Gbps) accelerates the optical flow estimation taking only $0.986 \pm 0.09$ s. Additionally, using the remote GPU brings a great value to the system because it allows us to reduce the overall computation time of the *Two Stream I3D* network by about 46%. Note in Figure 9 and Table 2 how using the remote GPU allows us to parallelize the optical flow computation and the *RGBI3D* inference. In this way, just after the inference of the *RGBI3D* is finished on the local GPU, the *Optical Flow* stream network starts with the inference on

Table 2: Two Stream I3D, optical flow estimation alternatives

| DL Architecture | Optical Flow Estimation | # Frames | Resolution | F1-Score | Inference (ms) |
|---|---|---|---|---|---|
| Two Stream I3D | CUDA | 64 | 112 × 112 | 83.68 | 5770.22 |
| Two Stream I3D | GPU RTX2080 Ti via PoCL-R | 64 | 112 × 112 | 83.68 | **3060.55** |

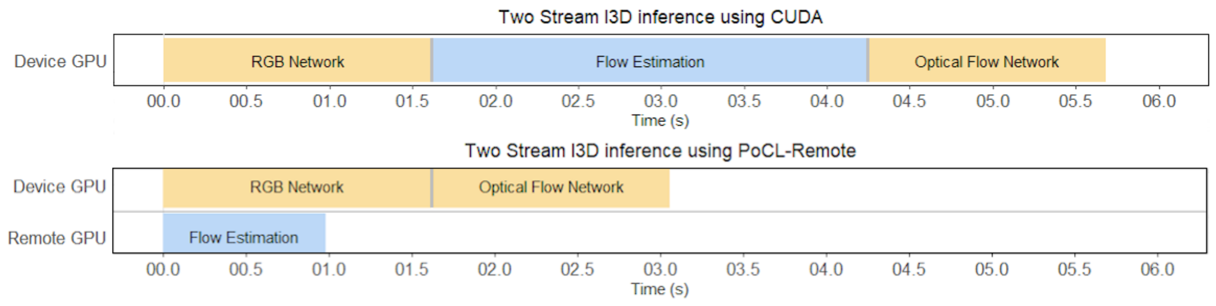


Figure 9: Comparison in time performance between using local GPU or the remote server via PoCL-R when performing an inference with the *Two Stream I3D* network. The alternative using the remote GPU reduces the *Two Stream I3D* inference time by 46% offloading the Optical Flow estimation. It also alleviates the energy consumption on the local node compared to the CUDA approach.

the local device. The use of the remote GPU via PoCL-R saves almost 2.6 s and reduces power consumption of the battery-powered local nodes.

After the analysis, 4 different action recognition models are chosen: *RGBI3D 16* - 112 × 112, *RGBI3D 32* - 112 × 112, *RGBI3D 64* - 112 × 112 and *Two Stream I3D (PoCL-R) 64* - 112 × 112. These alternatives provider the best F1-score vs GFlops trade-offs, leading also to low-power implementations. Figure 10 shows accuracy, time performance, and power for the HAR alternatives that are selected to be deployed to the local edge nodes. Although the *Two Stream I3D* model is the most accurate, it also comes with the burden of high energy consumption and higher inference times. In general, alternatives that require fewer computational resources (GFlops) and lower energy budgets, also require shorter inference times. Nevertheless, the power consumption of the four proposed alternatives is very low, without exceeding an average consumption of only 4.5 Watts (W) on the local nodes. Note that the power consumption is measured on the SoC device by performing 30 inferences over a video feed of 64 frames.

## 4.3 System Reconfiguration Policy

In general, DL models are optimised at design time to achieve state-of-the-art performance, but there may be many other objectives in real-world scenarios, and they may also change over time. To cope with multi-objective optimization, real-world systems therefore need to provide run-time adaptation mechanisms. These mechanisms are controlled by a reconfiguration policy whose design determines the specific optimization objectives.

In our case, the multi-objective optimization aims at maximizing the working time of the local nodes (batteries) in continuous operation while minimizing the occurrence of false alarms related to critical actions through enhanced action recognition accuracy. The policy is implemented as an analysis task running in FIVIS, which monitors and analyzes the relevant system and application qualities and issues reconfiguration commands to achieve the desired objectives and trade-offs.

Table 3 summarizes 5 different operation modes using the selected DL models described in Section 4.2. The presented operation modes are sorted from 0 to 4 depending on its F1 score and energy consumption (low to high). *Mode 0* is simply a *motion detector*, this operating mode avoids running any DL model, and it is the most power efficient. The *motion detector* only computes differences between frames separated by a time span of one second; no difference between frames means nobody is in the scene. Modes 1 to 4 run the selected DL network models: *mode 1* runs the simpler (*RGBI3D 16* - 112 × 112) DL network model, using less computational resources than any other alternative; *mode 2* runs the *RGBI3D 32* - 112 × 112 model, this operating mode offers a reasonable F1-score at a low energy cost; *mode 3* uses the most accurate *RGBI3D* model (*RGBI3D 64* - 112 × 112) that reaches real-time performance, offering a high recognition performance at medium energy cost; finally, *mode 4* uses the *Two Stream I3D 64* - 112 × 112 model to confirm whether a critical action really took place, reducing false emergency alarms.

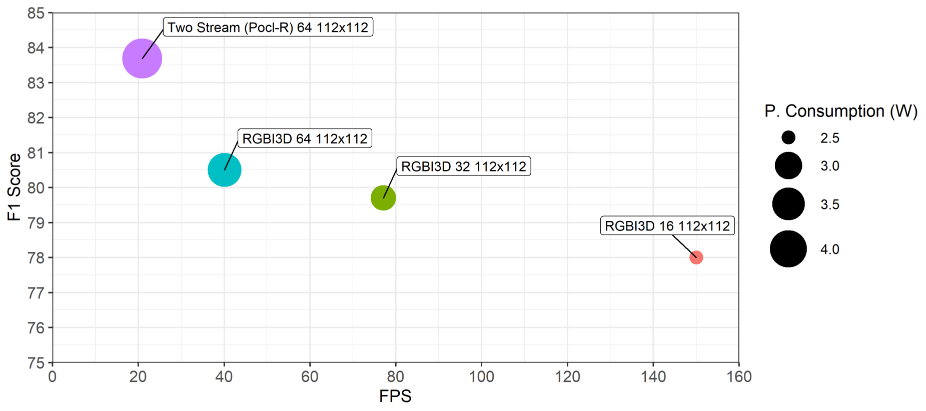


Figure 10: Performance comparison between the selected DL architectures in terms of F1 score, time performance, and power consumption. Architectures that reach higher F1 values comes with a higher energy cost and a reduced time performance. For instance, the *RGBI3D 16* - $112 \times 112$ model offers the best time performance with a low energy budget but at the expense of loosing around 5.6 points of F1 score compared to the *Two Stream I3D* network.

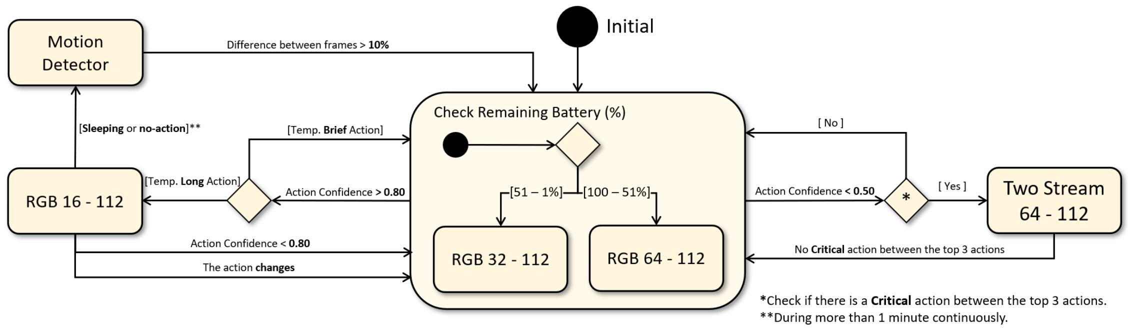


Figure 11: State Transition Diagram of the reconfiguration policy, describing the model for the run-time adaptation triggering reconfiguration commands. Left) operating *modes 0 and 1* are activated when temporally long actions are recognized; center) operating *modes 2 and 3* are used when brief actions are being carried out by the subjects, the decision of which one is active depends on the remaining edge battery level; right) operating *mode 4* is used to confirm whether a critical action has happened.

Our reconfiguration policy is described on a state transition diagram in Figure 11 that represents how the system is adapted automatically in runtime, triggering reconfiguration commands.

- In the state transition diagram, when subjects are performing brief actions, *mode 2* or *mode 3* are used: if the remaining battery level is above 50%, *mode 3* is used, offering high recognition performance; otherwise, *mode 2* is activated, a more energy-efficient alternative that will contribute to extend battery times with a limited loss in accuracy.
- When long actions are recognized with high confidence, *mode 1* is deployed to the edge nodes. The subject performing the long action is continuously monitored with this operating mode, using few computational resources. A change or recognized action confidence decrease switches back to *mode 2* or *mode 3*.
- When monitoring detects that the subject is sleeping or nobody is in the scene for a few minutes, *mode 0* is deployed to the edge nodes. This operating mode allows the system saving resources and a significant amount of energy. Motion in the scene triggers another switch to *mode 2* or *mode 3*.
- *Mode 4* is deployed to the edge nodes when a critical action is recognized. In order to confirm that a life-threatening situation occurred, the system re-analyzes the video using mode 4 that uses the *Two Stream I3D* network model. After that, if the system comes up with a false positive (negative confirmation), nodes are switched back to *mode 2* or *mode 3*.

Table 3: Operating modes of the local nodes

| Mode id | Name | F1-Score | Inference (ms) | Power (W) |
|---|---|---|---|---|
| 0 | Motion Detector | - | - | $1.76 \pm 0.03$ |
| 1 | RGBI3D 16 - $112 \times 112$ | 78.00 | **426.50** | $2.50 \pm 0.74$ |
| 2 | RGBI3D 32 - $112 \times 112$ | 79.70 | 829.78 | $2.86 \pm 0.96$ |
| 3 | RGBI3D 64 - $112 \times 112$ | 80.50 | 1595.42 | $3.61 \pm 0.98$ |
| 4 | Two Stream I3D (PoCL-R) 64 - $112 \times 112$ | **83.68** | 3060.55 | $4.44 \pm 0.55$ |

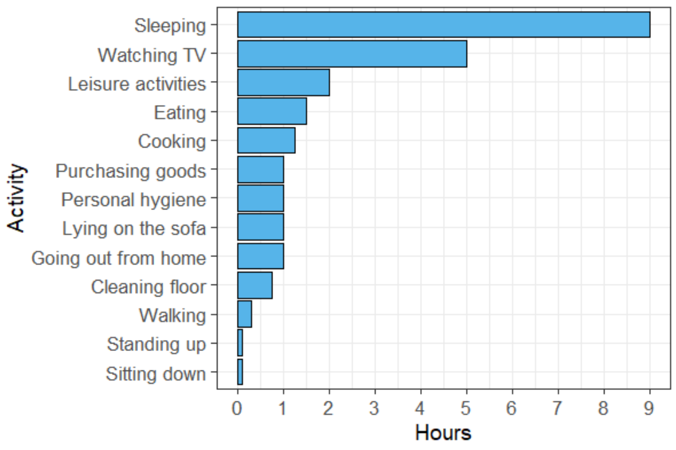


Figure 12: Average hours per day spent in selected activities of an elderly person.

### 4.4 System evaluation on real scenarios

To illustrate the benefits of the proposed system we define two real-world scenarios: 1) a conventional daily routine and 2) an accident that triggers an alarm. In this experiment, for the sake of simplicity, we assume only one edge node. To estimate the overall working time and performance, we consider common $5V$ $25000mAh$ batteries for our edge devices. With a capacity of $125Wh$, the system runs for up to 25 hours, draining 5W in average.

#### 4.4.1 Scenario 1: A day of an elderly person

This scenario assumes the monitoring of an elderly person on a typical day. Figure 12 shows the average hours per day spent by an elderly person in different activities according to the [61]. Considering the indoor activities selected for our system, Figure 13 shows a comparison of the system overall working time without and with the reconfiguration policy explained in Section 4.3. In the first case, the first three columns show the system running only one DL network model, respectively: *RGBI3D 16* - $112 \times 112$ (mode 1), *RGBI3D 32* - $112 \times 112$ (mode 2), and *RGBI3D 64* - $112 \times 112$ (mode 3).

For the described scenario, Figure 13 shows how the overall working battery time is extended for up to 63% when performing reconfiguration, running for 22 additional hours compared to only using *mode 3*. Also, working time is also extended by 13% respect to *mode 1*, the most power-efficient DL model. By performing reconfiguration, the most suitable *mode* according to the situation is activated. This *mode* adaptation ensures good recognition performance and optimal energy consumption of the overall system. As seen, adapting the use of resources of the local nodes enables large energy savings. It is possible thanks to the reconfiguration policy and the use of less resource-intensive alternatives when the room is empty or long actions take place (see Figure 14). This also ensures continuous operation without interruptions. Bear in mind that energy savings scale up with the number of edge nodes in a system.

Confidence of the recognized actions and other qualities of the system are continuously logged. This operational data in the dashboard in Figure 16 help the healthcare professionals to keep track on their patients' lifestyle habits. This dashboard shows the average time spent on every activity and notifies with an alarm when a critical action occurs. A further step would e.g. offer new routines for active lifestyle.

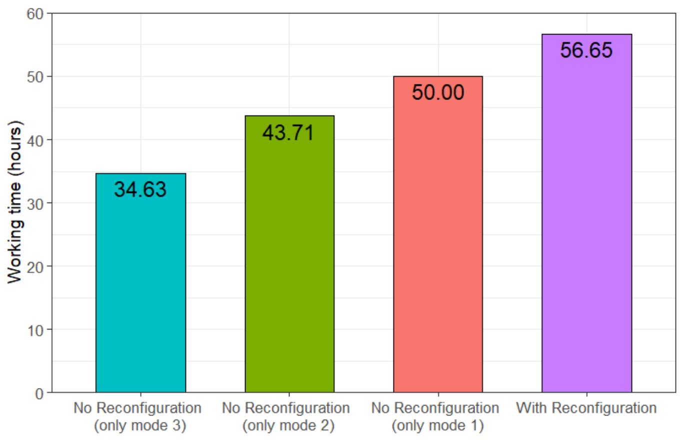


Figure 13: Estimated working times of local edge nodes using $5V$ $25000mAh$ batteries, considering times and activities shown in the left. From left to right, columns 1 to 3 show the overall working time for the operating modes 3 to 1 respectively, column 4 shows the extended working time of the system when runtime adaptation is implemented.

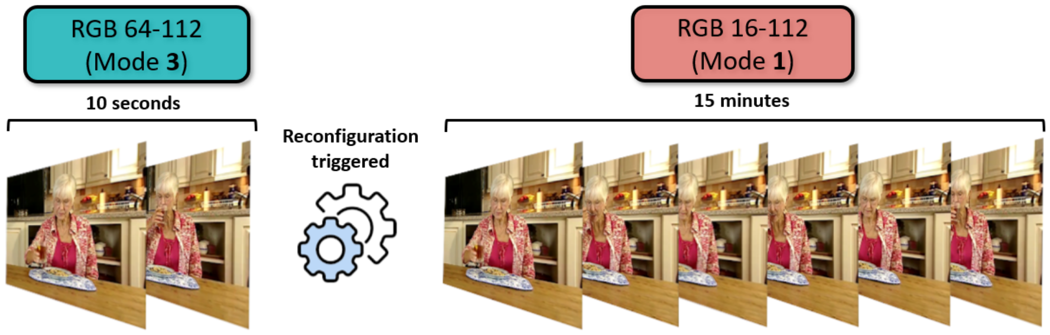


Figure 14: Scenario 1: Reduction of energy consumption through reconfiguration. A person is eating, and the system recognizes with high confidence this long action. FIVIS triggers a reconfiguration to use a more power efficient DL model to recognize the temporally long action. The *RGBI3D 16* - $112 \times 112$ continues properly identifying the action with lower confidence. It achieves up to 30% energy saving during the recognition of the long action compared to only using the *RGBI3D 64* - $112 \times 112$ model without reconfiguration.

### 4.4.2 Scenario 2: The elderly person suffers an accident

Reconfiguration also allows for providing more accurate predictions reducing false alarms. This is crucial when recognizing critical actions that trigger alarms for the emergency medical services.

In this second case, an elderly person falls down. It is assumed that the active DL network model is the operating *mode 2* or *mode 3*. In this case, the system provides an ambiguous result recognizing the label *falling down* with a confidence of 0.27 and *sitting* with a 0.24 confidence (see the real-world case in Figure 15). Applying the already described reconfiguration policy, the system is adapted in run-time deploying the (*Two Stream I3D*) network model. This alternative reaches the highest performance in action recognition in testing for the CIAV dataset thus, the new model is expected to provide a higher confidence confirming whether the person actually fell or not. The new model confirms the critical action with a confidence of 0.99 and also recognizes the current action as *falling down*, by adding the motion cues. Recall and precision for the action *falling down* is increased by 5% and 6.7% respectively using the *Two Stream I3D* model. After that, the system triggers the confirmed alarm. Reducing false alarms will eventually result in an increase of user trust in the system. Finally, the virtualization mechanism via *PoCL-R* is omitted to simplify the description, but it is done as explained in Section 4.2.

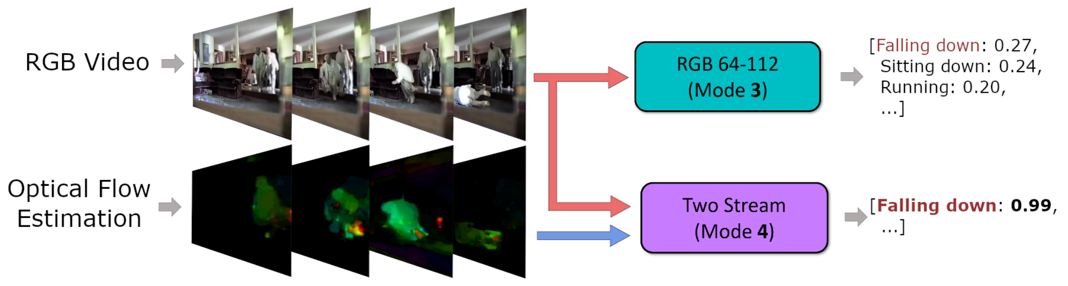

RGB Video
Optical Flow Estimation
RGB 64-112 (Mode 3)
Two Stream (Mode 4)
[Falling down: 0.27, Sitting down: 0.24, Running: 0.20, ...]
[Falling down: 0.99, ...]

Figure 15: Scenario 2: Critical action confirmation using reconfiguration. An elderly person falls down and the video is analyzed with the *RGBI3D 64* - 112 $\times$ 112 (mode 3) network. The confidence of the identified critical action is low (27%). A reconfiguration is triggered by FIVIS and the *Two Stream (mode 4)* process the input and confirm with a confidence of **99%** that the action *falling down* occurred. Note how the Optical Flow estimation of the video provides information about the motion of the person falling.

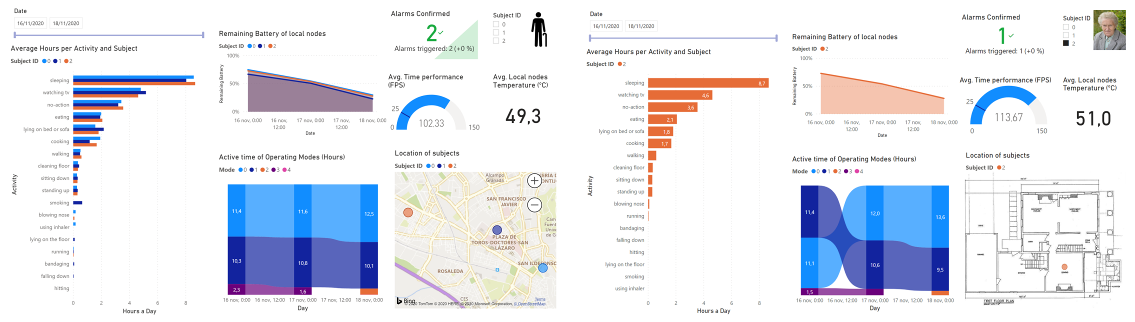

Average Hours per Activity and Subject
Remaining Battery of local nodes
Alarms Confirmed
Avg. Time performance (FPS)
Avg. Local nodes Temperature (°C)
49,3
51,0
Active time of Operating Modes (Hours)
Location of subjects
Hours a Day
Day

Figure 16: Life Style monitoring system dashboard. Left: Dashboard with the information of three subjects at the same time. Right: Dashboard showing only the information related to subject with ID 2. This dashboard can be used by healthcare professionals to review the lifestyle habits of the subjects. For example, it shows the average time each subject spend on each activity, where the person is and whether alarms have been triggered among others.

## 5 Conclusions

We have proposed a general framework for reconfigurable CPS for video processing using embedded devices. The CPS continuously monitors its operational data and performance to automatically adapt its computation to optimize the overall system performance. We have demonstrated its use within an indoor lifestyle monitoring system for the elderly, enabling them to live independently. Our solution targets several objectives such as the supervision of habits to promote healthy lifestyle and the rapid emergency response when a person suffers an accident.

The proposed demonstrator is built from the integration of several components: an aggregator high-end server for computationally intensive operations and to collect the global status of the system; local edge nodes that perform real-time action recognition with low power consumption in optimized embedded devices; a virtualization platform enabled via *PoCL-R* that offloads computation to the high-end server to reduce the computation load of the edge nodes; and the Quality and Resource Management tool FIVIS that also helps analyzing operational data and visualizing the parameters with a friendly interface. Thanks to this last component, changes in the environment may trigger reconfiguration to optimize energy consumption or action recognition confidence, sending a runtime reconfiguration command to deploy the most suitable models on the edge nodes. In fact, we have demonstrated how intensive computing tasks can be offloaded to high-end GPUs in remote servers via PoCL-R, benefiting inference times while helping local nodes to save energy, extending their batteries working times. Also, this technology enables seamless virtualization of resources enabling the use of more powerful and accurate DL models without affecting the local nodes power consumption while accelerating the computation.

In our case, reconfiguration provides flexibility to our system with the automatic adaptation of the active DL models on our local edges, the Jetson Nano devices. With the proposed reconfiguration policy the system working time is extended by 63% with a very limited loss of the overall system accuracy. Also, the recognition of critical actions by using DL architectures is improved, a crucial aspect of the s-Health monitoring system.

In future works, we plan to include multimodal information for s-Health system such as vital signs that will help in hospitalized patients. Also, new heterogeneous architectures for this kind of information and DL models capable of combining the multimodal information will be explored.

## Acknowledgements

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 783162. The JU receives support from the European Union's Horizon 2020 research and innovation programme and The Netherlands, Czech Republic, Finland, Spain, and Italy. The work was also supported by a Spanish National grant funded by MINECO through APCIN PCI2018-093184 and partially by the Research Network RED2018-102511-T.

## References

[1] United Nations, Department of Economic Social Affairs, and Population Dynamics. 2019 revision of world population prospects. In *World Population Prospects 2019*, page 46, 2019.

[2] Kadian Davis, Jun Hu, Loe Feijs, and Evans Owusu. Social hue: A subtle awareness system for connecting the elderly and their caregivers. In *2015 ieee international conference on pervasive computing and communication workshops (percom workshops)*, pages 178–183. IEEE, 2015.

[3] Juan M Murillo. Complex event processing for health monitoring. In *Gerontechnology: First International Workshop, IWoG 2018, Cáceres, Spain, and Évora, Portugal, 14 and 17 December, 2018, Revised Selected Papers*, volume 1016, page 3. Springer, 2019.

[4] Alaa Awad Abdellatif, Ahmed Emam, Carla-Fabiana Chiasserini, Amr Mohamed, Ali Jaoua, and Rabab Ward. Edge-based compression and classification for smart healthcare systems: Concept, implementation and evaluation. *Expert Systems with Applications*, 117:1–14, 2019.

[5] Shuo Tian, Wenbo Yang, Jehane Michael Le Grange, Peng Wang, Wei Huang, and Zhewei Ye. Smart healthcare: making medical care more intelligent. *Global Health Journal*, 3(3):62–65, 2019.

[6] Alaa Awad Abdellatif, Amr Mohamed, Carla Fabiana Chiasserini, Mounira Tlili, and Aiman Erbad. Edge computing for smart health: Context-aware approaches, opportunities, and challenges. *IEEE Network*, 33(3):196–203, 2019.

[7] Shafiq ur Rehman, Marcel Prehn, and Volker Gruhn. Smartchair: A realization of smart health care system based on cyber-physical systems. In *Proceedings of the Fourth International Conference on Engineering & MIS 2018*, pages 1–8, 2018.

[8] Parisa Rashidi and Alex Mihailidis. A survey on ambient-assisted living tools for older adults. *IEEE journal of biomedical and health informatics*, 17(3):579–590, 2012.

[9] Pau Climent-Perez, Susanna Spinsante, Alex Mihailidis, and Francisco Florez-Revuelta. A review on video-based active and assisted living technologies for automated lifelogging. *Expert Systems with Applications*, 139:112847, 2020.

[10] Hemant Ghayvat, S Mukhopadhyay, B Shenjie, Arpita Chouhan, and W Chen. Smart home based ambient assisted living: Recognition of anomaly in the activity of daily living for an elderly living alone. In *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–5. IEEE, 2018.

[11] Juan A Botia, Ana Villa, and Jose Palma. Ambient assisted living system for in-home monitoring of healthy independent elders. *Expert Systems with Applications*, 39(9):8136–8148, 2012.

[12] J Mikael Eklund, Jonathan Sprinkle, Shankar Sastry, and T Riisgaard Hansen. Information technology for assisted living at home: building a wireless infrastructure for assisted living. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 3931–3934. IEEE, 2006.

[13] Yin Zhang, Meikang Qiu, Chun-Wei Tsai, Mohammad Mehedi Hassan, and Atif Alamri. Health-cps: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal*, 11(1):88–95, 2015.

[14] Edward A Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE Int. Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE, 2008.

[15] Radhakisan Baheti and Helen Gill. Cyber-physical systems. *The impact of control technology*, 12(1):161–166, 2011.

[16] Wei Feng, Yu Qin, Shijun Zhao, and Dengguo Feng. Aaot: Lightweight attestation and authentication of low-resource things in iot and cps. *Computer Networks*, 134:167–182, 2018.

[17] Shah Ahsanul Haque, Syed Mahfuzul Aziz, and Mustafizur Rahman. Review of cyber-physical system in healthcare. *international journal of distributed sensor networks*, 10(4):217415, 2014.

[18] Zaid Al-Ars, Twan Basten, Ad de Beer, Marc Geilen, Dip Goswami, Pekka Jääskeläinen, Jiří Kadlec, Marcos Martinez de Alejandro, Francesca Palumbo, Geran Peeren, et al. The fitoptivis ecsel project: highly efficient distributed embedded image/video processing in cyber-physical systems. In *16th ACM Int. Conference on Computing Frontiers*, pages 333–338, 2019.

[19] Juan Isern, Francisco Barranco, Daniel Deniz, Juho Lesonen, Jari Hannuksela, and Richard R Carrillo. Reconfigurable cyber-physical system for critical infrastructure protection in smart cities via smart video-surveillance. *Pattern Recognition Letters*, 140:303–309, 2020.

[20] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[21] Zhengang Guo, Yingfeng Zhang, Xibin Zhao, and Xiaoyu Song. Cps-based self-adaptive collaborative control for smart production-logistics systems. *IEEE Transactions on Cybernetics*, 2020.

[22] Martijn Hendriks, Marc Geilen, Kees Goossens, Rob de Jong, and Twan Basten. Interface modeling for quality and resource management. *arXiv:2002.08181*, 2020.

[23] Young-Sik Jeong, Hyun-Woo Kim, and Haeng Jin Jang. Adaptive resource management scheme for monitoring of cps. *The Journal of Supercomputing*, 66(1):57–69, 2013.

[24] Tangbin Xia, Lifeng Xi, Ershun Pan, and Jun Ni. Reconfiguration-oriented opportunistic maintenance policy for reconfigurable manufacturing systems. *Reliability Engineering & System Safety*, 166:87–98, 2017.

[25] Vincenzo Eramo and Francesco Giacinto Lavacca. Proposal and investigation of a reconfiguration cost aware policy for resource allocation in multi-provider nfv infrastructures interconnected by elastic optical networks. *Journal of Lightwave Technology*, 37(16):4098–4114, 2019.

[26] Lubomír Bulej, Tomáš Bureš, Petr Hnětynka, Václav Čamra, Petr Siegl, and Michal Töpfer. IVIS: Highly customizable framework for visualization and processing of IoT data. In *Proceedings of EUROMICRO SEAA 2020, Portorož, Slovenia*, page 4. IEEE, 2020.

[27] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[28] NVIDIA. Jetson nano developer kit, 2020.

[29] Jan Solanti, Michal Babej, Julius Ikkala, and Pekka Jääskeläinen. Pocl-r: A scalable low latency distributed opencl runtime. In *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*. Springer, 2021.

[30] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *International conference on pervasive computing*, pages 1–17. Springer, 2004.

[31] Mohammed Mehedi Hassan, Md Zia Uddin, Amr Mohamed, and Ahmad Almogren. A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems*, 81:307–313, 2018.

[32] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209, 2012.

[33] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, 2019.

[34] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. A comprehensive survey of vision-based human action recognition methods. *Sensors*, 19(5):1005, 2019.

[35] L Minh Dang, Kyungbok Min, Hanxiang Wang, Md Jalil Piran, Cheol Hee Lee, and Hyeonjoon Moon. Sensor-based and vision-based human activity recognition: A comprehensive survey. *Pattern Recognition*, 108:107561, 2020.

[36] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE Int. Conference on Computer Vision*, pages 4489–4497, 2015.

[37] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[38] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint pattern recognition symposium*, pages 214–223. Springer, 2007.

[39] Philip P Dang and Paul M Chau. Image encryption for secure internet multimedia applications. *IEEE Transactions on consumer electronics*, 46(3):395–403, 2000.

[40] Pekka Jääskeläinen, Carlos Sánchez de La Lama, Erik Schnetter, Kalle Raiskila, Jarmo Takala, and Heikki Berg. pocl: A performance-portable opencl implementation. *International Journal of Parallel Programming*, 43(5):752–785, 2015.

[41] Khronos® OpenCL Working Group. The OpenCL™ Specification. `https://www.khronos.org/registry/OpenCL/specs/3.0-unified/pdf/OpenCL_API.pdf`. accessed: 2020-10-16.

[42] Peng Zhou, Decheng Zuo, Kun Mean Hou, Zhan Zhang, Jian Dong, Jianjin Li, and Haiying Zhou. A Comprehensive Technological Survey on the Dependable Self-Management CPS: From Self-Adaptive Architecture to Self-Management Strategies. *Sensors*, 19(5), January 2019.

[43] Grafana. Grafana, 2021.

[44] Kibana. Kibana, 2021.

[45] Chronograf. Chronograf, 2021.

[46] Suresh K. Peddoju and Himanshu Upadhyay. Evaluation of IoT Data Visualization Tools and Techniques. In *Data visualization: Trends and challenges toward multidisciplinary perception*, pages 115–139. Springer, 2020.

[47] Clinton Gormley and Zachary Tong. *Elasticsearch: the definitive guide*. O'Reilly, 2015.

[48] Lubomír Bulej, Tomáš Bureš, Adam Filandr, Petr Hnětynka, Iveta Hnětynková, Jan Pacovský, Gabor Sandor, and Ilias Gerostathopoulos. Managing latency in edge–cloud environment. *Journal of Systems and Software*, 172, 2021.

[49] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv:1705.06950*, 2017.

[50] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[51] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36, 2012.

[52] Gunnar A Sigurdsson, Olga Russakovsky, and Abhinav Gupta. What actions are needed for understanding human actions in videos? In *Proceedings of the IEEE international conference on computer vision*, pages 2137–2146, 2017.

[53] François Chollet et al. Keras. `https://keras.io`, 2015.

[54] NVIDIA. Tensorrt. `https://developer.nvidia.com/tensorrt`, 2020.

[55] Daniel Deniz, Francisco Barranco, Juan Isern, and Eduardo Ros. Reconfigurable cyber-physical system for lifestyle video-monitoring via deep learning. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1705–1712. IEEE, 2020.

[56] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 Int. Conference on Computer Vision*, pages 2556–2563. IEEE, 2011.

[57] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012.

[58] Imen Charfi, Johel Miteran, Julien Dubois, Mohamed Atri, and Rached Tourki. Optimized spatio-temporal descriptors for real-time fall detection: comparison of support vector machine and adaboost-based classification. *Journal of Electronic Imaging*, 22(4):041106, 2013.

[59] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.

[60] Yuya Yoshikawa, Jiaqing Lin, and Akikazu Takeuchi. Stair actions: A video dataset of everyday home actions. *arXiv:1804.04326*, 2018.

[61] Bureau of Labor Statistics and U.S. Department of Labor. Average hours per day spent in selected activities by age, 2019.