



Article

# Data Processing Unit for Energy Saving in Computer Vision: Weapon Detection Use Case

Marina Perea-Trigo <sup>1,\*</sup>, Enrique J. López-Ortiz <sup>2</sup>, Jose L. Salazar-González <sup>1</sup> , Juan A. Álvarez-García <sup>1</sup>   
and J. J. Vegas Olmos <sup>3</sup>

<sup>1</sup> Department of Languages and Computer Systems, Universidad de Sevilla, 41012 Sevilla, Spain

<sup>2</sup> Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, 41012 Sevilla, Spain

<sup>3</sup> NVIDIA Corporation, Ltd., Hermon Building, Yokneam 20692, Israel

\* Correspondence: mptrigo@us.es; Tel.: +34-652-137-075

**Abstract:** The growth of the Internet has led to the emergence of servers that perform increasingly heavy tasks. Some servers must remain active 24 h a day, but the evolution of network cards has facilitated the use of Data Processing Units (DPUs) to reduce network traffic and alleviate server workloads. This capability makes DPUs good candidates for load alleviation in systems that perform continuous data processing when the data can be pre-filtered. Computer vision systems that use some form of artificial intelligence, such as facial recognition or weapon detection, tend to have high workloads and high power consumption, which is becoming increasingly costly. Reducing the workload is therefore desirable and possible in some scenarios. The main contributions of this study are threefold: (1) to explore the potential benefits of using a DPU to alleviate the workload of a 24-h active server; (2) to present a study that measures the workload reduction of a CCTV weapon detection system and evaluate its performance under different conditions. We observed a 43,123% reduction in workload over the 24 h of video used in the experimentation, reaching more than 98% savings during night hours, which significantly reduces system stress and has a direct impact on electrical energy expenditure; and (3) to provide a framework that can be adapted to other computer vision-based detection systems.

**Keywords:** BlueField-2; data processing unit (DPU); deep learning; computer vision; weapon detection; green AI; energy saving



**Citation:** Perea-Trigo, M.; López-Ortiz, E.J.; Salazar-González, J.L.; Álvarez-García, J.A.; Vegas Olmos, J.J. Data Processing Unit for Energy Saving in Computer Vision: Weapon Detection Use Case. *Electronics* **2023**, *12*, 146. <https://doi.org/10.3390/electronics12010146>

Academic Editor: Antonio Lanata

Received: 28 October 2022

Revised: 22 December 2022

Accepted: 23 December 2022

Published: 28 December 2022



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Thanks to advances in Deep Learning, computer vision applications have reached heights that were unthinkable a decade ago. Deep Learning allows computational models composed of multiple layers to learn data representations with multiple levels of abstraction [1]. Over the years, the computational, energetic and infrastructure costs of Deep Learning architectures have grown to become unsustainable [2] and inaccessible to many small or medium companies that cannot pay the cost of computation (e.g., “DeepMind trained its system to play Go, it was estimated to have cost \$35 million”). In addition, in some real environments, usually in computer vision, inference tasks are designed to run real-time processing [3] and usually operate 24 h a day [4], which makes them energetically expensive. Especially since many of these models run on a Graphics Processor Unit (GPU), one of the most power-consuming components of a computer [5]. Add to this the increase in energy prices and constant fluctuations in these prices, and these systems can account for a large portion of the company’s total energy expenditure. This energy consumption has an impact not only on economics but also on the environment. Electricity consumption and its impact on climate change are one of humanity’s most significant challenges in the 21st century.

At the same time, the massive expansion in the use of the Internet and its data centers has caused servers to perform tasks that can be computationally expensive and to increase

the number of requests they have to handle [6]. In addition, many companies have started offering online services to perform tasks that used to be carried out on local machines. This increase in cloud computing and Internet use has increased demand for network performance. Standard NIC (Network Interface Card) devices assist the CPU in performing a variety of network functions, but in some scenarios, the tasks performed on the servers are too heavy, and even the combination of servers with standard NICs is not enough to provide performance that meets the speed demands of the network. To address these problems, an extension of the NIC device with the ability to solve more complex tasks was developed. These devices are called Data Processing Units (DPU) [7].

In this study, we will use a basic detection model running on the GPU with a DPU to reduce the working time and energy expenditure of a dedicated armed intrusion detection server designed to operate 24/7 on a CCTV.

Our key contributions are the following:

- We explore the potential benefits of using a DPU to alleviate the workload on a computer vision inference task.
- We present a study focused on measuring the reduced workload on a server dedicated to weapon detection on a real CCTV video stream recording.
- We offer a framework that could be adapted to any computer-vision-based detection system (<https://github.com/Deepknowledge-US/DPU-HumanDetection>, accessed on 28 September 2022).

The remainder of this paper is structured as follows, Section 2 presents the background to DPUs and discusses scenarios in which they have been used in the past. Section 3 explains the experimental setup, including the dataset used. Section 4 develops the approach followed, Section 5 contains experimentation, and finally, the findings and conclusions are drawn in Section 6.

## 2. Related Works

The most widely used definition of a DPU was coined by Alan Freedman (<https://www.computerlanguage.com/results.php?definition=Smart+NIC>, accessed on 28 September 2022) who described it as “A network interface card that offloads processing tasks that the system CPU would normally handle. Using its onboard processor, the DPU can perform any combination of encryption/decryption, firewall, and network processing, making it ideal for high-traffic web servers.”

We can find some examples of DPUs as workload relievers in the state-of-the-art. Liu et al. [7] conducted a study that focused on characterizing the network and computational aspects in a 100 Gbps Ethernet environment. The benchmark involved a series of tests that compared the performance of this DPU with that of a server. Although DPU failed to deliver server performance for specific tasks (intrusion detection, network address translation, virtual private networks), it was found to perform much better in encryption/decryption, memory operations that involve heavy contention, and interprocess communication tasks. In [8], it is demonstrated that the DPU is capable of offloading IPsec encryption to its hardware accelerators, which offer significant performance improvement compared to encrypting Ethernet traffic on workstation CPUs. This method is promising for improving network performance and adding confidentiality, integrity, and authentication to Ethernet traffic. The work of Barsellotti et al. [9] presents three different use cases for edge scenarios that exploit the recently introduced innovative programmability enabled by DPUs. Five ML models are, in turn, evaluated in the context of network intrusion detection, concluding that the use of DPUs enables the native integration of artificial intelligence-based operations directly into the network fabric.

In Deep Learning, DPU has been used in different phases of model training, such as data augmentation or model validation. With this approach, Jain et al. [10] were able to increase training performance by up to 15%. However, they did not find an offloading scheme that could optimally accelerate the training of all models in their study. In an extension of this work [11], the same authors show a consistent improvement for Convolu-

tional Neural Networks (CNNs) and Transformer models with weak and strong scaling on multiple nodes.

Many proposals can be found concerning computer vision detection, but they generally focus on detecting medium or large objects such as vehicles, cyclists, pedestrians or animals [12–15]. However, the detection of weapons by computer vision adds some difficulties, among other things, due to the small size of the objects to be detected [16,17]. In [18], Salazar et al. propose a challenging dataset that included these difficulties and a Deep Learning weapon detection model based on a Faster-RCNN (Region Convolutional Neural Network) with FPN (Feature Pyramid Network), which improved the state-of-the-art in two-phase training.

As in [10,11], our proposal makes use of the DPU in a Deep Learning environment, but unlike those where it is used in the training phases of the model, in our case, the card will be used to perform filtering tasks to help an already trained model to reduce the inference workload. We were motivated to try using a DPU as a filter for a video stream because of its ability to alleviate the load on the system. However, because the tasks and objectives of our experiment differed from those described in previously mentioned DPU articles, it is non-viable to make a direct comparison. Our goal in using a DPU as a filter was to reduce the working time of a server dedicated to video stream processing.

In previous works [19,20], frameworks have been proposed that use frame filtering-based methods to skip similar consecutive frames in order to reduce the computational cost of detection-driven applications using reinforcement learning. However, these solutions cannot be applied to our study on detecting weapons in public spaces because the appearance of weapons in such contexts is anomalous, making it impossible to perform reinforcement training. Additionally, using skip-length techniques in this context could potentially decrease detection accuracy. As far as we know, our work is the only one that uses a DPU device to reduce server workload and decrease power consumption in a CCTV weapon detection system.

### 3. Experimental Setup

#### 3.1. Implementation Details

To carry out the experimentation, we have used a machine with 32 GB of RAM, an AMD Ryzen™ Threadripper™ PRO 3955WX CPU and an NVIDIA 3090 ([https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/?nvid=nv-int-solr-791893#cid=\\_nv-int-solr\\_en-us](https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/?nvid=nv-int-solr-791893#cid=_nv-int-solr_en-us), accessed on 28 September 2022) GPU card with 24 GB of RAM. On this host, we have installed an Nvidia Mellanox BlueField-2 DPU (<https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-2-dpu.pdf>, accessed on 28 September 2022), which has 8 ARM cores and 16 GB DDR4 DRAM. These cards have previously been used in the state-of-the-art and tested in fields such as task offloading, isolation, and acceleration for security, networking, and storage [7,10,21].

A motion detection algorithm will be used on the DPU side, while an object detection model will be used on the server side, specifically a weapon detection model. In Deep Learning tasks, especially tasks focused on computer vision, it is common to use models running on the GPU since these devices spend less time on training and inference than a CPU, thus preserving real-time detection. We have chosen to use the GPU model described in Salazar et al. [18] because it has previously demonstrated high accuracy and fast detection times in CCTV weapon detection tasks, but it is worth noting that our methodology is not tied to this specific model and can be adapted to use a different model if needed. Therefore, we will not delve into the model's inner workings, as this can be found in the original article.

In [18], it was reported that the weapon detection model required 90 milliseconds to analyse a single frame using an NVIDIA GeForce GTX-1080Ti GPU. However, when using a more powerful NVIDIA RTX 3090 GPU, the model's processing time was significantly reduced to 47 milliseconds.

### 3.2. Dataset

To further test the feasibility of our proposal, we sought out a scenario where it would be beneficial to have a continuous surveillance system in place. While shootings in public buildings are thankfully non-casual events, it is critical to have early detection systems in place to minimise damage as much as possible if one does occur. In the final implementation, our hardware and software will be used to analyse a CCTV video stream in real-time, but for this experimentation, we have used a recording of a public video stream from the University of Michigan: the camera in the clock tower (<https://www.mtu.edu/webcams/clock/>, accessed on 28 September 2022). This camera was chosen for several reasons. Firstly, it is located in a public institution where security systems are critical. Secondly, the university setting provides a varied flow of people, with periods of high traffic and periods of low traffic. Finally, the building is closed at night, providing a period with practically no foot traffic. This variability in foot traffic allows us to evaluate the performance of our proposal under three different traffic conditions (high, medium, and low). We recorded this camera over 24 h, resulting in a video with a resolution of  $1024 \times 768$  pixels, a frame rate of 32 FPS, and 24 h. The position of this camera can be observed in Figure 1.



**Figure 1.** Clock tower camera.

On average, our server requires 47 milliseconds to process a single frame, so real-time processing is not feasible with a 32 FPS video stream. This processing time results in a backlog of frames waiting to be processed, and this backlog grows more prominent as the video continues, causing high stress on the server for the duration of the video and a long delay in the inference time. This backlog is a common problem in computer vision, and several partial solutions are available. One approach is to discard some frames and periodically process a subset of them in the stream [22]. This frame discarding must be adjusted based on the server's processing speed to avoid overflow. For example, if a server requires 0.05 s (on average) to process a single frame, it will not be able to process more than 18–20 frames per second, which is often sufficient for successful task completion. Another solution involves load sharing, such as the architecture for distributed computing of a real-time video encoding application presented in [23]. Given that we only have one server capable of weapon detection, we have chosen to use the frame-discarding approach and work with a 16 FPS stream, which allows the server to process the stream in real time.

## 4. Using DPU as Server-Intensive Task Offloader

### 4.1. Motivation: Reduce Server Workload on Computer Vision Tasks

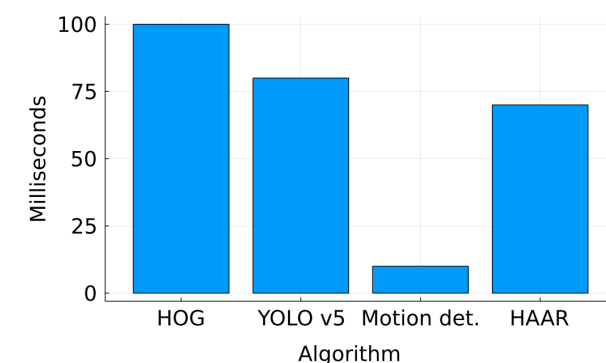
Since the server takes 47 milliseconds to process one frame, the detection model can handle 16 frames per second, leading to a server running 752 milliseconds every second. In other words, the GPU works about 75.2% of the time in a 24/7 system, which is a significant power consumption.

Given the results of [7,10,18], we decided to explore the advantages of using a DPU in this computer vision scenario. We aim to install a DPU on the server to reduce the

server's working time by using the DPU as a traffic light that alerts the server when needed. To do this, we need a mechanism to discriminate the video frames and classify them based on whether they need to be processed. Although the range of such mechanisms is not very wide, we have considered two options that could be effective. The first is to use a human detection model, as weapons need to be carried by people. Therefore, when the DPU's human detection model classifies a frame as positive, it will alert the server to apply weapon detection. Another option is to install a motion detector in the DPU that alerts the server when it detects changes in the image. The latter mechanism is usually faster, but it also has a slightly lower hit rate as it may be triggered by false positives, such as the movement of leaves or animals entering the camera's range of vision.

#### 4.2. Our Approach

To test these mechanisms on the DPU side, we decided to use OpenCV, a widely used library whose effectiveness has been previously demonstrated in literature [24,25]. This library has functions based on frame subtraction that we have used to perform motion detection. OpenCV also has some detection algorithms we have used in this work to test human detection performance on the DPU, such as the Histogram of Oriented Gradient (HOG) or its Haar cascade classifier algorithm [26,27], which are based on the gradient. We also tested other classification algorithms, such as Yolo, a popular Deep Learning object detector known for its excellent performance in terms of time and accuracy [28–30]. Nonetheless, Yolo is designed to run on GPU, and its performance on CPU (or DPU, as in our case) decreases significantly in terms of time. Figure 2 shows a comparison of the time required for each algorithm to analyse a single frame.



**Figure 2.** Processing time of each algorithm for a single frame in the DPU.

As we can observe from the comparison, motion detection takes significantly less time to perform than methods based on human detection, which is ideal for a device such as a DPU that is not powerful enough to support computationally expensive methods. Among the human detection algorithms, the Haar algorithm is the fastest, taking 70 milliseconds to process a single frame. The Haar cascade classifier algorithm [31] uses a cascading window approach and attempts to compute features in every window and classify whether it could be a human. However, even with this algorithm, we would not be able to process the 16 FPS of the  $1280 \times 720$  resolution video in real time. Therefore, we have decided to implement the motion detection algorithm on the DPU side, which only takes ten milliseconds per frame to perform the detection.

The motion detection algorithm applied is based on the classical Inter-frame difference [32] that uses the absolute value of the difference obtained from the subtraction of two consecutive frames (Equation (1)) to determine the movement using an established threshold (Equation (2)). The algorithm detects a movement if the difference is greater than the threshold.

$$\Delta_t = \sum_{u,v} |\text{Frame}_t(u, v) - \text{Frame}_{t-1}(u, v)| \quad (1)$$

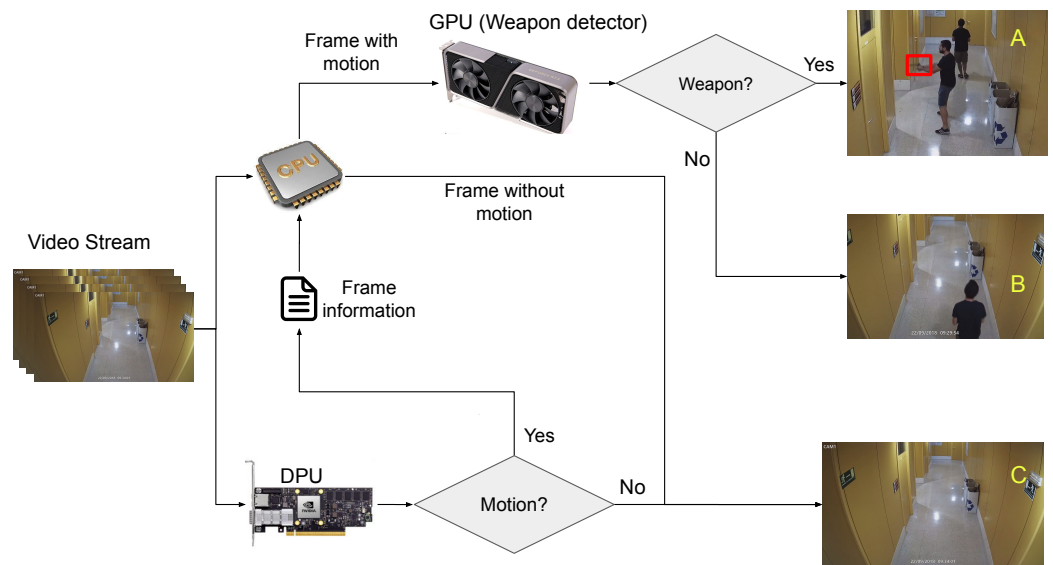
$$M_t = \begin{cases} \text{Frame}_t & , \Delta_t \geq T \\ 0 & , \Delta_t < T \end{cases} \quad (2)$$

The function  $\Delta_t$  calculates the absolute value of the difference between two consecutive frames ( $\text{Frame}_t$  and  $\text{Frame}_{t-1}$ ), where  $u$  and  $v$  represent the coordinates of a pixel in an image. This  $\Delta_t$  is calculated to determine motion ( $M_t$ ) based on the threshold  $T$  (empirically established for each camera using video intervals where no motion occurs).

The next step was to find a way to provide the necessary information to the server where fine detection is performed. When a server uses a DPU as a network card, two modes of operation are possible: either the information arrives at the server after passing through the DPU, or the information arrives simultaneously at both the server and the DPU. The first mode allows us to read the video stream on the DPU and, when fine processing is needed, bring the entire frame to the server using gRPC (Google Remote Procedure Calls). However, exchanging frames between the card and the server is a slow and cumbersome task, so we decided to use the second mode and have both the server and the DPU read the video stream. The following figure shows a graph of the information flow in our approach. By default, only the DPU analyses each frame. When the DPU detects motion in a frame, it passes the frame identifier to the CPU, and upon receipt, the GPU applies weapon detection to that frame.

In our final implementation, both the server and the DPU read the video stream, where each frame has an associated time identifier. However, instead of the server constantly applying the weapon detection model to each frame, the DPU does the hard work. As the frames arrive at the DPU, the DPU applies the motion detection model. If no motion is detected, it is unnecessary to perform fine processing of the frame, and the server will not apply the weapon detector to it. However, if any motion is detected, the DPU notifies the server and passes information (via gRPC) about the frame identifier. The server is also reading the video stream, so sending the frame from the DPU to the server is unnecessary; the frame identifier being the only one necessary. From a weapon detection perspective, type B and C frames are discarded, as no weapon has been detected in them; in fact, type C frames have not even applied the weapon detection model.

As shown in Figure 3, the action frames are from an indoor camera and belong to a public dataset [18] containing a set of frames tagged during a mock attack. For privacy reasons, no high-quality public streams are available for identifying people, so the tests were performed using the public video stream from the University of Michigan. Although detecting weapons at such long distances is unlikely, this public stream allows us to test the solution's efficiency using a DPU, and the system could be seamlessly transferred to private CCTV streams. Measurements were taken every 10 s using the "powertop" tool (<https://github.com/fenrus75/powertop>, accessed on 28 September 2022) and "nvidia-smi" command (<https://developer.nvidia.com/nvidia-system-management-interface>, accessed on 28 September 2022), respectively, to monitor the energy consumption of both the DPU and the GPU.



**Figure 3.** Video stream processing flow. Both the DPU and the CPU read the video stream. GPU applies fine processing only when motion is detected. Frames belong to the public dataset from Salazar et al. [18].

### 5. Results

As discussed in previous sections, our model uses a GPU to perform the processing. During the execution of the model, this card runs at full power, reaching its maximum power consumption level. However, since the server does not have a graphics interface, the power consumption of the graphics card is minimal when it is not used for video processing. It is crucial to consider that the benefits of workload reduction and energy conservation will depend on the flow of people, which implies whether there are activity periods. In a scenario with no flow of people, the most significant savings will be achieved as the GPU’s energy cost will be minimised to zero. On the other hand, in a scenario with a constant flow of people (i.e., activity in 100% of the frames), there will not only be any savings, but the energy cost of the DPU will also need to be added to the energy cost of the GPU. To determine the point at which this methodology becomes less energy effective, we will calculate the percentage of time with activity at which GPU and DPU consumption is higher than a traditional GPU-only system.

$$C_{inactivity} + C_{activity} < C_{GPU} \tag{3}$$

As shown in Equation (3), the consumption produced during inactivity periods, along with consumption during periods of activity in which the DPU and GPU are working, should be lower than the consumption of a traditional detection system without filtering.

In the following equation, the percentage calculation is shown in detail:

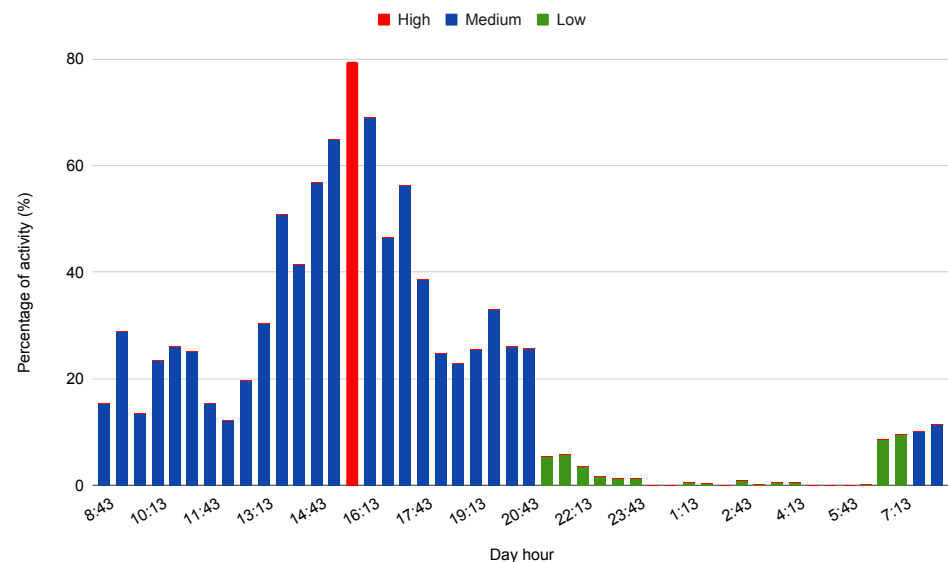
$$\theta \cdot (C_{GPU} + C_{DPU}) + (1 - \theta) \cdot C_{DPU} < C_{GPU} \tag{4}$$

where  $(1 - \theta)$  is the time percentage in which there is no motion, and  $\theta$  is the remaining period when DPU does detect motion. Moreover,  $C_{DPU}$  and  $C_{GPU}$  are the power consumption of both devices. Therefore, considering that GPU’s average consumption is 274.18 kWh and 6.4 kWh for the DPU, we have obtained that if the period of activity is above 97.67% of the time, our system is not energetically efficient. For example, in 24 h, it would be necessary to be motion over 23.5 h for both DPU and GPU consumption to be higher than a traditional GPU-only system.

Based on the flow of people, we have established two thresholds and the following ranges of low, medium, and high savings as follows:

- **High-Activity:** Busy scenario with a high flow of people. An interval of time is classified as High-Activity when there are people in the camera's viewing range for more than 75% of the time. In this situation, the stress on the server side could be as intense as usual.
- **Medium-Activity:** Not very busy scenario with an intermittent flow of people. We consider an interval of time to belong to this range of activity when the time at which there are people within the camera vision range is between 10% and 75%.
- **Low-Activity:** Clear or no-people-flow scenario. When there are people in the area of view of the camera for less than 10% of the total time.

The 24-h recording provided by the clock tower camera gives us intermediate activity for much of the day, but it also has intervals of time when the scenario has high-activity (e.g., school start time) and intervals of low-activity where the flow of people is virtually non-existent, such as in the night hours. However, security systems should remain active throughout the day in a public building. Figure 4 shows the percentage of GPU activity recorded throughout the day on the clock tower camera. Half-hour intervals have been considered.



**Figure 4.** Server activity in 30-min time windows over a day.

According to Figure 5, there is a direct relationship between server workload and energy consumption. When we let the server do the work, we get an average power consumption of 274.18 kWh as the frames are being processed non-stop for the entire process. However, when we use pre-filtering, we obtain a drastic reduction in consumption because specific frames do not need to be processed on the server. In the low-activity time interval (from 9:00 p.m. to 7:00 a.m.), only the DPU performs video processing, reducing the energy consumption of the entire system. We also have a small peak of high-activity between 15:45 and 16:15 due to the high flow of people. Note that there is a slight increase in power consumption during busy periods compared to the server working alone. This increase is because, in this situation, both the DPU and the GPU are working, so the consumption of the DPU has to be added to that of the server. However, due to the low power consumption of the DPU, this increase is not too high.



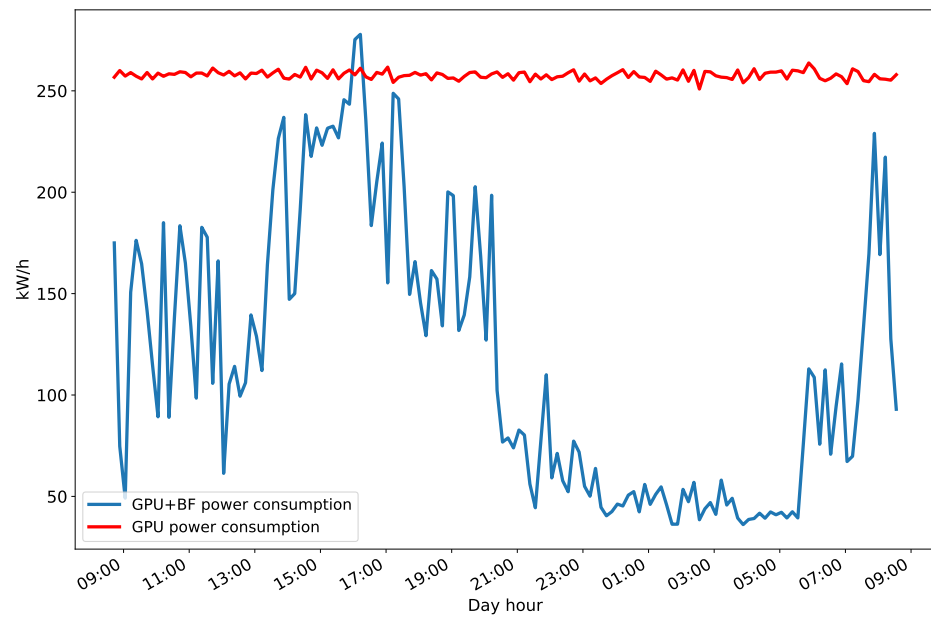


Figure 5. Energy consumption throughout the day.

Figure 6 shows the accumulation of the values in Figure 5. As expected, the constant consumption of the server working alone results in an almost uniform curve because at each instant of the process, the consumption is practically the same because the server analyses all the video frames. However, using DPU to preprocess the video reduces the number of frames in which fine processing is needed.

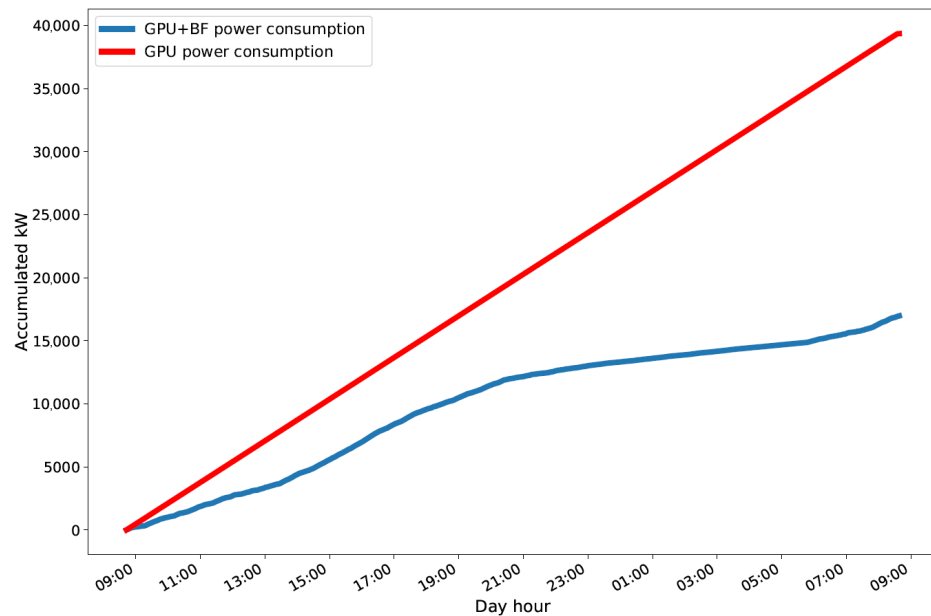
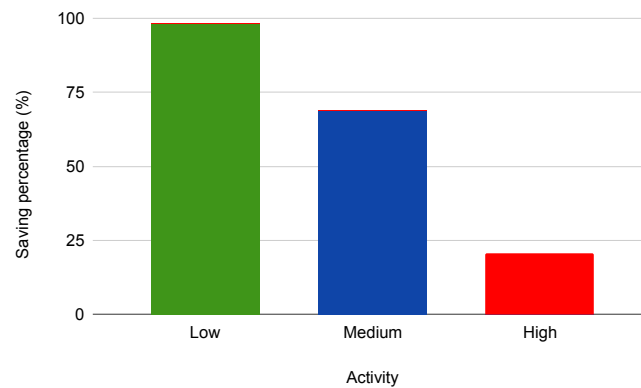


Figure 6. Energy consumption accumulation throughout the day.

In the recording used for the experimentation, a value of 39,353.58 kWh was obtained in the case of the server working alone and 16,970.57 kWh in the case of the server assisted by the DPU, which represents a saving of 43.12% over the 24 h recording when using this methodology. This difference in power consumption will depend on the scenario since, as mentioned above, the increase or decrease in the flow of people will have a direct relationship with the server’s workload and, consequently, with the power consumption.

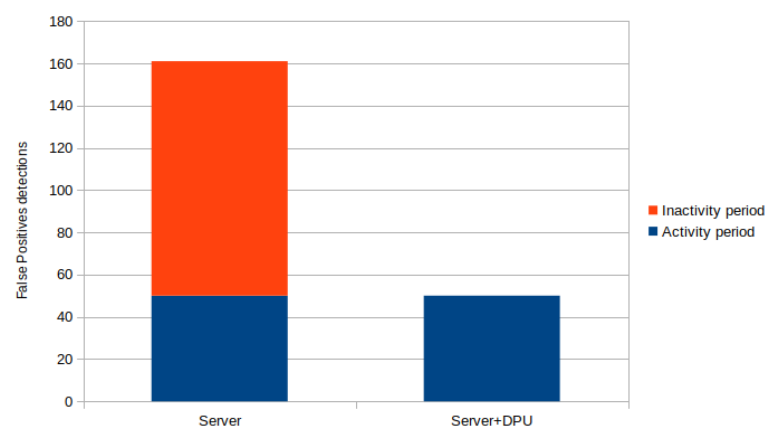
As shown in Figure 7, the most significant percentage of savings occurred during periods of low pedestrian traffic when detection tasks did not heavily tax the GPU.



**Figure 7.** Percentage of energy savings by activity ranges when using the BlueField.

In the low-activity intervals (which in our case make up 41.67% of the hours of the day), the savings achieved are enormous, reaching 98.07%. However, in the high-activity range, no significant savings have been achieved, as the number of frames to be analysed is very similar to those analysed by the server without the help of the DPU. In the video used in our test, a saving of 20.54% was achieved in the high-activity time intervals (which represent 2.08% of the total time of the day). For the medium-activity periods, an average saving of 68.7% has been achieved using pre-filtering (medium-activity intervals account for 56.35% of the total hours of the day).

As a final experimental study, the false positive ratio of our system was tested for the 24 h of video. As shown in Figure 8, many false positives occur during periods of inactivity, which is normal because this period is more prevalent than activity ones. These false detections in periods of inactivity would result in multiple incorrect alarms, as for a weapon to appear on the scene, movement by people must occur. Therefore, besides saving energy for the system, the method proposed in this study would avoid a higher rate of false positives.



**Figure 8.** False positive detections during activity and inactivity periods.

## 6. Conclusions and Future Work

As far as we know, this is the first time a DPU has been used as a pre-filter in a computer vision detection task. The code used in this work is available on our GitHub account and can be adapted for use in other computer vision systems where the pre-filtering of input data is possible. As demonstrated in the previous sections, using a pre-filtering device can significantly reduce server workload and energy consumption. In our tests,

energy savings ranged from 20.54% during periods of high-activity, to 68.7% during periods of medium-activity and up to 98.07% during periods of low-activity. This methodology is beneficial when there is no constant flow of people, but active surveillance is necessary at all times, such as in universities, public institutions, military buildings, and museums.

Additionally, the use of DPUs as load balancers allows for the distribution of workload in real-time video processing, as discussed in [23]. While we could not test this method due to only having one server capable of weapon detection, it is worth noting that the DPUs could distribute video frames among multiple servers rather than discarding them.

**Author Contributions:** Conceptualization, J.A.Á.-G., J.J.V.O.; Funding acquisition, J.A.Á.-G., J.J.V.O.; Investigation, M.P.-T., E.J.L.-O., J.L.S.-G.; Methodology, M.P.-T., E.J.L.-O., J.L.S.-G.; Software, M.P.-T., E.J.L.-O., J.L.S.-G.; Writing-original draft, M.P.-T., E.J.L.-O., J.L.S.-G. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research has been partially funded through the ID2PPAC project under contract 101007254 by the European Commission, the DISARM project (Automatic detection of armed individuals' Grant n. PDC2021-121197), and the HORUS project ('Hybrid Object Recognition Platform for Weapon Seeking' Grant n. PID2021-126359OB-I00) funded by MCIN/AEI/ 10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR".

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data available in a publicly accessible repository. The data presented in this study is openly available in <https://github.com/Deepknowledge-US/DPU-HumanDetection> (accessed on 27 October 2022).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
2. Thompson, N.C.; Greenewald, K.; Lee, K.; Manso, G.F. Deep learning's diminishing returns: The cost of improvement is becoming unsustainable. *IEEE Spectr.* **2021**, *58*, 50–55. [CrossRef]
3. Lee, C.Y.; Lin, S.J.; Lee, C.W.; Yang, C.S. An efficient continuous tracking system in real-time surveillance application. *J. Netw. Comput. Appl.* **2012**, *35*, 1067–1073. [CrossRef]
4. Esteve, M.; Palau, C.; Martínez-Nohales, J.; Molina, B. A video streaming application for urban traffic management. *J. Netw. Comput. Appl.* **2007**, *30*, 479–498. [CrossRef]
5. Collange, S.; Defour, D.; Tisserand, A. Power consumption of GPUs from a software perspective. In *Computational Science—ICCS 2009*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 914–923.
6. Gupta, O.; Raskar, R. Distributed learning of deep neural network over multiple agents. *J. Netw. Comput. Appl.* **2018**, *116*, 1–8. [CrossRef]
7. Liu, J.; Maltzahn, C.; Ulmer, C.; Curry, M.L. Performance Characteristics of the BlueField-2 SmartNIC. *arXiv* **2021**, arXiv:2105.06619.
8. Diamond, N.; Graham, S.; Clark, G. Securing InfiniBand Networks with the Bluefield-2 Data Processing Unit. In Proceedings of the 17th International Conference on Cyber Warfare and Security (ICWS 2022), Albany, NY, USA, 17–18 March 2022; pp. 459–468. [CrossRef]
9. Barsellotti, L.; Alhamed, F.; Vegas Olmos, J.J.; Paolucci, F.; Castoldi, P.; Cugini, F. Introducing Data Processing Units (DPU) at the Edge [Invited]. In Proceedings of the 2022 International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 25–28 July 2022; pp. 1–6. [CrossRef]
10. Jain, A.; Alnaasan, N.; Shafi, A.; Subramoni, H.; Panda, D.K. Accelerating CPU-based distributed DNN training on modern HPC clusters using bluefield-2 DPUs. In Proceedings of the 2021 IEEE Symposium on High-Performance Interconnects (HOTI), Santa Clara, CA, USA, 18–20 August 2021; pp. 17–24.
11. Jain, A.; Alnaasan, N.; Shafi, A.; Subramoni, H.; Panda, D.K. Optimizing distributed dnn training using cpus and bluefield-2 dpus. *IEEE Micro* **2021**, *42*, 53–60. [CrossRef]
12. Arcos-Garcia, A.; Alvarez-Garcia, J.A.; Soria-Morillo, L.M. Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing* **2018**, *316*, 332–344. [CrossRef]
13. Srivastava, S.; Narayan, S.; Mittal, S. A survey of deep learning techniques for vehicle detection from UAV images. *J. Syst. Archit.* **2021**, *117*, 102152. [CrossRef]

14. Kocamaz, M.K.; Gong, J.; Pires, B.R. Vision-based counting of pedestrians and cyclists. In Proceedings of the 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Placid, NY, USA, 7–10 March 2016; pp. 1–8. [CrossRef]
15. Dhulekar, P.A.; Gandhe, S.T.; Bagad, G.R.; Dwivedi, S.S. Vision Based Technique for Animal Detection. In Proceedings of the 2018 International Conference On Advances in Communication and Computing Technology (ICACCT), Sangamner, India, 8–9 February 2018; pp. 344–348. [CrossRef]
16. Debnath, R.; Bhowmik, M.K. A comprehensive survey on computer vision based concepts, methodologies, analysis and applications for automatic gun/knife detection. *J. Vis. Commun. Image Represent.* **2021**, *78*, 103165. [CrossRef]
17. Castillo, A.; Tabik, S.; Pérez, F.; Olmos, R.; Herrera, F. Brightness guided preprocessing for automatic cold steel weapon detection in surveillance videos with deep learning. *Neurocomputing* **2019**, *330*, 151–161. [CrossRef]
18. Salazar González, J.L.; Zaccaro, C.; Álvarez-García, J.A.; Soria Morillo, L.M.; Sancho Caparrini, F. Real-time gun detection in CCTV: An open problem. *Neural Netw.* **2020**, *132*, 297–308. [CrossRef] [PubMed]
19. Arefeen, M.A.; Nimi, S.T.; Uddin, M.Y.S. FrameHopper: Selective Processing of Video Frames in Detection-driven Real-Time Video Analytics. *arXiv* **2022**, arXiv:2203.11493. <https://doi.org/10.48550/ARXIV.2203.11493>.
20. Li, Y.; Padmanabhan, A.; Zhao, P.; Wang, Y.; Xu, G.H.; Netravali, R. Reducto: On-camera filtering for resource-efficient real-time video analytics. In Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, Virtual, 10–14 August 2020; pp. 359–376.
21. Karamati, S.; Young, J.; Conte, T.; Hemmert, K.; Grant, R.; Hughes, C.; Vuduc, R. *Computational Offload with BlueField Smart NICs*; Technical Report; Sandia National Lab. (SNL-NM): Albuquerque, NM, USA, 2021.
22. Liu, D.; Cui, Y.; Chen, Y.; Zhang, J.; Fan, B. Video object detection for autonomous driving: Motion-aid feature calibration. *Neurocomputing* **2020**, *409*, 1–11. [CrossRef]
23. Watanabe, H.; Ghatp, A.; Nakazato, H. Distributed Computing for Real-time Video Processing. 2003. Available online: <https://www.ams.giti.waseda.ac.jp/data/pdf-files/2003ICUC-hw.pdf> (accessed on 27 October 2022).
24. Pranav, K.; Manikandan, J. Design and Evaluation of a Real-time Pedestrian Detection System for Autonomous Vehicles. In Proceedings of the 2020 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia, 26–27 May 2020; pp. 155–159. [CrossRef]
25. Farag, W. A lightweight vehicle detection and tracking technique for advanced driving assistance systems. *J. Intell. Fuzzy Syst.* **2020**, *39*, 2693–2710. [CrossRef]
26. Soo, S. Object detection using Haar-cascade Classifier. *Inst. Comput. Sci. Univ. Tartu* **2014**, *2*, 1–12.
27. Setjo, C.H.; Achmad, B. Thermal image human detection using Haar-cascade classifier. In Proceedings of the 2017 7th International Annual Engineering Seminar (InAES), Yogyakarta, Indonesia, 1–2 August 2017; pp. 1–6.
28. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
29. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767. <https://doi.org/10.48550/ARXIV.1804.02767>.
30. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. <https://doi.org/10.48550/ARXIV.2004.10934>.
31. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, Kauai, HI, USA, 8–14 December 2001; Volume 1, p. I.
32. Lipton, A.J.; Fujiyoshi, H.; Patil, R.S. Moving target classification and tracking from real-time video. In Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision, WACV'98 (Cat. No. 98EX201), Princeton, NJ, USA, 19–21 October 1998; pp. 8–14.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.