Journal of Internet Services and Applications, 2025, 16:1, doi: 10.5753/jisa.2025.5156

People Counting Application with Crowded Scenarios: A Case Study with TV Boxes as Edge Devices

Gabriel Massuyoshi Sato [**Universidade Estadual de Campinas** | ***g172278@dac.unicamp.br***]
Gustavo P. C. P. da Luz [**Universidade Estadual de Campinas** | ***g271582@dac.unicamp.br***]
Luis Fernando Gomez Gonzalez [**Universidade Estadual de Campinas** | ***gonzalez@unicamp.br***]
Juliana Freitag Borin [**Universidade Estadual de Campinas** | ***jufborin@unicamp.br***]

Instituto de Computação, Universidade Estadual de Campinas (Unicamp), Av. Albert Einstein, 1251, Cidade Universitária, Campinas, SP, 13083-852, Brazil.

Received: 03 November 2024 • **Accepted:** 07 May 2025 • **Published:** 06 August 2025

Abstract Counting people in various urban spaces using artificial intelligence enables a wide range of smart city applications, enhancing governance and improving citizens' quality of life. However, the rapid expansion of edge computing for these applications raises concerns about the growing volume of electronic waste. To address this challenge, our previous work demonstrated the feasibility of repurposing confiscated illegal TV boxes as Internet of Things (IoT) edge devices for machine learning applications, specifically for people counting using images captured by cameras. Despite promising results, experiments in crowded scenarios revealed a high Mean Absolute Error (MAE). In this work, we propose a patching technique applied to YOLOv8 models to mitigate this limitation. By employing this technique, we successfully reduced the MAE from 8.77 to 3.77 using the nano version of YOLOv8, converted to TensorFlow Lite, on a custom dataset collected at the entrance of a university restaurant. This work presents an effective solution for resource-constrained devices and promotes a sustainable approach to repurposing hardware that would otherwise contribute to electronic waste.

Keywords: Internet of Things, Smart Cities, People Counting, Crowded Scenarios, Sustainability, TV Boxes, Repurposing.

1 Introduction

People counting is a computational task with diverse applications across various domains. In smart cities, accurate counting and detection of people are essential for enhancing public safety and mobility, managing disasters, and optimizing tourism strategies [Gao *et al.*, 2024]. The ability to make informed decisions based on real-time crowd detection can significantly improve urban planning and resource allocation [Collini *et al.*, 2024]. Moreover, this capability provides citizens with valuable information about the presence of queues or crowds in public spaces and events.

Although sensor-based solutions for people counting have been utilized [Metwaly *et al.*, 2020; Tang *et al.*, 2020], there is a growing trend toward artificial intelligence-driven approaches based on video analysis [Gao *et al.*, 2024]. Commonly, these Artificial Intelligence (AI) workloads for video analytics were conducted in the cloud; however, with the emergence of edge intelligence - integrating edge computing with AI - these workloads can now be shifted to the network edge. This transition enhances latency, privacy, and security while providing bandwidth savings [Badidi *et al.*, 2023]. Edge AI enables processing closer to the point of video capture using devices such as smartphones, single-board computers (SBCs), Internet of Things (IoT) devices, and edge servers [Himeur *et al.*, 2024].

While a smart city powered by edge AI can enhance governance and improve citizens' quality of life, it also raises significant concerns about energy demand and electronic waste (e-waste) production stemming from the widespread use of intelligent devices in urban environments [Dwivedi *et al.*, 2022]. In 2022, a concerning 62 million tonnes (Mt) of electronic waste were generated, representing an 82% increase since 2010. Projections indicate this figure could rise by another 32%, reaching 82 million tonnes by 2030 [Baldé *et al.*, 2024]. Such growth contradicts the Sustainable Development Goals (SDGs), particularly SDG 12.5, which states: "*By 2030, substantially reduce waste generation through prevention, reduction, recycling, and reuse*" [Nations, 2015].

In light of this context, this work investigates the repurposing of TV boxes as AI edge devices for the task of people counting. The Brazilian Federal Revenue Service (known as *Receita Federal* - RFB) confiscates thousands of TV boxes each year due to illegal imports, and it is estimated that around 2.5 million of these devices are currently stored in warehouses. By repurposing these devices, we can prevent them from becoming e-waste. Additionally, demonstrating their viability for this task allows us to avoid the costs - both financial and environmental - associated with purchasing or manufacturing new edge devices.

In a previous study [Sato *et al.*, 2024], we examined the feasibility of repurposing TV boxes as edge devices employing machine learning algorithms in smart city scenarios. The case study focused on a people-counting application that used images captured by a camera positioned in front of Unicamp's university restaurant to inform users about the access queue. The results demonstrated that two models of TV boxes were capable of running machine learning models with performance comparable to two widely used single-board computers commonly employed in IoT applications. Additionally, we found that converting a YOLOv8 model to the TensorFlow Lite (TFLite) format [TensorFlow Lite, 2024] significantly reduced memory usage compared to the original version, making it a viable option for all edge devices tested in the study, while meeting latency, CPU usage, and machine learning performance requirements.

One key challenge in people-counting applications is maintaining accuracy in crowded scenarios, where error rates tend to increase significantly. Our initial study did not specifically address this limitation. In the present work, we tackle this issue by investigating the performance boundaries of both larger and smaller models to assess their continued viability for the same application and hardware platforms.

Accordingly, one of the main contributions of this study is the proposal of a patching technique that enhances people-counting accuracy in crowded environments without relying on more complex models such as YOLOv8x. This method enables accurate inference even on resource-constrained hardware, including Raspberry Pi devices and repurposed TV boxes. Furthermore, our results show that the computational and time overhead introduced by applying the patching technique to lightweight models like YOLOv8n remains acceptable, with inference times under one minute on the most constrained devices tested.

In summary, the main contributions of this extended work are:

- the proposal of a patching method that improves accuracy in people counting;
- the demonstration that the patching method, when applied to lightweight models such as YOLOv8n, is compatible with the computational constraints of low-resource hardware;
- the validation that the processing time required for patching remains within acceptable limits for typical people-counting scenarios, even on constrained devices.

The remainder of the paper is organized as follows: Section 2 reviews related work; Section 3 details the proposed technique for improving people-counting accuracy in crowded scenarios; Section 4 describes the experimental setup; Section 5 presents and discusses the results; and finally, Section 6 concludes the paper and outlines directions for future research.

2 Related Work

This section compares related work on people counting, focusing on vision-based and non-vision-based approaches, the datasets used, and the methods employed.

[Gao *et al.*, 2024] discusses the existing techniques for crowd counting approaches in IoT environments and other applications such as healthcare and biotechnology, presenting the challenges in complex environments where the task is to count the number of existing people in a video or an image. The authors define three types of approaches: people estimation, object estimation and density estimation. Some IoT solutions involve people counting based on sensors in various forms, such as thermal, passive infra-red, ultrasonic and motion sensors.

One example of non-vision-based approaches is the work of [Tang *et al.*, 2020], which explored the feasibility of Passive WiFi Radar (PWR) for occupancy sensing and people counting, although it is limited to small crowds. Their method utilized Wi-Fi received signal strength and channel state information in a system that requires no modifications to the access point. Conducted in an office environment with four subjects, the system used a modified LeNet neural network.

Moving to vision based solutions for people counting, [Ruchika *et al.*, 2022] provided an evaluation of YOLOv5 in crowd counting applications, showing that it performs well on low and medium density ranges but not in a very dense crowd. On the highly crowded Shanghaitech dataset, the model's MAE was 11 percentage points higher compared to less crowded datasets.

[Bhatt *et al.*, 2024] tested some object detection models such as Faster R-CNN, YOLO, and SSD models. YOLO displayed the lowest MAE at the Shanghaitech dataset combined with a custom crowd counting dataset. The authors followed an approach of training the models, which can enhance the results however require additional steps than using pre-trained options.

One work that shows a different approach to enhance people counting is the one of [Liu *et al.*, 2019], that proposes a novel post-processing method (Adaptative Non Maximum Supression - NMS) to refine the model output based on a dynamic threshold according to the density of instances. Their method combined with one-stage and two-stage detectors was claimed to achieve state of the art results on the CityPersons and CrowdHuman datasets.

Some authors have already explored the use of TV Boxes as IoT devices. For instance, [Moreira *et al.*, 2022] proposed utilizing TV Boxes as edge devices for detecting plant diseases through image analysis, while [Shu and Shu, 2021] developed a system that detects falls to prevent injuries in settings such as senior homes and nursing facilities. In a previous work [da Luz *et al.*, 2025] we examined the resilience and robustness of TV Boxes by creating a testbed with 20 devices to evaluate their performance and carbon footprint. Our findings indicate that these devices could perform over 16 million inferences during three weeks of testing under stress conditions. This raises the discussion of how TV boxes can surpass commercially available devices in terms of carbon footprint when using the Brazilian energy matrix.

The aforementioned works encompass a variety of people-counting approaches, including both vision-based and non-vision-based methods. We also review studies that explore the use of TV boxes for computational tasks, as well as research focused on improving the accuracy of people-counting models. The present work combines these aspects by employing an enhanced vision-based people-counting technique on TV boxes, specifically aiming to improve detection performance in crowded scenarios. Table 1 summarizes the related works comparing their approach, dataset, and method used.

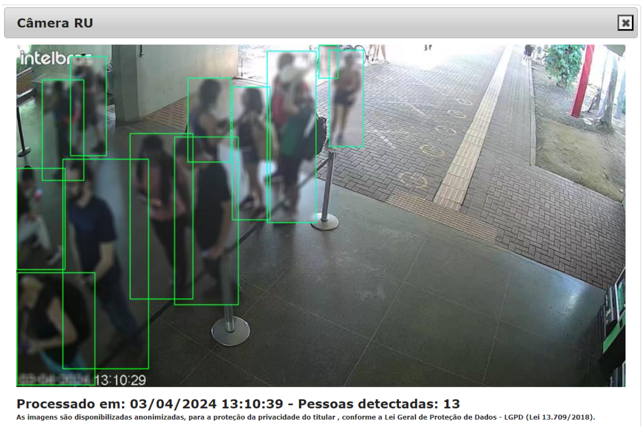
Table 1. Comparison of related works based on approach, dataset, and method.

Work	Approach	Dataset	Method
[Gao *et al.*, 2024]	Vision & Sensors	Multiple (review study)	People, Object, and Density Estimation
[Tang *et al.*, 2020]	Non-Vision (Wi-Fi)	Office Environment	Passive WiFi Radar for People Counting
[Ruchika *et al.*, 2022]	Vision	Shanghaitech + Others	Image Analysis for People Counting
[Bhatt *et al.*, 2024]	Vision	Shanghaitech + Custom	Image Analysis for People Counting
[Liu *et al.*, 2019]	Vision	CityPersons + CrowdHuman	NMS-Enhanced Image Analysis for People Counting
[Moreira *et al.*, 2022]	Vision & TV Box	Plant Disease Dataset	Image Analysis for Disease Detection
[Shu and Shu, 2021]	Vision & TV Box	Fall Detection Dataset	Fall Detection System
[da Luz *et al.*, 2025]	Vision & TV Box	TV Boxes Testbed	Performance & Carbon Footprint Study
[Sato *et al.*, 2024]	Vision & TV Box	Custom Dataset	Image Analysis for People Counting
This Work	Vision & TV Box	Custom Dataset	Patching-Enhanced Image Analysis for People Counting

3 Improving People Counting in Crowded Scenarios

Counting people is a very common computing operation and it is used in various applications [Gao *et al.*, 2024]. In smart cities, as an example, detecting people enables monitoring spaces for public safety, mobility, enhancing urban planning, managing crowds at events, and supporting retail analytics.

The current system at Unicamp's university restaurant entrance performs people counting and makes the results available to end users through the university app [Prefeitura Universitária Unicamp, 2024] (Figure 1). However, image analysis and storage are conducted in the cloud, requiring all photos to traverse the network, which incurs significant bandwidth costs. Additionally, there are security layers in place for communication between the camera and the server, as the images contain sensitive content, such as faces. Our work aims to enable local people counting using TV boxes as a way to repurpose them. By performing inference on edge devices, we can drastically reduce bandwidth consumption, as photos will no longer need to be transmitted, thereby enhancing privacy. Moreover, disconnecting the camera from the cloud mitigates the risk of network attacks during photo uploads.



Processado em: 03/04/2024 13:10:39 - Pessoas detectadas: 13

As imagens são disponibilizadas anonimizadas, para a proteção da privacidade do titular , conforme a Lei Geral de Proteção de Dados - LGPD (Lei 13.709/2018).

Figure 1. Example of people detection in Unicamp's university restaurant waiting line - [Prefeitura Universitária Unicamp, 2024]

Although our previous work [Sato *et al.*, 2024] demonstrated the feasibility of running machine learning models on TV boxes for this application, the results were unsatisfactory in scenarios with many people in the images, particularly during peak hours at the restaurant. To address this issue, we propose using a patching technique to enhance the performance of YOLO models in crowded environments. Patching involves cropping the image into smaller regions and performing inference on each of these blocks.

Algorithm 1 presents the pseudocode for the patching technique, where the input image is divided into six blocks, each conforming to the dimensions used for pre-trained YOLO models (640x640). Inference is conducted for each block separately. Given that the EfficientDet models yielded poorer results, we applied the patching method exclusively to the YOLO models.

Algorithm 1 Pseudocode for Patching Processing

```
1: Input: Image object img_object of shape $(1920, 1080)$, inference model model
2: Parameters: Block size $(640, 640)$, total blocks 6
3: Resize img_object to $(1920, 1280)$
4: Crop img_object into 6 blocks of size$(640, 640)$
5: Store each block in blocks
6: Initialize person_count to 0
7: for each block in blocks do
8:     Run inference on block using model, filtering for ”person” class
9:     Count detected people in the result and add to person_count
10: end for
11: Output: Total person_count
```

The proposed method is straightforward and can be further refined with additional parameters, such as overlap between blocks—an approach utilized by other methods to tackle similar challenges. One such method is Slicing Aided Hyper Inference (SAHI) [Akyon *et al.*, 2022], which allows for an overlapping ratio. While we evaluated the use of SAHI, initial results indicated that it would require an ablation study of the parameters, which we will leave for future work. Supporting this notion, the work of [Gia *et al.*, 2024] defined patch width, aspect ratio, and overlap ratio for a Co-DETR model applied to a fisheye camera dataset, demonstrating improvements in the detection of small objects.

4 Experimental Setup

4.1 Dataset

For our study, we collected 346 photos from various times between April 1, 2024, and April 2, 2024, in collaboration

Table 2. Comparison between used used devices specification

Device	CPU			RAM Memory	
	Architecture	Cores	Frequency	Technology	Capacity
Raspberry Pi 3B+	ARM Cortex-A53 (64-bit)	4	1,4 GHz	LPDDR2	1 GB
Raspberry Pi 4B	ARM Cortex-A72 (64-bit)	4	1,5 GHz [1]	LPDDR4	2 GB [1]
TV Box Gen 1 (Tx2)	ARM Cortex-A7MP (32-bit)	4	1,2 GHz	LPDDR3	2 GB
TV Box Gen 2 (E10)	ARM Cortex-A53 (64-bit)	4	1,9 GHz	LPDDR4	2 GB

with the Smart Campus project team at Unicamp. Those images constituted a dataset covering approximately a full day of operation at the university restaurant. The photos encompass scenarios of low, medium, and high crowd density, featuring at least one person, as including empty photos could skew our results. To minimize bias in evaluating model performance, the labeling process was conducted independently by three different annotators.

4.2 Hardware

Our goal is to repurpose the TV Boxes provided through the partnership between Unicamp and RFB. We explored two models: the Youit Tx2 and the BTV E10, due to their different hardware limitations, capabilities, and architecture. For comparison, we used the single-board computers Raspberry Pi 3 Model B+ and Raspberry Pi 4 Model B, which are widely used embedded devices in IoT applications. Figure 2 shows an image of each device, while Table 2 outlines their hardware specifications.

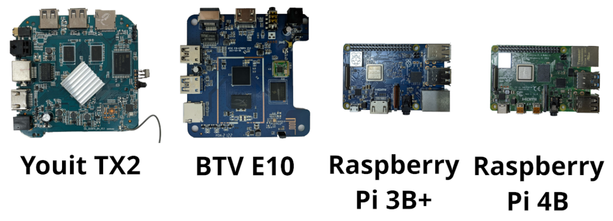


Figure 2. Picture of the four devices used in tests

To conduct tests on the TV Boxes, we followed the installation tutorials[2] [3] provided by the Smart Campus project team at Unicamp for setting up the Linux distribution on the devices. For TV Box Gen 1, we installed Armbian 24.2.5 Bookworm for the RK322X, and for TV Box Gen 2, we installed Armbian 24.5.0 Bookworm for the Aml-s9xx. This setup allowed us to transform both TV Boxes into general-purpose computers, albeit with some hardware limitations, while running a Linux distribution.

4.3 Inference Models

We are going to compare the performance of the two TV Boxes in a study case using two pre-trained machine learning algorithms:

1. YOLOv8 [Jocher *et al.*, 2023]: Coming from YOLO (*You Only Look Once*) models, this object detection deep learning model is a single-shot algorithm, being able to find all objects within an image in one pass. The YOLOv8 version uses a modified *CSPDarknet53* backbone with a Spatial Pyramid Pooling Fast (SPPF) layer. In this work, we tested two versions, YOLOv8n and YOLOv8x, corresponding to the lighter and the heavier model from the ”8” version available in Ultralytics packages.
2. EfficientDet [Tan *et al.*, 2020]: This network uses the *EfficientNet* as backbone combined with a Weighted Bi-directional Feature Pyramid Network (BiFPN) detecting objects and doing a fast multi-scale feature fusion. It is also a single-shot algorithm, which motivated the choice for this work, in addition to its popular use in edge devices. We tested two versions, ”D0” and ”D4”, corresponding to the lighter and the heavier model that fits in the devices with similar execution time to YOLOv8. Both models are available in Kaggle Hub [Kaggle Hub, 2024].

It is important to emphasize that we did not train any of those models used in this study. Both YOLOv8 and EfficientDet are pre-trained models based on the 2017 version of the Common Objects in Context (COCO) dataset. [Lin *et al.*, 2014] This choice facilitates ease of implementation and enhances the reproducibility of the proposed solution.

For running YOLOv8 models on the TV Box Gen 1 it was necessary to use a modified version using TensorFlow Lite (TFLite). Therefore, we converted the original pre-trained YOLOv8n and YOLOv8x models to the TensorFlow Lite (TFLite) format for deployment. This process was made with the tutorial provided by Ultralytics [4]. The conversion is beneficial for resource constrained edge devices, generating lightweight models with a lower latency, such as shown by [Verma *et al.*, 2021].

EfficientDet models already have their Lite version from Kaggle Hub. All TFLite models were run with TensorFlow Lite Runtime with the purpose of using the same interpreter for ARM 32-bit and ARM 64-bit. Since the models are simpler, we implemented image normalization as a pre-processing step and Non-Max Suppression (NMS) as a post-processing step.

[2]Linux Installing Tutorial for TV Box - Tx2 Model
[3]Linux Installing Tutorial for TV BOx - BTVE10 Model

[4]A Guide on YOLO Model Export to TFLite for Deployment

4.4 Metrics

Running all models, we could investigate inference performance and, as a result, the TV Boxes usage capability for image analysis. We provided an evaluation of the model using common machine learning metrics but also investigated the behaviour of the devices as inference should happen on the edge. To provide that analysis, we used the following metrics:

1. Mean Absolute Error (MAE), as the smallest the best;
2. Root Mean Square Error (RMSE), as the smallest the best;
3. Average Inference Time by each device;
4. Average Ram Memory Usage;
5. Average CPU Usage;

MAE measure the average of the absolute difference between the predicted number of people and the true value of people while RMSE penalize major errors, being more sensible to outliers.

Average inference time is measured to ensure that the application meets its requirements. Since our case study focuses on people counting at the university restaurant entrance, it is essential for end users to receive updates promptly. The current system has an update interval of two minutes, so our goal is to achieve the same rate.

Average RAM memory usage and average CPU usage were measured concurrently with the inferences. Given that we are using hardware with limited resources (1 to 2 GB of memory and ARM processors), it is crucial to assess the resource consumption during the inference process.

During our experiments, we implemented specific procedures to ensure a fair evaluation. For memory usage, we subtracted the memory consumption immediately prior to the start of each inference, allowing us to measure only the impact of the inference itself. In terms of CPU usage, we made the decision to utilize all four available cores during all tests, as this behavior is not guaranteed by default depending on the framework used. It's important to note that CPU usage is not a straightforward metric; the devices run the operating system alongside the inferences, which includes various I/O operations during swap memory usage, potentially lowering the reported CPU usage levels.

5 Results and Discussion

Table 3 presents MAE and RMSE metrics for the selected machine learning models in their standard forms as well as their TFLite versions. Meanwhile, Table 4 illustrates the impact of the patching technique on the evaluation metrics for the YOLO models. In both instances, the analysis is based on the same dataset of 346 images captured at the university restaurant entrance.

Table 3. Evaluation metrics from each machine learning model

Model	**MAE**	**RMSE**
YOLOv8n	8.78	14.30
YOLOv8n (TFLite)	8.77	14.18
YOLOv8x	7.63	13.38
YOLOv8x (TFLite)	7.75	13.56
EfficientDet D0	9.57	15.50
EfficientDet lite0	14.44	21.89
EfficientDet D4	8.44	14.47
EfficientDet lite4	13.02	20.05

Table 4. Evaluation metrics from each machine learning model with patching technique

Model	**MAE**	**RMSE**
YOLOv8n with patching	3.77	6.19
YOLOv8n (TFLite) with patching	3.77	6.24
YOLOv8x with patching	3.24	5.26
YOLOv8x (TFLite) with patching	3.33	5.36

After applying the patching technique, we observed a significant improvement in performance metrics. To explore this impact further, Figure 3 displays the difference between the predicted and actual number of people per photo, categorized by time of day. The data show that photos are generally more crowded during lunch (11 AM to 2 PM) and dinner (5 PM to 7 PM) times, with smaller spikes during breakfast (7 AM to 7:30 AM) and just before lunch (10 AM to 10:30 AM). These peak times are when machine learning algorithms struggle the most, highlighting the difficulty of predicting the number of people in crowded scenarios. Finding an inter-annotator agreement [Artstein, 2017] is a challenge present in object detection problems, especially in highly dense crowds.

Despite these challenges, various machine learning and image processing techniques can enhance prediction accuracy. In this work, we utilize pre-trained models in combination with the patching technique described in Section 3. By dividing the image into smaller blocks, the model can focus on specific regions, improving its ability to identify more individuals. This improvement is illustrated by the purple and blue lines in Figure 3. The red line indicates EfficientDet lite0 predictions, which was the worst evaluated model. We found that using either the extra-large version of the YOLOv8 model or the smallest version resulted in less than a 1 percentage point difference in MAE and RMSE. This suggests that the patching technique is more impactful than simply opting for a more complex model.

As shown in Table 4, all YOLO models demonstrated improved performance when using image patching. Notably, the results with patching for both YOLOv8n and YOLOv8x were very close, with an MAE difference of about 0.5, corresponding to approximately 15%. This indicates that applying patching can significantly enhance the accuracy of smaller models, bringing their performance closer to that of larger models while maintaining faster inference times.

We did not test patching on the EfficientDet models because they already exhibited time inefficiency during the in-

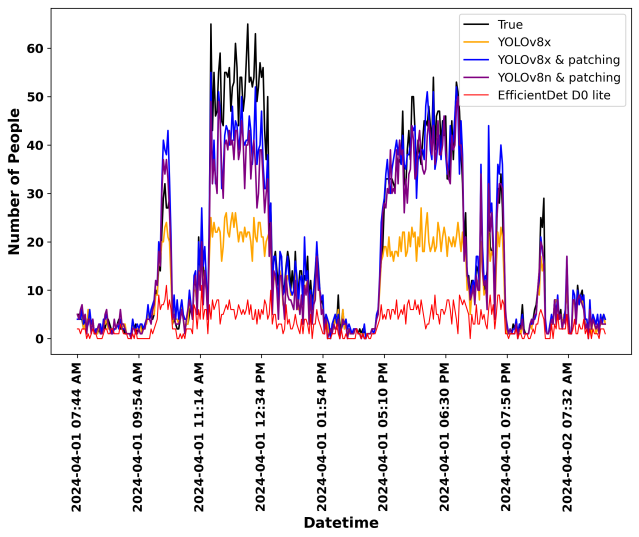


Figure 3. True value of people and models predictions by datetime on a typical day.

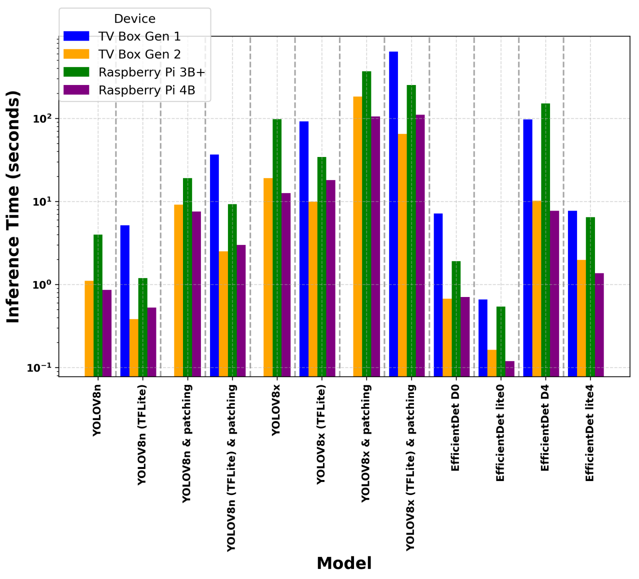


Figure 4. Inference time for different models and devices.

ference experiments without patching. However, we believe that they would benefit from patching on a similar scale as the YOLO models. Despite the potential accuracy gains, applying patching to the EfficientDet D4 model would increase their inference times even further, making them even less practical for real-time applications. One limitation found in the proposed patching method is the intersection between neighboring blocks, which can cause a person to be counted more than one time. Despite this limitation, it was still worth it to apply the technique.

Having a model that has good machine learning metrics is not enough to deploy a smart city solution, so we also measured the devices' performance. As the end users need to know how many people are in the line for the restaurant, we want the inference time to be at most two minutes, as this is the current update time in the university's app.

Table 5 shows results for inference time by each device, while Figure 4 indicates a comparison between each model and device. By analyzing the inference time, we observe the impact of model size, with more complex models showing substantially longer inference times compared to lighter models, as expected.

The Raspberry Pi 3B+'s time is comparable to the first generation of TV Box, with superior performance for simple models and inferior performance for models that require more than 1 GB of RAM. The inference time of the Raspberry Pi 4B is comparable with the second generation of TV Box. Furthermore, from the point of view of inference time, the YOLOv8x is comparable with the EfficientDet D4. The impact of using EfficientDet in the lite version is observed in both versions, which shows the gains of reducing the model size in RAM for edge devices [TensorFlow Lite Runtime, 2024]. For practical people-counting applications in smart city solutions, inference time is often one of the most critical variables.Therefore, optimizing models can lead to significant practical benefits for edge devices with limited computational resources. The patching technique can be selectively applied - used in combination with standard inference - to accommodate fluctuating crowd densities. For instance, patching could be activated only during peak hours, when higher accuracy is required.

For a more detailed understanding of device performance during model inference, we examine the average CPU and memory usage presented in Tables 6 and 7. In all cases, the devices had sufficient resources to support the patching technique. The worst-case scenario was observed on the Raspberry Pi 4B, running the YOLOv8x model with patching applied. This resulted in an increase of nearly 10 percentage points in CPU usage, reaching a still acceptable level of 87.17%. RAM usage also increased in this scenario by approximately 200 MB, which remains within acceptable limits given the device's 2 GB of RAM.

When running the YOLOv8x model, the largest among those tested, the Raspberry Pi 3B+ exhibited lower average CPU usage compared to both the TV Box Gen 2 and the Raspberry Pi 4B. This reduced CPU utilization can be attributed to the Raspberry Pi 3B+'s limited RAM capacity, which requires the use of swap memory and results in performance stalls due to storage access delays. In contrast, the YOLOv8n model is particularly appealing because of its low memory requirements. In a smart city scenario, this efficiency means that TV Boxes could perform additional tasks beyond running our application.

Of all the models tested, the best MAE and RMSE result was obtained by YOLOv8x with patching, closely followed by the same model converted to TFLite, also with patching. Considering applications that require results in less than 120 seconds, this is not a practical model for all devices, reaching 367 seconds at the Raspberry Pi 3B+ and 636 seconds in the TV Box Gen 1. Using smaller versions such as YOLOv8n showed to be a good option for applications in which a shorter inference time is required while keeping similar machine learning metrics. The performance obtained by the YOLOv8n model stands out, with an MAE approximately 4% lower than the EfficientDet D4 model, but with the inference time at least an order of magnitude lower, varying depending on the device used. Therefore, we verified that the YOLOv8 models in the TFLite version have a higher inference time performance than the EfficientDet models, also in the TFLite version, on all devices tested.

Based on these results, we conclude that the TFLite ver-

Table 5. Inference time in seconds by each device

Model	TV Box Gen 1	TV Box Gen 2	Raspberry Pi 3B+	Raspberry Pi 4B
YOLOv8n	-	1.11 ± 0.36	3.99 ± 1.31	0.86 ± 0.06
YOLOv8n (TFLite)	5.18 ± 0.04	0.38 ± 0.002	1.19 ± 0.03	0.53 ± 0.002
YOLOv8n with patching	-	9.14 ± 0.03	19.05 ± 1.14	7.56 ± 0.31
YOLOv8n with patching (TFLite)	36.6 ± 0.54	2.51 ± 0.01	9.25 ± 0.17	3.0 ± 0.09
YOLOv8x	-	19.03 ± 0.8	97.87 ± 21.06	12.63 ± 1.30
YOLOv8x (TFLite)	92.17 ± 0.89	9.93 ± 0.14	34.24 ± 0.84	18.08 ± 0.14
YOLOv8x with patching	-	182.84 ± 0.71	367.98 ± 9.48	105.23 ± 2.79
YOLOv8x with patching (TFLite)	636.46 ± 7.91	65.02 ± 0.79	252.17 ± 1.0	110.55 ± 1.83
EfficientDet D0	7.14 ± 0.04	0.68 ± 0.004	1.91 ± 0.07	0.71 ± 0.002
EfficientDet lite0	0.66 ± 0.01	0.16 ± 0.001	0.54 ± 0.05	0.12 +- 0.002
EfficientDet D4	97.24 ± 0.49	10.23 ± 0.01	151.61 ± 7.18	7.71 ± 0.08
EfficientDet lite4	7.71 ± 0.08	1.98 ± 0.02	6.47 ± 0.09	1.37 ± 0.02

Table 6. Processing and Memory Usage at YOLOv8 Models Inferences

YOLOv8 Model Inferences					
Model	**Metric**	**TV Box Gen 1**	**TV Box Gen 2**	**Raspberry Pi 3B+**	**Raspberry Pi 4B**
YOLOv8n	CPU (%)	-	56.11	44.89	54.70
	Mem (MB)	-	296.99	308.33	313.35
YOLOv8n (TFLite)	CPU (%)	54.56	59.22	61.85	80.92
	Mem (MB)	65.58	155.96	150.40	156.75
YOLOv8x	CPU (%)	-	82.20	37.36	76.81
	Mem (MB)	-	710.75	550.63	757.05
	Swap (MB)	-	-	372.91	-
YOLOv8x (TFLite)	CPU (%)	76.36	84.92	82.06	88.95
	Mem (MB)	629.66	846.41	559.04	852.46
	Swap (MB)	-	-	409.35	-

Table 7. Processing and Memory Usage at YOLOv8 Models Inferences with Patching

YOLOv8 Model Inferences with Patching					
Model	**Metric**	**TV Box Gen 1**	**TV Box Gen 2**	**Raspberry Pi 3B+**	**Raspberry Pi 4B**
YOLOv8n	CPU (%)	-	65.58	74.49	80.59
	Mem (MB)	-	339.82	321.91	324.07
YOLOv8n (TFLite)	CPU (%)	54.72	59.24	72.00	75.30
	Mem (MB)	92.34	192.05	203.94	205.27
YOLOv8x	CPU (%)	-	82.82	73.55	87.17
	Mem (MB)	-	836.27	496.98	957.59
	Swap (MB)	-	-	712.89	-
YOLOv8x (TFLite)	CPU (%)	74.57	83.94	81.01	88.67
	Mem (MB)	409.74	627.89	597.90	651.06
	Swap (MB)	-	-	219.99	-

sion of YOLOv8n with patching offers the best balance between machine learning performance metrics, inference time, CPU usage, and memory consumption. The patching method could reach a MAE of 3.77 and a RMSE of 6.24, while converting the model to TFLite version decreased memory allocation and CPU Usage. In addition, it could work for ARM based processors with 32-bit or 64-bit architecture.

6 Conclusions

In this work we explored the feasibility of repurposing TV Boxes for people counting application with crowded scenarios. We tested different models from the YOLOv8 and EfficientDet families on two configurations of TV Boxes and two mini-computers commonly used in IoT projects. The results demonstrate that the considered TV Boxes are capable of running one of the most recent state-of-the-art object detection models, YOLOv8, delivering good performance without exhausting the device's computational resources and with acceptable inference times for the application.

The models were evaluated using Pytorch, TensorFlow and TensorFlow Lite Runtime for inference. The use of TensorFlow Lite resulted in a significant reduction in processing time, with only a minimal penalty in detection performance. Among the two families of models tested, the YOLOv8 models exhibited better performance relative to inference time, making them our preferred choice for deployment on resource-constrained edge devices in people-counting applications.

To address the increased counting error observed in images with a higher number of people, we applied an image patching technique to the YOLO models and evaluated it using both the Pytorch and TensorFlow Lite frameworks. This approach involved dividing input images into smaller patches, allowing the models to focus on smaller regions and improving detection accuracy. The results showed that the patching technique systematically resulted in better MAE and RMSE, especially in more crowded images. This indicates that image patching effectively mitigates the limitations of the models in detecting larger numbers of people, which is crucial for applications where the accuracy of absolute counting is critical.

Finally, our findings highlight that even devices with older processors, such as the first-generation TV Boxes tested, can be utilized as edge devices even with additional image processing techniques, provided that their limitations, primarily due to 32-bit processors, are taken into account. In such scenarios, the use of the TensorFlow Lite Runtime is mandatory for the inference, as the full version of TensorFlow and PyTorch are not supported. While this could pose challenges for on-device model training, it can be circumvented for inference-only use, with the added benefit of lower RAM usage due to the 32-bit operating system.

From a general perspective, the YOLOv8n model with patching, converted to the TFLite format, presents a suitable option for all tested edge devices, offering a balanced trade-off between resource consumption and accuracy—achieving a Mean Absolute Error (MAE) of 3.77 on a custom people-counting dataset. This finding is particularly relevant for resource-constrained devices such as TV boxes. However, conclusions may differ in scenarios where greater computational resources are available. In non-edge-based systems, deploying more complex models such as the standard YOLOv8x may be worthwhile, as discussed in a similar application by [da Luz *et al.*, 2024]. Nonetheless, as shown in Table 5, applying patching to YOLOv8n consistently resulted in faster inference times than using the non-patched YOLOv8x across all evaluated devices.

As future work, newest versions of YOLO and also Transformers based real time models [Huang *et al.*, 2024; Peng *et al.*, 2024] can be used and compared with the YOLOv8. A comparison between finetuned models and different image processing techniques can be done to assess their impact in crowded datasets. We also aim to use other patching techniques that allow more parameters to deal with overlapping regions and reduce MAE even more.

Declarations

Acknowledgements

We acknowledge our initial findings published in [Sato *et al.*, 2024] as a foundational baseline. This extended study introduces a novel method providing new combinations that have enhanced the results, advancing the scope and performance beyond our previous work. The authors would like to thank the Smart Campus Unicamp Project team for providing the dataset.

Funding

This research was supported by CAPES (process 88887.999360/2024-00), CNPq (process 308840/2020-8 and 131653/2023-7), by the Brazilian Ministry of Science, Technology and Innovations, with resources from Law num. 8,248, of October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex and published Arquitetura Cognitiva (Phase 3), DOU 01245.003479/2024 -10, and by FAPESP (process 2023/00811-0).

Authors' Contributions

All authors contributed to interpreting the results and writing and reviewing the manuscript. G.M.S., G.P.C.P.D.L., and L.F.G.G. conducted the experiments. J.F.B. supervised the project.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The datasets generated and/or analysed during the current study are not available due to the sensitive nature of the research, however codes and materials that support the findings of this study are available upon reasonable request.

References

Akyon, F. C., Altinuc, S. O., and Temizel, A. (2022). Slicing aided hyper inference and fine-tuning for small object detection. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 966–970. IEEE. DOI: 10.1109/icip46576.2022.9897990.

Artstein, R. (2017). Inter-annotator agreement. *Handbook of linguistic annotation*, pages 297–313. DOI: 10.1007/978-94-024-0881-$2_1$1.

Badidi, E., Moumane, K., and Ghazi, F. E. (2023). Opportunities, applications, and challenges of edge-ai enabled video analytics in smart cities: A systematic review. *IEEE Access*, 11:80543–80572. DOI: 10.1109/ACCESS.2023.3300658.

Baldé, C. P., Kuehr, R., Yamamoto, T., McDonald, R., D'Angelo, E., Althaf, S., Bel, G., Deubzer, O., Fernandez-Cubillo, E., Forti, V., Gray, V., Herat, S., Honda, S., Iattoni, G., Khetriwal, D. S., di Cortemiglia, V. L., Lobuntsova, Y., Nnorom, I., Pralat, N., and Wagner, M. (2024). The global e-waste monitor 2024. Technical report, International Telecommunication Union (ITU) and United Nations Institute for Training and Research (UNITAR). Available at:`https://ewastemonitor.info/the-global-e-waste-monitor-2024/`.

Bhatt, C., Kukreti, A., Pratap, A., and Dhondiyal, S. A. (2024). Deep learning for crowd counting: Addressing crowd density with advanced methods. In *2024 Second International Conference on Advances in Information Technology (ICAIT)*, volume 1, pages 1–5. IEEE. DOI: 10.1109/icait61638.2024.10690493.

Collini, E., Palesi, L. A. I., Nesi, P., Pantaleo, G., and Zhao, W. (2024). Flexible thermal camera solution for smart city people detection and counting. *Multimedia Tools and Applications*, 83(7):20457–20485. DOI: 10.1007/s11042-023-16374-x.

da Luz, G. P., Sato, G. M., Gonzalez, L. F. G., and Borin, J. F. (2024). Smart parking with pixel-wise roi selection for vehicle detection using yolov8, yolov9, yolov10, and yolov11. *arXiv preprint arXiv:2412.01983*. DOI: 10.48550/arXiv.2412.01983.

da Luz, G. P., Sato, G. M., Gonzalez, L. F. G., and Borin, J. F. (2025). Repurposing of tv boxes for a circular economy in smart cities applications. *Scientific Reports*, 15(1):22638. DOI: 10.1038/s41598-025-97379-4.

Dwivedi, Y. K., Hughes, L., Kar, A. K., Baabdullah, A. M., Grover, P., Abbas, R., Andreini, D., Abumoghli, I., Barlette, Y., Bunker, D., Chandra Kruse, L., Constantiou, I., Davison, R. M., De', R., Dubey, R., Fenby-Taylor, H., Gupta, B., He, W., Kodama, M., Mäntymäki, M., Metri, B., Michael, K., Olaisen, J., Panteli, N., Pekkola, S., Nishant, R., Raman, R., Rana, N. P., Rowe, F., Sarker, S., Scholtz, B., Sein, M., Shah, J. D., Teo, T. S., Tiwari, M. K., Vendelø, M. T., and Wade, M. (2022). Climate change and cop26: Are digital technologies and information management part of the problem or the solution? an editorial reflection and call to action. *International Journal of Information Management*, 63:102456. DOI: 10.1016/j.ijinfomgt.2021.102456.

Gao, M., Souri, A., Zaker, M., Zhai, W., Guo, X., and Li, Q. (2024). A comprehensive analysis for crowd counting methodologies and algorithms in internet of things. *Cluster Computing*, 27(1):859–873. DOI: 10.1007/s10586-023-03987-y.

Gia, B. T., Khanh, T. B. C., Trong, H. H., Doan, T. T., Do, T., Le, D.-D., and Ngo, T. D. (2024). Enhancing road object detection in fisheye cameras: An effective framework integrating sahi and hybrid inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7227–7235. DOI: 10.1109/cvprw63382.2024.00718.

Himeur, Y., Sayed, A. N., Alsalemi, A., Bensaali, F., and Amira, A. (2024). Edge ai for internet of energy: Challenges and perspectives. *Internet of Things*, 25:101035. DOI: 10.1016/j.iot.2023.101035.

Huang, S., Lu, Z., Cun, X., Yu, Y., Zhou, X., and Shen, X. (2024). Deim: Detr with improved matching for fast convergence. *arXiv preprint arXiv:2412.04234*. DOI: 10.48550/arxiv.2412.04234.

Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics YOLO. Available at: `https://github.com/ultralytics/ultralytics`.

Kaggle Hub (2024). Efficientdet object detection model. `https://www.kaggle.com/models/tensorflow/efficientdet/tensorFlow2`. Accessed in 03/03/2024.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer. DOI: 10.1007/978-3-319-10602-$1_4$8.

Liu, S., Huang, D., and Wang, Y. (2019). Adaptive nms: Refining pedestrian detection in a crowd. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6459–6468. DOI: 10.1109/cvpr.2019.00662.

Metwaly, A., Queralta, J. P. n., Sarker, V. K., Gia, T. N., Nasir, O., and Westerlund, T. (2020). Edge computing with embedded ai: Thermal image analysis for occupancy estimation in intelligent buildings. In *Proceedings of the INTelligent Embedded Systems Architectures and Applications Workshop 2019*, INTESA2019, page 1–6, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3372394.3372397.

Moreira, R., Rodrigues Moreira, L. F., Munhoz, P. L. A., Lopes, E. A., and Ruas, R. A. A. (2022). Agrolens: A low-cost and green-friendly smart farm architecture to support real-time leaf disease diagnostics. *Internet of Things*, 19:100570. DOI: 10.1016/j.iot.2022.100570.

Nations, U. (2015). Transforming our world: The 2030 agenda for sustainable development. *New York: United Nations, Department of Economic and Social Affairs*, 1:41. DOI: 10.1891/9780826190123.ap02.

Peng, Y., Li, H., Wu, P., Zhang, Y., Sun, X., and Wu, F. (2024). D-fine: redefine regression task in detrs as fine-grained distribution refinement. *arXiv preprint arXiv:2410.13842*. DOI: h10.48550/arxiv.2410.13842.

Prefeitura Universitária Unicamp (2024). Fila do restaurante em tempo real. `https://www.prefeitura.unicamp.br/cardapio/`. Accessed in 04/03/2024.

Ruchika, Purwar, R. K., and Verma, S. (2022). Analytical study of yolo and its various versions in crowd counting. In *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2021*, pages 975–989. Springer. DOI: 10.1007/978-981-16-7610-$9_7$1.

Sato, G., Luz, G., Gonzalez, L., and Borin, J. (2024). Reaproveitamento de tv boxes para aplicação de contagem de pessoas na borda em cidades inteligentes. In *Anais do VIII Workshop de Computação Urbana*, pages 197–209, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/courb.2024.3375.

Shu, F. and Shu, J. (2021). An eight-camera fall detection system using human fall pattern recognition via machine learning by a low-cost android box. *Scientific reports*, 11(1):2471. DOI: 10.1038/s41598-021-81115-9.

Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790. DOI: 10.1109/cvpr42600.2020.01079.

Tang, C., Li, W., Vishwakarma, S., Chetty, K., Julier, S., and Woodbridge, K. (2020). Occupancy detection and people counting using wifi passive radar. In *2020 IEEE Radar Conference (RadarConf20)*, pages 1–6. DOI: 10.1109/RadarConf2043947.2020.9266493.

TensorFlow Lite (2024). Guide for tensorflow lite. `https://www.tensorflow.org/lite/guide`. Accessed in 04/05/2024.

TensorFlow Lite Runtime (2024). Tensorflow - an end-to-end platform for machine learning. `https:`

`//blog.tensorflow.org/2020/10/optimizing-tensorflow-lite-runtime.html`. Accessed in 04/05/2024.

Verma, G., Gupta, Y., Malik, A. M., and Chapman, B. (2021). Performance evaluation of deep learning compilers for edge inference. In *2021 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pages 858–865. IEEE. DOI: 10.1109/ipdpsw52791.2021.00128.