Implementation of visual people counting algorithms in embedded systems

Rudolf, O.
Darmstadt University of Applied Sciences
Schöfferstraße 3
64295, Darmstadt, Germany
oskar.rudolf@h-da.de

Hecker, R.
Darmstadt University of Applied Sciences
Schöfferstraße 3
64295, Darmstadt, Germany
robert.hecker@stud.h-da.de

Thißen, M.
Darmstadt University of Applied Sciences
Schöfferstraße 3
64295, Darmstadt, Germany
martin.thissen@h-da.de

Sillekens, L.
Darmstadt University of Applied Sciences
Schöfferstraße 3
64295, Darmstadt, Germany
laurens.sillekens.h-da.de

Penner, I.
Thermokon Sensortechnik GmbH
Platanenweg 1
35756, Mittenaar, Germany
ina.penner@thermokon.de

Akelbein, J.-P.
Darmstadt University of Applied Sciences
Schöfferstraße 3
64295, Darmstadt, Germany
jens-peter.akelbein@h-da.de

Seyfarth, S.
Thermokon Sensortechnik GmbH
Platanenweg 1
35756, Mittenaar, Germany
stefan.seyfarth@thermokon.de

Hergenröther, E.
Darmstadt University of Applied Sciences
Schöfferstraße 3
64295, Darmstadt, Germany
elke.hergenroether@h-da.de

Abstract

Optimising the efficiency of HVAC systems represents a significant opportunity to reduce energy consumption in buildings and mitigate greenhouse gas emissions. This research evaluates low-resolution computer vision algorithms for occupancy detection on resource-constrained embedded systems. Our evaluation focuses specifically on the feasibility of deploying advanced AI object detection models on low-cost hardware platforms (under €10) with varying computational capabilities. We systematically compared 45 different pre-trained object detection models using the COCO dataset. Among the models evaluated, those with the YOLO backbone proved to be the most suitable for this task. Quantitative analysis showed that YOLOv5n achieved a favourable balance between accuracy (AP50 = 0.944; AP50-95 = 0.584), model size (2.6 MB in RKNN format) and inference time. Performance tests on three embedded platforms - ESP32-CAM (microcontroller), Raspberry Pi Zero 2 W and Luckfox Pico Mini A (single-board computer) - revealed significant differences in inference speed, with hardware-accelerated solutions up to 10,000 times faster than software-only implementations. We have verified real-world applicability using our own ceiling-mounted wide-angle camera dataset. Future work will focus on developing a full hardware prototype, optimising the training dataset with AI-generated synthetic data, and implementing sensor fusion with audio signals for a multimodal approach.

Keywords

Occupancy Detection, Computer Vision, YOLO, Embedded systems

1 INTRODUCTION

According to the 2023 Global Status Report of the International Energy Agency (IEA), buildings make up a significant global energy balance. For example, the German Federal Ministry for Economic Affairs and Climate Protection (BMWK) buildings in Germany are responsible for about 35% of total national energy consumption [Umw24]. This energy consumption statistic can be partly explained by the inefficiencies of HVAC systems [Ost+19]. Improving their efficiency by enabling intelligent and precise control is important in achieving the European Green Deal's 2050 goal of climate neutrality. Reducing electricity and heating costs benefits companies. Therefore, improving occupancy detection can play a crucial role in optimizing HVAC operation, defined as the process of detecting if a space is occupied. Applications span

buildings, security systems, energy management, and analytics. Traditional methods commonly use PIR, ultrasonic, or near-field radar sensors.

However, recent developments in microprocessor technology have produced chips that are both compact and affordable for use with Artificial Intelligence (AI) in this scenario. This creates an opportunity to explore whether AI-based methods could provide alternative solutions for presence detection, especially with respect to cost-effectiveness and reliability, which can significantly reduce building energy consumption [Mun+24]. One promising solution that we want to focus on in this research is the use of ordinary cameras in combination with computer vision techniques. Ongoing improvements in object detection algorithms allow reliable people and activity detection in a room and allow HVAC systems to be controlled according to demand. Moreover, the use of computer vision in smart buildings is not limited to occupancy detection but can also enhance security, space utilization, and overall building management. Techniques like region-based convolutional neural networks (R-CNN) and YOLO models (you only look once) are used for real-time detection of occupancy and activities, achieving high accuracy rates (up to 99.3%) in identifying multiple occupant behaviors, such as computer usage and light adjustments [TWC20], [Li+23].

We want to evaluate the complementary added value of machine learning object detection methods to traditional sensor systems in building automation for climate control. A particular focus lies on practical feasibility: First, the minimal hardware resource constraints in the form of computing and memory capacities have to be identified to ensure a close to real-time performance. Second, the machine learning models which the algorithms are based on have to be applicable to the specific context of being used on a central ceiling mounted camera with a wide camera lens indoors. Third, algorithms need to be further optimized and tailored for application in an embedded system. The aim is to provide a baseline for further development of a robust and cost-efficient video-based solution for occupancy detection that can be integrated into existing building automation systems and enables significant energy savings.

2 BACKGROUND

2.1 Related work

Recent advances in artificial intelligence have significantly improved the capabilities of embedded systems in the realm of real-time object detection in applications such as autonomous driving [Ngu+24], intelligent surveillance [SB24], and precision agriculture [MRP24]. Collectively, these studies highlight the transformative impact of integrating AI into embedded systems for near real-time object detection. Key benefits of employing embedded systems are low latency, increased processing speed, and the capacity to function without reliance on centralized cloud services — an essential factor for applications that demand short decision-making or work with sensitive data. However, new challenges emerge, such as optimizing deep learning models to fit the constrained computational resources of embedded devices without compromising accuracy. Ongoing research focuses on developing and refining optimization techniques such as pruning, quantization, knowledge distillation, lightweight models, hardware acceleration, and energy-efficient algorithms to overcome these hurdles, thus broadening the scope of AI applications in embedded systems. However, up to today, research in the field of embedded deployment of AI for computer vision is quite limited. Solutions for running person detection models on embedded hardware often rely on more powerful hardware such as the NVIDIA Jetson Nano board ([SC23]) or Raspberry Pi 4 Model B ([Saf+21]), which do not fit both the space and budget constraints.

2.2 Hardware constraints

While the overarching research initiative aims to develop a small encapsulated ceiling-mounted system with commercial viability, the present study focuses specifically on benchmarking embedded inference methods. In order to keep in line with the objective of developing a low-cost detection unit for integration within a Building Management System (BMS), approximately €100 has been set for the complete hardware system. Consequently, the processing hardware should be kept well below this price point, as more powerful processors tend to increase the need for additional resources, such as cooling fans. The hardware cost constraint thus directly affects the selection of viable processing platforms. They must provide both sufficient processing resources (clock speed and memory) to perform AI inference on images in a reasonable time, as well as come with available toolchains to deploy the models in the first place.

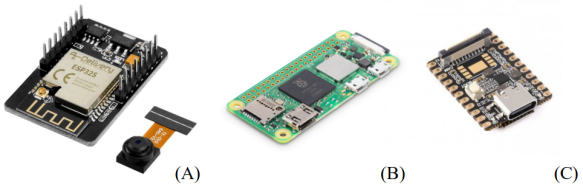
Two classes of hardware devices that fit our set constraints are microcontrollers (MCUs) and single-board computers (SBCs). While the distinction between both classes is imprecise and varies slightly between authors, the most important distinction for our goals is that MCUs usually possess very limited hardware resources; SBCs generally contain more Random Access Memory (RAM), Read-Only Memory (ROM), and computing power. The hardware capabilities of smaller microcontrollers do not suffice to run sophisticated machine learning models such as those required for occupancy detection. However, stronger MCUs with larger amounts of RAM can

potentially meet the requirements to run computer vision models. For our experimentation, we selected three devices with varying capabilities, reaching from lower to higher-end performance capabilities: the ESP32-CAM (MCU), Raspberry Pi Zero 2 W (SBC) and the Luckfox Pico Mini A (SBC) (see Table 1 for technical details and Figure 1 for respective device pictures).

Device	RAM	CPU	Cost	Size
ESP32-CAM	4 MB	2x 240 MHz	9.00 €	55mm×28mm
Raspberry Pi Zero 2 W	512 MB	4x1 GHz ARM	8.00 €	51mm×21mm
Luckfox Pico Mini A	64 MB	2x1.2 GHz ARM	6.50 €	42mm×21mm

Table 1: Processing unit details.

In contrast to both the ESP32-CAM and the Raspberry Pi Zero 2 W, the Luckfox Pico Mini A also provides hardware acceleration for neural network computations through a dedicated neural processing unit (NPU) for machine learning applications (up to 0.5 trillion operations per second on Int8 data). Even though all of the presented devices do have some sort of operating system, it is important to note that the RTOS on the ESP32-CAM is much more rudimentary than the Linux variants on the Raspberry or Luckfox devices, which adds a layer of complexity to the implementation of neural network pipelines.



(A) (B) (C)

Figure 1: (A) ESP32-CAM; (B) Raspberry Pi Pico 2 W; (C) Luckfox Pico Mini A. The images were taken from the manufacturers product websites .

3 MODEL SELECTION

Deep learning methods, particularly deep convolutional neural networks (CNNs), have proven to be a reliable method for use in object detection, offering high precision and robustness even under challenging conditions such as varying lighting [HMT22]. Ensuring valid model predictions requires both the availability of suitable training data and the selection of an optimal model architecture. Before implementing and optimizing a detection algorithm for occupancy monitoring in a restricted hardware environment, we structured the approach to model selection in several stages.

First, suitable public data sources, the Commons Objects in Context (COCO) dataset [Lin+14] and the WiseNET dataset gathered by Marroquin et al. [MDN19] were identified to help with model preselection. Next, we aimed to evaluate multiple trained models on the gathered data, comparing their performance in terms of accuracy and inference speed while also accounting for computational resource efficiency in terms of processing and memory usage. Finally, to assess the real-world applicability of the selected model, we conduct controlled experiments using our own lab data that reflect our specific use case — a ceiling-mounted, wide-angle camera setup for comprehensive room coverage.

3.1 Data

Obtaining application-specific training data can be a major challenge in research due to its inherently sensitive nature. Large-scale video data collection is often impractical for ethical, legal, and logistical reasons. Given these limitations, we decided to leverage already publicly available pretrained models and a large-scale image dataset as a starting point to reduce the need for expensive and time-consuming data collection on a large scale for training. The overall key criteria to select a suitable dataset included diversity in perspectives and scenarios, consistent annotation standards, dataset size, and practical relevance, e. g. the presence of human beings. We decided to use the COCO dataset for its extensive collection of images including people, all of which are already annotated and divided into training, test, and validation sets. Conveniently, there are also already numerous pretrained models and frameworks for their use available for object detection based on the COCO dataset, which makes them easily comparable.

To further improve the model performance through application of transfer learning, additional data was collected. Recordings of nine participants in groups of two to four people were obtained to augment the contextual data of the already pre-trained models and to refine their predictions according to our use case. The laboratory environment included office furniture and a central ceiling-mounted camera view. To allow comparisons between camera types, we used a 3860×2160 resolution camera with a 120° lens and a 1920×1080 resolution camera with a 180° lens. Each group of participants was recorded multiple times with different movement instructions and furniture settings (e.g., one large conference table versus several smaller tables). Because successive frames had too little difference and would artificially increase the dataset without introducing additional diversity, we limited the video data to one frame per second. In the final dataset, consisting of $N = 28$ frames, each person was then manually labeled with rectangular bounding boxes.

3.2 Model comparison

To identify the most suitable model architecture for our application, we evaluated 45 different object

detection models using TensorFlow as the backend [Mar+15]. The evaluation included architectures from the ResNet (50, 101, 152 with different resolutions), YOLO (up to version 8), MobileNet (v1 and v2), and EfficientDet (d0-d7) families. The three key criteria for comparison included model size, inference time, and average precision (AP). The AP metric summarizes the relationship between precision ($P = \frac{T_p}{T_p+F_p}$) and recall ($R = \frac{T_p}{T_p+F_n}$) by computing a weighted average of precision across different recall levels, where the weight corresponds to the increase in recall between thresholds ($APn = \sum_n((R_n - R_{n-1}) * P_n)$). For determining valid detections, we adopted a standard approach for object detection tasks, using a fixed Intersection over Union (IoU) threshold of 50%.

Since the primary goal of this research stage was to select a model architecture rather than implement it on embedded hardware, we conducted our evaluations on an upscaled system using Nvidia A5000 GPUs with 32 GB RAM. This approach helped to avoid compatibility issues and minimized time spent adapting each model to embedded environments. To make sure to create a comparable basis for evaluation, we used the pre-trained weights based on the same COCO data subset for all models without further fine-tuning.

Lastly, beyond the quantitative metrics, we also considered the models' compatibility with the target hardware platform to ensure feasibility for future deployment. Although newer architectures - such as YOLOv9 and beyond - have demonstrated marginally better performance in recent benchmarks [Jeg+], we excluded them from our evaluation. This decision was based on practical limitations, as vendor support for these newer models is limited, and custom porting them to microcontroller-compatible formats would have been too time-consuming and costly.

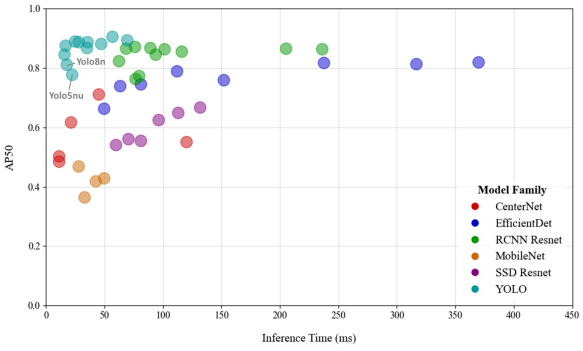


Figure 2: Comparison of inference time for occupancy detection vs. AP50 grouped by different pretrained architecture model families using an upscaled system.

The evaluation results (comp. Figure 2) indicate a dominant performance of network architectures explicitly from the YOLO-CNN model family. In particular, versions 5 and 8 in the smallest model variant ("nano") showed a particularly good balance between inference time, resource requirements, object detection rate, and also hardware compatibility compared to other models. As the training data is different from the application context, we also investigated the extent of distortion sensitivity of each model by additionally performing various common image augmentations, e.g. edge smoothing, brightness changes, image scaling, and grayscale conversion (see Figure 3). We could not find any significant differences in inference time of augmented vs. non-augmented images, as well as no difference in between any of the augmentation types.

Figure 3: Augmentation example with a picture taken from the COCO-dataset.

3.3 Field evaluation

Given the limitations of using non-context-specific data for model training, it is crucial to evaluate the generalizability of pretrained models when actually deployed in real-world environments. Specifically, wide-angle lenses introduce significant distortions that may affect the performance of object detection systems trained on datasets with minimal distortion. To address this challenge, we applied the pretrained models to a centered ceiling-mounted wide-angle camera to assess its ability to adapt to these lens-induced distortions and to the specific top-down perspective (see Figure 4).

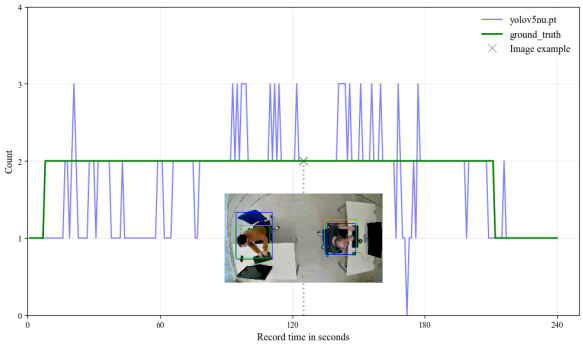


Figure 4: Example prediction of model YOLOv5n in a live lab recording with a wide-angle camera lens.

Even though the overall metrics reached reasonably high values (e.g.AP50 = 0.944; AP50-95 = 0.584 with

the pretrained model architecture YOLOv5n [Joc20]), it was not possible to reliably return the exact room occupancy at each given moment (see Figure 4). This discrepancy could be mitigated by smoothing the average detection over multiple frames within a given time window (e.g. 5 seconds) to help improve consistency, but implementing this logic in an embedded system would result in a significant slowdown due to the use of multiple inferences. This places a lot of strain on the system's workload, especially because real-time applications typically are designed to simultaneously manage additional sensor data as well.

3.4 Transfer Learning

To further refine the most effective model (YOLOv5nu), the Ultralytics framework was utilized, which provides a user-friendly high-level interface for tasks such as object detection, segmentation, classification, and pose estimation specifically for YOLO models. The dataset was not augmented, as the framework applies geometric, color space and flipping augmentations by default. The default hyperparameters were not adjusted and can be found in the Ultralytics documentation [Ult23].

Transfer learning helped to further improve the YOLOv5nu detection results from 78% AP50 and 64% AP50-95 to about 99.5% and 94.8% on the training dataset respectively. However, the transferability of this relearned model to a real-time environment still needs fine-tuning. Initial attempts to apply it to the test dataset showed signs of overfitting (leading to no improvement compared to the initial pretrained model). Unfortunately, despite changed room configurations within the recording, the collected data context (background, floor) is still too narrow in variance to investigate the impact of training the models solely from context-near images with initial random weights (training from scratch).

4 APPLY TO EMBEDDED SYSTEMS

Based on the model evaluation results, we have decided to implement occupancy detection using the YOLOv5nu architecture. This architecture is considered as "nano"-size as it has a relatively low number of weights totaling around 1.9 million parameters, which can be stored in less than 8 MB of memory in Float32 format.

4.1 Algorithm Optimizations

Once suitable networks were identified, optimizations were explored to further reduce the resource requirements for embedded computing. This involved quantization (converting Float datatype to Integer datatype) and reducing model parameters (pruning). While quantization resulted in a model size reduction of approximately 75% with minimal impact on prediction quality, pruning of model parameters - achieved by adjusting the number of convolution filters of the YOLO model by 50% lead to notable decrease in recognition performance. Tests conducted on a compact Linux microcontroller development board with a Rockchip RV1103 chip (Luckfox Pico Mini) showed average inference times of around 800 milliseconds per image at an image resolution of 640 x 640 pixels.

4.2 Deployment in embedded devices

Executing inference on these models on embedded hardware poses certain challenges related to the limited board memory and computational resources on the devices. Addressing these challenges is highly dependent on the specific device being used, often requiring different frameworks and runtime environments to be used for different devices. The table 2 provides a concise overview of the deployment formats and toolchains for three representative embedded platforms.

	ESP32-CAM	**RPi Zero 2 W**	**LuckFox Pico Mini A**
Toolchain	TFLite Micro / ESP-DL	TFLite / ONNX RT	RKNN TK / TFLite
Offline Conversion	Yes	Optional	Yes (to RKNN)
Quantization Support	INT8 only	INT8, Float32	INT4/8/16 (RKNN)
Deployment Effort	High	Medium	Low-Medium

Table 2: Model deployment overview. For a more detailed overview refer to documentations provided by the vendors.

For executing machine learning models on microcontrollers such as the ESP32-CAM we used TensorFlow Lite Micro [Dav+21] (TFLM). TFLM accounts for the reduced capabilities of MCUs e.g. by only supporting a subset of all defined TensorFlow operations or by not creating any dynamic memory allocations for its calculations. TFLM requires a model that is stored in the specialized TensorFlow Lite file format, requiring conversion of a TensorFlow model into this form. The YOLOv5 toolkit already offers a conversion path to export the internally used PyTorch models into the TensorFlow and TensorFlow Lite formats. To keep the size of the model weights within the 4 megabytes of available flash memory on the ESP32-CAM, model weights had to be quantized to Int8 format. This quantization only minimally degraded the detection performance, resulting in reductions of at most two percentage points of the AP50 object detection metric at a resolution of 512×512 pixels. Setting up and invoking the inference process of TFLM was accomplished through a custom C++ program.

For the Raspberry Pi Zero 2 W, the default Debian-based Raspberry Pi OS contains a Python runtime environment, meaning that the Python dependencies for YOLOv5n could be installed directly and run without any modifications to any existing model files. The Luckfox Pico Mini also runs a Linux-based operating system, although by default not

shipping a Python runtime environment. Due to this, running model inference through the YOLOv5n Python toolkit is not easily possible, further complicated by the significantly smaller RAM of the Luckfox Pico Mini (64 MB) compared to the Raspberry Pi Zero 2 W (512 MB). In order to efficiently run model inference using the hardware acceleration of the NPU, the manufacturer Luckfox maintains its custom "RKNN" solution. Similar to TensorFlow Lite Micro, it expects model weights to be provided in a specialized file format, requiring conversion from YOLOv5n's PyTorch format to ONNX and finally into the RKNN format. To invoke the RKNN runtime, Luckfox provides example C++ code for YOLOv5n models, although the code had to be tweaked to ensure compatibility with the modified YOLOv5n variants with reduced class count.

4.3 Inference performance

Performance measurements were made based on the different implementations on the three hardware devices. Inference was performed on ten input images randomly selected from the COCO dataset. To determine the effect of image size on execution speed, different image resolutions were tested, ranging from 128×128 to 512×512 pixels with a step of 32 pixels. Figure 5 shows the average inference time for a single model created on the three hardware devices, averaged over all ten input images. Note that except for the input layer, the original YOLOv5nu architecture with 1.9 million parameters has not been changed. The RAM requirements for holding pixel information increased quadratically with the number of pixels. As expected, performance metrics improved with increasing image size, with AP50 values ranging from $AP50 \approx .394$ at 128×128 pixels up to $AP \approx .442$ at 512×512 pixels. Quantizing the weights from float32 to int8 did not significantly affect any of the performance metrics.

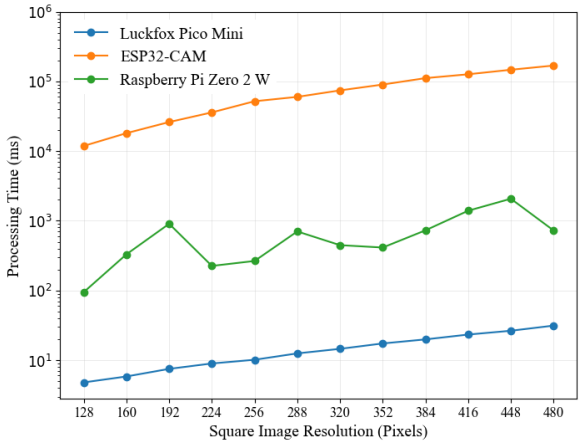


Figure 5: Inference times with YOLOv5n variant (1.9M Parameters) on the hardware devices.

There are noticeable differences in runtime between the devices. For smaller image sizes, the ESP32-CAM took several seconds to process a frame. Larger resolutions resulted in inference times of up to three minutes per image; resolutions upwards of 480×480 pixels required more memory than was available on the device, making inference impossible. The Raspberry Pi Zero 2 W exhibited more unstable average runtimes, which are to be attributed to measurement error of the used timing routine and not to actual significant runtime differences between images. Aside from these inaccuracies, the Raspberry Pi Zero 2 W was able to process at least one image per second, being significantly faster than the ESP32-CAM. The Luckfox Pico Mini achieves the lowest runtimes of all devices through usage of its hardware acceleration, which would even allow real-time processing of video data. In general, these experiments illustrate the significant differences in neural network inference performance among embedded devices of similar size and price point. Although object detection models can be run on powerful microcontrollers like the ESP32-CAM, the inference is too slow for most use cases. Single-board computers with hardware acceleration can execute models $10,000$ times faster without being significantly larger or more expensive.

5 CONCLUSIONS

In accordance with the findings of this study, recent research supports the efficacy of YOLO-based models for real-time occupancy detection within embedded Building Management Systems (BMS) [Ver+23]. Further, as demonstrated in the study by [Zha+24], YOLOv8x demonstrated the highest level of accuracy in tests of lecture-room occupancy, with predictions within approximately ≈6.7% of the ground truth energy usage. This result indicates that YOLOv8x outperforms SSD and Faster R-CNN. In the field of embedded applications [Saf+21] have reported the deployment of YOLOv5s on a Raspberry Pi 4 (2 GB RAM) utilising TensorFlow Lite, achieving a reported accuracy rate of over over 90% accuracy, ≈100 ms inference time per frame (≈10 FPS), and a compact 14 MB model size (≈7.25M parameters). In a similar manner, [Mon+22] fine-tuned a YOLOv3 model on a Raspberry Pi using transfer learning, achieving a mean absolute error (MAE) of 1.23 people in two classroom settings.

5.1 Implications

Our research demonstrates the feasibility of implementing a low-cost, video-based embedded sensor system for occupancy detection in modern building automation.Recent advancements in the market provide fast and reliable hardware solutions that can complement or even replace existing systems. Given the increasing global demand for energy, such technologies are expected to play a crucial role in future

smart home systems, especially in office buildings. Our findings indicate that machine learning models trained on non-contextual data can be applied to some extent. As usability improves and costs continue to decrease, the adoption of smart building technologies is likely to accelerate. This, in turn, could encourage more companies to invest in automated systems for HVAC optimization, ultimately contributing to greater energy efficiency and supporting global efforts to combat climate change.

5.2 Limitations

The development and implementation of the presented systems are subject to several technical limitations. A key restriction is the insufficient availability of representative training data. While the YOLO model architecture with pre-trained weights on the extensive COCO data already ensures a reliable and resource-saving object detection method, it still needs to be refined to our application-specific context to further increase its reliability. As the transfer learning analysis showed promising results, we are currently working on an adapted model to specifically detect people from a ceiling-mounted camera and collecting custom data to reinforce this approach.

The use of wide-angle cameras is recommended in our use case to ensure that the entire room is covered by the field of view. This adds another layer of difficulty to occupancy detection. We either have to retrain the models or try to reduce the distortion mathematically. For the first approach, Datasets such as the COHI dataset [BAV23] could help by providing annotated fisheye imagery for training and evaluation. For the latter approach, we could use different model, commonly used in calibration tools like OpenCV and MATLAB to account for radial (e.g., barrel, pincushion) and tangential (e.g., skew due to misalignment) distortions using polynomial coefficients. Instead of distortion-free images, modern object detection approaches increasingly adapt to distortion directly through methods such as distortion-aware augmentation, network-level adaptation and custom bounding-box representations.

Acquiring new real-world training data with subjects in an office environment involves considerable planning and cost. Among other difficulties, compliant participants need to be mobilized and office buildings need to be rented or reserved. The use of generative AI models for synthetic data generation could be a promising solution to partially address the small amount of accessible data in the office application context needed for transfer learning. Through targeted prompting, additional images with a variety of scenarios can easily be generated automatically, thus diversifying the training base. In addition to self-collected and open-sourced real-life datasets, generative AI models can further increase the amount of training data by generating synthetic images in different office environments with a variable number of people within a very short time. Studies were able to show that the integration of synthetic data can significantly improve the accuracy of object recognition models, especially when few real images are available [Lu+23], [Fra+24]. Parameters such as lighting or the design of the office building can also be precisely controlled via the input prompt, which supports the development of robust recognition models. Advances in generative models are leading to a steady improvement in the quality of synthetic images, so that they are visually almost indistinguishable from real images. Given the possible benefits of this technology, our research group is currently working on ways to utilize synthetic data to further enhance custom models for occupancy detecion from a ceiling-mounted perspektive.

5.3 Further Research

In future research, we aim to integrate all our findings into a prototype and further enhance the models' accuracy by incorporating more live data and AI-generated images. This approach will allow us to refine the system and improve its performance, ultimately providing a more robust and reliable solution for the target applications.

Another way to automatically capture room occupancy is sensor fusion. Sensor fusion is a multimodal approach that combines different types of sensors, e. g. audio and visual data. This approach helps to improve the occupancy detection system even further by compensating errors of a single sensor technology, and thus increase the robustness of the people count. In addition, as soon as we can confidently predict occupancy in a room, it would be possible to integrate occupancy information with lighting control or security systems. However, sensor fusion introduces additional challenges, such as increased overhead and complex resource management. We aim to address these hurdles and implement a computationally efficient sensor fusion method at the feature and decision level under the constraints of available resources in an embedded system.

6 ACKNOWLEDGMENTS

The authors thank the Federal Ministry for Economic Affairs and Climate Action (BMWK, Germany) for financial support of this research project.

REFERENCES

[BAV23] Zarema Balgabekova, Muslim Alaran, and Hüseyin Atakan Varol. "A Data-Centric Approach for Object Recognition in Hemispherical Camera Images". In: (May 2023). DOI: 10.36227/techrxiv.23016185.v1.

[Dav+21] Robert David et al. "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems". In: *Proceedings of Machine Learning and Systems*. Ed. by A. Smola, A. Dimakis, and I. Stoica. Vol. 3. 2021, pp. 800–811.

[Fra+24] J. Frank et al. "A Representative Study on Human Detection of Artificially Generated Media Across Countries". In: *2024 IEEE Symposium on Security and Privacy*. 2024, pp. 55–73.

[HMT22] Khaled Ben Abbes Hassen, José J. M. Machado, and João Manuel R. S. Tavares. "Convolutional Neural Networks and Heuristic Methods for Crowd Counting: A Systematic Review". In: *Sensors* 22.14 (2022), p. 5286. DOI: 10.3390/s22145286.

[Jeg+] Nidhal Jegham et al. "YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions". In: *Yolo11, and Their Previous Versions* ().

[Joc20] Glenn Jocher. *Ultralytics YOLOv5*. 2020. DOI: 10.5281/zenodo.3908559.

[Li+23] Kangting Li et al. "A study on multiple occupant behavior detection based on computer vision technique". In: *Building Simulation 2023*. Vol. 18. IBPSA. 2023, pp. 676–680.

[Lin+14] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014).

[Lu+23] Zeyu Lu et al. "Seeing is not always believing: Benchmarking human and model perception of ai-generated images". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 25435–25447.

[MDN19] Roberto Marroquin, Julien Dubois, and Christophe Nicolle. "WiseNET: An indoor multicamera multi-space dataset with contextual information and annotations for people detection and tracking". In: *Data in Brief* 27 (2019), p. 104654. DOI: 10.1016/j.dib.2019.104654.

[Mar+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015.

[Mon+22] Lorenzo Monti et al. "Edge-based transfer learning for classroom occupancy detection in a smart campus context". In: *Sensors* 22.10 (2022), p. 3692.

[Mun+24] Balakumar Muniandi et al. "AI-Driven Energy Management Systems for Smart Buildings". In: *Power System Technology* 48.1 (2024), p. 280. DOI: 10.52783/pst.280.

[MRP24] Canicius Mwitta, Glen C. Rains, and Eric Prostko. "Evaluation of Inference Performance of Deep Learning Models for Real-Time Weed Detection in an Embedded Computer". In: *Sensors* 24.2 (2024), p. 514. DOI: 10.3390/s24020514.

[Ngu+24] H. Nguyen et al. "Accurate Real-time Detection of Vehicles and Pedestrians on Edge Device with Scaled-YOLOv4 and TOPST AI". In: *2024 24th International Conference on Control, Automation and Systems (ICCAS)*. 2024, pp. 1293–1298. DOI: 10.23919/ICCAS63016.2024.10773348.

[Ost+19] M. Ostadijafari et al. "Smart Building Energy Management using Nonlinear Economic Model Predictive Control". In: *2019 IEEE Power & Energy Society General Meeting (PESGM)*. 2019, pp. 1–5. DOI: 10.1109/PESGM40551.2019.8973669.

[Saf+21] Ali Saffari et al. "Battery-free camera occupancy detection system". In: *Proceedings of the 5th international workshop on embedded and mobile deep learning*. 2021, pp. 13–18.

[SB24] Anıl Sezgin and Aytuğ Boyacı. "Advancements in Object Detection for Unmanned Aerial Vehicles: Applications, Challenges, and Future Perspectives". In: *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*. 2024, pp. 1–6. DOI: 10.1109/ISDFS60797.2024.10527339.

[SC23] Tsu-Chuan Shen and Edward T.-H. Chu. "Edge-Computing-Based People-Counting System for Elevators Using MobileNet-Single-Stage Object Detection". In: *Future Internet* 15.10 (2023), p. 337. DOI: 10.3390/fi15100337..

[TWC20] Paige Wenbin Tien, Shuangyu Wei, and John Calautit. "A computer vision-based occupancy and equipment usage detection approach for reducing building energy demand". In: *Energies* 14.1 (2020), p. 156.

[Ult23] Ultralytics. *Configuration - Ultralytics YOLO Docs*. 2023. URL: https://docs.ultralytics.com/usage/cfg/.

[Umw24] Umweltbundesamt. *Energiesparende Gebäude*. 2024. URL: https://www.umweltbundesamt.de/themen/klima-energie/energiesparen/energiesparende-gebaeude.

[Ver+23] V. M. Vergara et al. "Comparing Deep Learning Performance for Aircraft Detection in Satellite Imagery". In: *NAECON 2023 - IEEE National Aerospace and Electronics Conference*. 2023, pp. 86–91. DOI: 10.1109/NAECON58068.2023.10366019.

[Zha+24] Wuxia Zhang et al. "Deep learning models for vision-based occupancy detection in high occupancy buildings". In: *Journal of Building Engineering* 98 (2024), p. 111355.