Experiments of Crowd Detection for Crowd Digital Twins

Experiments of Crowd Detection for Crowd Digital Twins

Kuan Pok Chong[1], Chon Hou Lai[1], Weibo Ling[1], Zhuoqian Lu[1], Rui Wang[1], Ruoqi Wang[1], Yanjun Yu[2], Chan-Tong Lam[3], Su-Kit Tang[3], Alex Testa[4], Giovanni Delnevo[4], Roberto Casadei[4], Roberto Girau[4], Silvia Mirri[4]

[1]Bachelor of Science in Computing, Macao Polytechnic University, Macao SAR, China.
[2]Guangdong University of Foreign Studies, China.
[3]Faculty of Applied Sciences, Macao Polytechnic University, Macao SAR, China.
[4]Department of Computer Science and Engineering, University of Bologna, Bologna, Italy.

Contributing authors: p2304402@mpu.edu.mo; p2304281@mpu.edu.mo; p2320609@mpu.edu.mo; p2320502@mpu.edu.mo; p2111671@mpu.edu.mo; p2211317@mpu.edu.mo; 20231203669@gdufs.edu.cn; ctlam@mpu.edu.mo; sktang@mpu.edu.mo; alex.testa2@unibo.it; giovanni.delnevo2@unibo.it; roby.casadei@unibo.it; roberto.girau@unibo.it; silvia.mirri@unibo.it;

Abstract

The development of a crowd digital twin offers significant potential for enhancing public safety, urban planning, and event management. A key challenge in creating such a digital twin lies in the efficient and accurate acquisition of crowd-related data, particularly through object detection models deployed on resource-constrained devices. Through a series of experiments, we compare TinyML and Edge approaches in terms of detection accuracy, inferencing time, and resource utilization. Our findings highlight the trade-offs inherent in selecting detection models for crowd digital twin applications, underscoring the importance of aligning model choice with specific deployment needs.

Keywords: human crowds detection, crowd digital twin, tinyml, edge ml, crowd management, crowd safety

1 Introduction

Digital twins and digital shadows have proven to be effective in numerous domains, providing valuable insights for monitoring, simulating, and optimizing complex environments and systems [1–3]. Crowd detection is a crucial component of modern urban management and public safety systems, enabling the real-time monitoring and analysis of large groups of people in various environments [4]. By leveraging advanced computer vision techniques, crowd detection algorithms can identify and track individuals within a crowd, estimate crowd density, and detect abnormal behaviors or patterns that may indicate potential safety hazards [5]. The integration of TinyML [6] and Edge ML [7] into crowd detection systems presents significant opportunities for advancing real-time monitoring and analysis of public spaces. TinyML, which enables machine learning models to run on microcontrollers and low-power devices, offers substantial benefits in terms of efficiency and deployment versatility [8]. Similarly, Edge ML, which involves executing machine learning algorithms directly on edge devices, brings substantial advantages in terms of real-time data processing and reduced dependence on cloud infrastructure [9].

In previous work, the concept of Crowd Digital Twin has been proposed [10]. It can be defined as a digital replica of a group of co-located humans, capturing collective behaviors and spatio-temporal relationships within the group and with their environment

In this work, we put the focus on crowd detection. Sanchez et al. presented a taxonomy of crowd behaviour analysis through deep learning [11] based on two dimensions. The former distinguishes depending on how the crowd is treated. In microscopic approaches, the crowd is treated as a collection of individuals while in macroscopic one the crowd is treated as a whole single entity. The latter dimension is the pipeline for crowd behaviour analysis that is composed of four main stages: detection, tracking, feature extraction, and crowd behaviour classification and anomaly detection.

In this work-in-progress paper, we focus on microscopic approaches for the detection of crowd density with emphasis on low-resource devices. We evaluate the performance of three different detection models deployed on distinct hardware platforms: FOMO on the OpenMV7 camera and YOLOv5 small on the Nvidia Jetson Nano 2. This work aims to assess the trade-offs between these models in terms of accuracy, inferencing time, and resource usage, providing valuable insights into the feasibility and efficiency of different approaches to data acquisition in the context of a crowd digital twin. By contrasting the performance of these models, we seek to identify the optimal configuration for various deployment scenarios, considering the specific requirements of crowd-monitoring applications.

The rest of the paper is structured as follows. Section 2 details our approach while Section 3 illustrates the results obtained in the various experiments. Finally, Section 4 concludes the paper, paving the way for future works.

2 Materials and Methods

This Section describes the dataset used in this study, the detection models together with the corresponding devices, and the experimental settings employed.

2.1 Dataset Description

We combined two different datasets. The former one is the Penn-Fudan Dataset [12]. It is an image database for pedestrian detection. The images are taken from scenes around campus and urban streets. All these images were labelled as 'walking'. The latter one is the Human Activity Recognition [13]. It features 15 different classes of Human Activities such as calling and drinking. We filtered just the images with labels 'cycling' and 'sitting' to be able to recognize different situations in crowds. Ultimately, our dataset comprised 406 images, containing 915 *objects* to detect. Data were split into 80% (324) for the training and 20% (82) for the testing.

2.2 Hardware and Detection Models

The deployment of a crowd digital twin necessitates the integration of advanced data acquisition systems capable of real-time processing and high-accuracy detection. This subsection delineates the specific hardware and detection models utilized in our experiments.

The OpenMV7 camera serves as the primary TinyML device in our experimental setup [14]. This compact, efficient camera is designed for embedded computer vision applications, making it an ideal candidate for edge-level processing in our crowd digital twin framework. The OpenMV7 camera is equipped with a microcontroller capable of executing machine learning models, thus enabling on-device inference and reducing the need for continuous data transmission to centralized servers.

For detection algorithms, the OpenMV7 camera employs the FOMO (Faster Objects, More Objects) variant of MobileNetV2 [15, 16]. MobileNetV2 is a lightweight convolutional neural network optimized for mobile and embedded vision applications. The FOMO adaptation is specifically tailored to enhance the real-time object detection capabilities on resource-constrained devices like the OpenMV7. This combination ensures that the device can detect and track multiple individuals in a crowd with minimal latency, thus contributing to the real-time aspect of the digital twin.

The Nvidia Jetson Nano 2 is the edge device within our setup, providing a more powerful platform for data processing and inference than the TinyML device. The Jetson Nano 2 is equipped with a Quad-core ARM Cortex-A57 CPU and a 128-core Maxwell GPU, delivering significant computational power to handle complex machine learning tasks.

On the Jetson Nano 2, we utilize YOLOv5, a state-of-the-art object detection algorithm renowned for its speed and accuracy. YOLOv5 small is specifically designed for scenarios requiring fast and accurate object detection while operating under constrained computational environments. This version of YOLOv5 reduces the number of parameters (i.e., 7.2M) and computational demands compared to larger variants, making it a suitable choice for the Jetson Nano 2

By comparing YOLOv5 with MobileNetV2 implementations on both the OpenMV7 camera and the Nvidia Jetson Nano 2, we aim to highlight the trade-offs between edge and TinyML processing. Moreover, we add to the comparison YOLOv5 medium, a bigger version that would require more computational capabilities. This

comparison sheds light on the practical considerations of deploying a crowd digital twin, such as latency, bandwidth requirements, and scalability.

2.3 Experimental Settings

The experimental setup for evaluating the detection models across different hardware platforms was carried out on Edge Impulse [17]. Data were preprocessed using standard techniques, including resizing images to the required model input dimensions, normalising pixel values, and data augmentation techniques such as random rotation, flipping, and scaling. The training process was conducted within the Edge Impulse platform, leveraging its automated machine learning pipeline. Key settings during the training included learning rate and number of epochs. To comprehensively assess the performance of the detection models across different hardware platforms, we employ a set of standard evaluation metrics for object detection [18], including mean Average Precision and Recall, with different confidence thresholds. An important difference between FOMO and other object detection algorithms is that it does not output bounding boxes. In fact, FOMO is trained on the centroids of objects, making it easier to count closer objects. Hence, it requires the use of other standard classification metrics, such as precision, recall, and f1-score. Furthermore, real-time performance metrics, including inference time, peak RAM usage, and flash usage, were logged and analyzed throughout the experiments. These metrics were crucial in comparing the effectiveness of the TinyML and edge devices, providing insights into the trade-offs between different deployment strategies for a crowd digital twin.

3 Results

The detection accuracy of the models (i.e., FOMO, YOLOv5 small, and YOLOv5 medium) are discussed in isolation in the subsections 3.1, 3.2, and 3.3 while subsection 3.4 details the comparison of on-devices resources.

3.1 FOMO on OpenMV7

The training process of FOMO has been carried out varying two main hyperparameters, number of epochs and learning rate. Four different learning rates were tested: 0.002, 0.004, 0.006, and 0.008. Lower learning rates generally allow for more fine-grained adjustments, while higher rates can speed up training but risk overshooting the optimal model weights. Instead, experiments were conducted with 60 and 70 epochs to observe the impact on model performance. More epochs generally allow the model to learn more complex patterns but can lead to overfitting if extended too long.

Table 1 reports the results obtained on the validation set, composed of 20% of the training samples. The best overall performance, as indicated by the highest F1-score of 0.70, was achieved with 70 epochs and a learning rate of 0.002. This configuration also had a high precision of 0.67 and a recall of 0.74, suggesting a balanced and effective detection capability. Lower learning rates (0.002 and 0.004) combined with more epochs (70) generally yielded better precision and F1-scores, indicating that a

slower and more deliberate learning process helped the model achieve better overall accuracy.

Table 1 Results of the training process of FOMO

Epochs	Learning Rate	Precision	Recall	F1-score
60	0.006	0.4	**0.76**	0.53
60	0.008	0.42	0.75	0.54
70	0.002	**0.67**	0.74	**0.7**
70	0.004	0.59	0.68	0.63

Once selected the best combination of hyper-parameters, we moved to the test set evaluation. Results are reported in Table 2. All the metrics are computed excluding the background. The accuracy consists in the percentage of all samples with a precision score above 80%. The low metrics indicate that the model struggles with general classification tasks in the test set, possibly due to the complexity of the crowd scenes or the limitations of the model architecture in handling varied environments. This suggests that while the model might be useful in specific scenarios, its current state may not be sufficient for more demanding real-time crowd monitoring applications without further improvements.

Table 2 Results obtained with FOMO on the test set.

Metrics	Value
Precision (non-background)	0.58
Recall (non-background)	0.42
F1 Score (non-background)	0.49
Accuracy	0.23

3.2 YOLOv5 small on Nvidia Jetson Nano 2GB

The training of YOLOv5 small has been conducted without a tuning phase. The number of epochs has been set to 100 together with an early-stopping criterion. Results are reported in Table 3, column Training. The overall mean Average Precision (mAP) across all Intersection over Unit (IoU) thresholds is 0.57, indicating that the model has a moderate level of precision and recall averaged across different IoU thresholds. Anyway, the mAP at an IoU threshold of 50% is 0.92, suggesting that the model performs excellently when the overlap between predicted and ground truth bounding boxes is relatively lenient. Then, the model has been evaluated on the test set. The results are reported in column Testing in Table 3. Also in this case, the accuracy represents the percentage of all samples with a precision score above 80%. Overall, the model achieved an accuracy of 0.78 on the testing set. This suggests a good performance, though there is still room for improvement.

Table 3 Results obtained with YOLOv5 small on the training and testing sets.

Metrics	Training	Testing
Accuracy	-	0.78
mAP	0.57	0.5
mAP@[IoU=50]	0.92	0.8
mAP@[IoU=75]	0.62	0.55
mAP@[area=small]	0.65	0.16
mAP@[area=medium]	0.49	0.42
mAP@[area=large]	0.63	0.59
Recall@[max_detections=1]	0.4	0.32
Recall@[max_detections=10]	0.63	0.55
Recall@[max_detections=100]	0.63	0.55
Recall@[area=small]	0.65	0.17
Recall@[area=medium]	0.57	0.47
Recall@[area=large]	0.68	0.63

3.3 Comparison with YOLOv5 medium

The medium version of YOLOv5 has been also evaluated. Training and testing results are reported respectively in Table 4.

Table 4 Results obtained with YOLOv5 medium on the training and testing sets.

Metrics	Training	Testing
Accuracy	-	0.8
mAP	0.58	0.52
mAP@[IoU=50]	0.93	0.82
mAP@[IoU=75]	0.61	0.54
mAP@[area=small]	0.53	0.28
mAP@[area=medium]	0.48	0.42
mAP@[area=large]	0.65	0.61
Recall@[max_detections=1]	0.41	0.32
Recall@[max_detections=10]	0.65	0.56
Recall@[max_detections=100]	0.65	0.56
Recall@[area=small]	0.65	0.31
Recall@[area=medium]	0.56	0.47
Recall@[area=large]	0.7	0.64

As shown, there is no significant difference with respect to the small version. This suggests that to fully harness the model's capabilities, it is essential to incorporate more data into the dataset. By expanding the dataset, especially with a diverse range of scenarios and object variations, the model can learn more effectively and improve its ability to generalize across different conditions.

3.4 On-Device Resources Comparison

The comparison of on-device resource usage focuses on three metrics: Inferencing Time, Peak RAM Usage, and Flash Usage. FOMO has the fastest inference time at 326ms, ideal for real-time processing. YOLOv5s follows with 375ms, slightly slower but still

quick. YOLOv5m has the longest time at 1059ms, making it less suitable for time-sensitive applications. With regard to RAM usage, FOMO is lightweight, using just 892K, making it fit for devices with limited memory like TinyML. The statistics for YOLOv5s and YOLOv5m aren't available but are expected to be much higher due to their complexity. Finally, FOMO uses only 78.5K of flash memory, while YOLOv5 models require significantly more, reflecting their more complex architectures.

Table 5 On-device resources comparison.

Model	Inferencing Time	Peak RAM Usage	Flash Usage
FOMO	326ms	892K	78.5K
YOLOv5s	375ms	N/A	13.5M
YOLOv5m	1059ms	N/A	39.9M

The comparison highlights the trade-offs between model complexity and resource demands. FOMO, with its quick inference time and low resource usage, is ideal for environments with limited resources. YOLOv5s offers a balance between performance and resource consumption, suitable for scenarios needing moderate efficiency [19]. YOLOv5m, while offering the best detection capabilities, requires more time and storage, making it better suited for applications prioritizing accuracy over speed and resource efficiency.

4 Conclusion and Future Works

In this paper, we explored crowd detection approaches for developing a crowd digital twin, focusing on the deployment and evaluation of different detection models across various hardware platforms. Our study involved the use of FOMO on the OpenMV7 camera and YOLOv5 small on the Nvidia Jetson Nano 2. The results of our experiments demonstrated the trade-offs between detection accuracy, inference time, and resource usage. Even if FOMO is ideal for resource-constrained environments, it has low detection accuracy, suggesting that it is better suited for macroscopic approaches rather than microscopic ones. On the other hand, while YOLOv5 small requires more computational resources, it achieves an accuracy level that meets the standard for microscopic approaches.

While this study offers valuable insights into detection models for crowd digital twin applications, further research is needed. More extensive testing across diverse datasets and environments is essential to validate the generalizability of our findings, particularly under varying conditions like lighting, weather, and crowd density. Additionally, future work should explore other dimensions suggested by Sanchez et al. [11], including the macroscopic scale and stages like tracking, feature extraction, crowd behavior classification, and anomaly detection. Moreover, the use of other deep learning approaches, both supervised [20] and unsupervised [21], could be investigated.

Acknowledgement

This work was supported by project ECOSISTER (ECS_00000033) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU.

References

[1] Furini, M., Gaggi, O., Mirri, S., Montangero, M., Pelle, E., Poggi, F., Prandi, C.: Digital twins and artificial intelligence: As pillars of personalized learning models. Communications of the ACM **65**(4), 98–104 (2022)

[2] Ferretti, S., Furini, M., Palazzi, C.E., Roccetti, M., Salomoni, P.: Www recycling for a better world. Communications of the ACM **53**(4), 139–143 (2010)

[3] Ceccarini, C., Chan, K.K., Lok, I.L., Tse, R., Tang, S.-K., Prandi, C.: Fostering user's awareness about indoor air quality through an iot-enabled home garden system. In: 2021 International Conference on Computer Communications and Networks (ICCCN), pp. 1–4 (2021). IEEE

[4] Gündüz, M.Ş., Işık, G.: A new yolo-based method for real-time crowd detection from video and performance analysis of yolo models. Journal of Real-Time Image Processing **20**(1), 5 (2023)

[5] Altowairqi, S., Luo, S., Greer, P.: A review of the recent progress on crowd anomaly detection. International Journal of Advanced Computer Science and Applications **14**(4) (2023)

[6] Delnevo, G., Mirri, S., Prandi, C., Manzoni, P.: An evaluation methodology to determine the actual limitations of a tinyml-based solution. Internet of Things **22**, 100729 (2023)

[7] Murshed, M.S., Murphy, C., Hou, D., Khan, N., Ananthanarayanan, G., Hussain, F.: Machine learning at the network edge: A survey. ACM Computing Surveys (CSUR) **54**(8), 1–37 (2021)

[8] Banbury, C., Njor, E., Stewart, M., Warden, P., Kudlur, M., Jeffries, N., Fafoutis, X., Reddi, V.J.: Wake vision: A large-scale, diverse dataset and benchmark suite for tinyml person detection. arXiv preprint arXiv:2405.00892 (2024)

[9] Tan, Y., Li, Q., Peng, J., Yuan, Z., Jiang, Y.: Air-cad: Edge-assisted multi-drone network for real-time crowd anomaly detection. In: Proceedings of the ACM on Web Conference 2024, pp. 2817–2825 (2024)

[10] Casadei, R., Delnevo, G., Girau, R., Mirri, S.: Modelling groups of humans: Towards crowd digital twins. In: 29th IEEE Symposium on Computers and Communications (ISCC). IEEE, ??? (2024)

[11] Sánchez, F.L., Hupont, I., Tabik, S., Herrera, F.: Revisiting crowd behaviour analysis through deep learning: Taxonomy, anomaly detection, crowd emotions, datasets, opportunities and prospects. Information Fusion **64**, 318–335 (2020)

[12] Wang, L., Shi, J., Song, G., Shen, I.-f.: Object detection combining recognition and segmentation. In: Asian Conference on Computer Vision, pp. 189–199 (2007). Springer

[13] Human Action Recognition, HAR dataset - Kaggle. https://www.kaggle.com/datasets/meetnagadia/human-action-recognition-har-dataset. Accessed: 2024-07-01

[14] OpenMV Cam h7. https://openmv.io/products/openmv-cam-h7. Accessed: 2022-07-22

[15] Edge Impulse FOMO. https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/object-detection/fomo-object-detection-for-constrained-devices. Accessed: 2022-12-22

[16] Edge Impulse FOMO. https://www.edgeimpulse.com/blog/announcing-fomo-faster-objects-more-objects. Accessed: 2022-12-22

[17] Janapa Reddi, V., Elium, A., Hymel, S., Tischler, D., Situnayake, D., Ward, C., Moreau, L., Plunkett, J., Kelcey, M., Baaijens, M., et al.: Edge impulse: An mlops platform for tiny machine learning. Proceedings of Machine Learning and Systems **5** (2023)

[18] Padilla, R., Netto, S.L., da Silva, E.A.B.: A survey on performance metrics for object-detection algorithms. In: 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 237–242 (2020)

[19] Ceccarini, C., Prandi, C.: Escapecampus: exploiting a game-based learning tool to increase the sustainability knowledge of students. In: Proceedings of the 2022 ACM Conference on Information Technology for Social Good, pp. 390–396 (2022)

[20] Ling, Z., Delnevo, G., Salomoni, P., Mirri, S.: Findings on machine learning for identification of archaeological ceramics-a systematic literature review. IEEE Access (2024)

[21] Liu, L., Delnevo, G., Mirri, S.: Unsupervised hyperspectral image segmentation of films: a hierarchical clustering-based approach. Journal of Big Data **10**(1), 31 (2023)