Crowd Analysis on Edge Devices: A Comparative Study of Neural Networks on Blurred Images

SAMUEL COSTE, University of Twente, The Netherlands

Crowd counting is an important task in various applications such as public safety, traffic management, and surveillance. In recent years, edge devices have become increasingly popular for crowd counting due to their low cost and high efficiency. However, traditional image processing techniques for crowd counting often struggle to handle images taken in challenging environments, such as low light or crowded scenes. In this paper, we propose a new approach for crowd counting on edge devices using blurred images. The use of blurred images allows for crowd counting in challenging environments while also preserving privacy. This paper will com- pare the performance of different neural network architectures on a data set of blurred images of crowded scenes and evaluate their accuracy, robustness, and efficiency. We also examine the impact of different factors on the performance of our approach, such as image resolution, blur type and blur level. Our results show that crowd counting on edge devices using blurred images is a promising approach with potential applications in various fields such as crowd management in public events, retail, transportation, surveillance and security, public health, and emergency and disaster management.

Additional Key Words and Phrases: Crowd Analysis; Neural Networks; YOLOv7; TensorFlow; Edge Devices; Blurred Images

1 INTRODUCTION

In recent years, the use of neural networks for image analysis has gained significant attention in various fields such as surveillance, crowd management, and traffic monitoring. One of the challenging tasks in image analysis is counting the number of objects, particularly humans, in blurred images. In this research paper, we aim to investigate the best neural network architecture for counting humans in blurred images. We will compare the performance of different neural network architectures on a data set of blurred images of crowded scenes and evaluate their accuracy, robustness, and efficiency. This research will provide insights into the best practices for using neural networks for counting humans in blurred images and can be applied in various real-world applications. This research was motivated by the lack of insight in the usage of public spaces. One of the problems this paper aims to solve is to provide this insight in spaces like university libraries. By using blurred images in image analysis, it can help to maintain the privacy of individuals. This is because the (physical) blurring of the images makes it difficult to recognise individuals and their characteristics, such as facial features, which can be sensitive information. Therefore, using blurred images in image analysis can be a valuable solution for maintaining privacy while still achieving the goal of counting the number of humans present in the scene. Additionally, the use of physically blurred images could be an effective way to respect the privacy laws and regulations and to comply with the data protection standards. The main question that will be answered in this paper is: How does the use of blurred images affect the accuracy and efficiency of crowd counting on neural networks when running on edge devices? To answer this a comparison of different neural network architectures will be made, as well as an evaluation of their performance on blurred images using edge devices. In the conclusion section other ideas for real-world scenarios that could benefit from these techniques will be proposed.

TScIT 38, Februari 3, 2022, Enschede, The Netherlands

© 2022 Association for Computing Machinery.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in , https://doi.org/10.1145/nnnnnnn.nnnnnnn.

2 APPROACH

2.1 Research Design

For this research we used an experimental design to perform a comparative study. The basis was the collection of images, which would be blurred using different techniques and degrees of blurring, which would then each be assessed by several neural network models, to determine the effect the blurring had on the accuracy and performance of the models. The purpose of doing it this way was to isolate one variable at the time, making it possible to pinpoint which cause-and-effect relationships existed between the variables and the obtained results.

2.2 Tools used

Several tools were used to perform this research, in this subsection we will briefly discuss each, explain the choices made, and their contribution to this project.

2.2.1 Data sets. Two data sets were used during this research. The most relevant one for the main use case was the Mall data set first introduced in a paper by Chan et al. (2015) [7]. The data set includes a total of 2000 images, each with a resolution of 640x480 pixels. The images were captured by a webcam in a shopping mall, resulting in pictures with similar lighting conditions. They contain all contain people in different postures and sometimes partially covered. The number of people present in each frame ranges from 15 to 48, making it perfectly suited for this research. An example of a untreated image can be found in Appendix A.2, fig.5.

These picture were then modified, using Adobe Photoshop's Batch functionality, applying different blurring techniques and degrees. This resulted in new data sets that could be used to evaluate the ability of the neural networks to analyse them. As a Gaussian Blur is a good approximation for a lens which is out of focus [9] this was the primary blurring technique used. Another Photoshop filter that was used was one called "Lens Blur", but because it is not clear by the Adobe Documentation what this blurring effect exactly does, it was not possible to judge if this could be replicated physically. The same holds for the "Box Blur" technique, which was abandoned for these reasons after the initial tests. By varying the radius of the blur different blurring degrees could be achieved. This ranges from a very light blur where people are easily recognised as can be seen in fig.6 to images where it is difficult even for humans to identify the shapes of persons.

The second data set was called 'Human Crowd Dataset' [1] and it was used to verify the initial results. The data set is a collection of over 16.000 images of various groups of people in very varying settings. It was only tested using a Gaussian blur with a radius set to 3px. In the discussion section this specific choice will be elaborated. An example picked at random after the blurring was applied can be found in Appendix A.2, fig.18.

2.2.2 Hardware. As mentioned before, Adobe Photoshop was used to achieve a blurring effect on the images, this was running on a powerful PC as it does take a lot of computing power to process all these images. All the results gathered during this research came from a Raspberry Pi 4 Model B with 1GB of RAM. This was of course limiting the selection of models that could be chosen from, which will be further elaborated on in the next subsection. The results were then imported back on a PC, where Excel was used to analyze the results and plot the graphs.

2.2.3 Neural Networks. As this research was completely performed on a Raspberry Pi, the neural networks explored were chosen with this limitation in mind. The first model used was YOLOv7-tiny (For the purpose of readability, we will refer to Yolov7-tiny as "Yolo" throughout this paper). This model was chosen for it's known accuracy on the COCO-Dataset as can be seen here: [3]. Because of the high performance and accuracy of this model, as well as the availability of code [5] this model was used. Because the full model would be to heavy for the Pi, a smaller version of Yolov7, Yolov7-tiny [18] was used. This model uses fewer layers and parameters, which allows it to run on edge devices such as the Raspberry Pi.

2.2.4 TensorFlow Lite. TensorFlow and TensorFlow Lite are open source frameworks that were developed by Google. These frameworks include tools for building, training and running neural network models. TensorFlow Lite is a lightweight version of TensorFlow, specifically designed for edge devices. As these devices have less power and memory, it was designed to be more efficient than TensorFlow (both in terms of power and memory). More details about TensorFlow Lite and the optimisation techniques it offers can be found in these papers: [5, 16, 24]. TensorFlow Lite was used to run two different models, both developed by Google. The first [10] is based on SSD Mobilenet [17]. This model had some limitations, as it would not detect more than ten objects in an image. The second one, the "Google Mobile Object Localizer" [11] was capped at 100 object, making it much better suited for our application.

3 RESULTS

The results collected during this research were for each image the count as perceived by the neural networks, and to be able to confirm that the objects detected were indeed persons, the annotated images with bounding boxes around each detected person were saved as well. This was used to compare the accuracy of each run against the labeled data for each set. The results were then imported in Excel to calculate the average accuracy and standard deviation. The complete sheet with all the data points is available on GitHub [2]. The results are represented in the three box plots [1, 2, 3], one for each algorithm. In the graphs abbreviations are used to denote the blurring techniques applied. The 'Control' label is for the unaltered version of the image, this batch has not been modified from it's original image definition. 'GB' stands for 'Gaussian Blur', which is followed by the pixel radius, the degree of blurring, at which it was applied. 'LB' means 'Lens Blur' and the same logic was used to define it's degree of blurring.

3.1 Yolo

The first results will cover running Yolo on the Mall Data set. In these results the pictures were ordered in descending order based on the number of people present.

On the results graph [1] we can see for each blurring technique and blurring degree applied to the data set how many persons were detected by Yolo. We can clearly observe that when the degree of the blurring is increased the neural network is less likely to give an accurate count. For these tests a probability threshold of 0.1 was used. Meaning Yolo would count a human starting from 10% certainty. This was used because the certainty of Yolo goes down very quickly when analysing a blurred image. The downside to this technique is that Yolo might count non-humans as being humans, which explain that on some images the counted number of people surpasses the actual number of humans as stated in the labels. A balance has to be found, depending on the blurring degrees to find a appropriate probability threshold which will exclude things like mannequins (often miscounted in these examples) from real humans.

The graph also shows results higher than 100%. This shows that for some blurring degree the acceptance threshold was to low, resulting in objects being counted as persons.

The accuracy is not impacted by the amount of people in the pictures, as for each blurring technique the percentage of missed people stays constant.

3.2 SSD Mobilenet

As can be seen in our graph [2] this model never exceeds 10 persons detected per image. As our data ranges from 15 to 48 people per image this is insufficient. This results in very skewed results. The model seems to perform better on images with less persons in them, but this can not be verified, as it the results are mainly caused by the model being capped at 10 persons. It is interesting to see that this model performs better than Yolo when applied to a high blur. Where Yolo almost incapable of detecting persons, The SSD Mobilenet still manages to identify a few.

3.3 Google Mobile Object Localizer

The results for the Object Localizer were more similar to the results achieved on when running Yolo. However, when looking at the images with the bounding boxes it is clear that the Object Localizer spotted a lot of objects and counted them as humans. This has not been accounted for in the results table, but the effects are visible in the Standard Deviation Table, as the deviation is a lot higher for the Object Localizer compared to Yolo.

4 DISCUSSION

4.1 Blurring Techniques

During this research the main blurring technique applied was a Gaussian Blur. This choice, as stated in the Approach section was

Table 1. Average Accuracy of each algorithm applied on the different blurring techniques. Each cells indicates the average accuracy, defined as a percentage of identified persons on the total number of persons in the image, ± the Standard Deviation of the algorithm as applied on the set of images with the blurring degree/ technique mentioned in the first column. For every row, the highest score has a green background and the lowest score is red. The last row is the average performance measured in 'Frames Per Second' for each algorithm performing the detection on a set of 6000 images

Blurring Technique	YOLO	SSD Mobilenet	Object Localizer
Control	83,2% ± 12,3	35,2% ± 8,4	76,1% ± 43,8
Gaussian Blur 1px	76,0% ± 10,4	35,6% ± 8,9	56,4% ± 31,8
Gaussian Blur 2px	64,7% ± 11	34,4% ± 8,6	33,0% ± 16,9
Gaussian Blur 3px	48,6% ± 11,4	33,3% ± 8,1	21,3% ± 10,8
Gaussian Blur 4px	18,6% ± 10,6	29,0% ± 7,4	14,4% ± 6,5
Gaussian Blur 5px	4,3% ± 5,3	27,8% ± 8,2	10,8% ± 5,6
Gaussian Blur 6px	1,2% ± 2,8	29,2% ± 9,1	5,7% ± 3,3
Gaussian Blur 7px	0,4% ± 1,4	31,7% ± 9,5	4,6% ± 2,1
Gaussian Blur 8px	0,3% ± 1,1	34,6% ± 10,6	4,4% ± 1,7
Lens Blur 2px	83,2% ± 12,3	35,2% ± 8,4	76,1% ± 43,8
Lens Blur 4px	79,4% ± 11,7	35,6% ± 8,5	64,7% ± 38,7
Lens Blur 6px	71,5% ± 10,3	35,6% ± 9,4	52,4% ± 29,2
Performance	0,80 FPS	6,27 FPS	12,9 FPS

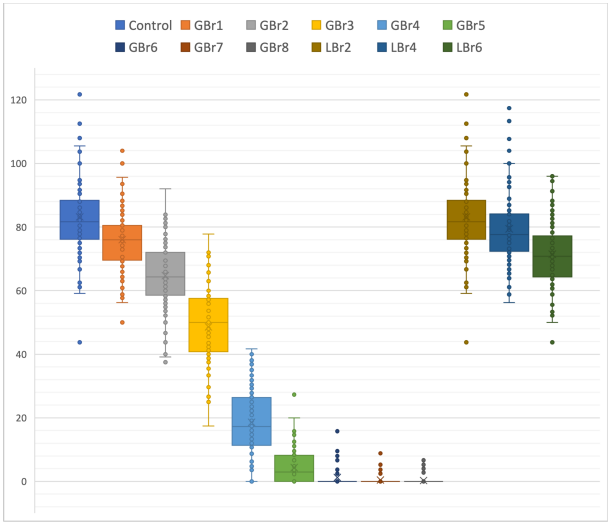

Fig. 1. Box plot showing the results of Yolo on the different blurring degrees. This shows a clear drop in accuracy when going from GBr3 to GBr4.

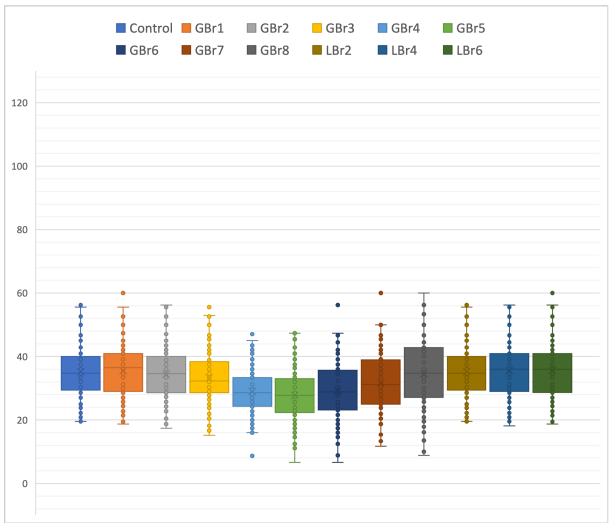

Fig. 2. Box plot showing the results of SSD Mobilenet on the different blurring degrees. Because the algorithm is capped at detecting 10 objects the results are limited.

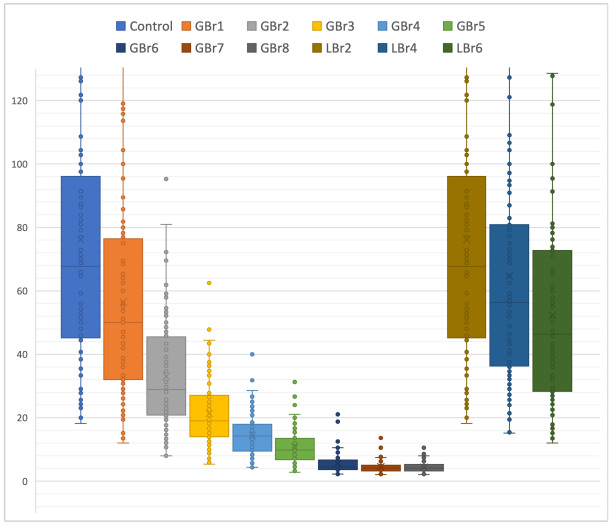

Fig. 3. Box plot showing the results of Object Localizer on the different blurring degrees. The high standard deviation shows that the precision is low. On the same blurring degree the model will vary a lot in accuracy.

made to closely match real world experiments that could follow. A three pixel blurring radius was chosen to test on the larger data set, as at this degree of blurring Yolo was still able to detect around 50% of the persons present. After experimenting with various probability thresholds and examining the results, it was concluded that a 10% certainty threshold on the blurred images performed best for Yolo and SSD Mobilenet and 18% was best suited for the Object Localizer. Going lower than these thresholds would lead to a big increase in false positives, while increasing this threshold would cause the accuracy to drop too quickly on blurred images.

4.2 Neural Networks

This research used Yolo, based on a convolutional neural network (CNN), as well as two models based on a Vision Transformer (ViT). The average accuracy for these models was highly dependent on the situation. For clear images Yolo was the best performing, but at higher degrees of blurring the Object Localizer would perform better. The SSD Mobilenet was very difficult to judge due to it's cap on ten objects detected simultaneously. It should be noted that these were all light versions of the models because they had to run

on a Raspberry Pi, and the performance of the full models could be different.

4.3 Limitations and Future Research

This brings us to the limitations of this research. First of all, research should be performed on the blurring degrees to define how much blurring is needed to protect someone's privacy. During this research it did not make sense to use a blurring degree higher than three pixels for a Gaussian Blur, as the accuracy would go down very quickly, and comparing results beyond this threshold was difficult as the rate of false positives would go up. But this does not mean that a blurring radius of three pixels is sufficient to render someone unrecognisable.

Another limitation that could be researched further are the models used. The SSD Mobilenet used was very limited, but maybe that the full version running on a powerful enough computer would boast great accuracy. DINO is momentarily the state-of-the art object detector as tested on the COCO data set [24], but was not tested in this research due to lack of available code.

Due to time constraints, the models used were all pre-trained models, but maybe that a model specifically trained on blurred images could outperform these models both in accuracy and performance.

5 CONCLUSION

5.1 Summary

In this research we compared a three neural networks performing crowd counting on blurred images using a Raspberry Pi. It is clear that blurred images have a significant impact on the accuracy of the researched models. The models are still able to detect persons in the images, but especially smaller people (at the back of the images) were hard to detect in blurred circumstances. It can be concluded that an edge device such as the Raspberry Pi is able to run a well performing model to count persons on non-blurred images (83.2% accuracy), but the affects of blurring are very noticeable, with the accuracy quickly dropping as low as 1.2% on a Gaussian Blur with a radius of 6px.

The performance of the models was unaltered when running on blurred images versus non blurred images. For all three models the performance would be good enough to run this in a real-time application, updating once every second.

5.2 Practical Applications

Several practical applications can be thought of. The reason for this research paper was the usage in (university) libraries. But this can of course be used in many different settings. Some real-world scenarios where this would be applicable could be as follows:

1. Public Events: Monitor Crowd Density and ensure Public Safety. During public events it would be beneficial to monitor the density of crowds. This type on non intrusive monitoring could help municipalities when large events are organised to react in a more timely manner. This could help them manage large crowds and avoid disasters like the Seoul Halloween Crowd Crush.

2. Retail: Monitor Customer Traffic, optimise Staffing. By using crowd analysis shops could optimise their staffing depending on the time and day of the year, as well as fore large surfaces where employees could be dispatched to locations with more customers to be more efficient. Another idea would be to use this data to optimise the product placements in shops based on where people spend most of their time.

3. Surveillance: The same technology could be use in surveillance to detect the presence of persons without compromising the privacy of your own personnel.

ACKNOWLEDGMENTS

Papers that helped me in my research but I did not cite explicitly in this paper: [4, 6, 8, 12–15, 19–23]

REFERENCES

[1] [n. d.]. Human Crowd Dataset. https://www.kaggle.com/datasets/hilongnguyn/human-crowd-dataset Accessed on: January 20, 2023.
[2] [n. d.]. Mod-12. https://github.com/Samuel-FC/Mod-12 Accessed on: January 26, 2023.
[3] [n. d.]. Object Detection on COCO. https://paperswithcode.com/sota/object-detection-on-coco accessed January 29, 2023.
[4] Maha Hamdan Alotibi et al. 2019. CNN-Based Crowd Counting through IOT: Application for Saudi Public Places. *Procedia Computer Science* 158 (2019), 095–102. https://doi.org/10.1016/j.procs.2019.12.095
[5] Alexey Bochkovskiy. 2021. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2104.13647* (2021).
[6] Mathilde Caron, Hugues Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging Properties in Self-Supervised Vision Transformers. *arXiv preprint arXiv:2104.14294v2* (2021).
[7] Sam Kwong Chan, Xiaogang Wang, and Chen Change Loy. 2015. A data set and benchmark for crowd counting in still images. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*.
[8] K Chen, CC Loy, S Gong, and T Xiang. 2012. Feature Mining for Localised Crowd Counting. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 21.1–21.11. https://doi.org/10.5244/C.26.21
[9] JC Dainty. 1984. Modeling the point spread function in optical microscopy. *Journal of the Optical Society of America A* 1, 6 (1984), 624–630.
[10] Google. 2018. SSD Mobilenet. https://storage.googleapis.com/download.tensorflow.org/models/tflite/coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip Accessed on January 27, 2023.
[11] Google. 2021. Google Mobile Object Localizer. https://tfhub.dev/google/lite-model/object_detection/mobile_object_localizer_v1/1/default/1 Accessed on January 27, 2023.
[12] Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*. 1–12.
[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2018. Structured pruning of convolutional neural networks. In *International Conference on Learning Representations*. 1–14.
[14] P Jawale, H Patel Chaudhary, and N Rajput. 2020. Real-Time Object Detection using TensorFlow. *Journal name* volume (2020), pages.
[15] H Jiang and S Wang. 2016. Object Detection and Counting with Low Quality Videos. (2016).
[16] T Lin, W Liu, C Shen, and J Jia. 2021. InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions. *arXiv preprint arXiv:2104.13865* (2021).
[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision*. Springer, Cham, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
[18] Qengineering. 2022. YoloV7-ncnn-Raspberry-Pi-4. https://github.com/Qengineering/YoloV7-ncnn-Raspberry-Pi-4 accessed January 15, 2023.
[19] Mohamed Sayed and Gabriel Brostow. 2021. Improved Handling of Motion Blur in Online Object Detection. In *Proceedings of the Computer Vision Foundation Conference*, Vol. volume. pages.
[20] Mingxing Tan, Quoc V Le, and Boqing Gong. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*. 6105–6114.
[21] Google AI Team. 2019. TensorFlow Lite: A lightweight library for deploying TensorFlow models on mobile and embedded devices. https://www.tensorflow.org/lite
[22] Google AI Team. 2020. TensorFlow Lite Micro: A Lightweight Edge TensorFlow Library for Microcontrollers. https://www.tensorflow.org/lite/microcontrollers

Fig. 6. YOLOv7-tiny result with Gaussian Blur r = 1px

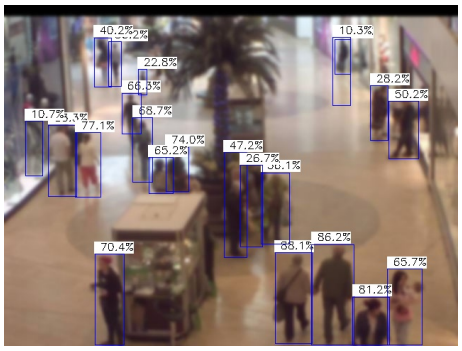

Fig. 7. YOLOv7-tiny result with Gaussian Blur r = 2px

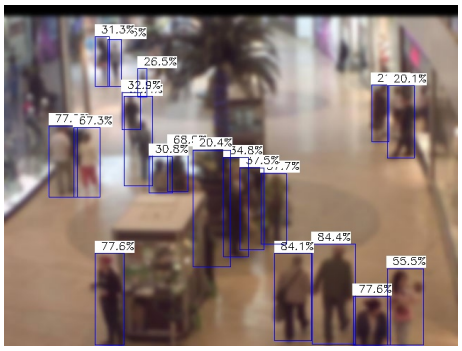

Fig. 8. YOLOv7-tiny result with Gaussian Blur r = 3px

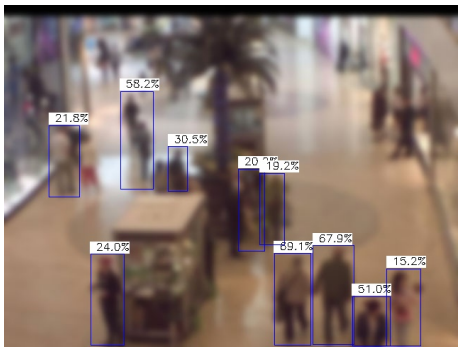

Fig. 9. YOLOv7-tiny result with Gaussian Blur r = 4px

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[24] W Wang, J Dai, Z Chen, Z Huang, Z Li, X Zhu, X Hu, T Lu, L Lu, H Li, X Wang, and Y Qiao. 2022. InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions. *arXiv preprint arXiv:2211.05778v2* (2022).

A APPENDICES

A.1 Appendix A.1

These are samples from the data sets showing the blurring and the bounding boxes drawn by the different neural networks.

Fig. 4. First Image in the Mall Data set

Fig. 5. YOLOv7-tiny result on the control image.

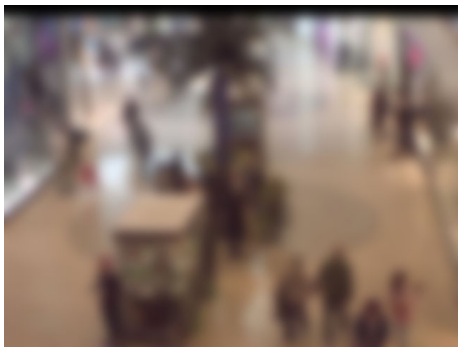

Fig. 13. YOLOv7-tiny result with Gaussian Blur r = 8px

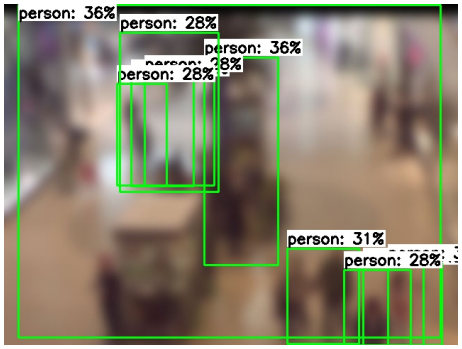

Fig. 14. SSD Mobilenet result with Gaussian Blur r = 8px

Fig. 15. YOLOv7-tiny result with "Lens Blur" r = 2px

Fig. 16. YOLOv7-tiny result with "Lens Blur" r = 4px

Fig. 17. YOLOv7-tiny result with "Lens Blur" r = 6px

Fig. 18. Example from the Human Crowd set