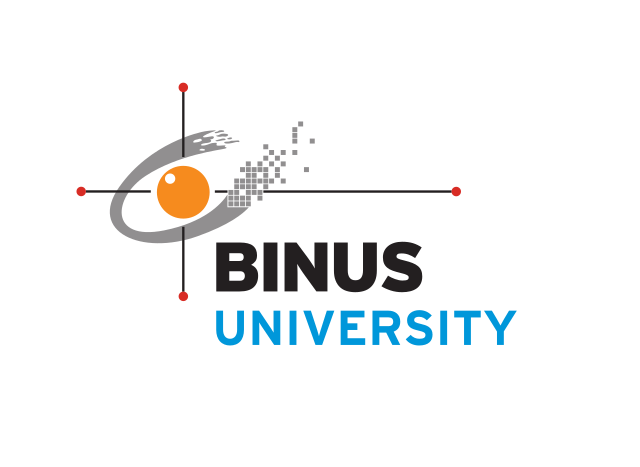
COMP7116001 – Computer Vision

**Lecturer: WAWAN CENGGORO, S.Kom., M.TI**

**LA08 - AOL**

**Efficient Elevator System by Amount of People Recognition using YOLOv8**



**Nama:**

1. **2540119545 - Felicia Angelica**
2. **2540118555 - Tora Sangputra Yopie Winarto**
3. **254012499 - Kevin Wijaya**
4. **2540125485 - Zefanya Delvin Sulistiya**
5. **2540124974 - Romington Hydro**

**BINA NUSANTARA UNIVERSITY**

**2023/2024**

# Table of Contents

Contents

[Table of Contents 2](#_Toc153557985)

[1. Introduction 3](#_Toc153557986)

[2. Methodology 3](#_Toc153557987)

[2.1. Workflow 3](#_Toc153557988)

[2.2. Dataset 3](#_Toc153557989)

[2.3. YOLOv8 4](#_Toc153557990)

[2.4. Streamlit 5](#_Toc153557991)

[3. Results and Discussions 5](#_Toc153557992)

[3.1. Project UI and Explanation 5](#_Toc153557993)

[3.2. Metrics 6](#_Toc153557994)

[3.3. GitHub Link 7](#_Toc153557995)

[4. Conclusion 7](#_Toc153557996)

[References 8](#_Toc153557997)

# 1. Introduction

The making of this project is primarily inspired by the author’s experience of the inefficient and crowded system of rush-hour elevator traffic at Binus University. One of the root problems is the elevator stopping unnecessarily even if there are no more people waiting (the people have left or gotten on another elevator via a different button) or a full or near-full elevator stopping when there are way too many people that they need to carry and ending up not carrying anyone from that floor at all.

This project aims to ease the traffic of people via delivering a robust machine learning-based amount of people recognition via image and video to inform the elevator system whether it is sufficiently capable of stopping and carrying more people on any given floor by recognizing the amount of people via camera both in the elevator and the amount of people waiting outside the elevator. This data will then be used to determine whether or not an elevator needs to stop at any given floor despite the elevator button being pressed. It can also be used to turn off the button if the system detects that there are no more people waiting for the elevator on that particular floor to increase the efficiency of the elevator.

# 2. Methodology

## 2.1. Workflow

The first step in this research is looking for a suitable model and dataset to test the model on. After finding an unlabeled dataset, we proceed to label them manually. After that, we import the model and build a functional app that can accept image input and produce the number of people output using Streamlit. We also built a separate code for testing the model’s performance.

## 2.2. Dataset

**Table 1. Dataset Example (.csv)**

|  |  |
| --- | --- |
| Path | Number\_of\_People |
| elevator\_dataset\01.jpg | 4 |
| elevator\_dataset\02.jpg | 3 |
| elevator\_dataset\03.jpg | 3 |
| elevator\_dataset\04.jpg | 3 |
| elevator\_dataset\05.jpg | 2 |



Fig 1. Dataset Example (images)

The unlabeled image dataset is taken from Rashid Ali’s project on GitHub containing images of people on elevators or other similar settings. In total, the dataset consists of 221 images. We then proceed to label these images manually by the number of people visible on each image in a .csv file.

## 2.3. YOLOv8

The model that we have chosen to be the backbone of this system is YOLO (You Only Look Once), specifically its latest variant (version 8). YOLO is a light pre-trained CNN-based computer vision model that can identify various classes of people. We chose this algorithm because YOLO algorithms have been known to have good accuracy while maintaining real-time detection capability on lower-end devices. They managed to do this by framing the image detection task as a single regression problem solved by using a single convolutional layer to predict multiple classes and objects and generate bounding boxes. This made it possible to reduce the complexity and thereby increase detection speed. Most other algorithms use multiple steps to achieve prediction, thus increasing processing time.

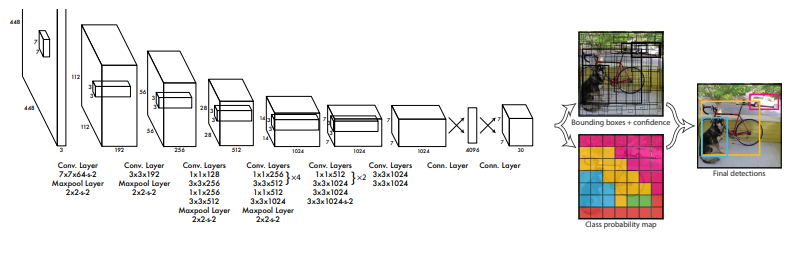


Fig 2. YOLO Architecture

It works by first doing pre-processing to the image by resizing the image into the same resolution. Then it starts to do image inference using YOLO’s convolutional neural network by splitting the image into many grids, running inference on the image, and combining the results. In this project, we limit the model to only infer on the “person” class instead of the full hundreds of classes available to optimize processing power. This will result in bounding boxes and probability for each object detected. Out of those bounding boxes, YOLO will use non-maximum suppression to eliminate all the overlapping bounding boxes with low confidence score producing a final detection with a single bounding box, class, and confidence for every object detected.

Non-maximum suppression works by sorting all bounding boxes from the highest confidence score to the lowest, then selecting the highest confidence score (S), removing it from the other overlapping bounding boxes (P), and adding it to the final prediction list. Compare the S's predictions with all the predictions in P. Calculate the Intersection over Unions (IoU) of the predictions and if the IoU is greater than the threshold then remove the predictions from P. If there are still predictions left in P we loop from the first step or else just return the final prediction list that contains the filtered predictions. This will leave only the highest confidence bounding boxes to remain for each object.

For this project, we specifically used the medium variant of YOLOv8 (yolov8m.pt) as we found that the small version has difficulties recognizing some objects, while the large model tends to detect reflections as people.

## 2.4. Streamlit

Streamlit is a python framework used to quickly make a web-based application for displaying and taking data. We used this framework to speed up development and facilitate direct input-output user experience in elevator people prediction.

# 3. Results and Discussions

## Project UI and Explanation

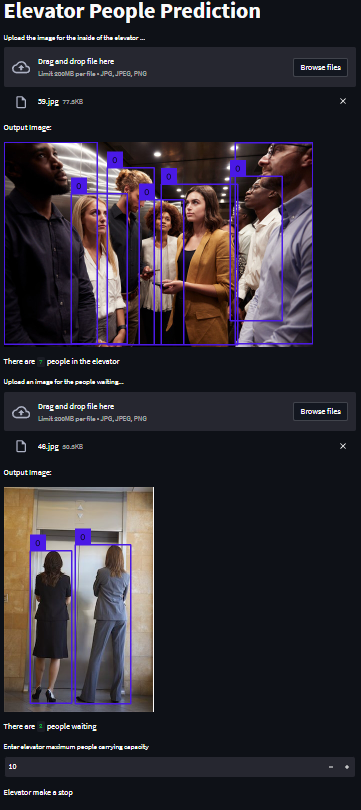
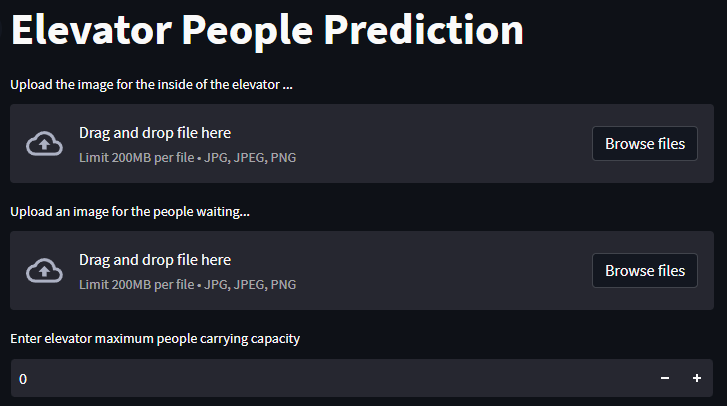


Fig 3. Project UI

This application will take two image input, one for the amount of people currently inside the elevator and one currently waiting outside the elevator. It will then detect how many people are in both images and print out the result. Then, we decide and input the maximum carrying capacity of the elevator. If the amount of people in the elevator is less than the maximum amount of people that the elevator can carry, then the elevator will stop and pick up the people waiting. If not, then the elevator will ignore the people waiting.

## Metrics

**Table 2. Result Regression Metrics**

|  |  |  |  |
| --- | --- | --- | --- |
| MAE | MSE | RMSE | R2-Score |
| 0.471 | 1.602 | 1.266 | 84.85% |

The experiment is carried out using python with YOLOv8m object detection model. We compared the amount of people detected from the YOLO model to the manually labeled dataset to get the resulting metrics in Table 2. We decided first to evaluate this model via regression metrics. Looking at the MAE, the model seems to be able to predict the amount of people with good accuracy with it only being wrong by 0.471 people on average. The 84.85% R2-score confirms that this model does indeed perform well in its task. However, the MSE and RMSE are a lot higher than its MAE suggesting that the model has problems detecting several outlier images.

The model seems to struggle the most when it comes across reflections as many elevators have mirror-like surfaces. This causes the model to sometimes misidentify a reflection as an actual person, thereby increasing the number count of people. The model also struggles in detecting huge crowds of people where some people might be obscured by other people in the image causing it to fail in identifying them. The third most common confusion in the model is when a person is only partially in the image, the model will sometimes identify these objects as a person, but sometimes also don’t classify it as an object.

**Table 3. Result Classification Metrics**

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Weighted Precision | Weighted Recall | Weighted F1-Score |
| 73.30% | 76.67% | 73.30% | 73.86% |

A chart of numbers and a number

Description automatically generated with medium confidence

Fig 4. Confusion Matrix

After analyzing the regression metrics of this model, we got curious and wanted to confirm our hypothesis about high outliers and investigate further using classification metrics. The resulting metric in Table 3 seems decent with it being able to achieve 73-76% performance in all metrics. From the confusion matrix in Fig 4 you can also see that most of the pictures contain less than 8 people. So, this test dataset might be unable to fully capture the accuracy of predicting large crowds of people. However, we felt that this is representative enough considering the fact that most elevators can only carry a maximum of 10 people.

**Table 4. Result Classification Metrics (Tolerance = 1)**

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Weighted Precision | Weighted Recall | Weighted F1-Score |
| 90.95% | 92.97% | 90.95% | 91.16% |

Considering the fact that a lot of the inaccurate results are mostly off by a small amount. We decided to check the model’s performance using an incorrect tolerance of 1. In this instance, if the prediction is only off by 1 compared to the manual label it will still be counted as accurate. We felt that this represents the model’s capability somewhat better than the baseline result because a difference of 1 is negligible in real-world applications. Using this tolerance, we find that the model actually performs a lot better than it seems scoring above 90% in all metrics. This also confirms our hypothesis that this model actually performs really well in most cases, however, struggles with some edge cases or outlier images.

## GitHub Link

<https://github.com/xxKeyaxx/Elevator-amount-of-People-Prediction>

# 4. Conclusion

The efficiency of elevators is important, especially in large buildings as it can lead to energy, time, and cost savings for its users and operators. This is especially true because most elevators already have CCTV cameras, so the cost of implementing this system is minimal in most situations. This project has managed to prove that YOLOv8 can be implemented as a reliable amount of people detector in elevators to maximize the system efficiency through better decision-making. The project also produced a demo application for showcasing how such a system could be implemented by detecting the number of people both in the elevator and the people waiting for the elevator to make better decisions in controlling the elevator.

In the future, we would like to see a full implementation of this program in real-world escalators to see how effective it is in practice.

# References

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Jiang, P., Ergu, D., Liu, F., Cai, Y. and Ma, B., 2022. A Review of Yolo algorithm developments. *Procedia Computer Science*, 199, pp.1066-1073.

Patel, Jagrat. 2023. Top 10 Object Detection Models in 2023!. Accessed from [www.linkedin.com/pulse/top-10-object-detection-models-2023-jagrat-patel](http://www.linkedin.com/pulse/top-10-object-detection-models-2023-jagrat-patel).

Ali, Rashid. 2019. Elevator-Passenger-Counting-using-Yolov3. Accessed from [github.com/rashidch/Elevator-Passenger-Counting-using-Yolov3](https://github.com/rashidch/Elevator-Passenger-Counting-using-Yolov3).

Streamlit Team. 2023. Streamlit • A faster way to build and share data apps. Accessed from [streamlit.io](https://streamlit.io/).

Ultralytics Team. 2023. Ultralytics YOLOv8 Docs. Accessed from [docs.ultralytics.com/models/yolov8](https://docs.ultralytics.com/models/yolov8/).

David, Lv. 2017. Passenger Elevator – The Complete FAQ Guide. Accessed from [dazenelevator.com/passenger-elevator-complete-faq-guide](https://dazenelevator.com/passenger-elevator-complete-faq-guide/).

Prakash, J., & Prakash, J. (2021, October 18). *Non maximum Suppression: Theory and Implementation in PyTorch*. LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Examples and Tutorials. <https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/>.

Zivkovic, N.M. 2021. Machine Learning with ML.NET – Object detection with YOLO. Accessed from <https://rubikscode.net/2021/04/05/machine-learning-with-ml-net-object-detection-with-yolo/>.