ASM Sc. J., 21(2), 2026

https://doi.org/10.32802/asmscj.2026.1025

Implementation of YOLOv8 Algorithm for Car Seat Detection in Automotive Industry with GUI

F.D.H. Fazlin Hakimie, S.Z. Mohd Muji*, M.S. Abdul Kadar and M.H. Hussin Chee

Department of Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor

In automobile production, precise car seat position detection is critical to prevent mistaking labels and to improve the production efficiency. In this paper, an improved car seat detection system using YOLOv8 object detection algorithm combined with a real time Graphical User Interface (GUI) and SQLite database is proposed. Extending from earlier models based on YOLOv5, the models perform better accuracy and add bar code validation automatically to car seat type A, B, and C. The dataset contains 1,406 labelled images, which is trained by Roboflow and later on Google Colab. Built on CustomTkinter, the GUI supports real-time detection and database retrieval. After evaluation, the system achieved an average accuracy of 87%, 84%, and 89% in seat types A, B, and C, respectively. The results reveal the superiority of the proposed system on the detection accuracy, reduces human labelling errors, and accelerates verification process on automotive production lines.

Keywords: YOLOv8; GUI; SQLite Database; car seat; automative industry

I. INTRODUCTION

In recent years, the Malaysian automotive industry has faced both notable challenges and significant growth. Due in large part to the end of the PENJANA Sales Tax Exemption scheme and shorter working months during the Hari Raya Aidilfitri holidays, new car sales in April 2023 fell 19.1% to 46,583 units when compared to the same month in 2022. In the same time frame, automobile production fell by 24.8% to 41,160 units. Nonetheless, the first few months of 2023 were encouraging, with automobile production and sales rising by 39.0% and 24.4% in February and 7.6% and 27.0% in March, respectively, over the prior year (Sin *et al.,* 2024). The growing demand for auto parts has driven a sharp increase in production, but with that comes a greater risk of human error. In manufacturing, repetitive tasks often lead to mistakes that can compromise product quality and disrupt delivery schedules (Real & Torres, 2024). One common issue across many industries is labelling errors, which can happen at any point, from early assembly to final packaging. In the auto parts sector, these mistakes can be especially costly, causing major disruptions to the supply chain (Sekhar *et al.,* 2023). For example, mislabelled barcodes might result in the wrong parts being sent to assembly lines, leading to delays, wasted resources, and higher operational expenses. On top of that, incorrectly labelled or defective components might not meet regulatory standards, requiring extra inspections and quality checks (Ferreira *et al.,* 2022). These problems not only slow down production but also reduce efficiency and can harm customer satisfaction.

To tackle these challenges, many industries are turning to automated vision systems for more efficient defect detection and quality control (Morales *et al.,* 2025). While traditional methods like barcode scanning are still used to identify parts, such as car seats. Computer vision systems offer a more advanced way to spot defects, especially in smaller components. For instance, machine vision setups using Charge-Coupled Device (CCD) cameras combined with Support Vector Machine (SVM) algorithms have achieved up to 97.1% accuracy in identifying surface flaws like scratches and dents (Abagiu *et al.,* 2023).

*Corresponding author's e-mail: szarina@uthm.edu.my

Convolutional Neural Networks (CNNs), when integrated with robotic controls, are also helping improve traceability and reduce the time needed for inspecting engine blocks. More recently, advanced models like YOLO-DD have pushed the boundaries of defect detection by adding transformer modules and attention mechanisms, reaching 82.4% accuracy while maintaining real-time performance (Wang *et al.,* 2024).

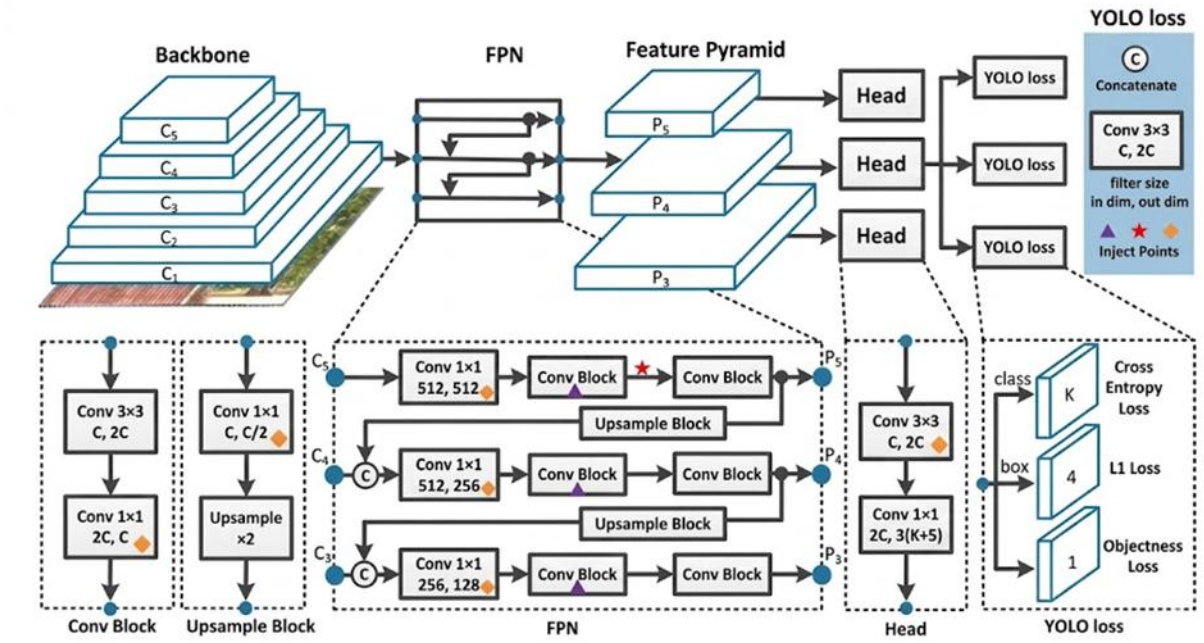


Figure 1. YOLOv8 architecture (Ali, 2023)

The YOLO (You Only Look Once) series has been a gamechanger in manufacturing, enabling real-time detection of defects and anomalies right on the production line. The latest version, YOLOv8, introduces a redesigned architecture that includes Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) modules (Zhang *et al.,* 2024). FPN helps the model detect defects at different scales by creating multi-resolution feature maps, while PAN enhances accuracy by combining features from multiple layers of the network. Figure 1 illustrates the YOLOv8 architecture. This combination delivers better performance than YOLOv5, which is based on a modified CSPDarknet architecture. With these improvements, YOLOv8 provides more accurate and efficient defect detection, making it a valuable tool for boosting quality and productivity in auto parts manufacturing (Ding *et al.,* 2024).

II. METHODS

This project focuses on developing a car seat class detection system using YOLOv8 to help reduce human errors that can occur after barcode tagging. The system's overall architecture is shown in Figure 2 and is built around three main components: a database, a camera, and a graphical user interface (GUI).

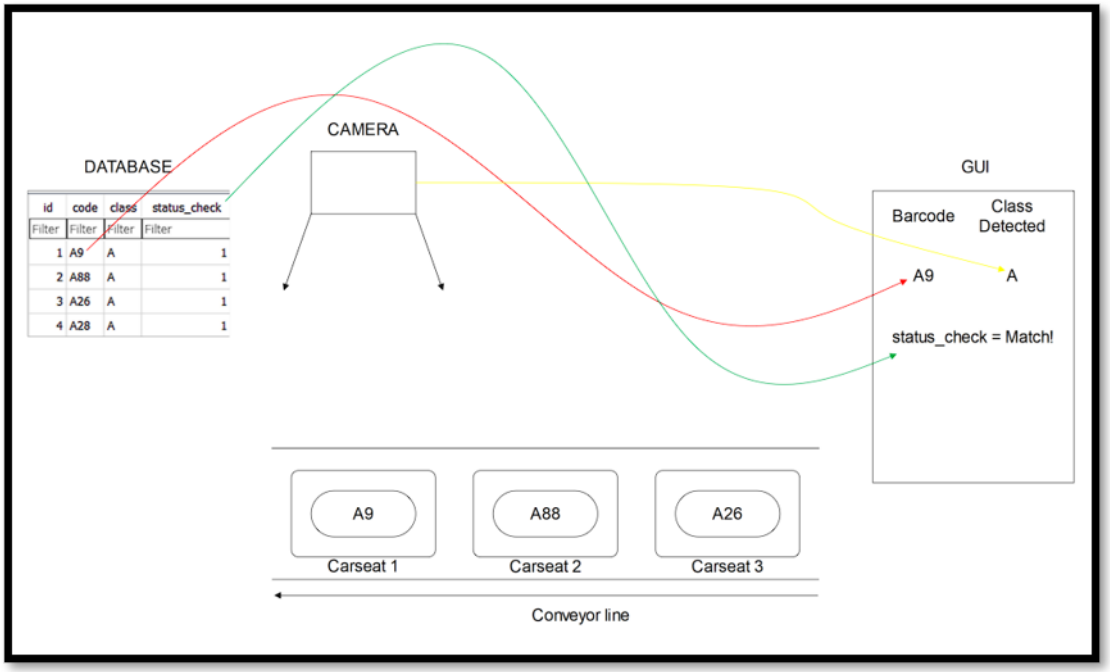


Figure 2. Block diagram of car seat detection process using the YOLOv8 algorithm

The database keeps a record of all scanned barcodes along with their unique identifiers. For class detection, the system uses YOLOv8, a cutting-edge computer vision model developed by Ultralytics (Khan *et al.,* 2024). Once a class is detected, the system uses that information to look up related details in the database. To ensure accuracy, a built-in function called "status_check" verifies whether the information retrieved matches the detected object, helping maintain consistency and reliability throughout the process. Figure 3 illustrates the flowchart of the car seat class detection process. It all starts with preparing the dataset, which includes annotating, augmenting, and preprocessing the images using Roboflow. Once this step is complete, Roboflow automatically organises the data into separate folders for training, validation, and testing.

These datasets are then uploaded to Google Colab, where the YOLOv8 algorithm is used to train the model and generate the necessary weight files. The system also includes a graphical user interface (GUI) that connects to the database, enabling real-time monitoring and analysis.

When the system detects a car seat, for example, type A, it checks the barcode against the database entries. If a match is found, the GUI displays a “Match” status. The same process applies for all other seat types, helping ensures the correct classification and tracking of each item.

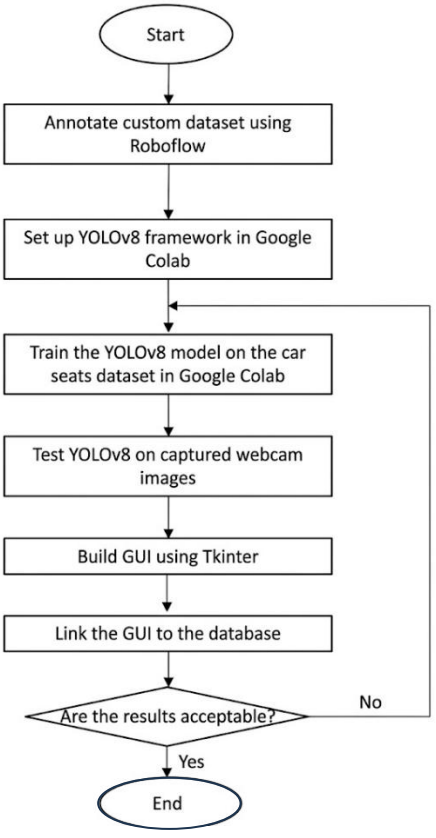


Figure 3. Flowchart of the overall system development using the YOLOv8 algorithm

A. Training Process using Google Colab

The dataset is carefully organised to make the annotation process smoother and to avoid any confusion between car seat classes. Each image is given a specific name based on its seat type to keep everything clearly labelled. Roboflow is used throughout the process; for annotation, class labelling, data splitting, preprocessing, and data augmentation. The dataset includes a total of 1,406 images representing three types of car seats: type A with 496 images (shown in Figure 4(a)), type B with 646 images (Figure 4(b)), and type C with 264 images (Figure 4(c)). All images are resized to 640 × 640 pixels for consistency during training. To ensure the model performs accurately, the annotation process must be done carefully, with each bounding box tightly drawn around the object. Once this step is complete, Roboflow automatically generates the necessary annotation files, including the bounding box coordinates, and compiles them into a data.yaml configuration file which ready for training.

The training process involves several key parameters, including image size, batch size, number of epochs, the data.yaml file, and the initial model weights. All training is carried out using Google Colab for convenience and speed. At the end of the training, a weight file named "best.pt" is generated. This file holds the optimised set of weights that the model has learned, and it plays a critical role in object detection. The better these weights, the more accurate the model's predictions will be on new data. In this project, the "best.pt" file is used within the graphical user interface (GUI) to detect car seat classes in real time. By using the most accurate version of the trained model, we ensure that the system delivers high performance and reliability during actual deployment. Table 1 outlines the training parameters and arguments used during the model training process in Google Colab.



(a) (b) (c)

Figure 4. Data preparation for (a) car seats type A, (b) car seat type B and (c) car seat type C (c)

Table 1. Argument with parameter that is being used during the training process in Google Colab

Argument	Description	Parameters
--imgsz	Input image size	640
--batch size	The number of batch images for the validation process	16
--epochs	Times of learning algorithms to process the entire dataset	10
--data	Data files consist of annotation values from datasets	data.yaml
--weights	Initial weight	yolov8n.pt

B. Database using SQLite

SQLite is a lightweight, file-based relational database that makes an excellent choice for storing data in many applications. One of its biggest strengths is that it's simple to use and it doesn't require a separate server and can be embedded directly into software. It's also cross-platform, portable, and easy to set up, making it ideal for quick and efficient deployment. When using SQLite to store datasets, it's important to think about how the database is structured. Good schema design, proper indexing, and storage optimisation all play a role in maintaining performance. Using efficient techniques like batch insertion and prepared statements can also help speed up data entry and processing. Figure 5 shows how the DB Browser for SQLite

tool is used to create and manage the car seat database in this project.

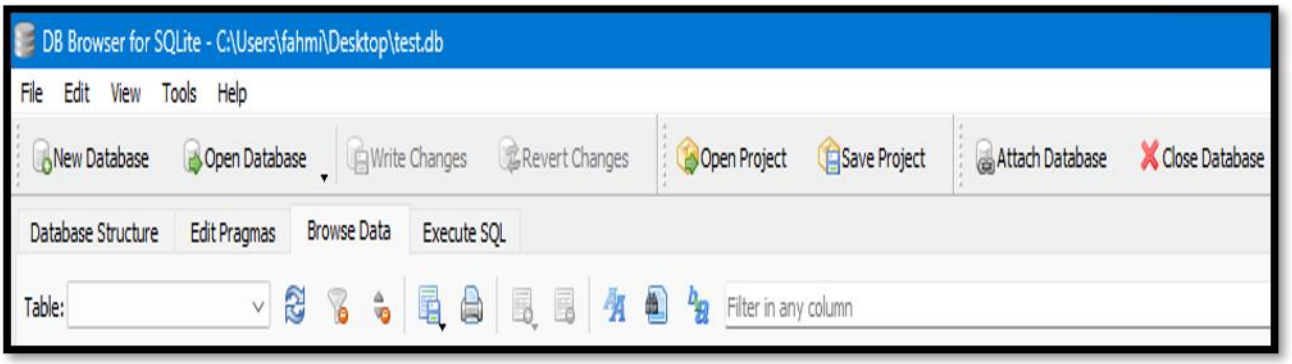


Figure 5. DB Browser for SQLite main interface

C. Data Splitting

Roboflow is a powerful platform that helps with annotating, preprocessing, and deploying computer vision models. It also supports cross-validation as part of the workflow (Isaiev & Kysil, 2024). Splitting the dataset properly is crucial to avoid overfitting, where the model performs well on training data but struggles with new, unseen data. Cross-validation helps with this by dividing the dataset into multiple parts, allowing the model to be trained and tested on different portions to get a better sense of its overall performance.

Roboflow simplifies this entire process. By automatically managing the training and testing of the model on various data subsets, it helps ensure that the model can generalise well to real-world scenarios and not just the data it has already seen. This increases confidence in the model's reliability and performance outside the training environment.

In this project, the dataset is divided into three parts: 70% for training, 20% for validation, and 10% for testing, as shown in Figure 6. The training set is used to teach the model, the validation set helps monitor how well it's learning during training, and the testing set is used to measure its final performance. This balanced split gives the model plenty of data to learn from while still leaving enough for accurate evaluation both during and after training.

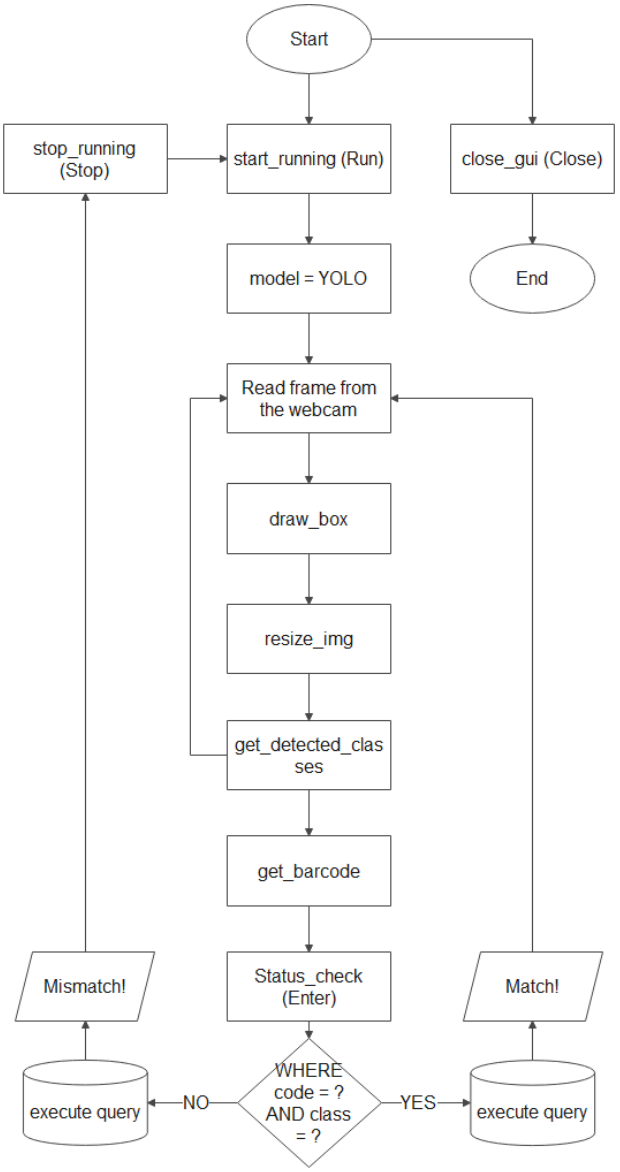


Figure 6. 1406 data sets are split into training, validation, and testing sets with percentages of 70%, 20%, and 10%, respectively

D. Software Development

This section focuses on the design and development of a graphical user interface (GUI) for real-time object detection using Tkinter and YOLOv8. The goal is to create an easy-to-use platform that works seamlessly with the powerful YOLOv8 algorithm. This report walks through the entire GUI development process, from the layout and key features to how it integrates with the detection model. A key priority is making the interface simple and clear, so users can easily interact with the system and understand the results being displayed.

To enhance the look and functionality of the interface, this project uses the CustomTkinter module, which is available through its GitHub repository (Schimansky, 2026). This module builds on the basic Tkinter library by offering more customisation options, better-looking widgets, and additional interactive elements. With CustomTkinter, the GUI gains a more modern and polished appearance, helping to create a more engaging and user-friendly experience.

Table 2 lists the widgets used in the front-end design. By combining standard Tkinter elements with the enhanced features of CustomTkinter, the interface delivers a smooth, interactive, and intuitive experience for anyone using the object detection system.

Table 2. Main Tkinter widgets implemented in the application

Argument	**Description**
TkFrame	Creates container frames within the GUI to organise and group other widgets
TkLabel	Displays text or informative messages with customisable font styles, sizes, colours, and alignment
TkButton	Interactive buttons that trigger specific actions or tasks within the object detection application

III. EXPERIMENTAL SETUP

This section can be organised in several ways, such as by research objectives, chronological sequence, grouped case studies, experimental setups, or any other logical structure that best supports clarity and understanding.

A. GUI System

The GUI system has been successfully developed and is capable of displaying either a Match or Mismatch status by comparing the barcode ID with the ID detected from the image. Each car seat type has a unique barcode prefix: car seat A barcodes begin with a capital "A", car seat B with "B", and car seat C with "C". The process begins when the user enters a barcode simulating the real scenario where a worker scans the barcode on the car seat. Simultaneously, an image of the car seat is captured using a camera. If the captured image is detected as car seat type A, and the entered barcode also starts with "A", a Match status will be displayed, as shown in Figure 7. In contrast, a Mismatch status is shown when the barcode and detected seat type do not align. For example, as illustrated in Figure 8, the barcode entered is "B110", but the image captured by the camera is detected as car seat type A resulting in a Mismatch status.

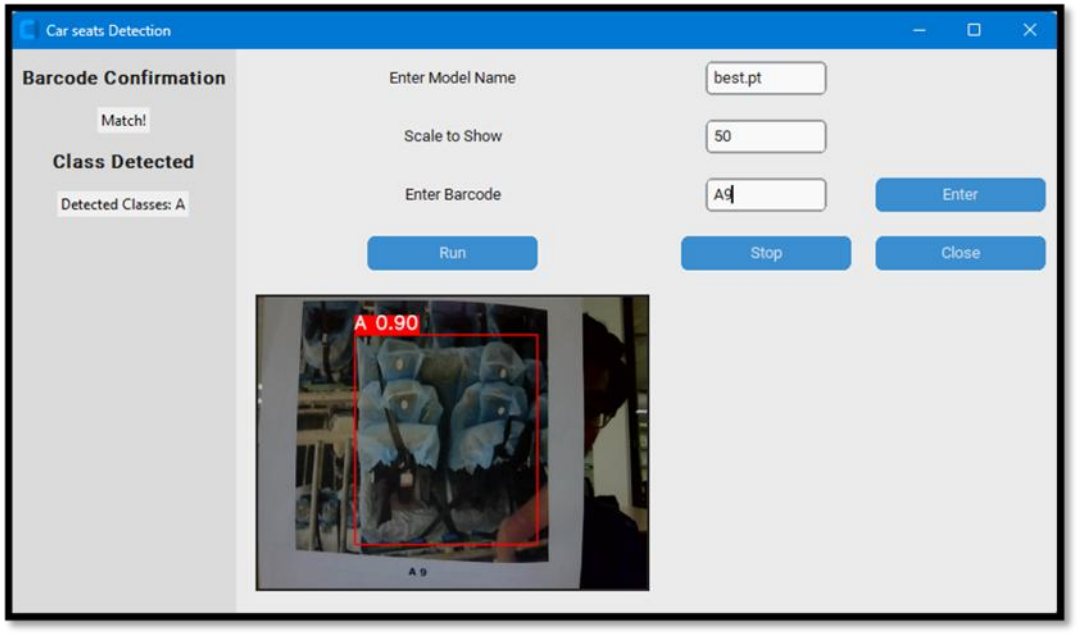


Figure 7. Match condition at GUI

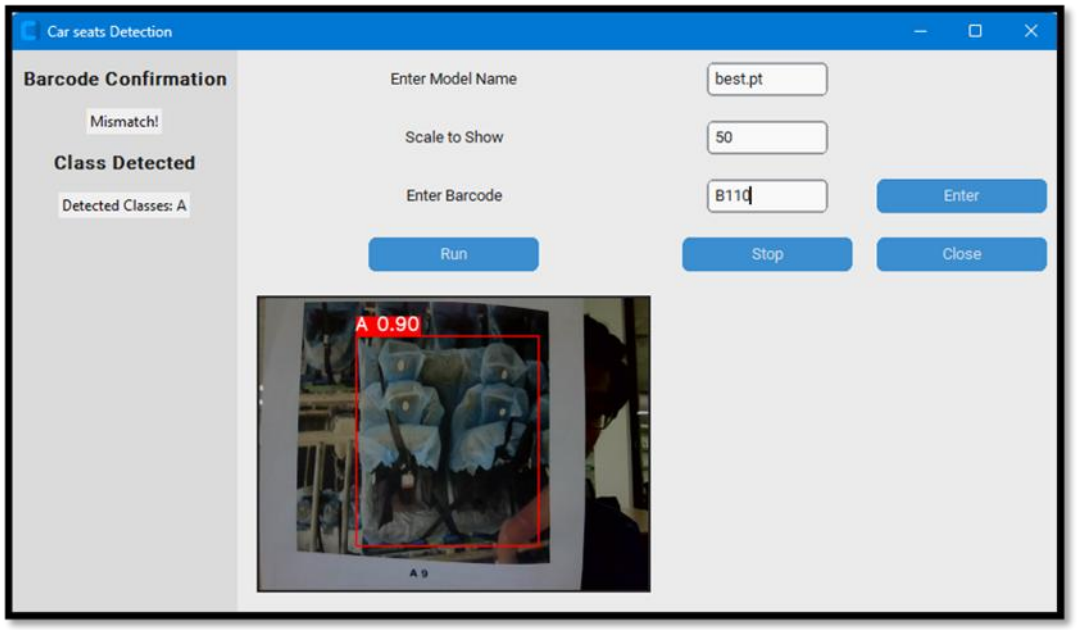


Figure 8. Mismatch condition at GUI

B. Evaluation Metrics for YOLOv8

In this project, the dataset consists of 1,406 images of different car seat types: 496 images for type A, 646 for type B, and 264 for type C. Due to the larger number of type B images, the model may learn to localise type B more effectively. However, if type C images are too few, the model may struggle to learn its features properly, potentially leading to overfitting or poor generalisation.

Dual Focal Loss mitigates this issue by ensuring the model learns from both frequent and infrequent classes more effectively. Object detection models are often evaluated using a combination of metrics that provide a more comprehensive view of their performance. Common metrics include:

1) Precision: Precision measures the fraction of true positive detections out of all positive detections. It focuses on how accurate the model's positive predictions are.
2) Recall: Recall (or true positive rate) measures the fraction of true positive detections out of all actual positive instances. It assesses the model's ability to find all relevant objects.
3) F1 Score: The F1 score is the harmonic mean of precision and recall and provides a balanced measure that considers both false positives and false negatives.
4) Intersection over Union (IoU): IoU measures the overlap between predicted bounding boxes and ground truth bounding boxes. It's crucial for assessing the accuracy of bounding box localisation.
5) Average Precision (AP): AP is a metric that considers precision-recall trade-offs at various confidence score thresholds. It quantifies how well the model performs across different confidence levels.

IV. RESULT AND DISCUSSION

Figure 9 shows the graph of training loss and validation loss for bounding boxes, measured using IoU (Intersection over Union). Figure 10 illustrates that the training and validation losses for object detection decrease and stabilise around the 10th epoch, indicating a good fit without signs of overfitting. Figure 11 presents the graph of training and validation loss using Dual Focal Loss, which addresses class imbalance, a situation where some classes appear significantly more frequently than others. Dual Focal Loss is effective in improving object detection performance under such imbalanced conditions. It helps the neural network focus more on underrepresented classes during training. Class imbalance occurs when one class occurs too frequently and another occurs less.

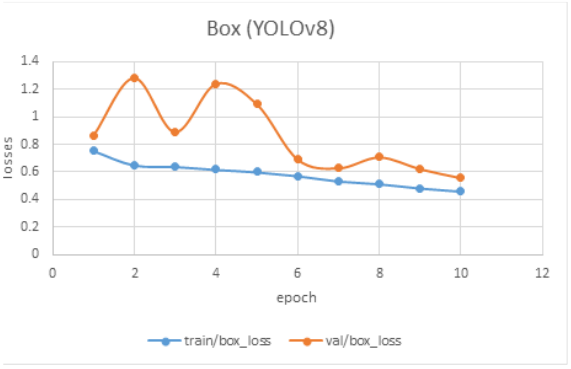


Figure 9. Training loss and validation loss of Box (YOLOv8)

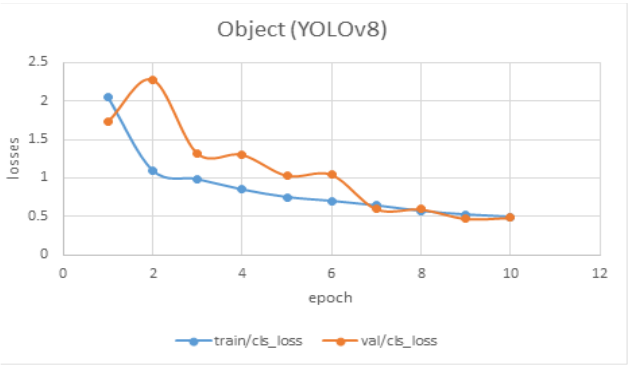


Figure 10. Training loss and validation loss of Object (YOLOv8)

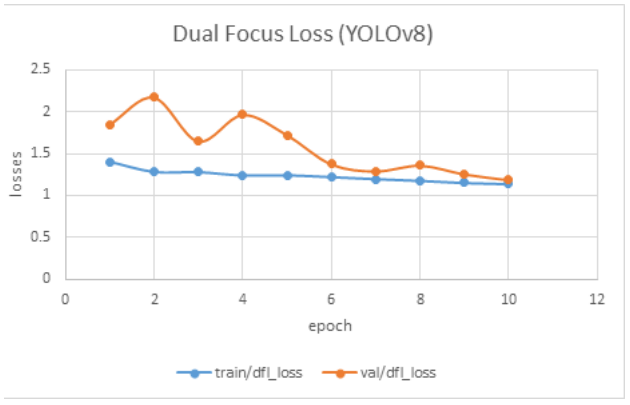


Figure 11. Training loss and validation loss of Dual Focus Loss (YOLOv8)

According to Table 3, the average accuracy for class A is 0.87, indicating that most class A images are detected accurately and with high confidence. Table 4 shows an

average accuracy of 0.84 for class B, while Table 5 reports the highest average accuracy of 0.89 for class C. These results can be attributed to the advancements in the architecture and training techniques employed in YOLOv8, which enable more precise object detection. The result shows an improvement compared to previous research, as presented in Table 6. Although previous studies (Wang *et al.*, 2024; Zhang *et al.*, 2024) used larger datasets than ours, our model achieved better overall performance. This suggests that the proposed approach is more effective and represents a notable improvement over existing methods.

Table 3. Accuracy for class A

Item	**Accuracy for Class A**									
Barcode	A9	A88	A26	A28	A33	A36	A65	A75	A217	A213
Class	A	A	A	A	A	A	A	A	A	A
Detected Barcode	Match!	Match!	Match!	Match!	Match!	Match!	Match!	Match!	Match!	Match!
Validation Accuracy	0.92	0.89	0.82	0.85	0.80	0.90	0.94	0.93	0.90	0.82

Table 4. Accuracy for class B

Item	**Accuracy for Class B**									
Barcode	B110	B120	B299	B4	B5	B6	B13	B145	B183	B313
Class	B	B	B	B	B	B	B	B	B	B
Detected Barcode	Match!	Match!	Match!	Match!	Match!	Match!	Match!	Match!	Match!	Match!
Validation Accuracy	0.88	0.88	0.89	0.72	0.78	0.68	0.91	0.92	0.90	0.92

Table 5. Accuracy for class C

Item	**Accuracy for Class C**									
Barcode	C233	C92	C241	C19	C47	C79	C118	C119	C186	C238
Class	C	C	C	C	C	C	C	C	C	C
Detected Barcode	Match!	Match!	Match!	Match!	Match!	Match!	Match!	Match!	Match!	Match!
Validation Accuracy	0.92	0.92	0.90	0.89	0.88	0.84	0.90	0.91	0.92	0.90

Table 6. Comparison of this work and previous research

Ref	**Type A**	**Type B**	**Type C**
(Muji *et al.*,2024)	30%	53%	38%
(Sum *et al.*, 2024)	79.3%	83.3%	81.1%
This work	87%	84%	89%

V. CONCLUSION

In conclusion, the development of the car seat detection application has been successfully outlined, covering key aspects such as GUI design, model integration, database interaction, and barcode recognition. The application effectively utilises the YOLO model for real-time object detection and class recognition. The GUI offers an intuitive interface for user interaction, displaying the detected seat types along with the corresponding barcode information. The integration of the SQLite database enables efficient querying and matching between barcode data and detected classes. However, to further improve the application, it is recommended to incorporate barcode scanning functionality for automatic data capture, thereby enhancing both accuracy and operational efficiency. By implementing these enhancements, the application will provide a more seamless and user-friendly experience for accurate barcode recognition and class comparison, delivering valuable feedback to users. Overall, the car seat detection application demonstrates a functional and effective solution for real-time barcode and class detection in automotive scenarios.

VI. ACKNOWLEDGEMENT

This research was supported by Universiti Tun Hussein Onn Malaysia (UTHM) through Tier 1 (Vot Q354).

VII. REFERENCES

Abagiu, MM, Cojocaru, D, Manta, F & Mariniuc, A 2023, 'Detecting machining defects inside engine piston chamber with computer vision and machine learning', *Sensors*, vol. 23, no. 2, doi:10.3390/s23020785.

Ali, SZ 2023, 'Principles of YOLOv8', *Medium*, viewed 29 May 2026, <https://medium.com/@syedzahidali969/principles-of-yolov8-6a90564e16c3>.

Ding, P, Zhan, H, Yu, J & Wang, R 2024, 'A bearing surface defect detection method based on multi-attention mechanism YOLOv8', *Measurement Science and Technology*, vol. 35, no. 8, p. 086003, doi:10.1088/1361-6501/ad4386.

Ferreira, R, Barroso, J & Filipe, V 2022, 'Conformity assessment of informative labels in car engine compartment with deep learning models', *Journal of Physics: Conference Series*, vol. 2278, no. 1, p. 012033, doi:10.1088/1742-6596/2278/1/012033.

Isaiev, T & Kysil, T 2024, 'Method of creating custom dataset to train convolutional', *Computer Systems and Information Technologies,* no. 4, pp. 37–44, doi:10.31891/csit-2024-4-5.

Khan, S, Vohra, S, Siddique, SA, Abro, AA & Ebrahim, M 2024, 'A computer vision-based vehicle detection system leveraging deep learning', *In: 2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC)*, pp. 1–7, doi:10.1109/KHI-HTC60760.2024.10482163.

Morales, MO, Takeo Nava, JG, Moreno Escobar, JJ & Ceballos Chávez, BA 2025, 'Artificial intelligence for quality defects in the automotive industry: a systemic review', *Sensors*, vol. 25, no. 5, doi:10.3390/s25051288.

Muji, SZM, Tat, TS, Hakimie, FDH, Fazlin, M, Abd Kadir, M, Mohd Norzali, Suhaila, S, Chua King, KL & Bakht, MP 2024, 'Image recognition using YOLOv5 for automotive industry', *International Journal of Integrated Engineering*, vol. 16, no. 3, pp. 212–222. doi:10.30880/ijie.2024.16.03.019.

Real, C & Torres, Y 2024, 'Effect of changes in the sequence of assembly operations on error rates: a case study from the car manufacturing industry', *IEEE Access*, vol. 12, pp. 34644–34655, doi:10.1109/ACCESS.2024.3371980.

Schimansky, T 2026, *CustomTkinter*, GitHub, viewed 29 May 2026, <https://github.com/TomSchimansky/CustomTkinter>.

Sekhar, R, Solke, N & Shah, P 2023, 'Lean manufacturing soft sensors for automotive industries', *Applied System Innovation*, vol. 6, no. 1, doi:10.3390/asi6010022.

Sin, MS, Anbarasen, S, Yew, TS & Yuan, WZ 2024, 'Effect of economic factors on the automotive industry in Malaysia', *Journal of Sustainability Science and Management*, vol. 19, no. 2, pp. 109–125, doi:10.46754/jssm.2024.02.006.

Sum, LC, Muji, SZM & Sari, S 2024, 'Object recognition using YOLOv8 for car seat identification', *In 2024 International Conference on Future Technologies for Smart Society (ICFTSS),* Kuala Lumpur, Malaysia, 7–8 August, pp. 17–22. doi:10.1109/ICFTSS61109.2024.10690270.

Wang, J, Wang, W, Zhang, Z, Lin, X, Zhao, J, Chen, M, Luo, L 2024, 'YOLO-DD: improved YOLOv5 for defect detection', *Computers, Materials & Continua*, vol. 78, no. 1, pp. 759–780, doi:10.32604/cmc.2023.041600.

Zhang, T, Wang, F, Wang, W, Zhao, Q, Ning, W & Wu, H 2024, 'Research on fire smoke detection algorithm based on improved YOLOv8', *IEEE Access*, vol. 12, pp. 117354–117362, doi:10.1109/ACCESS.2024.3448608.