

Class-Specific Dataset Splitting for YOLOv8: Improving Real-Time Performance in NVIDIA Jetson Nano for Faster Autonomous Forklifts

Chaouki Tadjine^{1,2}^a, Abdelkrim Ouafi²^b, Abdelmalik Taleb-Ahmed¹^c and Yassin El Hillali¹^d

¹Univ. Polytechnique Hauts-de-France, CNRS, Univ. Lille, UMR 8520 - IEMN - Institut d'Electronique de Microelectronique et de Nanotechnologie, Valenciennes, F-59313, Hauts-de-France, France

²Univ. Mohamed Khider Biskra, Department of Electrical Engineering, Faculty of Sciences and Technology, LVSC - Lab. Vision et Systèmes de Communication, 07000, Biskra, Algeria
{chaouki.tadjine, abdelmalik.taleb-ahmed, yassin.elhillali}@uphf.fr, {chaouki.tadjine, a.ouafi}@univ-biskra.dz

Keywords: YOLOv8, LOCO Dataset, Object Detection, Autonomous Forklifts, Real-Time Inference, NVIDIA Jetson Nano.

Abstract: This research examines a class-specific YOLOv8 model setup for real-time object detection using the Logistics Objects in Context dataset, specifically looking at how it can be used in high-speed autonomous forklifts to enhance obstacle detection. The dataset contains five common object classes in logistics warehouses. It is divided into transporting tools (forklift and pallet truck) and goods-carrying tools (pallet, small load carrier, and stillage) to meet specific task needs. Two YOLOv8 models were individually trained and implemented on the NVIDIA Jetson Nano, with each one specifically optimized for a tool category. Using this approach tailored to specific classes resulted in a 30.6 percent decrease in inference time compared to training a single YOLOv8 model on all classes. Task-specific detection saw a 74.4 percent improvement in inference time for transporting tools and 56.2 percent improvement for goods-carrying tools. Furthermore, the technique decreased the hypothetical distance traveled during inference from 45.14 cm to 31.32 cm and even as low as 11.55 cm for transporting tools detecting while still preserving detection accuracy with a minor drop of 1.25% in mean average precision. The integration of these models onto the NVIDIA Jetson Nano made this approach compatible for future autonomous forklifts and showcases the potential of the technique to improve industrial automation. This study demonstrates a useful and effective method for real-time object detection in intricate warehouse settings by matching detection tasks with practical needs.

1 INTRODUCTION

The technology of object detection is crucial for automating logistics warehouses, allowing self-driving forklifts to move through intricate surroundings, monitor products, and dodge obstacles instantly (Zaccaria et al., 2020). Quick and precise object detection is crucial for keeping operations efficient, avoiding accidents, and ensuring smooth processes, particularly when forklifts are moving at increased velocities. The Logistic Objects in Context (LOCO) dataset, created for logistics object detection tasks, consists of five equipments in warehouse settings: forklift, pallet truck, pallet, small load carrier (SLC), and stillage (Mayershofer et al., 2020). The dataset has unbal-

anced annotations, whereas the preliminary annotations are demonstrated in Figure 1.

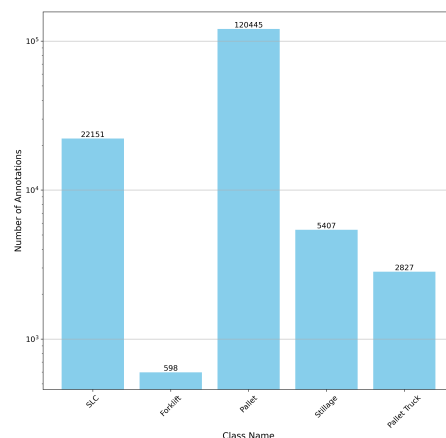






Figure 1: LOCO dataset annotations.

^a <https://orcid.org/0009-0000-6110-9956>

^b <https://orcid.org/0000-0002-6083-1688>

^c <https://orcid.org/0000-0001-7218-3799>

^d <https://orcid.org/0000-0002-3980-9902>

In the state-of-the-art works related to the LOCO dataset (Savas and Hinckeldeyn, 2022; Khalfallah et al., 2024; Clavero et al., 2024), they train an object detection model to detect all five object classes at once, which is not the optimal and efficient choice to use in for every specific warehouse task. For instance, it is crucial to identify equipment such as pallets, small load carriers, and stillages used for transporting goods in picking operations that prioritize identifying items for shipping. Identifying moving equipment like forklifts and pallet trucks is essential for safely navigating and avoiding obstacles in warehouse activities.

In order to meet these specific requirements, we suggest categorizing the objects based on their usage: either for transporting (forklift and pallet truck) or carrying goods (pallet, small load carrier, and stillage). This categorization helps in focusing on object detection by aligning it with task requirements, which may reduce computational burden.

Our approach utilizes this categorization by creating individual YOLOv8 models for each category, enabling detection specific to the task. The effectiveness of this approach was assessed in improving inference efficiency and preserving detection accuracy. By utilizing the trained models on the NVIDIA Jetson Nano, we assessed how well they can achieve quicker inference times while maintaining the necessary practicality for autonomous forklift operations within dynamic warehouse settings.

Lightweight models like YOLOv8n are ideal for edge devices like the NVIDIA Jetson Nano to attain real-time performance due to their blend of speed and accuracy (Asdikian et al., 2024). In this research, we taught YOLOv8n models using the class-specific partition of the LOCO dataset and assessed their results in relation to real-time object tracking and obstacle evasion. The focus on the Jetson Nano was on achieving fast inference time and high accuracy, essential for high-speed forklifts. Quicker inference times are important because they enable vehicles to react faster to obstacles, enhancing safety and efficiency in logistic operations. By looking into the prototyping works for forklift automation (Mohamed et al., 2018; Behrje et al., 2018; Cidal et al., 2019; Zaccaria et al., 2021), the only forklift that has publicly available top speed is Jungheinrich EVT 216, whereas this forklift has maximum speeds of 11 km/h, which is around 3.055 m/s. Therefore, we considered it as reference speed in our evaluations.

2 METHODOLOGY

The proposed method in our approach is to separate the dataset objects leading to train two fine-tuned YOLOv8 models, whereas they can be used depending on the requested task, however they can be combined to work same as single model trained for all the dataset at once. The figure 2 summarizes the functionality of the proposed method.

2.1 Dataset Preparation

The LOCO dataset has five classes with unbalanced annotations, it was split into 60% for training, 25% for validation, and 15% for testing. Due the unbalanced nature of the dataset, the splitting was an annotation-based split, ensuring that each class was proportionally represented in the training, validation, and testing sets. This approach preserves the class distribution across all phases of model development.

From this annotated split, we divided the dataset by isolating the annotations of forklifts and pallet trucks (transporting tools) to train it in one model. The remaining three classes (goods carrier tools) were used to train a second model. This approach ensured that we maintained the same annotations for each class across all models, which was crucial for conducting a fair comparison between the combined and split model approaches. Table 1 represents the data fed into the models.

2.2 Model Settings

We chose YOLOv8n for our tests because of its compact design, which makes it ideal for running on the limited resources of the Jetson Nano. YOLOv8n normally uses a 640x640 input image resolution, finding a middle ground between detection precision and computational speed.

Initially, we trained YOLOv8n on the complete dataset with a standard resolution of 640x640 for all five classes in our evaluation. We fine-tuned the class-specific split models by adjusting the resolution according to the class distribution. The model trained on the two-class subset (tools for transporting) utilized a smaller image size of 256x256, while the model trained on the three-class subset (tools for carrying goods) used a larger resolution of 384x384. This method guaranteed that the total resolution of both divided models was 640x640, ensuring a fair comparison in total resolution.

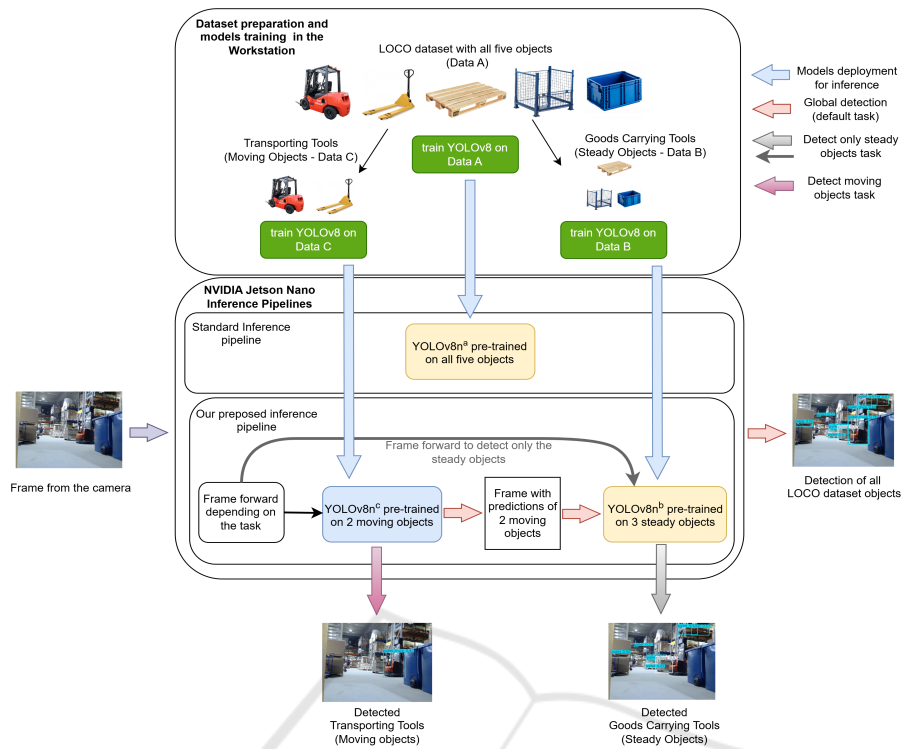


Figure 2: Proposed class-specific dataset splitting method on YOLOv8 for LOCO dataset.

Table 1: The used data for training, validation, and testing for YOLOv8n evaluation. YOLOv8n^a is trained on data A. YOLOv8n^b is trained on data B, and YOLOv8n^c is trained on data C. Data A is a combination of DATA B and C that contains all classes.

Class	Data A					
	Data B			Data C		
	Train	Val	Test	Train	Val	Test
SLC	13303	5532	3316	-	-	-
Forklift	-	-	-	353	153	92
Pallet	72306	30097	18042	-	-	-
Stillage	3247	1351	809	-	-	-
Pallet Truck	-	-	-	1695	708	474

(-): No annotations for this class in the subset.

2.3 Experimental Setup

For model training, we utilized a workstation equipped with a powerful GPU to ensure efficient processing and faster training times for the YOLOv8 models. The specifications of the workstation are in Table 2.

After the models training, we moved them to the Jetson Nano for validation. We utilized the designated ultralytics docker container for the Jetson Nano (Jocher et al., 2023). This method is required since the Jetson Nano officially operates on Ubuntu 18.04, which includes an incompatible Python version for the YOLOv8 setup. The Docker container has Python

Table 2: Workstation Specifications for Model Training.

Component	Specification
CPU	Intel Core i7-12700K
Memory	64GB DDR5 6000 MHz
GPU	NVIDIA RTX 3090
Python version	3.8.19
PyTorch version	2.0.1
CUDA version	11.7

3.8.0 and CUDA 10.2, as well as PyTorch 1.11.0, to guarantee compatibility and efficient execution of the YOLOv8 models. Table 3 displays the Jetson Nano specifications.

Table 3: NVIDIA Jetson Nano 4GB Specifications.

Component	Specification
GPU	Tegra X1 (128-core Maxwell)
CPU	Quad-core ARM Cortex-A57
Memory	4GB LPDDR4
Storage	32GB
Python Version	3.8 via docker
PyTorch Version	1.11.0
CUDA version	10.2

During validation, we set the batch size to one to emulate the performance of frame-by-frame real-time video processing. This configuration allows us to assess how effectively the models can detect in a continuous video stream, simulating real-world conditions that autonomous forklifts would encounter in logistics environments. Furthermore, we evaluated the model's efficiency for rapid forklift operations by calculating the distance covered in each frame prediction.

In model assessment, we used Precision (P), Recall (R), and Mean Average Precision (mAP) as the main metric, which are commonly used for comparing object detection models. These metrics offer a thorough evaluation of precision among various classes and aid in evaluating the efficiency of models in identifying different objects in the logistics setting.

3 RESULTS AND DISCUSSION

We trained YOLOv8n in three settings and named our pretrained models as YOLOv8n^a, YOLOv8n^b, and YOLOv8n^c. YOLOv8n^a is the model that was trained for all the objects in dataset. YOLOv8n^b is trained on goods-carrying objects. YOLOv8n^c is model trained on transporting tools. Table 4 showcases the accuracies and inference times obtained in the NVIDIA Jetson Nano.

3.1 Detection Accuracy Comparison

The YOLOv8n^b and YOLOv8n^c Outperformed YOLOv8n^a when evaluated with precision, which means the detections has less confusion over YOLOv8n^a. YOLOv8n^c reached higher accuracy for forklift detection in all evaluation matrices and outperformed the YOLOv8n^a on this class. However, the other objects showed slightly lower mAP@50 and recall results when using YOLOv8n^c or YOLOv8n^b compared to YOLOv8n^a. Specifically, the accuracy loss across classes ranged between approximately 1% to 5% per class in mAP. Despite the modest accuracy losses in detecting other objects, our method demonstrated enhanced detection performance for

high-resolution (1080p) for forklift detection using YOLOv8n^c compared to YOLOv8n^a as demonstrated in figure 3. Nevertheless, the YOLOv8n^b and YOLOv8n^c models showed lower performance when 480p resolution footage was employed, resulting in more objects being missed compared to YOLOv8n^a. The reduced accuracy in YOLOv8n^b and YOLOv8n^c models is due to the smaller image sizes (256 and 384), leading to decreased detection reliability for objects at lower resolutions and far distances. Figure 4 displays the findings from detecting images with a 480p resolution.

3.2 Inference Time Comparison

YOLOv8n^b demonstrated a noteworthy enhancement with an inference time of 64.7 ms, which is a significant improvement of approximately 56.2% compared to YOLOv8n^a with 147.7 ms. Specifically, the time it takes to detect specialized forklifts and pallet trucks using the YOLOv8n^c model has been reduced by 74.4%, now taking only 37.8 ms, showing a significant increase in efficiency with customized models that have smaller image sizes. The data in table 5 displayed the overall time taken for inference by two models compared to a model trained on 5 classes.

3.3 Inference Impact over High-Speed Forklift

The combined inference time for YOLOv8n^b and YOLOv8n^c models, trained on 3 and 2 classes respectively, was significantly faster than the inference time of YOLOv8n^a, which was trained on all 5 classes. Together, YOLOv8n^b and YOLOv8n^c have an inference time of 102.5 ms (64.7 ms for YOLOv8n^b and 37.8 ms for YOLOv8n^c), compared to 147.7 ms for YOLOv8n^a. This leads to a 30.6% improvement in inference time, demonstrating that dividing the detection task among specialized models with lower resolution can efficiently decrease processing time. This increase in efficiency is particularly advantageous for real-time tasks, as quicker detection can enhance the responsiveness of autonomous systems, like forklifts in warehouses with varying speeds.

The inference time of each YOLOv8 model used in a forklift moving at 11 km/h (around 305.56 cm/s) directly affects how frequently the system can identify and respond to obstacles. As an illustration, YOLOv8n^b takes 64.7 ms for inference, allowing the forklift to move 19.77 cm in each inference cycle. This enhances the forklift's ability to detect objects effectively while moving. YOLOv8n^c allows the forklift to move 11.55 cm in one inference, de-

Table 4: Performance results for YOLOv8n^a, YOLOv8n^b, and YOLOv8n^c across evaluation metrics (Precision, Recall, and mAP50) with their inference times on NVIDIA Jetson Nano.

Class	YOLOv8n ^a			YOLOv8n ^b			YOLOv8n ^c		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
SLC	0.749	0.424	0.579	0.771	0.307	0.541	–	–	–
Forklift	0.854	0.497	0.685	–	–	–	0.887	0.562	0.725
Pallet	0.872	0.449	0.663	0.887	0.332	0.613	–	–	–
Stillage	0.881	0.508	0.711	0.900	0.442	0.685	–	–	–
Pallet Truck	0.820	0.503	0.666	–	–	–	0.837	0.442	0.643
Inference (ms)	147.7			64.7			37.8		
Image size	640			384			256		

(-): Model not trained for this class.

Figure 3: Comparative detection results for 1080p image. YOLOv8n^c gave best performance to detect pallet trucks.Figure 4: Comparative detection results for 480p image. YOLOv8n^a detected a pallet truck as a forklift.

Table 5: Combined models performance over single model.

Model	Precision	Recall	mAP50	mAP50-95	Inference (ms)
YOLOv8n ^a	0.8350	0.476	0.661	0.397	147.7
YOLOv8n ^{b+c} (ours)	0.8575	0.431	0.6485	0.392	102.5

creasing inference time to 37.8 ms giving faster updates. This particular frequency is very beneficial for quickly detecting nearby objects, minimizing the chances of missing obstacles.

When YOLOv8n^b and YOLOv8n^c are combined, their cumulative inference time of 102.5 ms leads to a distance of 31.32 cm per inference cycle. Although this is slower than YOLOv8n^c alone, it remains a substantial improvement over the YOLOv8n^a model, trained on all five classes, which has an inference time of 147.7 ms and results in a distance of 45.14 cm per inference. The increased time interval between frame

updates in YOLOv8n^a may reduce the accuracy of obstacle detection, as the forklift may travel a considerable distance before the next frame is analyzed by the model. In fast-paced environments, this delay could heighten the chance of quickly appearing obstacles. Table 6 illustrates the distance covered during each inference with a forklift moving at a speed of 11 km/h.

Table 6: Theoretical distance per inference for Forklift with speed of 11km/h.

Model	mAP50	Inference (ms)	Distance (cm)
YOLOv8n ^a	0.661	147.7	45.14
YOLOv8n ^{b+c} (ours)	0.6485	102.5	31.32
YOLOv8n ^b (ours)	0.613	64.7	19.77
YOLOv8n ^c (ours)	0.684	37.8	11.55

4 CONCLUSIONS

This research findings confirmed that training the YOLOv8 model with a class-specific dataset split from the LOCO dataset greatly improved inference efficiency, resulting in a 30.6% decrease in overall inference time. Significantly, there were even more improvements in targeted detection tasks, with inference times decreasing by 74.4% for transporting tools and 56.2% for carrying tools. When applied to a forklift moving at a top speed of 11 km/h, this method reduced the distance covered per inference round from 45.14 cm to 31.32 cm, resulting in a minimum travel distance of 11.55 cm when identifying transporting equipment. Hence, these improvements were made with only a 1.25% decrease in mAP, ensuring adequate accuracy for real-world use.

In addition, the research discovered that decreasing the image size setting in YOLOv8 resulted in a notable decrease in inference times, which enhanced its efficiency in real-time object detection. Yet, the decrease in resolution led to failures in detection, especially for smaller objects. The results show that decreasing image size is most advantageous for datasets with bigger object annotations, while the accuracy of detection remains mostly unchanged. Hence, it is advised to utilize this technique in situations with high-resolution photos and bigger objects to strike a perfect equilibrium between speed of inference and performance of detection.

ACKNOWLEDGEMENTS

We acknowledge the use of ChatGPT4o to enhance the readability of our paper.

REFERENCES

Asdikian, J. P. H., Li, M., and Maier, G. (2024). Performance evaluation of YOLOv8 and YOLOv9 on custom dataset with color space augmentation for Real-time Wildlife detection at the Edge. In *2024 IEEE 10th International Conference on Network Softwarization (NetSoft)*, pages 55–60. ISSN: 2693-9789.

Behrje, U., Himstedt, M., and Maehle, E. (2018). An Autonomous Forklift with 3D Time-of-Flight Camera-Based Localization and Navigation. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1739–1746.

Cidal, G. M., Cimbek, Y. A., Karahan, G., Böler, O. E., Özkardesler, O., and Üvet, H. (2019). A Study on the Development of Semi Automated Warehouse Stock Counting System. In *2019 6th International Conference on Electrical and Electronics Engineering (ICEEE)*, pages 323–326.

Clavero, C., Patricio, M. A., García, J., and Molina, J. M. (2024). DMZoomNet: Improving Object Detection Using Distance Information in Intralogistics Environments. *IEEE Transactions on Industrial Informatics*, 20(7):9163–9171. Conference Name: IEEE Transactions on Industrial Informatics.

Jocher, G., Qiu, J., and Chaurasia, A. (2023). Yolov8 by ultralytics. original-date: 2022-09-11T16:39:45Z.

Khalfallah, S., Bouallegue, M., and Bouallegue, K. (2024). Object detection for autonomous logistics: A yolov4 tiny approach with ros integration and loco dataset evaluation. *Engineering Proceedings*, 67(1).

Mayershofer, C., Holm, D.-M., Molter, B., and Fottner, J. (2020). LOCO: Logistics Objects in Context. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 612–617.

Mohamed, I. S., Capitanelli, A., Mastrogiovanni, F., Rovetta, S., and Zaccaria, R. (2018). Detection, localisation and tracking of pallets using machine learning techniques and 2D range data.

Savas, R. and Hinckeldeyn, J. (2022). Critical Evaluation of LOCO dataset with Machine Learning. arXiv:2209.13499 [cs].

Zaccaria, M., Giorgini, M., Monica, R., and Aleotti, J. (2021). Multi-Robot Multiple Camera People Detection and Tracking in Automated Warehouses. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–6.

Zaccaria, M., Monica, R., and Aleotti, J. (2020). A Comparison of Deep Learning Models for Pallet Detection in Industrial Warehouses. In *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 417–422.