Extracting Miles and Kilometers From Odometer Pictures

Ali Aykut ARIK, Melih AKSOY, Mert DOĞRAMACI
Rock of MAM
Supervisor: Asst. Prof. Dr. Ahmet Selman BOZKIR
Computer Engineering, Hacettepe University

Our commitment lies in developing an application that uses an object detection model that was trained by us to revolutionize the collection of odometer readings for insurance providers.

PROJECT JOURNEY

1

Creating the Dataset

Creating a high-quality dataset is a crucial initial step in training an object detection model for odometer reading collection. Our dataset consists of annotated images that accurately represent various real-world scenarios, including different vehicle types, odometer designs, lighting conditions, and perspectives.

2

Training the Model

We trained two different models and eleven different architectures:
- For YOLOv6, we trained the S, M, and L architectures twice.
- For YOLOv8, we trained the N, S, M, L, and X architectures.

The overall best result is the result of the X architecture of the YOLOv8.

3

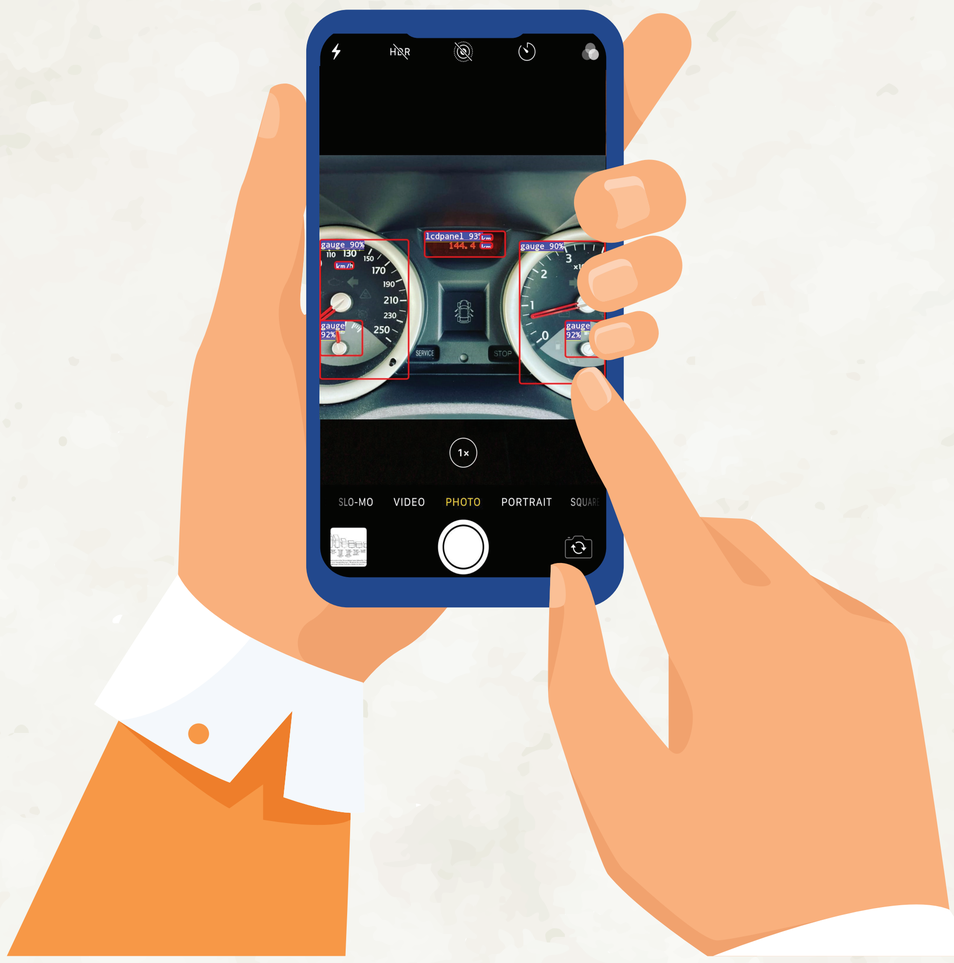
Developing the App

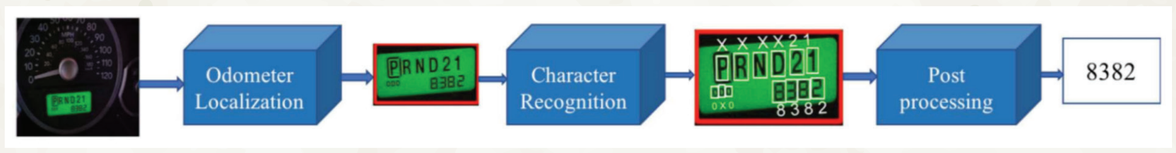
We have developed a comprehensive solution that includes both a mobile application and a web application. The mobile application can read both with its embeded model and via information obtained from the server. The web application is used to manage the user data, tracking processes and more.

4

Conclusion

In this system, which we created to minimize user errors, we ensured that the responsible or fleet managers could easily follow the processes. With a user-friendly interface, we send the data of users who take pictures of their odometers to the server for processing. It allows administrators to follow the processes with the filters they want (such as date, user, mileage differences in a certain range) by keeping the processed data.





Training Results

Model	Dataset	Architecture	Average Precision	Average Recall	F1 Score	mAP50	mAP50-95
YOLOv8	custom_dataset	N	0.815	0.789	0.802	0.836	0.554
		S	0.867	0.832	0.849	0.876	0.578
		M	0.874	0.841	0.857	0.886	0.584
		L	0.908	0.842	0.874	0.903	0.598
		X	0.924	0.831	0.875	0.905	0.598
	modified_dataset	N	0.902	0.822	0.860	0.893	0.556
		S	0.901	0.857	0.878	0.920	0.569
		M	0.890	0.894	0.892	0.926	0.578
		L	0.888	0.898	0.893	0.927	0.587
		X	0.877	0.903	0.890	0.933	0.590
	removed_large_panels	N	0.786	0.783	0.784	0.828	0.504
		S	0.875	0.831	0.852	0.878	0.532
		M	0.898	0.823	0.859	0.892	0.546
		L	0.885	0.855	0.870	0.899	0.553
		X	0.877	0.867	0.872	0.902	0.551

The overall best result is the result of the L architecture of the YOLOv8 that trained with the modified dataset. We can say that we expect some of those results, however there are some parts that are surprisingly interesting. If we consider the results in terms of differences between the N, S, M, L, and X, the bigger network should give better results, and we got what we expected. Hence, the biggest and best model gave the best results.