# **License plate detection model YOLOv8 using OCR**

**YoloV8:**

YOLOv8 is a powerful deep-learning model that has been pre-trained on a vast dataset, and its unique architecture allows it to detect objects at various scales and resolutions within an image. As a result, it is more accurate than other models that rely on a fixed set of anchor boxes.

Another significant advantage of YOLOv8 is its support for transfer learning, which enables customization and fine-tuning of the model to perform specific tasks, such as license plate detection. Although we experimented with other models, such as VGG16, we found that YOLOv8 outperformed them in terms of speed and accuracy, making it the ideal choice for our license plate detection model.

Therefore, we selected YOLOv8 as the primary model for our license plate detection system.

**Python:**

Python is a popular programming language for deep learning and computer vision applications. One of the main reasons for using Python with YOLOv8 is that it has a large number of libraries and frameworks that can be used for deep learning, such as TensorFlow, PyTorch, and Keras.

**Dataset and training process:**

The data set consisted of approximately 258 images, and data augmentation was applied to increase the size of the data set. The purpose of data augmentation was to enhance the overall performance of the model. The data set was divided into three parts for training, validation, and testing, with the percentages of 70%, 17%, and 13%, respectively.

**Training Process:** We trained our data using various epoch batch-sizes and optimizers. Initially, we trained the model for 75 epochs, but we found that it was under fitted since the training loss was increasing while the validation loss remained constant. The accuracy curve also did not show any improvement. Next, we increased the epoch value to 150, but this led to overfitting as there was a significant difference between the training and validation accuracy. To overcome overfitting, we used early stopping which is a good technique for this problem. Eventually, we found that training the model for 100 epochs yielded a 94% accuracy, and the model performed well in several epochs. We used the standard batch size of 16, which was suggested by the pretrained YOLOb8 weights, and kept the patience value at 50, along with the SGD optimizer.

**Data augmentation:** In order to improve the accuracy of license plate detection on our dataset, we applied several effective image augmentation techniques. First, we used random cropping, this technique involves randomly cropping the input image to create multiple variations of the same image, therefore it can help the model generalize better by introducing more variability in the training data, and it can also help reducing overfitting. Further, we applied rotation, as it can teach the model to detect slightly tilted or angled license plates, it can also help to improve the model's ability to handle images that are not perfectly aligned. Moreover, we applied exposure, due to the fact that exposure can help the model better handle images with different lighting conditions, and Gaussian blur can reduce the effect of noise and improve the detection of partially obscured plates. By using a combination of these augmentations, the license plate detection model can become more accurate and robust, better able to handle real-world scenarios.

**Image enhancement and reading the license plates using OCR.**

In order to improve the accuracy of OCR we have to apply image filters on license plates as filters can help by reducing noise, enhancing contrast, and increasing clarity. After getting the results from our yolov8 model, we crop the original image to only get the bounding box of the license plate based on the coordinates we get from the prediction. Further, now that we have a cropped image with only the license plate, we can apply filters. We tried multiple filters such as edge detection, Gaussian blur, thresholding, morphological operations (dilation and erosion), and grayscale. However, some filters didn't perform as expected and didn't provide the desired results for instance: Gaussian blur and edge detection, therefore we had to remove them. Furthermore, morphological and thresholding performed better than previous filters, however if we combine both it would not give us the ideal results. Further, combining grayscale with morphological or thresholding always resulted in better results. At the end, we decided to use thresholding combined with grayscale, as it performed better than the other filters

Based on the achieved 94% accuracy, we can safely assume that our license plate detection model is performing well. Furthermore, we enhanced the images to improve the readability of the number plates using OCR (Optical Character Recognition) technology. Allowing it to identify the characters on the number plates. However, in low-resolution images, the model may encounter some difficulties in reading the characters correctly. Overall, we can be confident in the performance of our model and its ability to accurately recognize license plates.

## **Notebooks:**

There are two notebooks that we used for our project. The first Google Colab is where we trained our custom dataset on the YoloV8 model, and in this notebook, you will find various monitoring diagrams such as the confusion matrix and training/validation results. If you are interested in reviewing the training process, you can find the link to the Google Colab below.

<https://colab.research.google.com/drive/1OQL3Og_VfS3f3ZVt5OZ1Tb1z4GrHCx_l?usp=sharing>

The second file is a Jupyter notebook (License-Plate-Model-With\_OCR) where we used the weights obtained from the training process to test our model on new images. In this notebook, we also applied image enhancement and OCR to improve the accuracy of license plate detection. It is attached to the project files.

## **Contribution**

**Individual contribution (Musavir Ahmed)**

As part of our big group project, we divided tasks among ourselves. I took on the responsibility of researching available object detection models specifically for license plate detection. After conducting thorough research, I found that the YOLOv8 model would be the best fit for our project. I delved into the YOLOv8 documentation and tested it with pre-trained weights, and it performed well. As a result, we decided to use the YOLOv8 algorithm for our project.

In addition to my research on the model, I also explored different libraries such as Easy OCR and Keras OCR for reading license plates. Also, I contributed to the collection of images and videos, and I annotated them on Roboflow.

Finally, I participated in creating all technical and functional diagrams, including class, sequence, activity, and use case diagrams, with my group members.

**Individual contribution (Mohammed Al-Selmi)**

Firstly, I was responsible for data collection, which involved gathering the dataset for license plate detection. During collecting the dataset, I made sure to collect a high-quality dataset that was suitable for the project's objectives. Secondly, I was involved in annotation. Annotation involves marking up the data with labels that will help the machine learning algorithms learn from the data. For one approach, I annotated the license plates in the dataset. Additionally, I collaborated with Alexander to annotate both the logos and characters in the license plates for another approach.

Thirdly, before starting to train the annotated dataset, I independently searched for a suitable model to train the data. Moreover, I was responsible for training a model from scratch using VGG16 for license plate detection. I did this without any pre-trained weights, which is a challenging task as it requires significant computational resources and expertise. Despite these challenges, I was able to train a model with an accuracy of 76, however this model didn't preform as expected compared to the model found by Musavir, therefore, I collaborated with Musavir to work on the license plate detection model and OCR using the YOLOv8 approach. I worked closely with Musavir to fine-tune the models, which required considerable effort.

For the TFGD, I played an active role in developing both the functional and technical design of the application. In terms of the functional design, my primary focus was on the use case and activity diagrams. Specifically, I contributed significantly to the development of the use case for license plate detection and OCR, while also providing insights to the group working on brand detection and how can we use both models. As we ultimately merged the TFGD of both the brand detection and license plate detection models together, it was essential that we worked collaboratively and understood each team member's goals. Regarding the technical design, I was heavily involved in the creation of the sequence diagram. While my main focus was on the license plate detection and OCR aspect of the diagrams, I also provided feedback to the other group as we merged the components together.

**Collaborative contribution (Musavir Ahmed and Mohammed Al-Selmi )**

We collaborated on various key tasks in this project. As we both tested different algorithms for license plate detection, we explored models such as YOLOv8 and VGG16 to determine the best-performing and user-friendly one. Together, we decided to use YOLOv8, which had an accuracy rate of 94%.

After selecting the model, we collaborated on refining our approach. For instance, we decided on the training process techniques, including the number of epochs, batch size, and optimizer to use, as well as how to prevent overfitting and underfitting our model. We examined the results and added image enhancements and morphological processes to improve character recognition, which significantly improved the results.