Parking Spot Detection and Classification: Assessing Legality and Slot Type Using Deep Learning and Rule-Based Approaches

Caterina Belluti,[1] Sara Gherardi,[2] Federico Gualandri[3]
University of Modena and Reggio Emilia
Emails: {300357[1], 306432[2], 301116[3]}@studenti.unimore.it

***Abstract*—The objective of this paper is to present our solution to the problem of identifying the correct use of parking spaces. Starting from an image of a parking area from above, first we employ detection to identify single parking spots, then classification to differentiate between occupied and empty spaces; additionally, if a car is present, we classify each slot as either correct or incorrect parking, otherwise we classify the type of empty space like normal, paid, disabled or pregnant parking. To tackle this issue, we use many approaches that we will discuss more in detail later: we employed mainly neural networks for object detection and classification, but also rule-based approaches. We assume that the starting images of parking areas are taken from above, for example by a drone, and our main goal is to identify wrongly parked cars to eventually report them, but also to recognize empty parking spaces and classify their types; this last count can be useful for users to know if there is a free parking slot they can use. Our code is available on GitHub. [1]**

I. Introduction

Considering images of a parking area taken from above, for example by a drone if it's an outdoor parking or by a camera if it's an indoor parking, our main goal is to optimize the usage of parking slots by correctly identifying and reporting wrongly parked cars. Additionally, incoming drivers who want to park their cars would benefit from knowing how many parking spots are available and their types; the solution presented implements also a procedure that does just this.

We took into account that the images may contain some disturbances, like salt and pepper and Gaussian noises, so we used a light preprocessing that takes care of it. Some constraints of the system are:

- it assumes that all cars in the image are parked
- we considered parked cars, but it works with other vehicles like trucks too, even though with lower accuracy
- the lines need to be visible, especially for the task of classifying the types of empty parking spaces

To tackle these problems, our pipeline consists of four tasks: preprocessing of the images to remove noise and enhance the visual properties that interest us, object detection of parking spots and classification between free and busy spaces, cropping the results to obtain images of single slots and finally classification again: parked correctly or not for the occupied ones and according to the type of parking spot for the empty ones. Our pipeline is shown in Figure 1; now we'll talk more in detail about each of the steps.

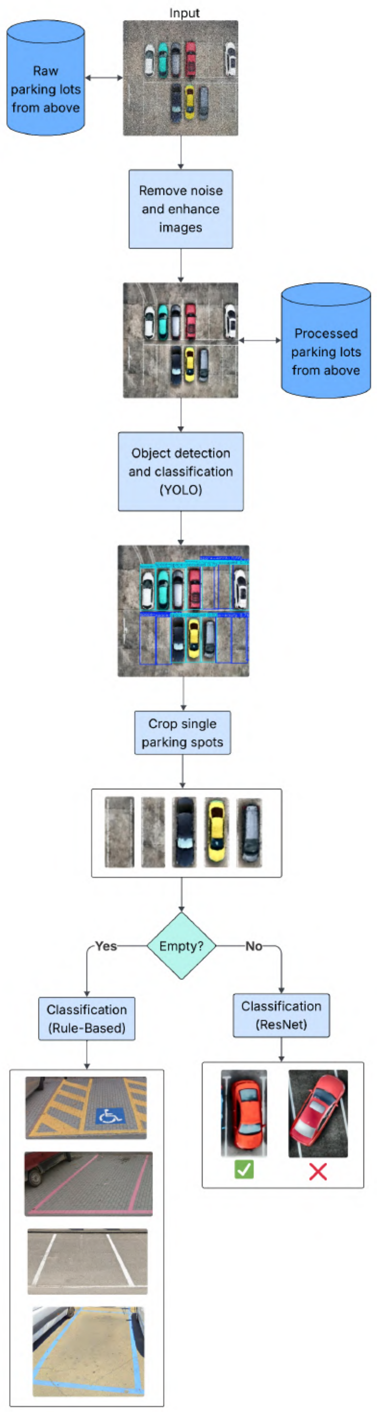
Input
Raw parking lots from above
Remove noise and enhance images
Processed parking lots from above
Object detection and classification (YOLO)
Crop single parking spots
Empty?
Yes
No
Classification (Rule-Based)
Classification (ResNet)

Fig. 1. Pipeline

A. Related Work

We took inspiration from other existing works in the parking detection and classification field, even if with different purposes. To build our system, we relied on solutions already tried and documented by others, combining them in a way that better helped reaching our specific goals. For example we employed public datasets widely used and tested in different papers, like CNRPark and PKLot [2] [3] [4]. We also examined and tried different models for object detection and classification obtaining good results with yolov8 [5], and we even found good validation for the choice of ResNet18 [6] for classification in the field of computer vision regarding parking lots.

II. Data

A. Dataset

The main datasets that we employed are:

- CNRPark [7], provides many images of single parking spaces either empty or occupied which we mainly used for classification
- PKLot [8], made up of many images of parking areas taken from above which we mainly used for object detection
- other smaller datasets taken primarily from Roboflow [9]

Furthermore, for images that we couldn't find a large number of, such as those of cars parked incorrectly or of empty parking spaces of different types, we integrated some taken by ourselves using tools like Google Earth.

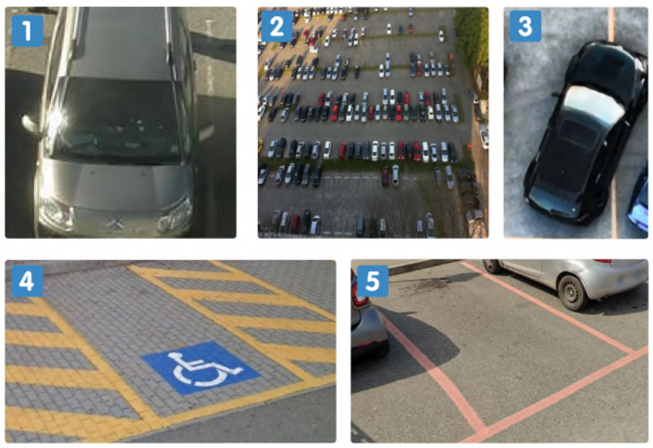

Fig. 2. 1. Image taken from CNRPark [7], 2. Image taken from PKLot [8], 3. Image taken from Roboflow [9], 4-5. Images taken from Google Earth

B. Preprocessing

In general we assume that there might be some noise present when the images are acquired, in particular we found that many images of the datasets were affected by salt and pepper noise; a median filter is applied to the input images to solve this issue.

The median filter replaces the value of a pixel in an image with the median value of the intensities of its neighbours. The filtered image I' at pixel (x, y) is defined as:

$$I'(x,y) = \text{median}\left\{I(i,j) \mid (i,j) \in \mathcal{N}(x,y)\right\} \qquad (1)$$

where I is the original image, and $\mathcal{N}(x,y)$ is the set of pixels in the neighborhood around the original pixel (x, y).

A general problem when talking about parking slots is that the parking lines might be faded or poorly visible: to solve this issue we employed a filter to increase the contrast of the images.

For contrast enhancement, we decided to use CLAHE (Contrast Limited Adaptive Histogram Equalization), which operates on the lightness channel in the LAB color space, separating lightness from color better than BGR. This allows us to better visualize parking lines without affecting the color information that can be useful later. After adjustment, the modified channel is merged back with the original ones, and the image is converted back to the BGR color space.

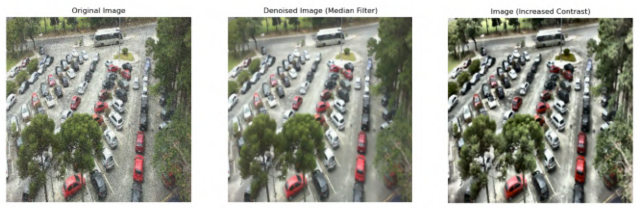

Fig. 3. Example of an image denoised and enhanced

III. Approach

A. Object Detection and Classification

Now that we have the preprocessed images of parking areas, the first task we have to do is to isolate single parking slots; to do this we employed an object detection model to identify bounding boxes of the parking spaces and classify them as either empty or occupied. In particular, we decided to use yolov8n [10] because of its lightness and low number of parameters, which provides fast inference times: we used a pre trained model that we fine tuned with our dataset consisting of PKLot [8] integrated with other smaller datasets, taken mainly from Roboflow [9]. In total we have 15717 images for training and 3041 for validation.

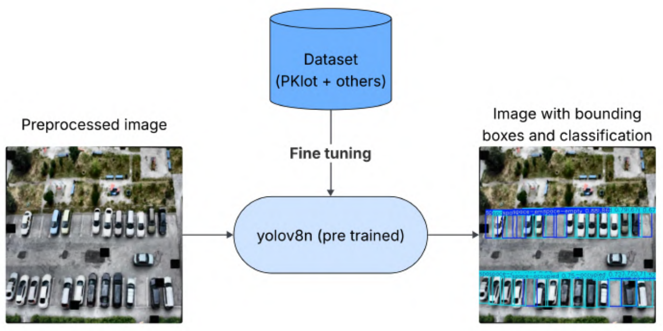

Fig. 4. YOLO model

Let's consider the training results of our YOLO model shown in Figure 5.

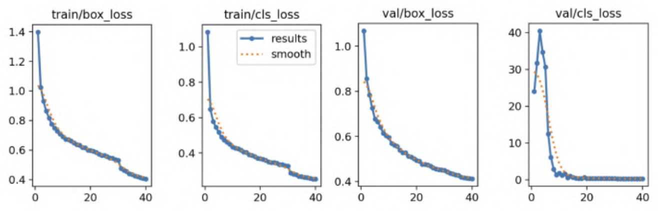

Fig. 5. YOLO model training curves

We decided to stop the training after forty epochs because we saw from the graphs in Figure 6 that the precision and the recall were getting worse.

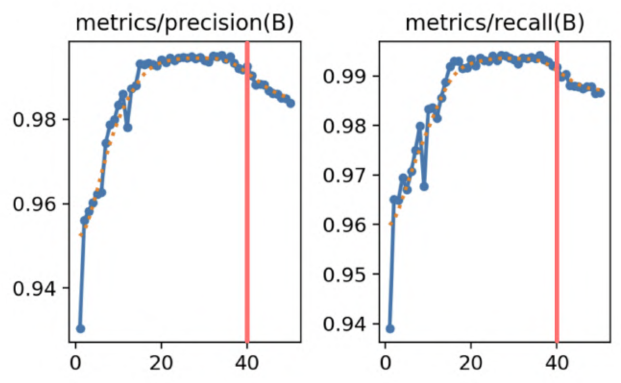

Fig. 6. YOLO model precision and recall curves

Where

$$precision = \frac{TP}{TP + FP} \tag{2}$$

and

$$recall = \frac{TP}{TP + FN} \tag{3}$$

with TP = True Positives, FP = False Positives and FN = False Negatives.

It's also important to consider other metrics, but before let's recall:

- Average Precision (AP) which measures how well the model identifies the correct objects with correct classes and bounding boxes
- Mean Average Precision (mAP) is often considered for object detection, being the average of AP values on all classes

$$mAP = \frac{1}{N}\sum_{i=1}^{N} \mathrm{AP}_i \tag{4}$$

- Intersection over Union (IoU) measures how well the predicted bounding box (A) corresponds to the real one (B)

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{5}$$

We can now consider mAP50, which is the AP calculated with a threshold of IoU equal to 0.50, and mAP50-95, which is stricter and computes the AP on ten different thresholds of IoU, from 0.50 to 0.95, and then considers the average during the training of our model. These metrics are shown in Figure 7.

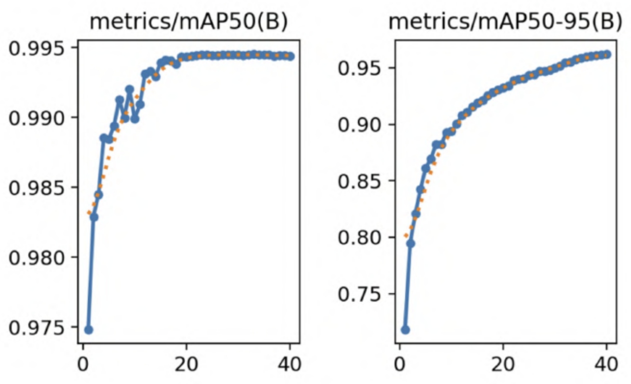

Fig. 7. YOLO model mAP curves

B. Cropping

The single parking slots are then extracted from the annotated images: the cropping takes place according to the bounding boxes coordinates that the object detection model predicted, then the resulting images are separated in two groups according to the labels of the bounding boxes, one for the empty slots and another for the occupied ones. This serves as preprocessing for the subsequent classifications explained in the next sections.

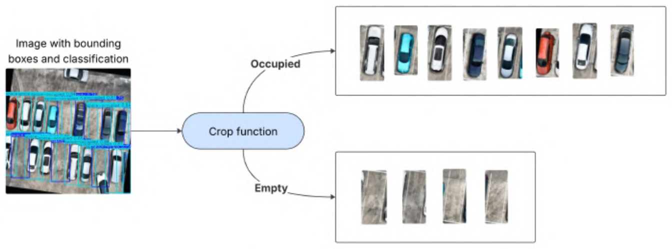

Fig. 8. Cropping of single parking slots

C. Classification of occupied spots

For the occupied parking slots we need to classify between cars parked correctly and incorrectly; to do this we employed a classification model, in particular we decided to use a pre-trained ResNet18 [11] model that we fine tuned using smaller datasets taken from Roboflow integrated with other images found by us on the web. In total, we have 2015 images for training and 215 for validation.

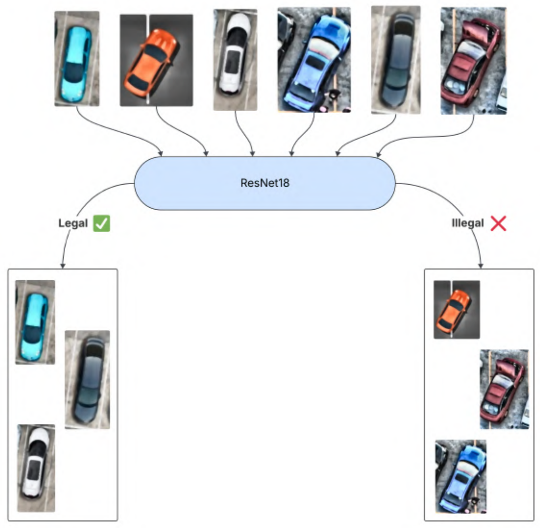

Fig. 9. ResNet18 [11] model

Let's consider the training results of our ResNet18 [11] model:

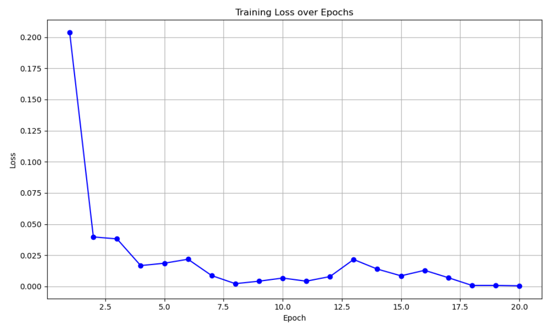

Fig. 10. ResNet18 [11] model training results

D. Classification of empty spots

For empty parking spots we need to classify between different kinds of parking spaces:

- white lines = free parking
- blue lines = paid parking
- yellow lines = disabled/reserved parking
- pink lines = parking for pregnant women

To do this, we employed a rule-based approach considering the color of the parking lines to do the classification; as a constraint we considered the Italian guidelines for parking colors, so the model might need to be adapted for other countries.

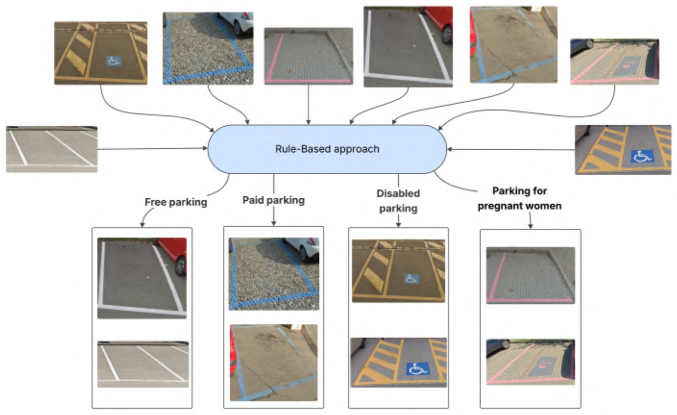

Fig. 11. Rule-based approach to classify empty parking spaces

Since the image is that of an empty slot we assume that the background is either black or grey, and we use a rule-based approach to look for the colors of the parking lines in the image. We compute the presence of each color in the image and then classify the type of the parking slot as the one with the highest percentage. To do this, we project the cropped empty slots in the HSV space, which is a color model that describes colors in terms of their Hue, shade (Saturation) and their brightness (Value). After trying out different thresholds for the values for these three parameters, we kept the four ranges that gave us the best results. This approach is fast and effective but has some limitations as it considers only color: a very light or dark image, as well as the presence of other colored objects in the scene, could lead to errors.

For the empty parking slots with white lines, the hue is not meaningful, so we focus on the saturation that we estimated between 0 and 40 as shown in Figure 12 and also on the value that needs to be high in the range of 220 and 255.

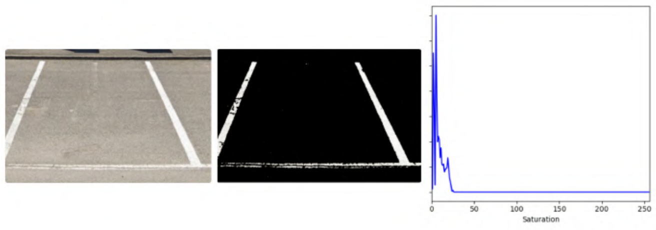
Fig. 12. Example of an empty parking slot with white lines followed by its mask and the relative histogram with saturation values

Instead, for the other types of parking lines, the discriminant for classification is the value of the hue. For the blue ones we consider a range of 90 and 130, for the yellow ones between 15 and 40 and for the pink ones between 150 and 180. As shown in the figures below, for each range we have a corresponding peak in the histogram.

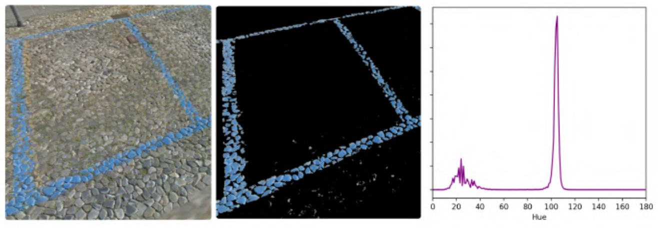
Fig. 13. Example of an empty parking slot with blue lines followed by its mask and the relative histogram with hue values

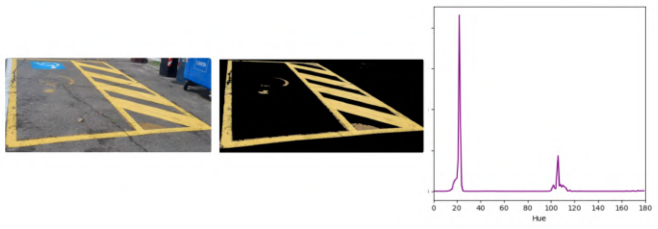
Fig. 14. Example of an empty parking slot with yellow lines followed by its mask and the relative histogram with hue values

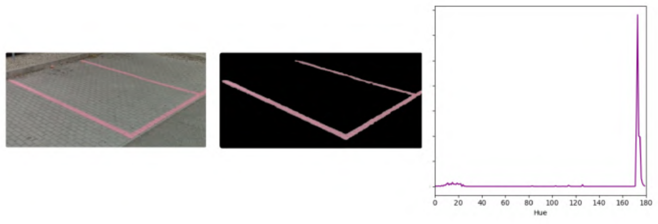
Fig. 15. Example of an empty parking slot with pink lines followed by its mask and the relative histogram with hue values

IV. Evaluation

For now, we've only looked at the training results of our models and approaches, so let's now consider the actual performance of our system.

A. *Object Detection and Classification*

For evaluating the performance of the yolov8n [10] model for object detection, we used a test set consisting of 1468 images, taken mainly from PKLot [8] integrated with other smaller datasets from Roboflow [9]. Let's start by considering the performance of the model in drawing the bounding boxes; the main metric used for this task is the Mean Average Precision (mAP) (4), in particular looking at the values of mAP50 and mAP50-95:

TABLE I
MAP FOR EACH CLASS

Class	**mAP50**	**mAP50-95**
space-empty	0.994	0.970
space-occupied	0.994	0.962

These values are pretty high, this is because most of the images of our dataset are taken from a fixed camera at different times and so our images are similar, albeit different. This could lead to a problem of overfitting but for our task, since we take images of parking areas from above, for example from a drone, we consider this acceptable, as we're going to have a similar scenario.

The YOLO model can also be used for the classification task between empty or occupied parking slots; we also tried a rule-based approach that can be used if we assume that the input images are of single parking spots and not of large parking areas. It can also be used just for the classification task between empty or occupied slots after the object detection model has detected the bounding boxes and these have been cropped into separate images. We tested this rule-based model on CNRPark: this dataset consists of 144965 images of single parking slots labeled either occupied or empty. We can compare the performances to understand the reason why in the end we chose the YOLO model both for object detection and for classification.

TABLE II
PRECISION AND RECALL COMPARISON

Model	**Precision**	**Recall**
yolov8n	0.988	0.991
Rule-Based	0.742	0.952

We also computed the confusion matrix for both methods to verify the results. The values that are on the diagonal represent the number of examples classified correctly, while the values that are not represent the classification errors: being the first ones high with respect to the others, and considering the number of images that we have, the results are good.

TABLE III
CONFUSION MATRIX - YOLOV8N

Predicted \ True	space-empty	space-occupied	background
space-empty	37950	58	554
space-occupied	506	38372	584
background	138	64	-

The background is an additional class: if both the scores for classifying a parking spot as either empty or occupied are low, the image gets classified as neither, but instead as background.

TABLE IV
CONFUSION MATRIX - RULE-BASED

Predicted \ True	space-empty	space-occupied
space-empty	39380	26304
space-occupied	3827	75454

The rule-based approach tries to classify between occupied and empty parking slots by computing a score and comparing it with a threshold. The score is a weighted sum of different components:

$$\begin{aligned}\text{score} = \alpha \cdot \text{edge\_density} \\ + \beta \cdot \text{contour\_score} \\ + \gamma \cdot \text{texture\_score} \\ + \delta \cdot \text{diff\_score}\end{aligned} \tag{6}$$

- diff_score is the difference of the target image from an empty parking space; we considered an image made of gray pixels to represent the asphalt
- texture_score is a value that takes into account the luminosity of the image and is computed using the average brightness of the image
- edge_density is a value that considers the importance of the edges detected on the image. An edge is a local property of a region, it's a point where there's a strong change of luminosity in a direction so it's a discontinuity in the image. The Canny [12] operator computes the gradient for every pixel of the image: if the magnitude of the gradient is higher than a threshold, then it's considered as a strong edge otherwise it's discarded. To compute the gradients, the Sobel mask is employed:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{7}$$

  This is only a brief summarization, the complete algorithm is more complex
- contour_score is a value that considers the importance of the contours in the image, in particular we keep only the strongest ones; the concept of contours is useful to identify objects present in the images. To find the contours, we used the OpenCV [13] function `findContours()` [14].

The parameters α, β, γ and δ are constants that we defined heuristically by trying different values.

For the threshold, we used a function to iteratively try different values to find the best one. This rule-based method is faster but it performs worse, so we decided to keep it as a second option.

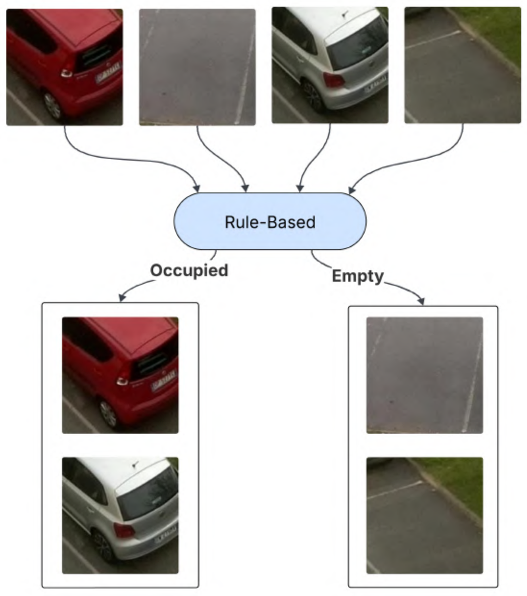

Fig. 16. Rule-based approach

B. Classification of occupied spots

For evaluating the performance of the ResNet18 [11] model for classification, we used a test set consisting of 200 images, taken mainly by integrating smaller datasets from Roboflow [9]. The bounding boxes computed by the YOLO model for object detection are used to crop the image into single parking spots, thus the quality of this step depends on the performance of the previous one. The ResNet model is applied on the occupied parking slots to identify if the car is parked correctly or not.

To consider the performance of this model we can consider different metrics:

TABLE V
RESNET18 [11] PERFORMANCES

Precision	Recall	Accuracy	F1-score
0.945	0.915	0.935	0.930

As these metrics are high, it can be said that the ResNet model can correctly classify between an illegal parking and a correct one.

Previously we have already talked about precision (2) and recall (3), let's also consider accuracy and F1-score.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

Accuracy is an evaluation metric that measures how often a model correctly predicts the target label, but this alone can be misleading especially if the data is unbalanced.

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{9}$$

The F1-score is a performance metric that combines precision and recall, it's computed as an harmonic mean between the two of them.

C. *Classification of empty spots*

Classification of the empty parking slots is done through a function that considers the color of the lines of the parking spaces, as described above. The test set consists of around 200 images; it's smaller than the others since we mainly collected it ourselves.

Let's consider some performance metrics:

TABLE VI
PERFORMANCE METRICS PER CLASS

Class	Accuracy	Precision	Recall	F1-score
Pay	0.84	0.69	0.84	0.75
Disabled	0.70	0.86	0.70	0.77
Pregnant	0.75	0.94	0.75	0.83
Normal	0.69	0.71	0.69	0.70

The performances are not very high because the color of the images and of the lines are heavily influenced by other factors such as: weather conditions, presence of other colored objects in the image, brightness, ... Being a rule-based approach, it's not able to generalize and requires a certain level of quality in the images to be able to discern the right colors, for example the parking lines are assumed to be clearly visible and not faded. As we already mentioned, we didn't find many images available for this task and we had to mainly gather them ourselves, for example from Google Earth; to increase the number of test samples we also employed data augmentation by modifying the colors and brightness of the images we did possess.

We also tried to build a simple convolutional neural network to classify images according to visual features like color:

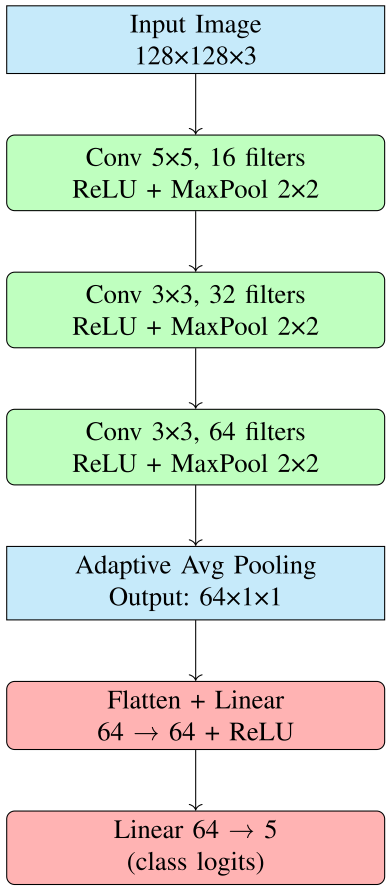

Fig. 17. Architecture of *SimpleColorCNN*

We trained this model with 150 images while the validation set and the test set consisted of 50 images each. The scarcity of the available data made it much more unreliable than the rule-based approach, let's compare the results:

TABLE VII
ACCURACY COMPARISON

Model	Accuracy
Rule-Based	0.751
Simple CNN	0.660

We can see that the performance is slightly worse but it's a much slower method and it also generalizes poorly because of the lack of data. If we had more images of empty parking slots with parking lines of different colors this would definitely be the preferred choice, but considering our situation and the results we decided to pick the rule-based approach.

V. CONCLUSION

This paper describes a system that assists in parking lot management by using different approaches assembled in a pipeline which has to:

- identify parking slots and classify them as either empty or occupied

- distinguish between occupied spots where a car is parked correctly and those where a vehicle is parked incorrectly
- recognize different types of empty parking spots according to the color of the parking lines

This might be useful in scenarios like the management of public or private parking lots. For example, it can help in finding transgressions of the parking guidelines and to promptly ask the drivers to move their cars. Additionally, by displaying the number of empty parking spaces together with their type, it can assist the incoming drivers that need to park. With this system, it's possible to increase the efficiency in the usage of a parking lot by detecting and reporting wrongly parked cars, but also to help the drivers by counting and then showing the number and kind of empty parking spots.

A. Future Improvements

In this system we only work with single still images; cars arriving, leaving or in general moving in the parking lot could be detected even though they shouldn't. A more refined approach could be achieved using videos as input and thus considering the frames as part of a sequence. Since our system has been trained with images of parked cars, it generalizes well enough such that it's also capable of recognizing other vehicles but it's not optimized for this task. If the parking lot is designed not just for cars but also, for example, motorcycles or trucks, to obtain better performances and generalize more it would be optimal to train the models with more images of different vehicles. Another issue is the use of a rule-based approach in the final classification of empty parking slots according to the color of the lines. This is light, fast and works pretty well, but comes with many limitations which we discussed previously. To achieve better performances we could employ more sophisticated approaches, like a classification model, but to do this we would need more data to train such a model like the CNN that we talked about earlier. Let's remember that we considered just four classes: normal parking, paid parking, disabled / reserved parking and parking for pregnant women, so to obtain better results and to represent the real scenario we'd need more images of empty parking spots of different types. Another improvement could be to check the type of parking slot according to the color of the parking lines even for occupied spots, furthermore we could check the presence of a disabled badge or a parking ticket on the windshield according to the respective type of parking. In order for a model to be able to recognize these tiny details we would need many high resolution images of parked cars from above with these characteristics.

References

[1] Caterina Belluti, Sara Gherardi, Federico Gualandri, "Github park project." https://github.com/sarag25/ParkProject.git.

[2] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection." https://www.sciencedirect.com/science/article/pii/S095741741630598X, 2017.

[3] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning." https://ieeexplore.ieee.org/abstract/document/7543901, 2016.

[4] "A systematic review on computer vision-based parking lot management applied on public datasets." https://www.sciencedirect.com/science/article/pii/S0957417422002032, 2022.

[5] N. Gonthina, S. Katkam, R. A. Pola, R. T. Pusuluri, and L. V. N. Prasad, "Parking slot detection using yolov8." https://ieeexplore.ieee.org/abstract/document/10435799, 2023.

[6] S. Deng, Y. Xia, W. Zuo, J. Xin, S. Zhou, and N. Zheng, "Parking space recognition methods based on visual image and deep learning." https://ieeexplore.ieee.org/abstract/document/10451942?casa_token=V0XI5MHkZZAAAAAA:oSSIYJG2bnCff_j3PDudsws7gC93xp3bzXOlKJWZepJ7-2Uw0xofdjMycln8qej3mEdrDhM, 2023.

[7] CNR Research Area of Bologna, "CNR area della ricerca di bologna." http://cnrpark.it/.

[8] Ammar Nassar Alhajali, "PKLot dataset." https://www.kaggle.com/datasets/ammarnassanalhajali/pklot-dataset.

[9] B. Dwyer, J. Nelson, T. Hansen, *et al.*, "Roboflow (version 1.0) [software]." https://roboflow.com/, 2024.

[10] "Yolov8: Real-time object detection." https://docs.ultralytics.com/it/models/yolov8/#overview, https://yolov8.com/.

[11] "Torchvision: ResNet-18." https://docs.pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html.

[12] "OpenCV: Canny." https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.

[13] "OpenCV: findContours." https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html.

[14] "OpenCV." https://github.com/opencv/opencv.