# Parking Space Classification Report

This report documents the process of developing a model to classify parking spaces as occupied or empty, based on a video of a parking space.

## Dataset Description

- Videos: Videos with the parking space and a clear view of the spots and their occupancy, including a longer video, a smaller version, and two cropped videos, all the same shots.
- Mask Images: Binary images containing a map of the parking spots.
- *Not  empty* Folder: Contains more than 3,000 images of occupied spots (for training purposes).
- *Empty* Folder: Contains more than 3,000 images of empty spots (for training purposes).

# Exploratory Data Analysis (EDA)

## Assumptions

Based on real world user experience, it was decided to consider that the final user would need to know in the video the spots that are occupied and free, flagging that visually and clearly, and having a count on the screen with the amount of spots occupied and empty.

## Project definition

In order to deliver the mentioned solution, we would need to:

1. Choose a video and a mask to work;
2. Extract some frames of the video to train our model;
3. Load the images on the folders (empty/not_empty) to train the model on which spots are occupied or empty;
4. Choose, define and train the model;
5. Find the contours in the mask to know the position of each parking slot;
6. Classify the parking spots in a frame;

7. Process the video frame by frame to get a final vision of the occupancy of each slot;

## Data selection and visualization

- Empty Images: Images of empty parking spots.
- Occupied Images: Images of occupied parking spots.
- Video: A video capturing the parking area.
- Mask: A binary mask image highlighting the parking spots in the video.

## Sample Visualization

The frame images captured from the video were resized to 64x64 pixels for uniformity, the mask image was used to identify and extract parking spots from each frame of the video, and the images available in empty and not_empty folders were loaded and concatenated and shuffled into a single folder.

## Normalization

The pixel values of images were normalized to the range [0, 1]. This helps stabilize and speed up the training process and guarantees that the data distribution is consistent across different models and training sessions. Also, it may improve the model to generalize better to unseen data.

## Label Encoding

Many machine learning algorithms and neural networks cannot directly work with categorical data in its raw form (empty or occupied classification for the parking slots). One-hot encoding transforms these categorical labels into a binary matrix representation that the models can understand. Also, one-hot encoding improves overall model performance and accuracy and ensures compatibility with various loss functions.

For this project, labels were one-hot encoded as follows: 0 for empty, 1 for occupied.

# Model Development

## Model selection

The model for this project would have to be a classification model, possibly binary (to classify whether a spot is empty or occupied). There were several that could fit, like Logistic Regression, Naive Bayes, Support Vector Machine (SVM), among others.

### Logistic Regression

Logistic Regression is a simple, yet powerful linear model used for binary classification tasks. It models the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead, or healthy/sick.

However, Logistic Regression assumes a linear relationship between the input features and the log odds of the output. Parking slot occupancy detection involves complex, non-linear patterns in the images which Logistic Regression cannot capture effectively. Therefore, it was probably not the best model for this problem.

### YOLO (You Only Look Once)

YOLO is a state-of-the-art object detection model that is fast and accurate. It frames object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities.

However, while YOLO is excellent for real-time object detection and could potentially work for parking slot detection, integrating YOLO into our existing pipeline would require significant changes in how features are extracted and processed. Additionally, YOLO has many specific parameters and complexities which would necessitate extensive adjustments to our current codebase.

### Support Vector Machine (SVM)

SVM is a supervised learning model that analyzes data for classification and regression analysis. It works by finding the hyperplane that best divides a dataset into classes.

However, SVM can be less effective for large datasets and might produce errors in complex image classification tasks. Many errors were encountered while trying to implement SVM, likely due to the high dimensionality and complexity of image data.

### Convolutional Neural Network (CNN)

CNNs are a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. CNNs are specifically designed to process pixel data and have proven to be highly effective in image classification tasks. They automatically detect important features without any human supervision and can handle the complexity and non-linearity in image data, making them well-suited for the task of classifying parking spots as occupied or empty.

## Model Architecture

A Convolutional Neural Network (CNN) was designed with the following layers:

- Conv2D: 32 filters, (3, 3) kernel, ReLU activation
- MaxPooling2D: (2, 2) pool size
- Conv2D: 64 filters, (3, 3) kernel, ReLU activation
- MaxPooling2D: (2, 2) pool size
- Conv2D: 128 filters, (3, 3) kernel, ReLU activation
- MaxPooling2D: (2, 2) pool size
- Flatten
- Dense: 128 units, ReLU activation
- Dropout: 50% dropout rate
- Dense: 2 units, Softmax activation

## Model Compilation

- Optimizer: Adam - chosen for its efficiency and less memory requirement.
- Loss Function: Categorical Crossentropy - suitable for multi-class classification problems.
- Metrics: Accuracy - chosen to evaluate the overall performance of the model.

### Model Training

- Epochs: 10 - a small number for initial training to quickly iterate and evaluate model performance.
- Batch Size: 32 - a standard size that balances the efficiency of training and model performance.
- Validation Split: 20% of training data used for validation

## Results and Evaluation

### Performance Metrics

- Validation Accuracy: Achieved a high accuracy of the model on the validation set.
- Test Set Metrics:
    - Accuracy: High accuracy indicating the model's effectiveness in classifying parking spots.
    - Precision, Recall, F1 Score: Further metrics to evaluate the model's performance in terms of true positives, false positives, and the balance between precision and recall.

      #### Model Evaluation Metrics:

      - Accuracy: 1.0
      - Precision: 1.0
      - Recall: 1.0
      - F1 score: 1.0

### Video Annotation

The result was a video generated upon the original video, where bounding boxes were designed in each of the parking slots (red for empty spots, green for occupied spots).

The video was saved with the bounding boxes and text annotations indicating the number of occupied and empty spots, and changing every time a car enters or leaves a parking slot.

# Further improvements and conclusions

## Further improvements

### Examine Misclassified Instances

One potential area for further improvement is to closely examine the instances where the model has misclassified parking spots. By analyzing these errors, we can identify specific scenarios that the model struggles with. For example, gray cars might be particularly challenging due to their lower contrast against the parking lot surface. Understanding these edge cases can help us refine the model or dataset to improve overall accuracy.

### Challenging Features

Another aspect to consider is the challenging features within the images. Poor lighting conditions, for instance, can significantly impact the model's ability to correctly classify spots as occupied or empty. Similarly, occlusions—where one vehicle partially blocks another—can lead to misclassifications. By identifying these challenging scenarios, we can look for ways to mitigate their impact, such as by enhancing image preprocessing techniques or collecting more diverse training data.

### Patterns and Peak Occupancy

Analyzing the temporal patterns of parking occupancy can also provide valuable insights. By monitoring how occupancy rates change over time, we can identify peak periods and potential bottlenecks. This analysis can help in optimizing parking space usage and improving overall traffic management within the parking area.

### Environmental Factors

Testing the model under different environmental conditions, such as varying weather conditions, can help assess its robustness. For example, the model's performance might differ between sunny and rainy days due to changes in lighting and visibility. By evaluating and adapting the model to perform well under diverse conditions, we can ensure its reliability in real-world applications.

### Feature Extraction with HOG

Another potential improvement involves extracting features from images using Histogram of Oriented Gradients (HOG). HOG is a powerful feature descriptor that captures edge information, making it effective for object detection. Incorporating HOG features into the model can enhance its ability to detect and classify parking spots, particularly in challenging conditions.

These areas of further improvement—examining misclassified instances, addressing challenging features, analyzing occupancy patterns, testing under different environmental conditions, and using advanced feature extraction techniques—can collectively help in refining the parking space classification model. By continuously iterating and improving the

model based on these insights, we can enhance its accuracy, robustness, and practical applicability in real-world scenarios.

Future work can also  include experimenting with more epochs, fine-tuning hyperparameters, and exploring more advanced architectures or transfer learning to further improve model performance.

## Conclusions

The CNN model demonstrated good performance in classifying parking spots as occupied or empty. The use of CNNs allowed us to effectively capture the complex patterns in the image data. Normalization and one-hot encoding were crucial preprocessing steps that contributed to the model's accuracy and generalization capabilities.

The chosen model architecture, with multiple convolutional layers and pooling layers, proved effective in feature extraction, while the dropout layer helped prevent overfitting. The Adam optimizer and categorical crossentropy loss function were appropriate choices for this classification task.