**Parking Space Detection**

**ECS 174  
Final Project  
Team Members: Athul Krishna Sughosh, Brian Dao, Rohan Kolappa**

**Abstract**

This project explores traditional computer vision techniques and machine learning models for a parking space detection system. The goal is to classify parking spaces as occupied or vacant in real-time from video footage. Initially, we used Convolutional Neural Networks (CNNs) trained on an augmented dataset. However, due to inefficiencies in prediction time for larger parking lots, we shifted to using traditional computer vision techniques with OpenCV, including adaptive thresholding and white pixel counting within predefined bounding boxes.

### **1. Introduction**

Efficient parking management is crucial in urban areas due to the rise in vehicles and limited parking spaces. Current methods often use costly hardware and can be inaccurate. This project aims to create a low-cost, robust parking space detection system using computer vision. We explore traditional computer vision techniques and machine learning models to find an accurate and efficient real-time solution. Effective parking management reduces congestion, search time, fuel consumption, and emissions, improving user experience and minimizing environmental impact. Our project offers a scalable solution for existing parking infrastructure.

### **2. Dataset Description**

* [PKLot Dataset](https://www.kaggle.com/datasets/ammarnassanalhajali/pklot-dataset): The PKLot dataset contains 12,416 images of parking lots extracted from surveillance camera frames. Images are split into train, validation, and test sets. The spaces are annotated in COCO format with bounding boxes and if the parking spot is occupied or not.
* [Parking Dataset](https://www.kaggle.com/datasets/mfaisalqureshi/parking): Annotated images of parking spaces indicating occupancy.
* Live Video Parking Lot Feed: 1920 x 1080 bird’s eye view feed of a parking lot.
* Custom Parking Dataset: Using a python script ParkingLotConfigure.py, we created a custom dataset by cropping out 69 x 30 images of occupied and vacant spots from the “Live Video Parking Lot Feed” to train the CNN. This dataset contains 1605 occupied parking space images and 400 vacant space images.

### **3. Methodology**

**PKLot Dataset:**The following pre-processing was applied to each image:  
 \* Auto-orientation of pixel data (with EXIF-orientation stripping)  
 \* Resize to 640x640 (Stretch)  
The model is trained based on cropped images using the annotated bounding box dimensions to predict is a space is empty or not. The model architecture is as follows:

* 2 Convolution Layers using ReLU as its activation function.
* 2 Pooling Layers
* 2 Fully Connected layers and an output layer

The hyperparameters for this model are:

| **Epochs** | 10 |
| --- | --- |
| **Batch Size** | 10 |
| **Optimizer** | Adam |
| **Loss** | Categorical Cross Entropy |
| **Metrics** | Accuracy |

These hyperparameters were chosen based on multiple rounds of tests and trials. The initial Sequential model was simplified to account for Google Colab’s RAM and GPU limitations. The images had to be resized to improve training time, the number of epochs and batch size were reduced as well.

**Custom Parking Dataset:**Initial Configuration  
The configuration process involved defining parking space boundaries using ParkingLotConfigure.py. This script allows users to manually set up bounding boxes for the parking lot by specifying the coordinates of parking spaces through mouse click (for regular spaces), scroll wheel click (for disabled spaces), and right mouse click to delete a box and saving these configurations for future use.

Data Preprocessing and Augmentation  
We applied normalization, horizontal flipping, shear transformation, and random zooming to improve model robustness. This helps the model generalize better by exposing it to various transformations of the original images. We set aside 20% of the dataset to validate and monitor model performance.

The model architecture used for this dataset is as follows:

* 3 Convolutional layers using ReLU activation function to add non-linearity to the network
* 2 Max Pooling layers to reduce spatial dimensions of the previous layer
* 1 Flatten layer to flatten the output of the previous layer to a one-dimensional array
* 1 Dense layer using ReLU
* 1 50% Dropout layer to regularize and prevent overfitting
* 1 Output dense layer with a single neuron using the sigmoid activation function for binary classification

The hyperparameters for this model are:

| **Epochs** | 50 |
| --- | --- |
| **Batch Size** | 32 |
| **Optimizer** | Adam |
| **Loss** | Binary Cross Entropy |
| **Metrics** | Accuracy |

Since our task at hand is a binary classification task (occupied or vacant parking space), we chose binary\_crossentropy as our loss function.

**Traditional Computer Vision Approach:**This approach is integrated into the main loop of the program and performs the preprocessing and occupancy detection in real-time, frame by frame. In the field, we aim to have the occupancy detection slightly faster than how long it would take to park a car. So we could reduce the detection to occur every 4-5 seconds instead of frame-by-frame, thus reducing computational load for the larger lots.

To compensate for camera movement in the footage, we dynamically adjusted the position of the bounding boxes in the video frames based on the time elapsed. We were able to figure out the points in time where the camera adjusts because the video is looped forward and in reverse every 28 seconds. In the real world, we assume that CCTVs are installed in place and are not supposed to move.

Preprocessing

We convert each frame to grayscale to reduce processing and apply adaptive thresholding to the grayscaled frames to segment the parking lot into binary regions. The distinguishing feature between occupied and vacant spaces is pixel intensity. Vehicles have higher pixel intensity compared to the asphalt vacant parking spaces and are thus classified as occupied.

Here are the hyperparameters of the adaptive threshold:

| **adaptiveMethod** | Adaptive Threshold Gaussian: This method calculates the threshold value for each pixel by using a weighted sum of nearby pixel values using a Gaussian window. We use this method over others as it is apt for dealing with varying lighting conditions. |
| --- | --- |
| **blockSize** | 53: the size of the neighborhood area used to calculate the threshold value for each pixel. |
| **C** | 30: constant subtracted from the mean of nearby pixels to compensate for variations in lighting. |
| **thresholdType** | Threshold Binary Inverse: pixels with intensities above the calculated threshold will be set to 0 (black) while pixels below are set to 255 (white). |

Occupancy Detection

Through fine-tuning, we found that a threshold of 250 was apt to distinguish between an occupied and a vacant space. We iterate through each frame of the video and extract the regions of interest (i.e. the binary frame based on the coordinates from the bounding box), calculate the number of white pixels, and classify them as occupied if there are more than 250 white pixels in the space and vice versa.

### **4. Experimental Results and Model Performance**

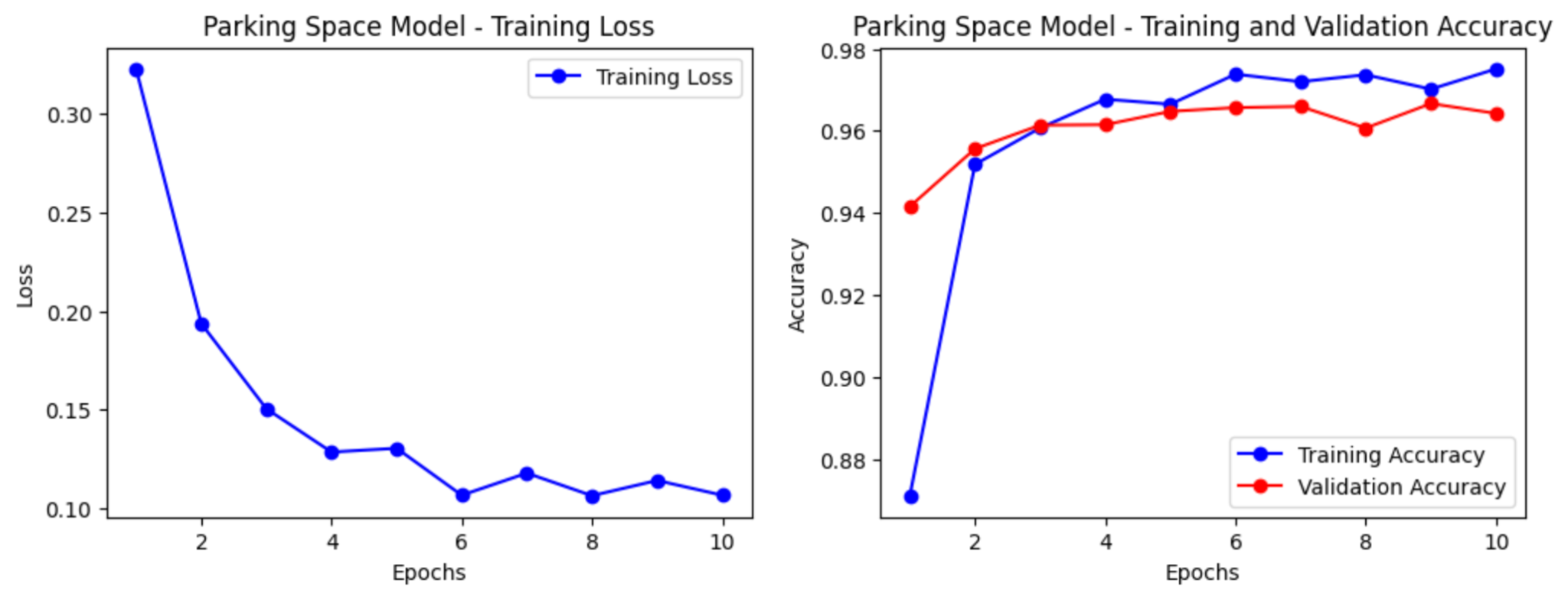
**PKLot Dataset:**

Fig 1. CNN Model Performance - PKLot Dataset

Test Accuracy: 99.84%

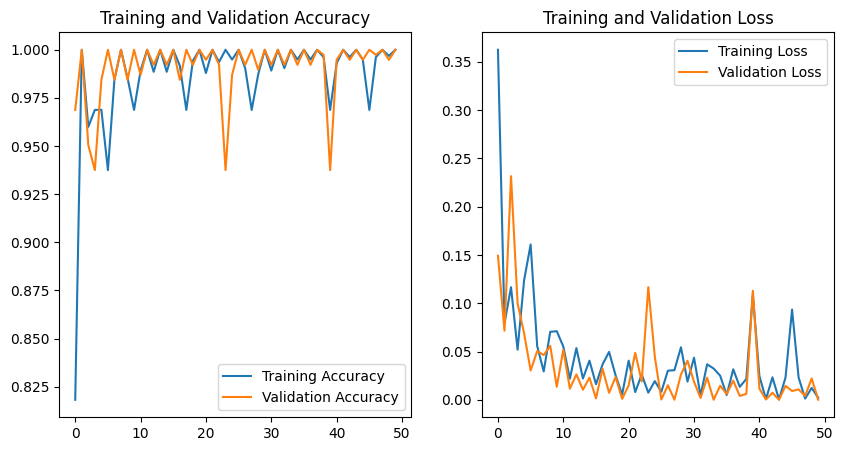
**Custom Parking Dataset from Video:**

Fig 2. CNN Model Performance - Custom Parking Dataset Test Accuracy: 99.74%

**Traditional Computer Vision Approach:**



Fig 3. Binary Frame after Adaptive Thresholding

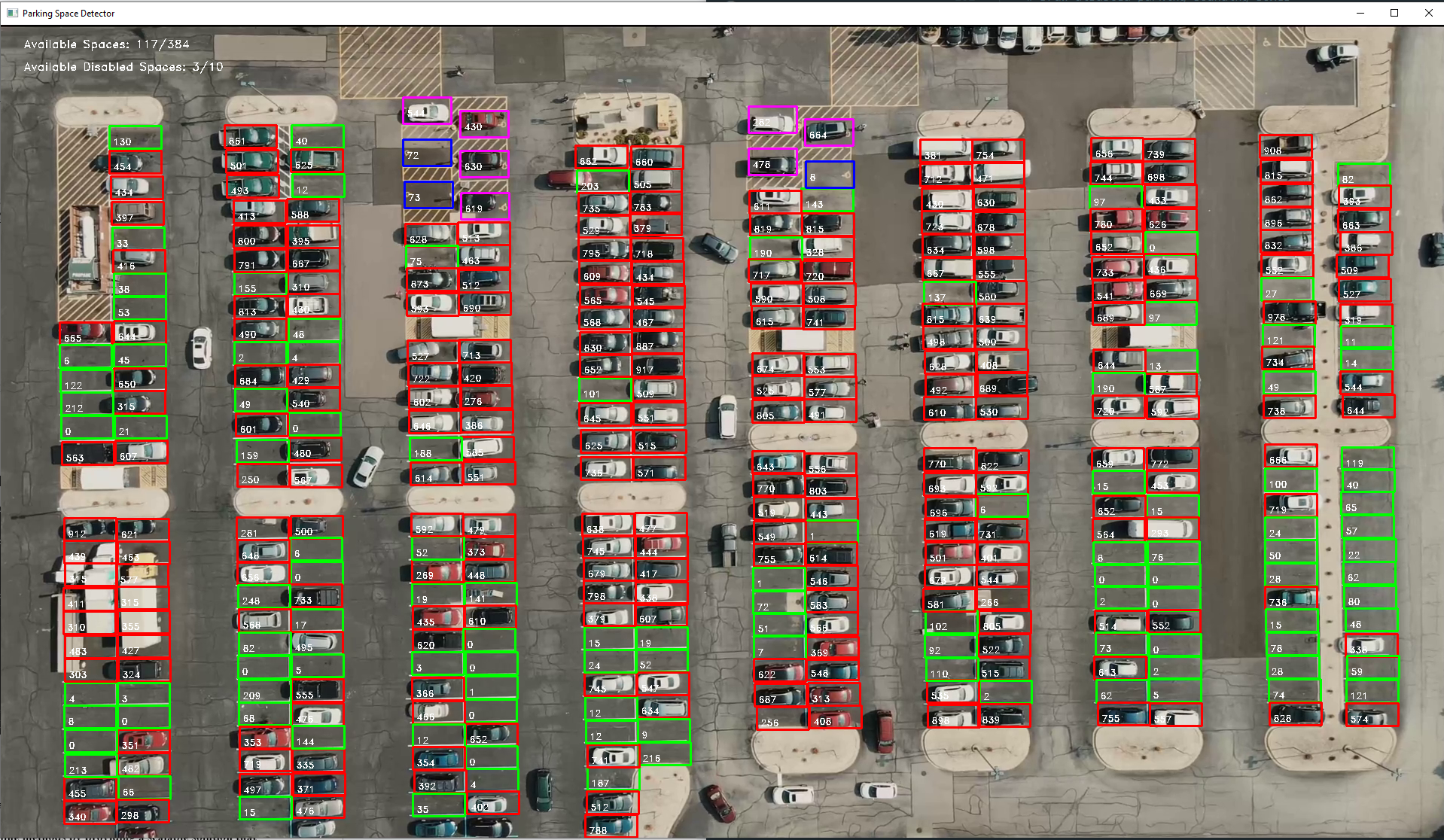


Fig 4. Parking Lot Feed Overlayed with Occupancy Detection

### **5. Discussion**

One major challenge in developing our parking space detection system was the impractical prediction time per space, especially in large parking lots, using the machine learning approach. Although the CNN model achieved high accuracy in classifying parking space occupancy, it required about 25 milliseconds per space. With 384 spaces, this totals approximately 9.6 seconds to detect occupancy. Our aim is to run occupancy detection faster than the average time it takes to park, making this solution unfeasible. The traditional computer vision approach performed accurate occupancy detection for the entire lot every frame, achieving 30 frames per second. This results in an efficiency gain of (9.6 seconds / 0.0333 seconds) \* 100% = 288 times. The traditional method not only proved more efficient but also balanced accuracy and scalability for real-world applications. Other challenges, such as diverse lighting conditions and occlusions, were addressed through data augmentation and model tuning. System limitations were accounted for by adjusting hyperparameters, resizing images, and reducing model complexity.

### **6. Contributions**

* Athul Krishna Sughosh: ML Model using PKLot dataset, Report
* Brian Dao: ML Model on CCTV video and Traditional CV Approach, Report
* Rohan Kolappa: Model Evaluation, Report, Slides

### **7. Conclusion**

Despite the CNN's high accuracy, its prediction time per space was impractical for larger parking lots. Therefore, the traditional computer vision approach was chosen for its efficiency and real-time performance. This method reliably detected parking occupancy under various conditions, providing an effective solution for automated parking management.

Our project successfully develops a scalable and efficient parking space detection system using computer vision. Future work includes incorporating more diverse datasets and enhancing model robustness to function under different conditions such as poor video quality, low lighting, and longer videos. This project addresses current inefficiencies in parking management and provides a foundation for future research and development in smart parking solutions.

### **References**

* Datasets:
  + [PKLot Dataset](https://www.kaggle.com/datasets/ammarnassanalhajali/pklot-dataset)
  + [Parking Dataset](https://www.kaggle.com/datasets/mfaisalqureshi/parking)
  + [Parking Video](https://drive.google.com/drive/folders/1CjEFWihRqTLNUnYRwHXxGAVwSXF2k8QC?usp=sharing)
* Code Repository:
  + [GitHub: Parking-Space-CV](https://github.com/daobrian/Parking-Space-CV)