

# Deepot: Parking Lot Identification Using Low-Resolution Satellite Imagery

Xianzhong Ding  
Lawrence Berkeley  
National Laboratory  
[dingxianzhong@lbl.gov](mailto:dingxianzhong@lbl.gov)

Wanshi Hong  
Lawrence Berkeley  
National Laboratory  
[wanshihong@lbl.gov](mailto:wanshihong@lbl.gov)

Zhiyu An  
University of  
California, Merced  
[zan7@ucmerced.edu](mailto:zan7@ucmerced.edu)

Bin Wang  
Lawrence Berkeley  
National Laboratory  
[wangbin@lbl.gov](mailto:wangbin@lbl.gov)

Wan Du  
University of  
California, Merced  
[wdu3@ucmerced.edu](mailto:wdu3@ucmerced.edu)

## Abstract

*Detecting parking lots is essential for siting electric truck charging infrastructure, ensuring convenient access for drivers. However, the private ownership of many parking lots limits public access to information for planners. Current object detection methods rely heavily on high-resolution satellite imagery, which is often restricted in availability. To address this, we introduce Deepot, a deep learning-based truck parking location identification approach using low-resolution satellite imagery. We begin by retrieving geographical data for the target area and optimizing image resolution for model training. The model produces initial parking lot locations, which are then converted to real global latitude and longitude coordinates using a custom coordinate reference systems transformer. These global coordinates facilitate querying Google Maps for high-resolution images to enhance detection performance. Deepot streamlines charging infrastructure planning from diverse satellite imagery and offers additional, high-fidelity candidate locations to medium- and heavy-duty infrastructure planning tool, HEVI-LOAD. Extensive experiments show the effectiveness of Deepot.*

**Keywords:** Parking lot detection, low-resolution satellite imagery, electric truck charging infrastructure, remote sensing, geospatial analysis, systems theory

## 1. Introduction

In recent years, the adoption of electric trucks in logistics has been accelerated due to their promise of reduced emissions, increased efficiency, and sustainability. As this transformation gains momentum, a critical challenge emerges: the strategic planning of electric truck charging infrastructure. Charging stations

are essential for keeping the electric truck ecosystem operational and for ensuring convenient recharging options for electric truck drivers. However, determining the optimal locations for charging stations is a complex task. A significant roadblock in this process is the lack of information regarding all existing truck parking lots. This restriction complicates the planning process for charging stations and creates an imperative need for innovative solutions to make electric truck charging more accessible and efficient. One potential solution is using object detection techniques on satellite images to identify parking lots.

Object detection (Liu et al., 2020, 2023) plays a crucial role across various domains, including environmental research, where it is vital for locating objects in satellite imagery. Current object detection methods rely on high-resolution satellite image datasets. These high-resolution datasets are limited in availability and scope, posing a significant obstacle for researchers who require precise and comprehensive data for accurate detection and analysis. Despite advances in semi-supervised learning and oriented object detection techniques, these methods are typically tailored to general object detection tasks and do not effectively address the specific challenges posed by low-resolution satellite imagery in aerial scenes. Existing high-resolution datasets, such as DeepGlobe and SpaceNet, focus on building and road detection but are not optimized for parking lot detection using low-resolution imagery (Demir et al., 2018; Van Etten et al., 2018). Furthermore, the limited availability of annotated datasets for parking lot detection, primarily composed of camera view images, complicates the task even further (De Almeida et al., 2015).

To address these challenges, we introduce Deepot, a novel approach for detecting parking lots using public low-resolution satellite imagery. Deepot begins by retrieving geographical data for the target area and

optimizing the image resolution for model training. With the low-resolution satellite imagery dataset, the primary challenge is the poor object detection performance due to the diverse visual appearance and varying sizes of parking lots. Our model tackles this challenge by leveraging both low-resolution and high-resolution imagery in a two-stage detection process. Initially, the model identifies parking lots in low-resolution images, which are then converted into real global latitude and longitude coordinates using a custom coordinate reference systems transformer. These global coordinates are used to query Google Maps for high-resolution images, which enhance detection performance through additional object detection.

We conduct extensive experiments to validate Deepot's performance. Our evaluation of the charging load demonstrates that detecting more parking lots with our method significantly reduces the overall charging energy required. By integrating the Medium and Heavy-Duty Electric Vehicle Infrastructure Load Operations and Deployment (HEVI-LOAD) tool, we show that our framework can seamlessly identify additional candidate locations for medium- and heavy-duty infrastructure planning. This integration improves the fidelity of simulation analysis and supports more efficient planning of charging infrastructure.

We summarize the main contributions as follows:

- We present the first dataset specifically designed for parking lot detection using low-resolution satellite imagery. This dataset fills a critical gap in the existing datasets, which are high-resolution and not optimized for this specific application.
- We propose Deepot, a two-stage framework that combines low-resolution satellite imagery for initial detection with high-resolution image verification. This method enhances detection performance by leveraging the strengths of both low- and high-resolution images.
- We conduct extensive experiments to validate the performance of Deepot. Our results demonstrate that Deepot outperforms baseline methods in terms of precision, recall, F1 score, and average precision, showcasing its effectiveness.
- Our framework can seamlessly integrate with the HEVI-LOAD tool, providing additional candidate locations for medium- and heavy-duty infrastructure planning. This integration enhances the fidelity of simulation analysis and supports more efficient charging infrastructure planning.

## 2. Related Work

**Aerial and Satellite Images Datasets.** Geoinformation extraction from aerial and satellite images has gained popularity in recent years. DeepGlobes (Demir et al., 2018) and SpaceNet (Van Etten et al., 2018) are two popular high-resolution satellite imagery datasets for building detection, road detection, and land cover classification. However, to the best of our knowledge, the datasets related to parking lot detection are mostly composed of camera view images only (De Almeida et al., 2015). Take CNRPark+EXT (Amato et al., 2017) as an example, this dataset, though consisting of 144,965 images, is collected by 9 fixed cameras covering a small parking area only. APKLOT (Hurst-Tarrab et al., 2020) is the only satellite imagery-based parking-lot dataset, which contains 500 images with varying sizes and resolutions. Grab-Pklot is a high-resolution satellite imagery dataset with 1344 images for parking lots.

**Deep Learning Models and Feature Extraction in Modern GIS Tools.** Modern GIS tools like ArcGIS Pro and ArcGIS Online <sup>1</sup> integrate advanced deep learning models (Ding et al., 2024; Zhang et al., 2024) and feature extraction techniques essential for geospatial analysis. Key models include Faster R-CNN, YOLO, and SSD for object detection, vital for identifying structures in satellite imagery (Ren et al., 2016). For semantic segmentation, tools like U-Net and DeepLab facilitate detailed land cover classification (Chen et al., 2017), while Mask R-CNN aids in distinguishing overlapping objects through instance segmentation (He et al., 2017). Feature extraction techniques such as edge detection and Gray Level Co-occurrence Matrix enhance classification accuracy by delineating and analyzing land cover types (Haralick et al., 1973). Object-Based Image Analysis segments imagery based on attributes like shape and texture (Blaschke, 2010). These capabilities in GIS tools complement traditional methods, making them critical for urban planning, environmental monitoring, and infrastructure development. Unlike GIS methods using high-resolution imagery, Deepot handles low-resolution data, integrating geographic information for real-world coordinates. It focuses on parking lot detection, distinct from general-purpose GIS models.

## 3. Motivation

In this section, we conduct experiments to understand the current limitations in the number of parking lots and motivate the use of deep learning-based object detection methods to identify

---

<sup>1</sup><https://pro.arcgis.com/>

Location Type	Energy (GWh)	Percentage (%)
Depot	94.51	65.76
Public	25.44	17.73
Enroute	23.61	16.50

Table 1: Summary of energy usage by location type

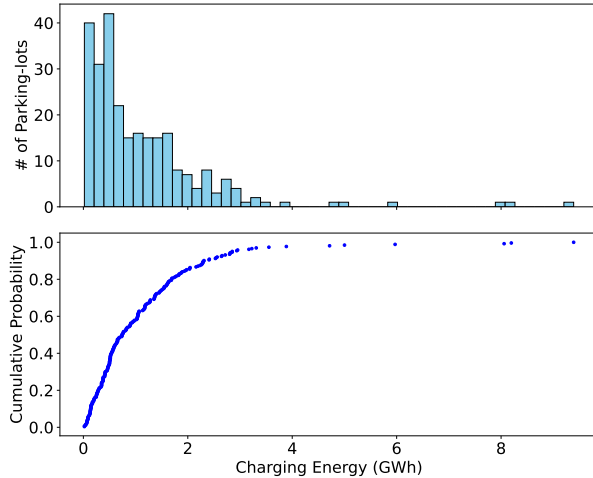


Figure 1: Charging energy distribution and CDF for 308 public parking lots.

more parking lots. Our experiments analyze the charging load of existing parking lots across California using the HEVI-LOAD tool. The HEVI-LOAD tool serves as a comprehensive solution to project infrastructure needs for the decarbonization of medium and heavy-duty vehicles, offering valuable insights into optimizing charger assignments and ensuring efficient operation of electric HDVs. This tool introduces an agent-based simulation method that integrates micro and macro-scale analyses. Not only does it address existing gaps in modeling capabilities, but also offers a tangible means to illustrate and demonstrate potential scenarios to key stakeholders and policymakers, facilitating more informed decision-making for a future shaped by electrified and automated freight transportation.

### 3.1. HEVI-LOAD and Experiment Setting

The HEVI-LOAD framework employs an agent-based simulation methodology. This approach models the trips and duty cycles of medium- and heavy-duty (MHD) zero-emission vehicles (ZEVs) and generates charging load profiles and infrastructure assessment with varied resolutions - at the site (location), traffic analysis zone (TAZ), county, state, freight corridors, and the national scale. It projects infrastructure deployment and operational needs,

including charging/refueling station type, quantity, and strategic locations for future ZEVs weighing more than 10,000 lbs (LBNL, 2024; Wang et al., 2024).

We consider the current scenario in HEVI-LOAD, which has 308 public parking lots in total across California. Using the HEVI-LOAD tool, we simulate 318,972 electric vehicles running in California for three days and calculate the total charging load required for each existing parking lot to meet the load requirements. There are three types of parking lots: Public parking lots, Depot parking lots which are private parking lots belonging to private users, and En-route parking lots, which are used when a vehicle cannot reach the next destination due to low remaining energy, triggering the rerouting algorithm in the HEVI-LOAD tool to find nearby public parking lots.

### 3.2. Results Analysis

Table 1 shows the total energy consumed by all vehicles. From this table, we see that vehicles consume 94.51 GWh at depot, 25.44 GWh at public parking lots, and 23.61 GWh at en-route parking lots. This accounts for 65.76%, 17.73%, and 16.50% of the total energy, respectively. The results indicate that due to the limited number of public parking lots, depots account for a large portion of the energy consumption, which places a significant burden on private parking lots. Public and en-route parking lots combined only account for 34.23% of the total energy. Figure 1 shows the charging energy distribution across these 308 public parking lots. From this figure, we observe that these results indicate a substantial concentration of parking lots with relatively high charging energy requirements. Specifically, the average charging energy for these parking lots is 1.12 GWh. 42.21% of the parking lots have a charging energy greater than 1 GWh, and 14.83% of the parking lots have a charging energy greater than 2 GWh. This distribution suggests that the current infrastructure has limited capacity to handle higher demands efficiently. The average charging energy of 1.12 GWh per parking lot represents a significant burden on each public parking lot. The analysis reveals a significant limitation in the current number of available public parking lots.

## 4. Design of Deepot

In this section, We describe the design of Deepot, including Low-Resolution Dataset Creation and High-Resolution Image Verification.

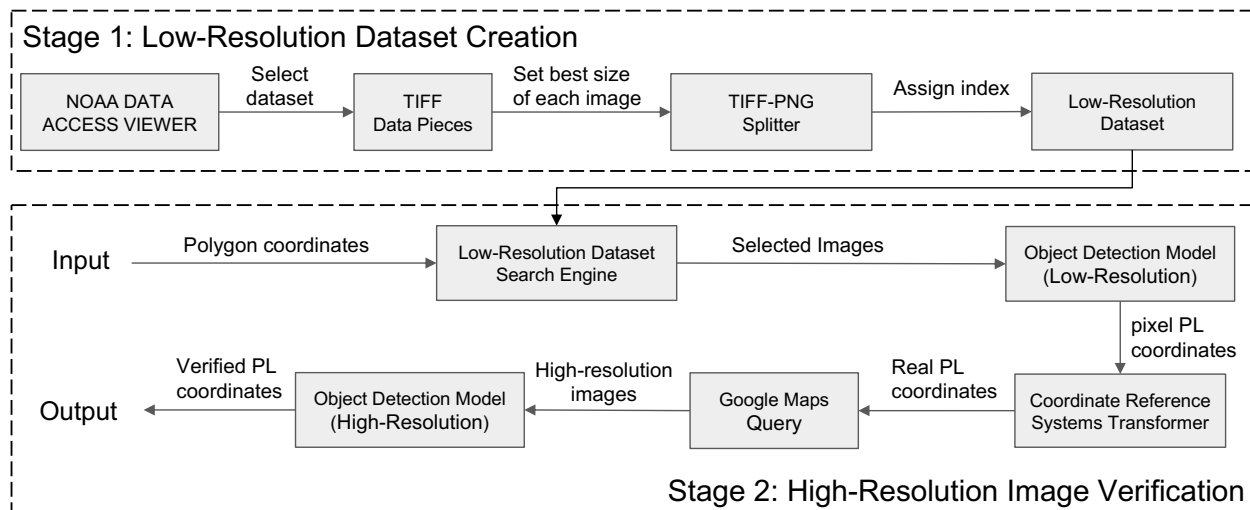


Figure 2: Overall of the Proposed Deepot Framework

#### 4.1. Deepot Overview

Figure 2 shows the data collection framework of Deepot, which consists of two stages. In **Stage 1: Low-Resolution Dataset Creation**, Deepot begins by acquiring geographic data for the target area from the NOAA Data Access Viewer. The acquired data is processed, and the best size for each image is determined. The images are then split into smaller pieces and converted to PNG format. An index is assigned to each sub-image, resulting in the Low-Resolution Dataset. This dataset is used to train the object detection model. In **Stage 2: High-Resolution Image Verification**, the user provides the specified geographic area defined by a polygon with latitude and longitude coordinates. The Low-Resolution Dataset Search Engine generates a dataset of sub-images that covers the specified area. The Object Detection Model detects parking lots (PL) coordinates in these images and outputs pixel coordinates. The Coordinate Reference Systems Transformer converts the pixel coordinates to initial real-world PL coordinates. To improve detection performance, these coordinates are used to query high-resolution images from Google Maps. These high-resolution images are then fed into the Object Detection Model again to obtain verified parking lot coordinates.

#### 4.2. Low-Resolution Dataset Creation

As shown in the top part of Figure 2, we illustrate our dataset selection process using the NOAA Data Access Viewer by inputting the location name. Given

the multitude of available datasets for each location (e.g., 77 datasets for California), we carefully select the most suitable one based on specific attributes detailed in Table 2. Our selection criteria prioritize high accuracy, detailed resolution, and comprehensive information, supported by substantial computing resources. We aim to balance these attributes, as some datasets excel in certain areas (e.g., cell size) but fall short in others (e.g., number of bands). Our computing resources enable us to handle more data-intensive datasets, facilitating thorough and detailed analysis.

We proceed to select an appropriate dataset containing numerous TIFF (Tagged Image File Format) files, each representing large image dimensions images of our location of interest. These TIFF files are replete with rich spectral data, essential for detailed analysis. However, our object detection model operates on JPEG or PNG formats, necessitating the conversion of TIFF files into PNG. This conversion often results in extremely large image dimensions, as seen with the California dataset which has dimensions of 12,084x12,084 pixels. Despite the large dimensions, the quality of these images is not optimal.

To mitigate this, we introduce a 'best-size' parameter, which aids in splitting the original converted data into more manageable segments. We employ a grid search method on a test dataset to determine the optimal 'best-size' parameter, aiming to maximize the accuracy of our object detection model. Since all TIFF files in a dataset typically share similar attributes, we apply the same 'best-size' parameter across the board. Subsequently, we divide all large image dimensions data into segments based on this defined 'best-size',

Attribute	General Description
Cell Size (m)	Indicates the spatial resolution of the image; smaller cell size means higher resolution.
Radiometric Resolution (bit)	Describes the radiometric resolution, representing the levels of intensity a pixel can show.
Sensor	Specifies the type of sensor used to capture the image.
Horizontal Accuracy (cm)	Refers to the precision of the location data within the image.
Number of Bands	The number of different spectral bands the image contains, used for various analyses.
Tide Controlled	Specifies whether the timing of image capture was adjusted for tidal conditions.
Date	The date when the dataset was created or the image was captured.

Table 2: General Description of Satellite Image Parameters

culminating in the creation of the Low-Resolution Dataset. As a result, from the Low-Resolution Dataset Creation stage, we created a vast dataset consisting of 8.73 TB, 11,068 TIFF files, and 1,487,163 PNG images covering the whole of California.

### 4.3. High-Resolution Image Verification

In this section, our objective is to improve the detection performance of the object detection model while simultaneously enhancing data quality.

#### 4.3.1. Low-Resolution Dataset Search Engine

During the model deployment stage, the user provides the specified geographic area defined by a polygon with latitude and longitude coordinates. These can be obtained by drawing a polygon over the area of interest in Google Earth and exporting the coordinates. The coordinates are then used as input to the Low-Resolution Dataset Search Engine.

The Low-Resolution Dataset Search Engine algorithm is designed to efficiently retrieve subimages within a specified geographic polygon from a vast dataset. The core of the algorithm leverages JSON metadata files, which contain coordinates of subimages, to quickly determine their geographic relevance without processing the entire image files. This significantly reduces I/O operations, as the algorithm only needs to check the metadata for each subimage. By using the `shapely` library for geometric operations, such as checking if a point lies within a polygon, the algorithm ensures that spatial queries are handled efficiently. Additionally, the selective copying of only the subimages that meet the geographic criteria minimizes the amount of data transferred, making the process faster and more efficient.

The algorithm traverses the directory structure to locate metadata files and uses a sanitization function to ensure that file names are safe for use in file paths. The `copy_files` function efficiently manages the copying of both the subimage and its corresponding

TIFF file to the result directory, maintaining the directory structure. By focusing on metadata first and copying files conditionally, the algorithm streamlines the search process and handles large numbers of files effectively. This approach, combined with efficient file management and optimized geometric operations, allows the Low-Resolution Dataset Search Engine to handle large datasets with minimal overhead, ensuring that relevant subimages can be retrieved and copied quickly and accurately.

#### 4.3.2. Coordinate Reference Systems Transformer

The Object Detection Model utilizes relevant subimages from the Low-Resolution Dataset Search Engine to detect parking lots. The initial detection results provide pixel coordinates of the subimages corresponding to the local areas of interest. To convert the pixel coordinates of Subimage to Global location for a whole place, we use Equation 1 for Subimage to Global Pixel Conversion. In this equation,  $g_x, g_y$  represents the global x and y pixel coordinates.  $idx$  is The subimage index in the form of a tuple (e.g.,  $(idx[0], idx[1])$ ).  $s_w, s_h$  is the width and height of the subimage, respectively.  $s_x, s_y$  The x and y pixel coordinates detected the result of the Initial Object Detection Model within the subimage.

$$\begin{pmatrix} g_x \\ g_y \end{pmatrix} = \begin{pmatrix} idx[0] \times s_w + s_x \\ idx[1] \times s_h + s_y \end{pmatrix} \quad (1)$$

$$e, n = rasterio.transform.xy(tf_obj, g_y, g_x) \quad (2)$$

$$tfn = Transformer.from_crs(src_crs, dst_crs) \quad (3)$$

$$lon, lat = tfn.transform(e, n) \quad (4)$$

To extract geospatial information from global pixel coordinates, we initially apply Equation 2. This equation is used to transform global pixel coordinates into easting and northing coordinates, a Cartesian coordinate format, utilizing the georeferencing data embedded in the TIFF file. Coordinate Reference

Systems (CRS) are essential in this context, as they delineate the mapping of 2D image coordinates to actual geographic locations.

Subsequently, we engage Equation 3 to facilitate the conversion from the Source CRS (*src\_crs*) to the Target CRS (*dst\_crs*). In our scenario, the *src\_crs*, defined as "EPSG:32610", corresponds to the North American Datum 1983 (NAD83) UTM Zone 10N system, a CRS typically employed within the TIFF image. Conversely, the *dst\_crs*, represented by "EPSG:4326", aligns with the World Geodetic System 1984 (WGS 84), a globally recognized coordinate system.

The transformer object is created using the `pyproj.Transformer.from_crs` method, which is a common approach for transforming coordinates between different CRS. The `pyproj` library<sup>2</sup> is widely used for such geospatial transformations. In this context, we use "EPSG:32610" as the source CRS (North American Datum 1983, UTM Zone 10N) and "EPSG:4326" as the destination CRS (World Geodetic System 1984), specifically chosen for California. This regional specificity is crucial, as different locations may require different CRS pairs. Finally, Equation 4 converts the Cartesian coordinates into actual latitude and longitude coordinates for parking lots. This conversion is facilitated by the `pyproj` transformer, which accurately reconciles differences between the source and destination CRS. The process also leverages `rasterio`<sup>3</sup>, a library for reading and writing geospatial raster data, to manage and extract geospatial information from the images.

**4.3.3. Google Map Querier** In the context of object detection results obtained from the Coordinate Reference Systems Transformer, we have developed Google Map Querier to generate high-resolution images using the Google API. One challenge we face is that the Google API has limitations on frequent user visits. To address this issue, we first calculate the center point of the bounding box using the average of the extreme latitudes and longitudes of the detected objects. The center latitude ( $Lat_C$ ) is calculated as  $\frac{Lat_N + Lat_S}{2}$ , where  $Lat_N$  is the northernmost latitude and  $Lat_S$  is the southernmost latitude. Similarly, the center longitude ( $Lon_C$ ) is calculated as  $\frac{Lon_E + Lon_W}{2}$ , where  $Lon_E$  is the easternmost longitude and  $Lon_W$  is the westernmost longitude.  $Lat_C$  is the average of the northernmost ( $Lat_N$ ) and southernmost ( $Lat_S$ ) latitudes, representing the vertical center of the bounding box. Similarly,  $Lon_C$  is the average of the easternmost ( $Lon_E$ ) and westernmost ( $Lon_W$ ) longitudes, representing the

horizontal center.

Then we use the Equation 5 to calculate the zoom level needed to encompass the entire bounding box within a single image. The zoom level is determined based on the distance between the furthest points and the desired resolution. The notations used are as follows:  $Lat_N$  is the northernmost latitude among all detected objects,  $Lat_S$  is the southernmost latitude,  $Lon_E$  is the easternmost longitude,  $Lon_W$  is the westernmost longitude,  $Lat_C$  is the center latitude of the bounding box (calculated as the average of  $Lat_N$  and  $Lat_S$ ), and  $Lon_C$  is the center longitude (calculated as the average of  $Lon_E$  and  $Lon_W$ ).

$$Z = f(\text{Distance}(Lat_N, Lon_N, Lat_S, Lon_S), \text{Resolution}) \quad (5)$$

In this formula,  $f$  is a function that correlates the maximum distance between the detected objects and the required map resolution to compute the appropriate zoom level. The distance is calculated using the coordinates of the extreme points. These calculated coordinates, along with the required image size and zoom level, serve as inputs for the Google API, enabling it to generate high-resolution images that meet the specified requirements. It's important to note that both the low-resolution images selected from the Low-Resolution Dataset Search Engine and the high-resolution images reference the same geographical location for a given image. This alignment is achieved because the images are generated using the geographical information, such as latitude and longitude, provided by the dataset. Subsequently, the high-resolution images generated by the Google Map Querier are fed into the Object Detection Model again to output the verified parking lot coordinates in JSON format.

## 5. Implementation

In this section, we illustrate in detail the implementation of Deepot including dataset preparation, Deepot training, and tuning hyper-parameters.

**Dataset Preparation.** We utilize low-resolution satellite imagery datasets enriched with detailed annotations, specifically marking the locations of parking lots using bounding boxes. This meticulous annotation process ensures the utmost precision and consistency throughout the datasets. As part of our data preparation pipeline, we normalize the input images by adjusting their pixel values to have a mean of zero and a variance of one. This normalization step plays a crucial role in aiding the convergence of our models. Furthermore, we apply data augmentation techniques to enhance the datasets and improve the generalizability of

<sup>2</sup><https://pyproj4.github.io/pyproj/stable/>

<sup>3</sup><https://rasterio.readthedocs.io/en/stable/>

our models. These techniques include random rotations, scaling, and horizontal flipping, all of which contribute to the robustness and versatility of our research in satellite image analysis. The satellite images parking-lot annotations in our dataset have a size of 1024 x 1024 pixels with a ground sampling distance of 0.25 meter/pixel. To train parking-lot detection models, we further divide the dataset into a training set of 2000 samples, and a testing set of 400 samples. The images in the training and testing sets are labeled with manually refined high-quality parking-lot annotations.

**Deepot Training.** Deepot is implemented using the PyTorch framework (Ding and Du, 2024). We use the AdamW optimizer with a fixed learning rate of  $1e - 4$ , which decreases by a factor of 10 every 10 epochs in our current training runs. This adaptive learning rate scheme ensures optimal model convergence. In our current training sessions, we employ a batch size of 16, striking a balance between computational efficiency and convergence rate, allowing our model to learn effectively while making efficient use of available resources. Our model is currently trained for 200 epochs, and we have a mechanism in place for early stopping. If the validation loss fails to improve for 5 consecutive epochs during our ongoing training, early stopping is triggered. This real-time safeguard actively monitors the model’s performance to ensure that the training process remains efficient and productive (Yang et al., 2023).

**Tuning Hyper-parameters.** The performance of Deepot remains sensitive to the choice of hyperparameter values. To tackle this challenge and further enhance Deepot’s performance (An et al., 2023, 2024; Guo et al., 2024), we employ a tuning approach to optimize its hyperparameters, including batch size and learning rates. Specifically, we utilize a grid search method that allows us to define the range of values for each hyperparameter. In this process, the grid search constructs and evaluates our model for every possible combination of hyperparameters, ensuring a comprehensive exploration of the parameter space. Finally, we assess each learned model’s performance using cross-validation, ensuring a robust evaluation of Deepot’s capabilities in our work.

## 6. Evaluation

### 6.1. Experiment Setting

**Baselines:** To evaluate our model’s performance, we compared it against object detection frameworks such as YOLO, which were similarly trained on our datasets. The comparison provides insight into the

Table 3: Comparison of Deepot with YOLO Baseline

Metric	Deepot	YOLO Baseline
Precision	64%	58%
Recall	67%	54%
F1 Score	65%	56%
Average Precision (AP)	66%	57%
Confidence Score	63%	55%

improvements offered by our High-Resolution Image Verification stage. YOLO (Ultralytics, 2021), short for You Only Look Once, revolutionizes object detection with its unique framework. It treats object detection as a single regression problem, directly predicting bounding boxes and class probabilities in one evaluation.

**6.1.1. Evaluation Metrics** We evaluate the performance of Deepot and YOLO baseline model detection methods using several key performance metrics tailored for single-class detection tasks. Model performance was quantified using Precision, Recall, F1 Score, Average Precision (AP) and Confidence Score.

**Precision and Recall:** Precision measures the accuracy of positive predictions, while recall measures the model’s ability to identify all relevant instances.

**F1 Score:** The F1 score is the harmonic mean of precision and recall, providing a balanced metric for evaluating both false positives and false negatives.

**AP:** AP computes the average precision across a range of recall values, summarizing the precision-recall curve into a single performance measure.

**Confidence Score Comparison:** This compares the models based on the confidence of their predictions, with higher scores indicating more reliable predictions.

### 6.2. Training and Evaluation Results

Based on the training results displayed in Figure 3 for the YOLOv8 parking-lot detection object detection model, we can observe that the training box loss, class loss, and DFL loss show a clear decreasing trend over the training epochs, indicating that the model is learning effectively. Initially, the losses are relatively high, suggesting that the model starts with a high degree of error. However, as training progresses, the losses steadily decrease and plateau, demonstrating that the model is converging and improving its predictions. The precision and recall metrics illustrate the model’s performance in terms of correctly identifying parking lot objects. Both precision and recall show significant improvements over the epochs. Initially, these metrics fluctuate, indicating instability and potential overfitting

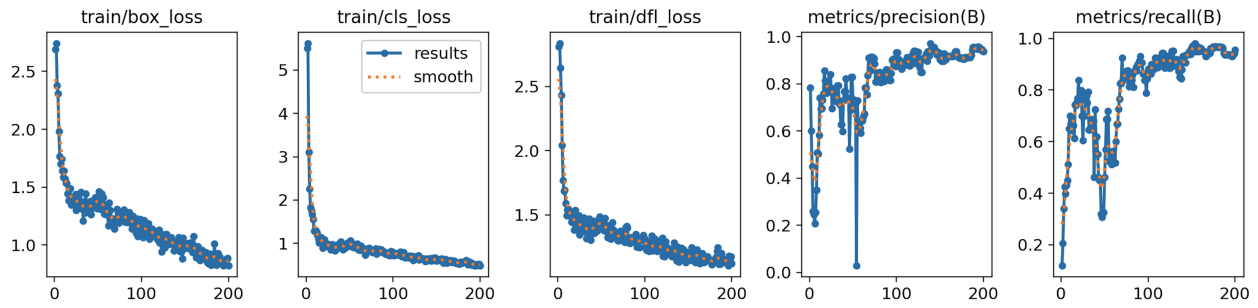


Figure 3: Training results of YOLOv8 on the parking-lot detection dataset.

Location Type	Energy for 308 Parking Lots	Energy for 1003 Parking Lots	Percentage Change (%)
Depot	94.51	82.43	↓ 12.64 %
Public	25.44	41.67	↑ 63.85 %
Enroute	23.61	22.98	↓ 3.05 %

Table 4: Energy Usage and Percentage Change Comparison between 308 and 1003 Parking-lots

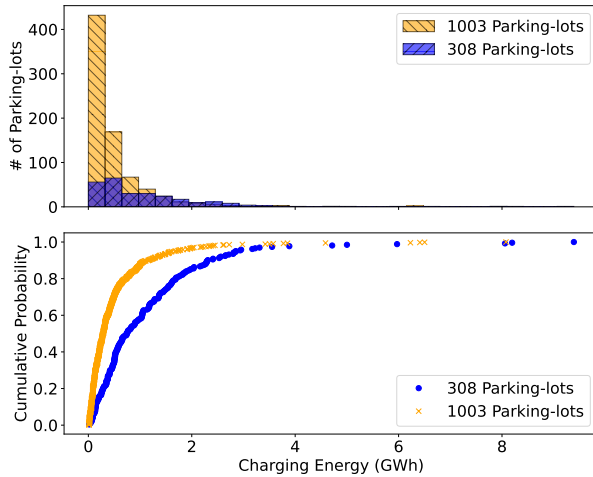


Figure 4: Comparison of Charging Energy and CDF for 308 and 1003 Parking-lots

in the early stages. As training continues, both metrics stabilize and improve, approaching values closer to 1.0, which signifies high accuracy in detection.

Table 3 presents a comparative analysis between Deepot and the YOLO Baseline across various metrics. Deepot outperforms the YOLO Baseline in all evaluated metrics. Specifically, Deepot achieves a Precision of 0.64 compared to the Baseline’s 0.58, marking an improvement of approximately 10.3%. The Recall metric shows a more substantial improvement, with Deepot at 0.67 versus 0.54 for the Baseline, representing a 24.1% increase. The F1 Score for Deepot is 0.655, which is 17% higher than the Baseline’s 0.56. Average

Precision (AP) for Deepot is 0.66, while the Baseline lags at 0.57, indicating a 15.8% enhancement. Finally, the Confidence Score of Deepot is 0.63, surpassing the Baseline’s 0.55 by approximately 14.5%. Overall, these results demonstrate that Deepot significantly improves performance across all key metrics compared to the YOLO Baseline.

### 6.3. Deepot Performance on Charging Load

To analyze Deepot’s performance on charging load, we use the same environment setting as in Section 3. Deepot detects 1003 parking lots. The detected parking lot location data is saved as a JSON file containing location IDs, latitude, and longitude. This JSON file is then used as input for HEVI-LOAD. The tool considers each location as a potential charging site, and its routing algorithm directs vehicles to these locations for charging when their battery levels are insufficient to complete the journey from the origin to the destination. Table 4 shows the energy usage and percentage change comparison between 308 and 1003 parking lots. From this table, we see that, compared to the 308 parking lots, the vehicles consume 82.43 GWh at depots, 41.67 GWh at public parking lots, and 22.98 GWh at en-route parking lots. The results indicate that due to the increased number of public parking lots, the energy consumption at depots decreases by 12.64%, which sheds the energy burden on depots. Meanwhile, the energy consumption at public parking lots increases by 63.85%, and the energy consumption at en-route parking lots is reduced by 3.05%. The reduction in en-route parking lot usage is beneficial as it means that vehicle drivers do not need

to find intermediate charging locations before reaching their next destinations. However, this percentage of en-route usage cannot be reduced to zero since, when the current battery energy is lower than a threshold value that is not enough to drive to the next destination, drivers must find nearby public parking lots.

Figure 4 shows the comparison of charging energy and CDF for 308 and 1003 parking lots. From this figure, we observe that compared to the 308 parking lots, the average charging energy for the 1003 parking lots is 0.50 GWh, which represents a 55.15% reduction in the charging load of each public parking lot even though the energy percentage increased by 63.85% as shown in Table 4. Additionally, we can see that only 12.44% of the parking lots have a charging energy greater than 1 GWh, and only 3.37% of the parking lots have a charging energy greater than 2 GWh. This distribution suggests that the infrastructure with 1003 parking lots can handle higher demands more efficiently.

## 7. Discussion

**Scalability and Practical Implementation Challenges.** For the California dataset, Deepot requires a server featuring an NVIDIA GeForce RTX 4090 GPU with 24GB VRAM for training. An Intel Xeon processor with 16 cores handles CPU operations, and the system has 128GB DDR4 RAM for efficient data and model processing. Training takes about 4.5 hours, with an inference time of around 30 milliseconds per image. A major bottleneck in scaling is the dataset's size which doubles after converting TIFF to PNG formats, as both are needed for geographic information and object detection purposes. Optimized PNG compression and distributed storage systems (Ding et al., 2017; Zhang and Stewart, 2020) can help manage larger datasets without compromising geospatial accuracy or detection.

**Assumptions and Limitations.** Deepot relies on the availability of both low- and high-resolution satellite imagery. While low-resolution images are generally accessible, high-resolution images may face availability constraints due to geographic coverage or access restrictions, potentially affecting detection accuracy, especially in regions with sparse or outdated data. Additionally, the model's performance can be influenced by satellite image variability, including sensor differences, atmospheric conditions, and occlusions. These factors may impact detection accuracy, necessitating further research into mitigation strategies such as data augmentation or incorporating synthetic data to enhance model robustness.

**Implementation Across Multiple Regions and Metropolitan Areas.** Deepot can be scaled to

different regions by first acquiring geographic data from sources like the NOAA Data Access Viewer. This data is processed and split into smaller images for the Low-Resolution Dataset, which trains the object detection model. Users then define the target area with polygon coordinates, and the system generates sub-images for that area. The detected parking lot coordinates are refined using high-resolution Google Maps imagery, enabling the method's application across various regions and metropolitan areas.

## 8. Conclusion

This paper presents Deepot, a novel approach for detecting truck parking lots using low-resolution satellite imagery. By leveraging both low- and high-resolution data, Deepot addresses the limitations of current detection methods reliant on high-resolution datasets. Our approach improves the accuracy and reliability of parking lot detection, facilitating more effective planning of electric truck charging infrastructure. Moreover, integrating Deepot with the HEVI-LOAD tool enhances the fidelity of infrastructure planning, providing additional high-fidelity candidate locations. Deepot's effectiveness is validated through a comprehensive dataset, with results showcasing its enhanced performance over existing methods.

For future work, improving detection algorithms to enhance accuracy and robustness, particularly through techniques like semi-supervised learning, transfer learning, and incorporating contextual data (e.g., road networks, traffic patterns), is a key focus. Expanding the dataset to cover more geographic regions and diverse environmental conditions is also crucial, as it would enhance the model's generalizability and scalability. Additionally, exploring new applications, such as urban planning, disaster management, and agricultural monitoring, could broaden the utility of low-resolution satellite imagery across various domains. These advancements will not only enhance the technical capability of Deepot but also extend its applicability, making it a valuable tool for infrastructure planning.

## 9. Acknowledgement

This work was funded by the U.S. Department of Energy Vehicle Technologies Office under Lawrence Berkeley National Laboratory Agreement No. #29749. Lawrence Berkeley National Laboratory is supported by the Office of Science of the United States Department of Energy and operated under Contract Grant No. DE-AC02-05CH11231.

## References

- Amato, G., Carrara, F., Falchi, F., Gennaro, C., Meghini, C., & Vairo, C. (2017). Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, 72, 327–334.
- An, Z., Ding, X., & Du, W. (2024). Go beyond black-box policies: Rethinking the design of learning agent for interpretable and verifiable hvac control.
- An, Z., Ding, X., Rathee, A., & Du, W. (2023). Clue: Safe model-based rl hvac control using epistemic uncertainty estimation. *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*.
- Blaschke, T. (2010). Object based image analysis for remote sensing. *ISPRS journal of photogrammetry and remote sensing*.
- Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation.
- De Almeida, P. R., Oliveira, L. S., Britto Jr, A. S., Silva Jr, E. J., & Koerich, A. L. (2015). Pklot—a robust dataset for parking lot classification. *Expert Systems with Applications*.
- Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., Hughes, F., Tuia, D., & Raskar, R. (2018). Deepglobe 2018: A challenge to parse the earth through satellite images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 172–181.
- Ding, X., Cerpa, A., & Du, W. (2024). Exploring deep reinforcement learning for holistic smart building control. *ACM Transactions on Sensor Networks*.
- Ding, X., & Du, W. (2024). Optimizing irrigation efficiency using deep reinforcement learning in the field. *ACM Transactions on Sensor Networks*.
- Ding, X., Zhang, Z., Jia, Z., Ju, L., Zhao, M., & Huang, H. (2017). Unified nvtcam and stcam architecture for improving packet matching performance. *ACM SIGPLAN Notices*.
- Guo, Z., Jing, X., Ling, Y., Yang, Y., Jing, N., Yuan, R., & Liu, Y. (2024). Optimized air quality management based on air quality index prediction and air pollutants identification in representative cities in china. *Scientific Reports*.
- Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics, SMC-3*(6), 610–621.
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*.
- Hurst-Tarrab, N., Chang, L., Gonzalez-Mendoza, M., & Hernandez-Gress, N. (2020). Robust parking block segmentation from a surveillance camera perspective. *Applied Sciences*, 10(15), 5364.
- LBNL. (2024). HEVI-LOAD: Heavy Electric Vehicle Infrastructure - Load Optimization and Assessment Design [(accessed: 14.06.2024)].
- Liu, M., Ding, X., & Du, W. (2020). Continuous, real-time object detection on mobile devices without offloading. *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*.
- Liu, M., Yang, K., Fu, Y., Wu, D., & Du, W. (2023). Driving maneuver anomaly detection based on deep auto-encoder and geographical partitioning. *ACM Transactions on Sensor Networks*.
- Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*.
- Ultralytics. (2021). YOLOv5: A state-of-the-art real-time object detection system.
- Van Etten, A., Lindenbaum, D., & Bacastow, T. M. (2018). Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*.
- Wang, R., Ju, Y., Allybokus, Z., Zeng, W., Obrecht, N., & Moura, S. (2024). Optimal sizing, operation, and efficiency evaluation of battery swapping stations for electric heavy-duty trucks. *2024 American Control Conference (ACC)*.
- Yang, K., Zhao, X., Zou, J., & Du, W. (2023). Atpp: A mobile app prediction system based on deep marked temporal point processes. *ACM Transactions on Sensor Networks*.
- Zhang, Y., He, Y., Gurukar, S., & Parthasarathy, S. (2024). Heteromile: A multi-level graph representation learning framework for heterogeneous graphs.
- Zhang, Y., & Stewart, C. (2020). Poster: Configuration management for internet services at the edge: A data-driven approach. *2020 IEEE/ACM Symposium on Edge Computing (SEC)*.