Monitoring Empty and Occupied Parking Spots: A Deep Learning Based Solution

S. Teja[*1], M. Sion Kumari[*2], P. Srividya Vaishnavi[*3],

T. Srija Sri[*4], T. Kavyasri[*5]

[*1,3,4,5]Student, Department of Computer Science and Systems Engineering Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh, India.

[*2]Project Guide, Department of Computer Science and Systems Engineering Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh, India.

I. ABSTRACT

The automobile industry has expanded rapidly with the growing population over the past few decades. With this exponential increase, there is a seamless flow of the traffic causing congestion and sometimes mismanagement of this heavy traffic. To solve this problem some hardware solutions have been introduced previously. These hardware-based solutions use sensors installed at various locations and monitor the traffic and status of the parking lots available at different sites. However, these are less scalable, difficult to maintain, and also costly at times. In the same way, there are also data-driven solutions being brought into the picture which use the data from the surveillance cameras installed at the parking areas. This overcomes the drawbacks of the sensor-based system but is not up to the mark. In order to overcome all these limitations, this paper brings up a solution that uses a deep learning model to classify and analyze the parking lots.

Keywords: Deep learning, OpenCV, Parking lot detection, video processing, real-time detection

II. INTRODUCTION

The world population has increased to 82 billion and there is a 0.85% increase from the previous year based on the statistics with this tremendous increase in the population, undoubtedly the usage of vehicles will also increase. So, there is a need to maintain and manage the congestion due to heavy traffic, accidents, and impaired circulation due to meandering for a parking lot. One of the main reasons for this traffic congestion is due to a hurdle in the maintenance of the parking areas by following the traditional systems. This is causing delays and heavy jamming of the vehicles. In the traditional parking management systems, which are mainly based on manpower involve only manpower checking for the empty slots available in the parking area, assigning the slot, and collecting the fare which is inefficient nowadays due to the growth of automobile usage day by day. The global smart parking market, valued at approximately $7.16 billion in 2023, is projected to reach around $27.85 billion by 2032, reflecting a compound annual growth rate (CAGR) of around 18.50% from 2024 to 2032 driven by urbanization and rising automobile ownership and also the demand for efficient parking management solutions. There are intelligent parking management systems that use IoT sensors and mobile apps to optimize and manage parking space utilization and enhance traffic flow. Their key components are the hardware sensors which detect whether the space is occupied or not and give real-time information. Drivers can also use mobile apps to find the available free parking lots book those lots and pay the fare in advance this reduces search time and also reduces manpower used in the traditional systems. These also involve automatic payment systems which allow drivers to pay through contactless methods or mobile apps the data collected from the sensors will be analyzed to optimize the parking operation identify the peak demand times and improve the overall performance however this intelligent parking management system is efficient to an extent but the usage of sensors is a drawback as they are costly to set up and maintain on a large scale. Deep learning models have gained demand in various domains hence we brought up a solution that uses OpenCV which is a

comprehensive open-source deep learning library designed for real-time computer vision applications. It provides various tools for image and video processing, object detection and tracking, feature detection and matching, image segmentation, and machine learning 3d reconstruction. It supports multiple programming languages including Python, C, and Java, and can run on various platforms like Windows, Linux, macOS, Android, and iOS

III. REVIEW OF LITERATURE

[1] Radiuk et al. (2022) proposed a solution that uses convolutional neural networks (CNN) which has managed to process 66 parking lots in 0.14 seconds and with an accuracy of 85.4%, their fine-tuned CNN model also has some weaknesses depending on the number of images, and the surveillance camera angle and weather conditions. With the growing innovations, other solutions are also being developed. [2] Abhigna et al. brought up a solution that emphasizes using the YOLOv8(You Only Look Once) Algorithm and OpenCV library, which enhances real-time parking management The YOLOv8 algorithm, which was developed by Ultralytics, facilitates video processing and finding out the available and occupied parking lots has high accuracy in object detection and can be used because of its fast response. [3] Elfaki et al. came up with an AI-based solution to provide dynamic allocation of parking lots and ensure that the car is parked in the correct slot. They have given two solutions, in which the first one is based on a motion sensor and the other one is a range-finder sensor. A plate detection and recognition system has been used where an IoT device is used to capture the number plate image, this system will recognize the numerals. [4] Dasari et al. proposed a solution using OpenCV parking spot received in the video is empty or not by combining the Prewitt Edge Detection Algorithm and coordinate-bound pixel sections, after the image processing, Tesseract is used to extract text from images. To get the best result different images are processed at different image processing levels. This can also be used to know whether the car is correctly parked or not. [5] Bhatia et al. proposed a solution using OpenCV, a Python library for image processing, and named it Counting Available Parking Space using Image Processing (CAPSuIP). This is low cost and it uses the Software Development Life Cycle to analyze, plan, and test the functionalities. [6]T.Mar et al. brought up a solution that uses video processing, they used a wide-angle lens camera equipped with a desktop. Different feature extractors and image processors are used to get accuracy and the final system has achieved an accuracy of 99.8%. [7] V.K. Boda et al. came up with a solution that uses wireless sensors that are equipped with magnetic sensors, and magnetic signatures given by sensors are presented. There is a small error probability. [8] Ichihashi et al. proposed a solution which is the enhancement of the camera-based parking systems. This has overcome some drawbacks in the camera systems where the camera lens was affected by raindrops, sunlight glare, etc. They brought up with a solution that uses Fuzzy C-Means clustering and hyperparameter tuning by swarm optimization. This was introduced at the underground parking in Tokyo at first and later it was tested outdoors also and got an accuracy of 99.6%. [9] Yusnita et al. came up with a solution where they used image processing techniques to process the images that are captured from the live streaming video of a camera. The pictures are captured when a vehicle leaves or enters the lot.

IV. METHODOLOGY

Libraries Used

1. CV2: cv2 is the main module in OpenCV that provides developers with an easy-to-use interface for working with image and video processing functions.

2. Pickle: This is a Python module used for serialization known as "pickling" and deserialization known as "unpickling". This is used to convert an object into a byte stream during the process of pickling and while deserialization original object will be reconstructed.

In our project we used pickle for serializing and deserializing the vehicle position list which is easier using pickle than manually reading/writing text files, also pickle makes the retrieval faster.

3. NumPy: Numerical Python, this library is used for performing mathematical manipulations on the data during the process of preprocessing the collected data. It has various built-in methods for easier manipulations.

4. Matplotlib: This library is used to get the visual representations i.e., graphical plots of the results. In our project, we used this library to get the plots of the model's accuracy and loss.

5. Keras: This is a high-level functional neural network API, written in Python and it is capable of running on top of other libraries like TensorFlow, CNTK, or Theano. This API eases the process of building and training the deep learning model. In this project, we used this library to build and train our model.

6. TensorFlow: This is an open-source library that was developed by Google, and is primarily used for building and deploying deep learning models. In this project, we used this over Keras to deploy the trained model.

7. VGG-16 Pretrained Model: This is a type of Convolutional Neural Network (CNN) architecture brought up by the Visual Geometry Group (VGG) at the University of Oxford. It has 16 layers i.e., 13 convolutional layers and fully connected layers. It is effective in computer vision tasks and image classification and also object recognition. In this project, we used this model to accurately classify the car objects in the snapshot from the video.

8. Flask

This is a web framework used for building web applications and APIs. It supports dynamic HTML rendering with embedded Python code. This is used for creating RESTful APIs for machine learning and web services.

Implementation

Step – 1: Data Collection & Preprocessing

In this step, a video of the parking lot is collected and preprocessed based on the model's requirements.

Step – 2: Model Building & Model Architecture

Firstly CV2 is used for resizing and normalizing the snapshots from the loaded video. VGG16 pre-trained model, a feature extractor is used for removing the top layer and adding the custom dense layer for classification task. The model is trained using Keras and TensorFlow. Model is saved using model.save() and label encodings are stored in a pickle file for later usage. Finally, accuracy will be optimized using fine-tuning and hyperparameter-tuning.

2.1 Model Architecture

- The 16 in the VGG16 model refers to the 16 layers that have weights. This model will comprise of 13 convolutional layers, and 3 dense layers, 5 max pooling layers which sum up to 21 layers in total, but it has only 16 weight layers i.e., learnable parameter layers.
- This model takes input tensor size as 224,244 with 3 RGB channels. In this model instead of having a large number of hyperparameters they focused on having convolutional layers of a 3x3 filter with stride 1 and always used the same padding and max pool layer of a 2x2 filter of stride 2.
- Both the max pool layer and convolutional layers are consistently arranged throughout the whole architecture.
- Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv-4 and Conv-5 has 512 filters.
- Three Fully Connected Layers follow a stack of convolutional layers: the first two have 4096 channels each, and the third performs 1000-way ILSVRC (ImageNet Large Scale Visual Recognition Challenge, which is a widely used dataset for computer vision and was created to facilitate research in image classification, object detection, and visual understanding tasks.)

classification and thus contain 1000 channels (one channel for each class). The final layer is the soft-max layer.

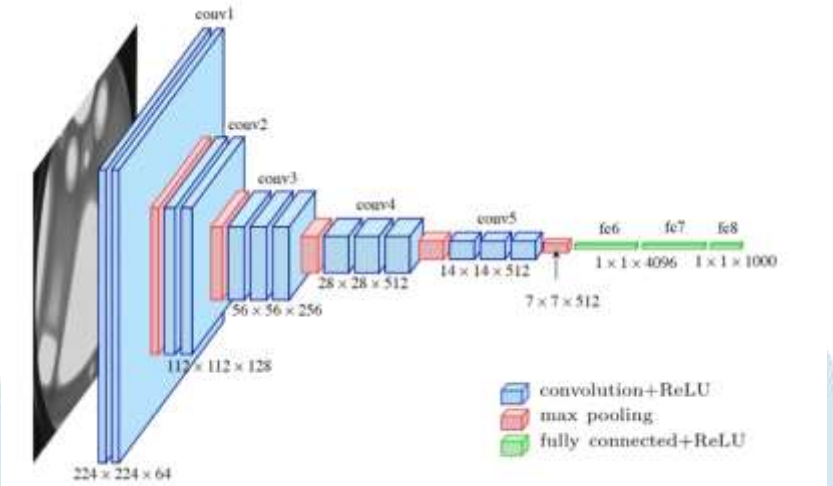

Fig. 4.1

Step – 3: Training & Testing model

Split the dataset into training and testing sets, to improve the generalization and apply data augmentation. VGG16 pre-trained model is used for freezing the convolutional layers and then adding the fully connected layers for classification. And then the model is compiled using Adam optimizer and categorical cross-entropy loss, then trained using model.fit(). For testing preprocess the images using cv2 and resize and normalize the images. Load the trained model using tensorflow.keras.models.load_model() and classify the parking spaces using model.predict(). Finally, evaluate the model's performance using accuracy for model validation.

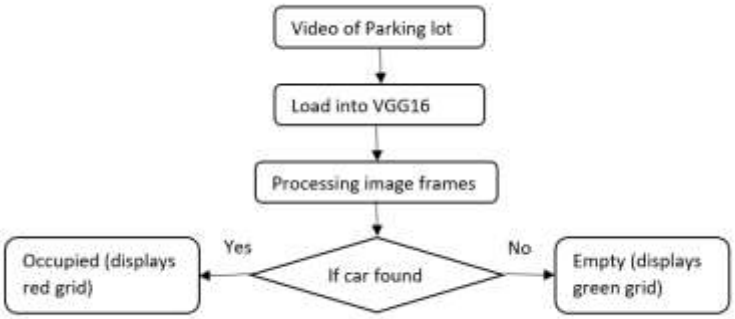

Fig. 4.2

Step – 4: Model Evaluation

The model's efficiency in classifying the parking lots accurately is assessed using the following metrics:

Metric	Score
Batch Size	32
Epochs	15
Number of classes	2
Accuracy	97.5%
Precision	87.36%
Recall	83.63%
F1 Score	85.45%
Loss	0.18%
Learning rate	0.001

V. RESULTS AND CONCLUSION

Input:

A video input is given to the pre-trained model for classifying the free and occupied parking lots. The model will take the snapshots from the input video, normalize it, and then store the label encodings in a pickle file.

Fig: Input Video

Fig: Image from input video

Output:

Flask is used for the front-end part of the project. This is used to show a proper picture of the output visually. It supports dynamic HTML rendering with embedded Python code.

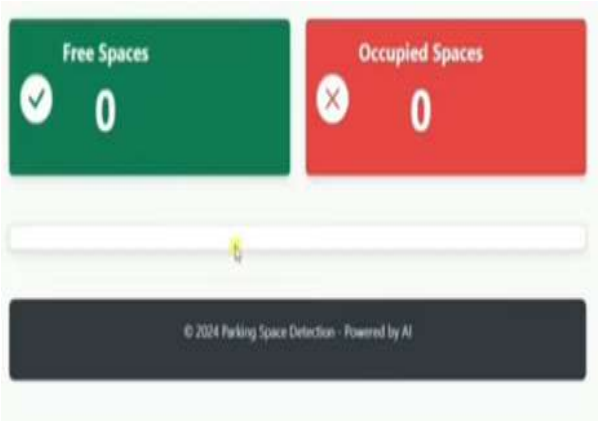

Fig: Output before loading the video

Fig: Output after loading the video

After successfully training and testing the model we have achieved an accuracy of 97.5% and a loss of 0.18%. This model is efficient in classifying the images taken from the video and giving out the desired output.

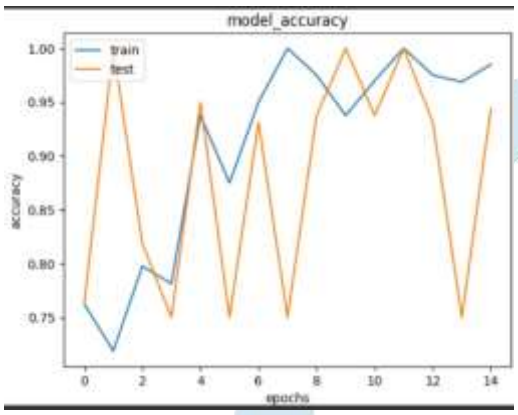

Fig: Model accuracy

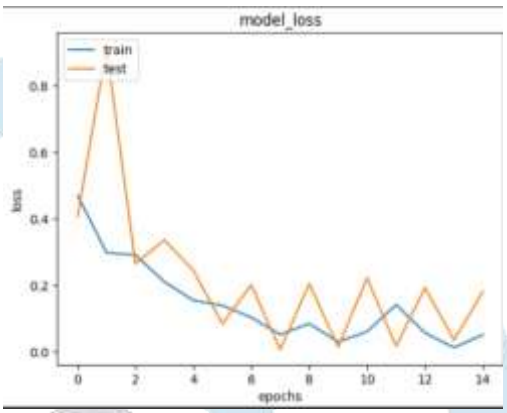

Fig: Model loss

CONCLUSION

This project demonstrates a vision-based solution for monitoring parking spaces using deep learning techniques. By converting video input into frames and analyzing them through the VGG16 model, the system successfully classifies empty and occupied spots without relying on physical sensors. The VGG16 model was chosen for its proven performance in image classification tasks and its ability to extract deep spatial features with high precision. The model achieved 97.5% accuracy, proving to be both efficient and scalable for real-time parking detection.

Future work may include testing other deep learning architectures such as ResNet or EfficientNet to further improve accuracy. Features like license plate recognition, dynamic parking suggestions, and mobile app integration could also be added to make it more interactive and useful in mart city environments.

VI. REFERENCES

[1] Radiuk, P., Pavlova, O., El Bouhissi, H., Avsiyevych, V., & Kovalenko, V. (2022). Convolutional neural network for parking slots detection.

[2] Abhigna, A., Sreeja, C., Mahesh, A., Varma, A. P., & Subhahan, D. A. (2024, August). Vacant Parking Slot Availability Detection Using Yolov8 and OpenCV-Integration with payment functionalities. In 2024 7th International Conference on Circuit Power and Computing Technologies (ICCPCT) (Vol. 1, pp. 1433-1438). IEEE.

[3] Elfaki, A. O., Messoudi, W., Bushnag, A., Abuzneid, S., & Alhmiedat, T. (2023). A smart real-time parking control and monitoring system. Sensors, 23(24), 9741.

[4] Dasari, J., Vodnala, S., & Enugala, S. (2023). Smart parking system using Python and OpenCV. International Journal for Research in Applied Science and Engineering Technology, 11(8), 1976-1985.

[5] Bhatia, S., Naib, B. B., Goel, A. K., Kumari, K., Harsh, U., & Mishra, S. (2023, November). Using OpenCV Space Detection System. In International Conference on Innovations in Data Analytics (pp. 259-271). Singapore: Springer Nature Singapore.

[6] T. Mar; N. Marcel; “Video-based parking space detection,” 2012 [Online]. Available: http://www.ini.rub.de/data/documents/ tschentscherneuhausen_parking_space_fbi2012.pdf

[7] Boda, V.K.; Nasipuri, A.; Howitt, I.; "Design considerations for a wireless sensor network for locating parking spaces," SoutheastCon, 2007.

[8] Ichihashi, H.; Katada, T.; Fujiyoshi, M.; Notsu, A.; Honda, K.; "Improvement in the performance of camera-based vehicle detector for the parking lot," Fuzzy Systems (FUZZ), 2010.

[9] Yusnita, R., Fariza Norbaya, and Norazwinawati Basharuddin. "Intelligent Parking Space Detection System Based on Image Processing." International Journal of Innovation, Management and Technology 3.3 (2012): 232.