A REPORT

ON

APPLICATIONS OF COMPUTER VISION: OBJECT DETECTION AND OPTICAL RECOGNITION

BY

Name of the student	ID No.
Aditya Khandelwal	2023A4PS0452P

AT

National Informatics Centre Services Inc. (NICSI) - India

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(July, 2025)

A REPORT

ON

APPLICATIONS OF COMPUTER VISION: OBJECT DETECTION AND OPTICAL RECOGNITION

BY

Name of the student Aditya Khandelwal
ID No. 2023A4PS0452P
Discipline(s) B.E. Mechanical

Prepared in partial fulfillment of the
Practice School-I Course Nos.
BITS C221/BITS C231/BITS C241

AT

National Informatics Centre Services Inc. (NICSI) - India

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(July, 2025)

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to everyone who has supported and guided me throughout. This experience has been immensely enriching, and it would not have been possible without the encouragement and assistance of several individuals.

Firstly, I extend my sincere thanks to the entire Practise School Division, for providing me with this incredible opportunity to gain practical exposure and for ensuring a well-structured program that facilitated my learning and growth.

I am also deeply grateful to Prof. Soumyananda Chakraborti, the faculty in charge, for their constant guidance and support.
A special note of appreciation goes to Mr. Abhinav Kaundal and Mr. Kumar Jyoti, my mentors at the workplace, for their patience, expertise, and mentorship. Their hands-on guidance and constructive criticism played a pivotal role in shaping my skills and understanding of the professional environment.

I would also like to thank my colleagues and peers at NICSI for their cooperation, camaraderie, and willingness to share their knowledge. Their support made my time at the workplace both enjoyable and productive.

Lastly, I am grateful to my family and friends for their endless encouragement and understanding during this internship. Their belief in me kept me motivated to give my best.

ABSTRACT SHEET

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI
(RAJASTHAN)
Practice School Division

Station: National Informatics Centre Services Inc.
Centre: NICSI, Bhikaji Cama Place, New Delhi
Duration: 8 Weeks
Date of Start: 26 May, 2025
Date of Submission: 15 July, 2025

Title of the Project

APPLICATIONS OF COMPUTER VISION: OBJECT DETECTION AND OPTICAL RECOGNITION

Name: Aditya Khandelwal
ID: 2023A4PS0452P
Discipline: B.E. MEchanical

Name & Designation of Experts: Mr. Kumar Jyoti, Senior Manager & Abhinav Kaundal, Technical Project Manager

Name of the PS Faculty: Prof. Soumyananda Chakraborti

Key Words: Computer Vision, Image Classification,Machine Learning, Convolutional Neural Network (CNN), Object Detection, Optical Character Recognition,Google Colab, Kaggle, RoboFlow, YOLOv8, Video Processing,
Project Areas: Machine Learning,Optical Character Recognition, Computer Vision, Image Processing, Real-Time Object Detection

ABSTRACT

This report presents the development and assessment of three computer vision projects that demonstrate practical applications of machine learning.
The first project centers on binary image classification using Convolutional Neural Networks (CNNs) to distinguish between images of cats and dogs. By employing techniques such as Image-preprocessing, augmentation, and hardware optimization, the system achieved strong classification performance.
The second project involved the detection of vehicles on a live video and classification of the incoming vehicles into various classes.
Then expanded to reading the license plate of these vehicles and extracting relevant data.

The third project involved the development of a computer vision system for detecting smoke emissions from vehicles using the YOLOv8 object detection algorithm. The system was trained on a publicly available dataset using the Roboflow platform The objective is to automate the detection of polluting vehicles and contribute to efforts aimed at environmental monitoring and air quality improvement.

All projects were implemented in Python, utilizing libraries including OpenCV, TensorFlow ,NumPy ,MatPlotLib, Ultralytics,YOLO and Roboflow. The report explores various facets of computer vision and machine learning—such as model architecture, training, evaluation, and deployment—highlighting the potential of computer vision to enable intelligent, responsive systems for safety and automation.

Signature of Student : Aditya Khandelwal
Date : 15 July, 2025

Signature of PS Faculty :
Date :

Table Of Contents

Introduction

Computer vision, which allows machines to interpret visual data, has the potential to transform industries like healthcare and transportation. This report offers an overview of my projects, focusing on methodologies, challenges, and outcomes.

My first project involved binary image classification, distinguishing between cats and dogs. This foundational exercise taught me image processing and machine learning principles, providing hands-on experience in data preprocessing, model training, and evaluation.

The second project expanded into real-time video analysis, focusing on vehicle detection. It required identifying and classifying vehicles like cars, trucks, and buses, enhancing my skills in object detection and real-time processing.

I further developed this project by adding smoke emission detection from vehicles, addressing environmental and safety concerns. This highlighted the practical impact of computer vision in solving societal issues.

In the report's next sections, I'll discuss the technical approaches, tools, and frameworks used, the challenges faced, solutions implemented, and key insights gained.

These projects allowed me to apply computer vision techniques to real-world challenges while also improving my technical, analytical and problem-solving skills.

PROJECT 1 : CAT VS DOG IMAGE CLASSIFICATION

The primary aim of the first project is to classify images into two categories: cats and dogs. To implement this project, various libraries were utilized, including TensorFlow for logic, OpenCV for image handling, NumPy for mathematical operations, MatPlotLib for Graph Plots and Keras for building the sequential model.

Dataset Preparation:

The dataset contained labelled images of cats and dogs, organised into separate folders for training and testing. The training set comprises approximately 12,000 images of each category, while the test set has around 1,000 images of each.
The images used were of higher than required quality and therefore needed to be resized to lower quality; leveraging OpenCV library for faster and optimized operations
After resizing, the Numpy's 'random' function was used to shuffle the order of these images to avoid any pattern formation.

```
for i in random.sample(glob.glob('cat/*.jpg') + glob.glob('cat/*.png') + glob.glob('cat/*.jpeg'),12000):
    shutil.move(i,'train/cat')
for i in random.sample(glob.glob('dog/*.jpg') + glob.glob('dog/*.png') + glob.glob('dog/*.jpeg'),12000):
    shutil.move(i,'train/dog')
for i in random.sample(glob.glob('cat/*.jpg') + glob.glob('cat/*.png') + glob.glob('cat/*.jpeg'),500):
    shutil.move(i,'valid/cat')
for i in random.sample(glob.glob('dog/*.jpg') + glob.glob('dog/*.png') + glob.glob('dog/*.jpeg'),500):
    shutil.move(i,'valid/dog')
for i in random.sample(glob.glob('cat/*.jpg') + glob.glob('cat/*.png') + glob.glob('cat/*.jpeg'),1000):
    shutil.move(i,'test/cat')
for i in random.sample(glob.glob('dog/*.jpg') + glob.glob('dog/*.png') + glob.glob('dog/*.jpeg'),1000):
    shutil.move(i,'test/dog')
```

Fig 1. Preparation of random training and test batches from available data

CNN Architecture:

Initially, a CNN model was built using the Keras Sequential API. Four convolutional layers were used to extract relevant features from the images such as edges, textures, and

shapes.The model recognizes and remembers these patterns and uses them to classify unseen images. Batch normalisation was applied after each layer to stabilise and speed up training by normalising the outputs of the previous layer. Max-pooling reduced the size of the feature maps, retaining only the most important information.

A flattening layer was used to arrange the pixel values in a linear array followed by a fully connected dense layer to compress the information into smaller format. A dropout layer was applied before the final dense output layer, which used softmax activation for binary classification. A fully connected dense layer learned to combine the extracted features and aided in decision-making for classification. An additional dropout layer was applied after this dense layer to further reduce the risk of overfitting. Finally, a dense output layer with softmax activation produced probabilities for each class (cat or dog), enabling the model to make a final prediction.

```
model=Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(128, 128, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))


model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))


model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2, activation='softmax'))

model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
model.fit(x=train_batches,validation_data=valid_batches, epochs=10, batch_size=32, shuffle=True, verbose=2)
model.save('cat_dog_model.h5')
```

FIgure. The sequential model leveraging Convolution layers to extract and store the information from images

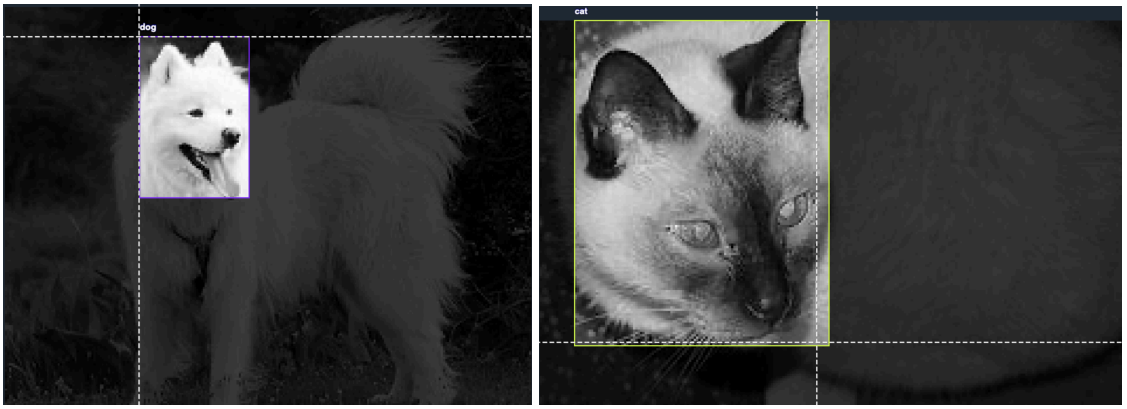
Model Evaluation:

The training and validation datasets demonstrated measurable levels of accuracy and data loss throughout the model development process. After training was completed, the model

underwent evaluation using previously unseen images, allowing for an objective assessment of its predictive performance

```
Epoch 1/10
/Users/aditya/Library/Python/3.9/lib/python/site-packages/PIL/TiffImagePlugin.py:950: UserWarning: Truncated File Read
  warnings.warn(str(msg))
688/688 - 170s - 247ms/step - accuracy: 0.6751 - loss: 0.8600 - val_accuracy: 0.7170 - val_loss: 0.5745
Epoch 2/10
688/688 - 199s - 290ms/step - accuracy: 0.7620 - loss: 0.4957 - val_accuracy: 0.7840 - val_loss: 0.4422
Epoch 3/10
688/688 - 216s - 314ms/step - accuracy: 0.7976 - loss: 0.4412 - val_accuracy: 0.8010 - val_loss: 0.4378
Epoch 4/10
688/688 - 211s - 307ms/step - accuracy: 0.8205 - loss: 0.3998 - val_accuracy: 0.7950 - val_loss: 0.4186
Epoch 5/10
688/688 - 211s - 307ms/step - accuracy: 0.8474 - loss: 0.3451 - val_accuracy: 0.8450 - val_loss: 0.3413
Epoch 6/10
688/688 - 211s - 307ms/step - accuracy: 0.8661 - loss: 0.3057 - val_accuracy: 0.8610 - val_loss: 0.3518
Epoch 7/10
688/688 - 188s - 274ms/step - accuracy: 0.8861 - loss: 0.2702 - val_accuracy: 0.8550 - val_loss: 0.3366
Epoch 8/10
688/688 - 191s - 278ms/step - accuracy: 0.9025 - loss: 0.2312 - val_accuracy: 0.8680 - val_loss: 0.3574
Epoch 9/10
688/688 - 193s - 281ms/step - accuracy: 0.9120 - loss: 0.2097 - val_accuracy: 0.8610 - val_loss: 0.3992
Epoch 10/10
688/688 - 196s - 285ms/step - accuracy: 0.9229 - loss: 0.1826 - val_accuracy: 0.8520 - val_loss: 0.5053
```

Outcomes: The output when the model was trained from tested on several unseen images are as follows:



Results : The model improved its accuracy every epoch and ended with a final accuracy of 0.9229 (or 92.29%) on validation sets and 89.5% on test sets. The data loss also decreased correspondingly from 86% in the 1st round to just 18% in the final epoch.

```
 Total params: 25,449,672 (97.08 MB)
 Trainable params: 8,483,074 (32.36 MB)
 Non-trainable params: 448 (1.75 KB)
 Optimizer params: 16,966,150 (64.72 MB)
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
Confusion matrix, without normalization
[[87 13]
 [ 8 92]]
Test Accuracy: 89.50%
```

PROJECT 2: VEHICLE DETECTION/LICENSE PLATE RECOGNITION

OBJECTIVE:

The second project focused on the automated detection of vehicles and their corresponding license plates from real-world traffic surveillance videos.

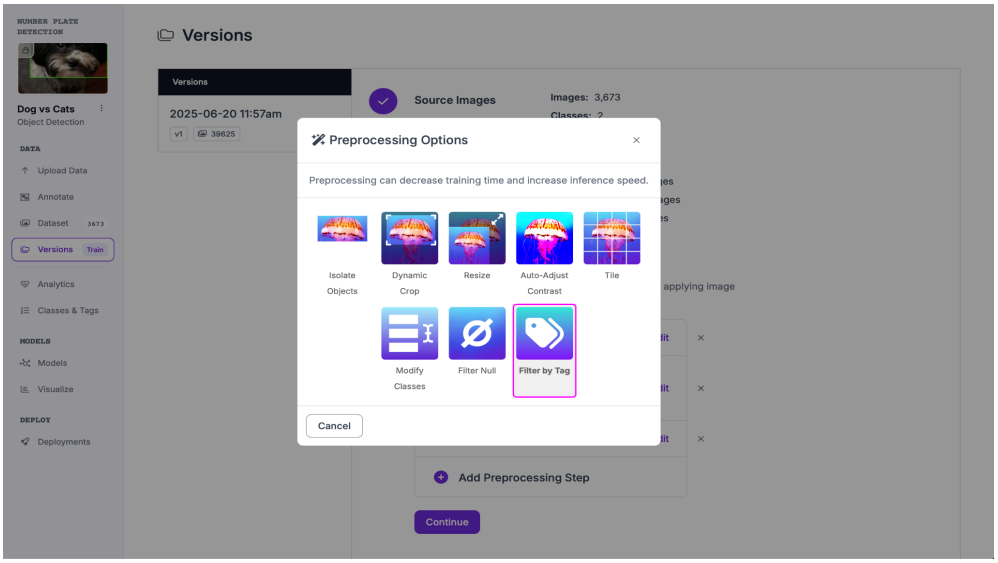
TOOLS USED:

For this project, several tools and libraries were employed to ensure efficient and accurate detection and recognition of vehicles and license plates from video data. YOLOv8, an object detection model by Ultralytics, was used for real-time detection of both vehicles and license plates.

Kaggle was used as the primary development and execution environment to leverage its free access to high-performance GPUs for training and inference. Dataset preparation, including annotation, augmentation, and formatting for YOLOv8, was facilitated using Roboflow, which streamlined the entire preprocessing operation. OpenCV was utilized for handling video processing tasks such as reading input video frames, drawing bounding boxes and class labels, and saving the processed output video. To extract readable text from the detected license plates, EasyOCR—an optical character recognition library—was used. Together, these tools formed a powerful and integrated pipeline for automated vehicle and license plate detection from video footage.

DATASET PREPARATION

1.Image preprocessing



The images available in the dataset were larger than required and therefore resized, allowing faster and smoother operations.Utilizing Roboflow's tools, the images were randomized,greyscaled, cropped and collaged to produce training data.

2.Image Augmentation

Image augmentation generates random images based on existing training data to improve the generalization ability of models. The operations that proved helpful for this project were:

- Random rotations up to 15 degrees
- Horizontal flipping
- Zooming, shearing, and shifting widths and heights
- Normalizing pixel values to a range of 0 to 1

CUSTOM TRAINING A MODEL:

Developing a model for Indian conditions required a custom model to be trained on a custom dataset.After selecting a suitable dataset from kaggle, it was iterated in 100 epochs by the model for learning patterns and different classes of vehicles like rickshaws, autos to heavy trucks on Indian roads.

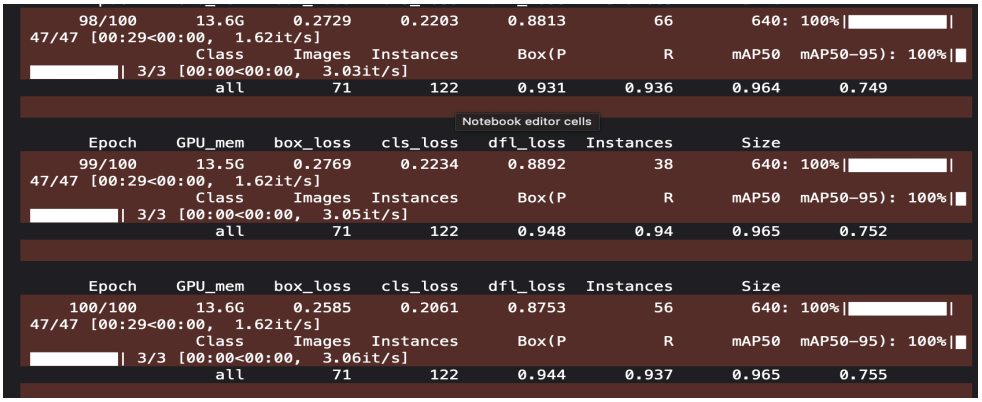
Since the model required learning a variety of classes altogether, 100 rounds/epochs were chosen to provide sufficient time for accurate training.

The training process, lasting 8 hours, was regularly monitored for accuracy and box loss after each round .

After successful training, the epoch with the high MAP score (equivalent to accuracy) was chosen and a **‘model.pt’** file was generated, which contains all the logics developed during training.

MODEL EVALUATION:

- Accuracy and the data loss were noted for the training and validation sets.
- The trained model was tested upon unseen images and predictions were evaluated.



OUTCOMES AND RESULTS:

The resulting model was able to successfully detect vehicles in a video and read their number plates.

The final test accuracy with IoU threshold between 50% to 95% came close to 75% or 0.755.


51G-205.36


DL3CC00107

PROJECT 3: DETECTION OF SMOKE EMISSIONS FROM VEHICLES

OBJECTIVE:

This project focused on detecting visible smoke emissions from vehicles using a YOLOv8-based object detection model. The objective was to develop a system that could identify polluting vehicles in traffic video feeds, contributing to automated environmental monitoring.

Tools and Technologies:

The model was implemented in Python using the Ultralytics YOLOv8 library for object detection. Roboflow was used to annotate and prepare the dataset, while OpenCV handled video processing tasks. Model training and inference were performed on Kaggle to utilize GPU resources.

Dataset Preparation and Annotation:

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="rgmTaPEi8oKgUYhLb1EH")
project = rf.workspace("smoke-detection").project("smoke100-uwe4t")
version = project.version(4)
dataset = version.download("yolov8")
```

A dataset containing 19,000 images of vehicles emitting visible smoke was uploaded to Roboflow. The classes were annotated as “vehicle” and “smoke,” and bounding boxes were drawn around both entities. Roboflow also provided augmentation features such as horizontal flipping, scaling, and brightness adjustment, which contributed to the model’s generalisation during training.

The images were resized to 640×640 resolution to ensure compatibility with YOLOv8. Following annotation, the dataset was exported in YOLOv8 format for incorporation into the training script.

MODEL TRAINING:

Out of the 5 YOLOv8 models, the ‘n’ model was chosen because of its low computational hardware requirements. The model was then trained on this custom data through 20 epochs, which provided a reasonable number of rounds without any major accuracy loss.

```
!pip install ultralytics
from ultralytics import YOLO
model=YOLO('yolov8m.pt')
model.train(data='/kaggle/working/Smoke100-4/data.yaml',imgsz=640,epochs=20,save=True)
```

MODEL EVALUATION:

After training, the model was evaluated using the validation set and unseen test images. Evaluation was based on the following metrics:

- **Precision**: Percentage of correctly detected smoke and vehicles out of all predicted instances.
- **Recall**: Percentage of actual smoke and vehicles correctly detected.
- **mAP@0.5**: Mean Average Precision at an IoU threshold of 0.5, commonly used to assess object detection models.

```
model=YOLO('/kaggle/working/runs/detect/train4/weights/best.pt')
model.predict(source='/kaggle/input/smokevehiclevideo/You Never Want to See This Color Exhaust
```

OUTCOMES AND RESULTS:

The final MAp score came out to 0.938 or 93.8% precision on 50 to 95% IoU threshold.The high mAP50 score indicated that the model was able to correctly detect smoke emissions in most cases with reasonably accurate bounding boxes. However, the lower mAP50-95 score showed some performance drop under stricter IoU conditions, which is expected for small and diffused targets like smoke.

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/20      8.05G     0.4356     0.1972      1.052          8        640: 100%|██████████|
944/944 [09:43<00:00,  1.62it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
          | 136/136 [00:56<00:00,  2.40it/s]
                   all       4322       4322          1      0.999      0.995      0.935

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/20      8.2G      0.4149     0.1862      1.026          8        640: 100%|██████████|
944/944 [09:42<00:00,  1.62it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
          | 136/136 [00:56<00:00,  2.41it/s]
                   all       4322       4322          1      0.999      0.995      0.938
```

CONCLUSIONS

This report presented the development and implementation of three distinct computer vision projects, each highlighting a specific application area: image classification, object detection, and environmental monitoring through visual data analysis. The first project, centered on binary image classification, provided foundational experience in handling datasets, applying convolutional neural networks (CNNs), and evaluating model performance.

The second project introduced real-time object detection using YOLOv8 to identify and classify vehicles from video frames. It also integrated optical character recognition (OCR) techniques using EasyOCR to read license plates. This required multiple components, including data preprocessing, model training, frame-by-frame video processing, and text extraction. The model was trained on a custom dataset relevant to Indian traffic conditions and achieved high accuracy across various vehicle classes, demonstrating the practicality of using object detection models for intelligent transportation systems.

The third project applied computer vision to an environmental use case—detecting smoke emissions from vehicles. A YOLOv8n model was trained on a labeled dataset of vehicles with and without visible smoke. Despite the challenge of detecting a diffuse and variable feature like smoke, the model achieved a high mAP@0.5 score, showing promising results in automated pollution monitoring. This project also demonstrated the value of leveraging cloud platforms like Kaggle for training computationally intensive models.

Collectively, these projects provided exposure to critical aspects of modern computer vision workflows—from data annotation and augmentation to deep learning-based model development and deployment. The use of tools such as Roboflow, OpenCV, Ultralytics YOLOv8, and cloud-based resources allowed for efficient prototyping and testing. These hands-on implementations not only deepened the understanding of theoretical machine learning concepts but also revealed the challenges and opportunities of applying them in real-world scenarios. This experience has strengthened core technical skills and highlighted the potential of computer vision in solving diverse practical problems in safety, automation, and sustainability.

GLOSSARY

- Computer Vision: A field of artificial intelligence that enables machines to interpret and understand visual information from the world, such as images and videos.
- Image Classification: The process of categorizing images into predefined classes or labels based on their visual content.
- Convolutional Neural Network (CNN): A type of deep learning neural network commonly used in image processing tasks, designed to automatically learn and extract features like edges and textures from images.
- Object Detection: A computer vision technique that involves identifying and locating objects within an image or video, often using bounding boxes.
- YOLOv8: A state-of-the-art object detection framework (You Only Look Once, version 8) known for its speed and accuracy in real-time detection tasks.
- Image Preprocessing: Techniques applied to raw images, such as resizing or normalization, to prepare them for model training or processing.
- Image Augmentation: A method to artificially expand a dataset by applying transformations like rotation, flipping, or zooming to existing images, improving model generalization.
- License Plate Recognition: A computer vision application that detects and reads text from vehicle license plates, often used in traffic monitoring systems.
- Smoke Emission Detection: A specialized application of computer vision to identify smoke or emissions from vehicles, often for environmental or safety monitoring.
- OpenCV: An open-source computer vision and machine learning software library used for tasks like image and video processing.
- TensorFlow: An open-source machine learning framework developed by Google, widely used for building and training deep learning models.
- Roboflow: A platform for preparing, annotating, and managing datasets for computer vision projects, streamlining model training workflows.
- Kaggle: An online platform that provides datasets, competitions, and computational resources like GPUs for machine learning and data science projects.
- Epoch: A complete pass through the entire training dataset during the training of a machine learning model.
- MAP Score (Mean Average Precision): A metric used to evaluate the accuracy of object detection models, considering both precision and recall across different classes.

REFERENCES

Datasets:

1.https://universe.roboflow.com/project/car-smoke-detection/dataset/1

2.https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/11

3.https://idd.insaan.iiit.ac.in/dataset/download/

4. https://universe.roboflow.com/smoke-detection/smoke100-uwe4t

GeeksforGeeks. (2024, May 13). *Keras Layers API*. GeeksforGeeks. https://www.geeksforgeeks.org/keras-layers-api/

TutorialsPoint. (2025, March 25). *Keras Introduction*. https://www.tutorialspoint.com/keras/keras_introduction.htm

GeeksforGeeks. (2024, May 13). *Keras Layers API*. GeeksforGeeks. https://www.geeksforgeeks.org/keras-layers-api/

Roboflow: Computer vision tools for developers and enterprises. (n.d.). Roboflow. https://roboflow.com/

Keras: The high-level API for TensorFlow. (n.d.). TensorFlow. https://www.tensorflow.org/guide/keras

TensorFlow. (n.d.). *TensorFlow*. https://www.tensorflow.org/

Sachinpatil. (2023b, September 5). *Cats vs Dogs : Image Classification using CNN(95%)*. Kaggle. https://www.kaggle.com/code/sachinpatil1280/cats-vs-dogs-image-classification-using-cnn-95

GeeksforGeeks. (2025, May 20). *Cat & Dog Classification using Convolutional Neural Network in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/deep-learning/cat-dog-classification-using-cnn-in-python/

Cats and Dogs Classification Dataset. (2023, October 7). Kaggle. https://www.kaggle.com/datasets/bhavikjikadara/dog-and-cat-classification-dataset

Cat and Dog. (2018, April 26). Kaggle. https://www.kaggle.com/datasets/tongpython/cat-and-dog

Vehicle Speed Detection System utilizing YOLOV8: Enhancing road safety and traffic management for metropolitan areas. (n.d.-b). https://arxiv.org/html/2406.07710v1

vehicle speed estimation Object Detection Dataset and Pre-Trained Model by me. (2024,June23). Roboflow. https://universe.roboflow.com/me-mt4xg/vehicle-speed-estimation

Skalski, P. (2024, April 15). *How to Estimate Speed with Computer Vision*. Roboflow Blog. https://blog.roboflow.com/estimate-speed-computer-vision/