Vehicle Re-Identification for Origin-Destination Flow Estimation:
Bengaluru Mobility Challenge - Phase II

***Abstract*—This report presents our approach for vehicle re-identification in the context of the Bengaluru Mobility Challenge (BMC) - Phase II. The objective is to re-identify vehicles seen at one location in another part of the network and estimate origin-destination (O-D) flows over a specific time period. Our solution integrates YOLOv10 object detection, OSNet for feature extraction, and Chroma for efficient vector storage. Using a custom dataset of Indian vehicles, our system achieved competitive performance in re-identification across multiple camera feeds. This document details our methodology, results, and conclusions.**

I. Introduction

The task of vehicle re-identification is essential for estimating origin-destination (O-D) flows, which are crucial for transportation planning and traffic management. The Bengaluru Mobility Challenge (BMC) Phase II evaluates the ability to re-identify vehicles across different locations within a network over a specific time period. Accurate O-D flow estimation is a key factor for effective traffic control and infrastructure development.

In this challenge, participants are tasked with re-identifying vehicles detected by multiple camera feeds, which helps understand traffic patterns, congestion, and vehicle movement between key points. Our approach leverages state-of-the-art detection and feature extraction techniques to enhance the accuracy of re-identification and prediction of O-D flows, improving the overall traffic management system in a dynamic city like Bengaluru.

II. Methodology

Our solution comprises three primary stages: detection, feature extraction, and matching.

A. Detection

We employ the YOLOv10 object detection framework [3] to identify vehicles in video streams. YOLOv10 is known for its real-time detection capability and balance between speed and accuracy.

B. Feature Extraction and Re-Identification

For the vehicle re-identification (ReID) task, we utilize the `torchreid` library [4], which provides state-of-the-art models and tools for deep learning-based ReID in PyTorch. Specifically, we employ the OSNet architecture, particularly the `osnet_x0_75` variant, for feature extraction. OSNet is designed for person and object re-identification tasks, and its omni-scale feature learning capability makes it particularly suitable for learning discriminative yet generalizable feature representations across various scales and conditions [5], [6].

OSNet's architecture enables robust ReID in complex environments with challenging variations, such as lighting and occlusions. The model combines information from both small and large receptive fields, ensuring that global and local features are well-represented. This is critical for vehicle ReID, where subtle differences between vehicles must be captured to achieve high matching accuracy.

For feature extraction, each vehicle is passed through the OSNet model, and the resulting feature embeddings are stored in the Chroma vector database. Matching between different camera feeds is performed using cosine similarity between the extracted feature vectors. A threshold is applied to determine whether the vehicles in different frames belong to the same identity.

C. Vector Storage with Chroma

Chroma, a vector database optimized for high-dimensional embeddings, is used to store the vehicle feature vectors. This ensures efficient similarity searches between frames captured by different cameras.

D. Matching

We compute cosine similarity between embeddings to match vehicles across multiple frames and locations. A threshold is applied to determine whether the vehicles in different frames are likely to be the same.

III. Training Metrics

We evaluated our solution on a dataset of 24,500 images of Indian vehicles, divided into 19,600 training images and 4,900 testing images. The performance of the models is summarized in Appendix (see Table II).

IV. Software and Libraries

Our project uses the following libraries:

- `json` - for handling data formats.
- `logging` - for logging program events.
- `os` - to interact with the operating system.
- `subprocess` - for running external commands.
- `sys` - to manipulate runtime environment.
- `time` - for time-related functions.
- `chromadb` - for storing feature vectors.
- `cv2` - for image processing.
- `h5py` - for handling large datasets.
- `numpy` - for numerical operations.
- `torch` - for deep learning and neural networks.
- `comet_ml` - for tracking model metrics.
- `PIL` - for image manipulation.
- `torchvision` - for computer vision tasks.
- `ultralytics` - for YOLO object detection.
- **PrenAbhi** - Custom library for the competition guidelines, data handling, and submission.

V. Conclusions

Our vehicle re-identification system demonstrated strong performance across various model configurations. The combination of YOLOv10 for detection, OSNet for feature extraction, and Chroma for vector storage allowed for accurate O-D flow estimation. Future work includes improving detection in challenging environments and optimizing the matching process for larger datasets. The authors of this paper believe that this model is foundational but requires large-scale, localized data to capture finer details in the images. Currently, the model performs well for commercial vehicles but needs improvement for identifying three-wheelers, bicycles, and other non-commercial vehicles.

References

[1] AI City Challenge, https://www.aicitychallenge.org/

[2] Z. Zheng et al., "Connecting Language and Vision for Natural Language-Based Vehicle Retrieval," *arXiv preprint*, 2021. Available at: https://arxiv.org/pdf/2105.14897

[3] A. Wang et al., "YOLOv10: Real-Time End-to-End Object Detection," *arXiv preprint*, 2024.

[4] K. Zhou and T. Xiang, "Torchreid: A Library for Deep Learning Person Re-Identification in Pytorch," *arXiv preprint arXiv:1910.10093*, 2019.

[5] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-Scale Feature Learning for Person Re-Identification," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[6] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Learning Generalisable Omni-Scale Representations for Person Re-Identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[7] M. Naphade et al., "The 5th AI City Challenge," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778.

APPENDIX

Here we provide detailed performance metrics for the models and configurations tested.

TABLE I: Comprehensive Results: Epochs, Loss, Model Parameters, Speed, GFLOPs, and mAP Across All Models

Name	Epochs	Val/Cls_Loss	Model/Parameters	Batch Size	Speed (ms)	Train/Box_Loss	Model/GFLOPs	mAP50(B)
prenag-xl	100	0.748	31,668,362	16	3.907	1.132	171.062	0.944
abhipren-aug	30	0.481	8,071,770	16	1.876	0.878	24.796	0.981
abhipren-noaug	30	0.394	8,071,770	16	1.748	0.613	24.796	0.984
prentrains-nonhyp	25	0.394	8,071,770	28	1.019	0.611	24.796	0.985
prentrains-hyp	28	0.425	8,071,770	-1	0.966	0.681	24.796	0.980

TABLE II: Model Performance Results

Model	mAP50(B)	Speed (ms)	GFLOPs	Val/Cls_Loss
prenag-xl	0.944	3.907	171.062	0.748
abhipren-aug	0.981	1.876	24.796	0.481
abhipren-noaug	0.984	1.748	24.796	0.394
prentrains-nonhyp	0.985	1.019	24.796	0.394
prentrains-hyp[1]	0.980	0.966	24.796	0.425