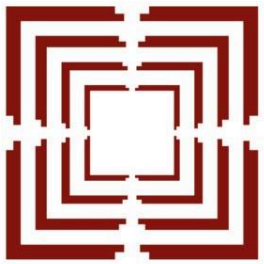

Ahmedabad University

CSE623 - Machine Learning Theory and Practice
Section - 1

Weekly Report 2
Identify Hard stop and momentary stop using the vehicle trajectory dataset

---

Team Name: **The Overfitters**

Name	Enrolment Number
Jinil Savaj	AU2240159
Jay Raval	AU2240151
Meet Suthar	AU2240198
Karan Prajapati	AU2240161
Vishesh Bhatia	AU2240027

Progress on Aerial Drone Dataset - Vehicle Detection and Feature Extraction

1. Introduction

The goal of this project is to process a road monitoring aerial drone dataset and identify vehicles, examining their movement through several frames. Every vehicle within a frame is tracked with an ID (trk), and its location is defined by bounding box coordinates (x, y), width (w), and height (h). The overall objective is to rectify any inconsistencies within the data and conduct feature extraction for insight into the behaviour of the vehicle, including classification, velocity, direction, acceleration, and other motion-related metrics.

This report describes the major steps we have taken thus far in pre-processing the dataset, fixing tracking errors, and removing useless features. Every section addresses the rationale behind the methods used and the outcomes obtained.

---

2. Dataset Overview

The data includes various frames, where each frame might have multiple cars tracked with specific IDs. The primary columns are:

- `Frame`: The frame number.
- `trk`: The ID of the car within the frame.
- `x`, `y`: The location of the top-left point of the car bounding box.
- `w`, `h`: The bounding box width and height.

Our first challenge was dealing with instances where a car went missing and reappeared in non-consecutive frames. This can interfere with continuity when tracking and analyzing. These gaps had to be corrected prior to proceeding with additional analysis.

---

3. Handling Tracking Errors

Issue Identified: The frequent disappearance and re-emergence of vehicles in non-sequential frames was a common issue. For example, a vehicle may appear in frames 1, 2, and 3, not show up in frame 4, and appear again in frame 5. This could result in discontinuity in tracking.

Method: The method used to correct this was interpolation of the missing data. The general concept was to estimate the missing frames for a particular vehicle (trk) by interpolating its location and dimensions (x, y, w, h). Linear interpolation was performed as this was easier, and the mean position and size between the missing and surrounding frames were calculated.

This ensured smooth transitions between frames where a vehicle went out of view, allowing a continuous track. The last corrected dataset only contained consistent vehicle positions per frame so that there was no gap in tracking.

```
# Interpolating missing frames for each trk
df_corrected = df.groupby('trk').apply(lambda group: group.interpolate())
```

This approach worked well in gap-filling and making sure that every tracked vehicle had continuous data across the frames.

---

4. Vehicle Classification (Overall_cls)

Issue Identified: The data did not categorize vehicles. To analyze the data further, I had to separate two-wheelers and cars.

Method: Vehicle classification was done based on bounding box size (width w times height h). Bigger bounding boxes were labelled as "car" and smaller ones as "2-wheeler". The cut-off for this classification was determined from the distribution of sizes in the dataset.

```
# Define a threshold to classify vehicle types (car or 2-wheeler)
size_threshold = 10000  # Arbitrary threshold based on bounding box size
df_corrected['overall_cls'] = np.where(df_corrected['w'] *
df_corrected['h'] < size_threshold, '2-wheeler', 'car')
```

To increase classification stability further, I calculated the mode of the class for each vehicle and used it as the ultimate label. This guaranteed that the vehicles were uniformly classified, even with slight variations in size.

```
# Mode-based classification for consistency
```

```
df_corrected['overall_cls'] =
df_corrected.groupby('trk')['overall_cls'].transform(lambda x:
x.mode()[0])
```

5. Feature Extraction

Once I cleaned and corrected the dataset, I proceeded to feature extraction, something that would enable us to investigate the behaviour of the vehicles in more detail. The features extracted are the centre coordinates of the vehicle, speed, direction, distance, acceleration, and stop patterns.

Center Coordinates: The centre coordinates of every vehicle's bounding box were calculated using the following equations:

```
df_corrected['center_x'] = df_corrected['x'] + df_corrected['w'] / 2
df_corrected['center_y'] = df_corrected['y'] + df_corrected['h'] / 2
```

Instantaneous Velocity: To calculate velocity, We used the difference in `center_x` and `center_y` from one frame to the next for every vehicle. This was converted to kilometres per hour:

```
df_corrected['Vx_Instant_Not_Smoothed(kmph)'] =
df_corrected.groupby('trk')['center_x'].diff() * 3.6

df_corrected['Vy_Instant_Not_Smoothed(kmph)'] =
df_corrected.groupby('trk')['center_y'].diff() * 3.6
```

Direction of Travel (Smoothed_dir): The direction of travel was calculated from the difference in positions (`dx`, `dy`). Depending on the angle between these differences, We gave the direction (e.g., E, NE, NW):

```
def get_direction(dx, dy):
    angle = np.arctan2(dy, dx) * 180 / np.pi
    # Map angle to cardinal directions
    ...
df_corrected['Smoothed_dir'] = df_corrected.apply(lambda row:
get_direction(row['Vx_Instant_Not_Smoothed(kmph)'],
row['Vy_Instant_Not_Smoothed(kmph)']), axis=1)
```

Smoothed Velocity: To gain insight into the speed of movement as a whole, we computed the total velocity (Vxy_smoothed) from the Euclidean distance:

```
df_corrected['Vxy_smoothed'] =
np.sqrt(df_corrected['Vx_Instant_Not_Smoothed(kmph)']**2 +
df_corrected['Vy_Instant_Not_Smoothed(kmph)']**2)
```

6. Additional Feature Extraction

In a second step, we pulled additional features to aid in analyzing stop and move patterns of vehicles:

Distance Traveled: We used the Euclidean formula to determine the total distance a vehicle had traveled:

```
df_corrected['distance'] = np.sqrt(df_corrected['x']**2 +
df_corrected['y']**2)
```

Velocity and Acceleration: Velocity was determined by the difference in distance between two frames, and acceleration was calculated by the difference in velocity:

```
df_corrected['velocity'] = df_corrected['distance'].diff()
df_corrected['acceleration'] = df_corrected['velocity'].diff()
```

Speed: The speed was computed based on the magnitude of the x-direction and y-direction velocity vectors:

```
df_corrected['speed'] =
np.sqrt(df_corrected['Vx_Instant_Not_Smoothed(kmph)']**2 +
df_corrected['Vy_Instant_Not_Smoothed(kmph)']**2)
```

Rolling Average Velocity: To identify trends over time, we calculated rolling averages for velocities for 50 and 100 frames.

```
df_corrected['avg_velocity_50'] =
df_corrected['velocity'].rolling(window=50).mean()
```

---

7. Conclusion

This advancement is a big step in trying to comprehend vehicle motion within the dataset. By fixing tracking mistakes and pulling out lots of features, we have now obtained an abundance of data where we can analyze vehicle classification, speed, acceleration, and stop behavior. This sets us up for subsequent analysis, i.e., predictive modeling or anomaly detection. In the future, I will narrow down the feature set, conduct exploratory data analysis, and possibly incorporate machine learning models to improve vehicle behavior prediction.

Next Week's Goals:

1. **Exploratory Data Analysis (EDA)**
    - Perform statistical analysis on extracted features to identify patterns.
    - Visualize vehicle movement, stops, and trajectory trends.
2. **Anomaly Detection and Behavior Classification**
    - Identify outlier movement patterns (e.g., sudden stops, abnormal acceleration).
    - Categorize vehicle behaviours based on extracted features.
3. **Initial Machine Learning Model Implementation**
    - Explore predictive models to classify vehicle movement.
    - Train a basic model to distinguish between normal and abnormal driving patterns.