Identify Abnormal driving behavior using Spatio-Temporal analysis

AU2140057 - Sanket Shah, AU2140125 - Khushal Trivedi, AU2140165 - Saumya Shah, AU2140233 - Harshal Prajapati.

***Abstract*— Understanding and identifying abnormal driving behavior is critical for strengthening road safety and traffic control systems. While it's common knowledge that drivers act consistently on some sections of the road, there are lots of circumstances in which this isn't the case. The purpose of this research is to identify unusual driving behavior by applying spatio-temporal analysis techniques. The study uses a trajectory dataset, to find abnormal tendencies. By building a binary classifier, trajectories are categorized as "normal" or "abnormal" based on their spatial and temporal characteristics. K-Nearest Neighbors (KNN) algorithm is the method used for classification. By looking at the trajectories, the model finds patterns that may indicate deviations from normal driving behavior. Temporal components like unexpected stops or irregular direction changes, as well as spatial characteristics like speed, acceleration, velocity are important indicators for anomaly identification. This research contributes to transportation efficiency and safety by providing a systematic approach to identify and address potentially hazardous driving habits.**

***Index Terms*— Normal, Abnormal, K-Nearest Neighbors algorithm, spatial, temporal, dataset.**

I. Introduction

In the domain of Transportation and Road Safety; the rules that are based on driving are crucial and essential and along with that detecting abnormal driving behavior beholds a major role and impact to it. Abnormal driving behavior has been a serious issue, and it has impacted society tremendously and requires it to be restricted. Any driving pattern which is not in accordance with rules and regulation of Indian Motor Vehicle Act is considered as abnormal driving and it needs to be identified and should be restricted. Henceforth, with reference to this problem; via using Machine Learning, we tend to solve this problem with the help of binary classifiers, which can detect abnormal driving patterns based on various features like trajectory points, height, altitude, time stamp, velocity, acceleration, etc. In our study, we aim to focus on binary classification using K-Nearest Neighbors (KNN) algorithm and previously using Random Forest Algorithm.

By considering factors like geographical location (longitude and latitude), height, altitude, velocity, bounding box and other topological features, our model becomes adept at distinguishing between normal and abnormal driving behaviors. This approach, combined with spatio-temporal analysis, allows us to emphasize abnormalities in driving behavior and classify them accurately. By doing so, we aim to contribute to safer roads and better transportation systems.

II. Methodolgy

In order to find the anomaly in driving behavior, we had a large data set consisting of features like time stamp, latitude, longitude, height, width and altitude which was analyzed using the K-Nearest Neighbors Algorithm. The detail methodology is as follows:

1) *Data Loading and Processing*
   - Extracts file names and their corresponding labels from a given directory.
   - Splits the data files into training and testing sets.
   - Processes the training data by loading CSV files, extracting features, and preparing the data for training.

2) *Model Training*
   - Performs a grid search to find the best number of neighbors for the K-Nearest Neighbors classifier.
   - Trains the KNN classifier on the training data using the best parameter found.

3) *Model Evaluation*
   - Loads and processes the test data similarly to the training data.
   - Makes predictions on the test data using the trained model and calculates accuracy and it separately predicts the for each file.
   - Identifies incorrectly predicted instances and stores their file names and true labels.

4) *Confusion Matrix Visualization*
   - Generates a confusion matrix.
   - Visualizes the confusion matrix using a heatmap.

5) *Annotated Image Generation*
   - Defines a function to annotate images with connecting lines based on data in CSV files.
   - Reads CSV files containing coordinate data and draws connecting lines between points.
   - Marks the first and last points with circles of different colors.
   - Overlays these annotations on base images and saves the resulting images.

It evaluates the model's performance and provides visualizations to aid in understanding the results. By predicting Abnormal Tracks, driving behavior is distinguished between normal and abnormal driving behaviors by evaluating the performance of the classifier.

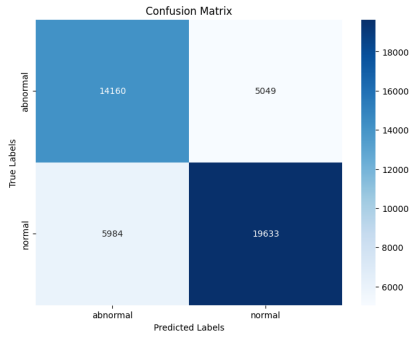

Fig. 1. Results image

III. Results

The accuracy we got is 75.38% and we have also done a grid search to find the number of neighbours for KNN where the accuracy is the highest. The accuracy is low as there is a pattern for which wrong predictions are happening which is mentioned as follows:

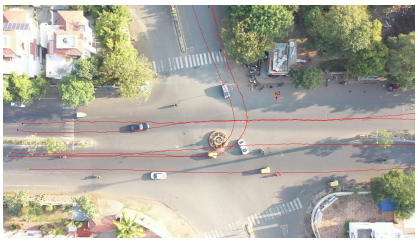

Fig. 2. 10th image and prediction is normal for abnormal

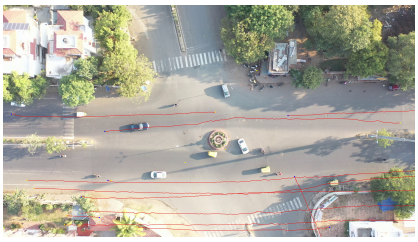

Fig. 3. 10th image and prediction is abnormal for normal

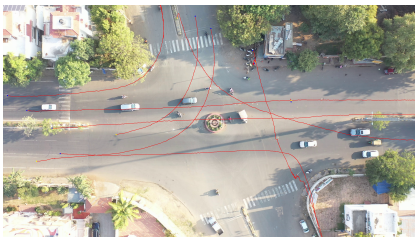

Fig. 4. 11th image and prediction is normal for abnormal

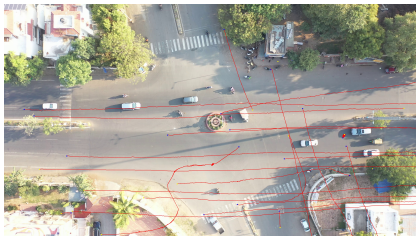

Fig. 5. 11th image and prediction is abnormal for normal

Fig. 6. 12th image and prediction is normal for abnormal

These are the images which provides the reason why the accuracy of our model is low where in image yellow point indicate staring point and black point indicate ending point.

Code:

```
from ctypes import sizeof
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
    , GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import matplotlib.patches as patches

# Step 1: Get all file names and their labels
def get_file_names_with_labels(directory):
    file_names = []
    labels = []

    # Loop through each file in the directory
    for file_name in os.listdir(directory):
        if file_name.endswith('.csv'):
            file_names.append(file_name)
            label = file_name.split('_')[0]  #
    Extract the label before the first underscore
            labels.append(label)

    return file_names, labels

# Example usage:
csv_directory = 'C:/Users/acer/Documents/CSES/Data/'
      # Adjust the directory path as needed  #
    Adjust the image directory path as needed

file_names, labels = get_file_names_with_labels(
    csv_directory)

# Print file names and their corresponding labels
for file_name, label in zip(file_names, labels):
    print("File:", file_name, "Label:", label)

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    file_names, labels, test_size=0.2, random_state
```

```
    =7)

# Load and process training data
train_data = []
train_label = []
for file_name, label in zip(X_train, y_train):
    file_path = os.path.join(csv_directory,
    file_name)
    df = pd.read_csv(file_path)
    features = df.iloc[:, :7].values
    train_data.extend(features)
    train_label.extend([label] * len(features))

# Define the range of neighbors to try
neighbors = list(range(1, 11))  # Trying neighbors
    from 1 to 10

# Perform grid search to find the best number of
    neighbors
param_grid = {'n_neighbors': neighbors}
grid_search = GridSearchCV(KNeighborsClassifier(),
    param_grid, cv=5)
grid_search.fit(train_data, train_label)

# Get the best number of neighbors
best_neighbors = grid_search.best_params_['
    n_neighbors']

# Train the model with the best number of neighbors
knn = KNeighborsClassifier(n_neighbors=
    best_neighbors)
knn.fit(train_data, train_label)

# Store the trained model for later use
models = {"KNeighborsClassifier": knn}

# Load and process test data, make predictions, and
    calculate accuracy
y_pred = []
test_data = []
test_label = []
incorrectly_predicted_files = []  # List to store
    file names that were incorrectly predicted
for file_name, label in zip(X_test, y_test):
    file_path = os.path.join(csv_directory,
    file_name)
    df = pd.read_csv(file_path)
    X_test_data = df.iloc[:, :7].values
    prediction = knn.predict(X_test_data)
    y_pred.extend(prediction)
    test_data.extend(X_test_data)
    test_label.extend([label] * len(X_test_data))

    # Check if prediction is correct
    incorrect_predictions = np.array(prediction) !=
    label
    if np.any(incorrect_predictions):
        incorrectly_predicted_files.append((
    file_name, label))

# Calculate accuracy
print(incorrectly_predicted_files)
accuracy = accuracy_score(test_label, y_pred)
print("K-Nearest Neighbors Classifier Accuracy:",
    accuracy)
print("Best number of neighbors:", best_neighbors)

# Print out the incorrectly predicted instances

for file_name, true_label in
    incorrectly_predicted_files:
    print(file_name)
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Generate confusion matrix
conf_matrix = confusion_matrix(test_label, y_pred)


# Visualize confusion matrix using seaborn heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='
    Blues',
            xticklabels=np.unique(y_test),
    yticklabels=np.unique(y_test))
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
import pandas as pd
import cv2

def draw_connecting_lines(image_path, csv_path, img,
     offset_x, offset_y):
    # Read the CSV file
    df = pd.read_csv(csv_path)

    # Convert center coordinates to integer
    df['center_x'] = df['center_x'].astype(int)
    df['center_y'] = df['center_y'].astype(int)

    # Plot connecting lines and mark points
    for i in range(len(df) - 1):
        point1 = (df.iloc[i]['center_x'] + offset_x,
     df.iloc[i]['center_y'] + offset_y)
        point2 = (df.iloc[i+1]['center_x'] +
    offset_x, df.iloc[i+1]['center_y'] + offset_y)
        cv2.line(img, point1, point2, (0, 0, 255),
    2)

    # Mark the first point in blue
    first_point = (df.iloc[0]['center_x'] + offset_x
    , df.iloc[0]['center_y'] + offset_y)
    cv2.circle(img, first_point, 5, (255, 0, 0), -1)

    # Mark the last point in yellow
    last_point = (df.iloc[-1]['center_x'] + offset_x
    , df.iloc[-1]['center_y'] + offset_y)
    cv2.circle(img, last_point, 5, (0, 255, 255),
    -1)

# Example usage:
path = 'Data/'

# Read the base image
base_image = cv2.imread('sample12.jpg')

# Set initial offsets for placing points from
    different files
offset_x = 0
offset_y = 0
print(len(incorrectly_predicted_files))
for filename, labels in incorrectly_predicted_files:
    if filename.startswith('abnormal_12'):
        draw_connecting_lines(path + filename.
    replace('.csv', '.jpg'), path + filename,
    base_image, offset_x, offset_y)
        # Adjust offsets for the next file
        offset_x += 50  # Adjust as needed
        offset_y += 50  # Adjust as needed

# Save the final annotated image
cv2.imwrite('combined_abnormal_12_image.jpg',
    base_image)
```

IV. Discussion

A. *Performance Evaluation:*

- Our model achieved an accuracy of 75.38% in distinguishing between normal and abnormal driving trajectories.

- While this accuracy demonstrates the effectiveness of the approach, it is essential to delve deeper into the factors contributing to both successful classification and misclassifications.

B. Identification of Limitations:

- Several factors may have influenced the performance of our classification model.
- Notably, the complexity and variability of driving behavior pose significant challenges in accurately capturing abnormal patterns.
- Our model may not have fully accounted for unforeseen events on the road, such as sudden obstacles or erratic behavior of other drivers, which could lead to misclassifications.

C. Analysis of Misclassifications:

- The visual examples provided in the results highlight specific instances where the model misclassified trajectories.
- These cases warrant further investigation to understand the underlying reasons for misclassifications.
- It is evident that certain driving scenarios may not have been adequately represented in the training data, leading to biases or inconsistencies in the classification process.

D. Consideration of Dataset Limitations:

- The limitations of the dataset itself may have also impacted the performance of our model.
- The trajectory data used may not have encompassed the full spectrum of driving behaviors, potentially limiting the generalizability of our findings.
- Furthermore, the choice of features and parameters in the KNN algorithm may not have been optimal for capturing all nuances of abnormal driving behavior.

E. Implications and Future Directions:

- Despite these challenges, our study represents a significant contribution to the field of transportation safety.
- By systematically identifying and addressing potentially hazardous driving habits, our approach has the potential to enhance road safety and traffic control systems.
- Future research endeavors could focus on addressing the identified limitations by incorporating more comprehensive datasets, refining feature selection, and exploring alternative machine learning algorithms to improve classification accuracy.

V. Conclusion

This research presents a robust approach to identifying abnormal driving behavior by integrating spatio-temporal analysis and K-Nearest Neighbors (KNN) algorithm. Leveraging trajectory data and various spatial and temporal features, our model accurately distinguishes between normal and abnormal driving patterns, contributing to road safety and traffic control systems.

Our findings underscore the importance of addressing abnormal driving behavior to enhance road safety and transportation efficiency. By detecting deviations from normal driving, our approach enables proactive intervention strategies to mitigate potential hazards and reduce accident risks. By improving the accuracy and making it more robust this model can contribute more in the domain of transportation and road safety domain.

VI. References

1) Ma, Y., Xie, Z., Chen, S., Qiao, F., & Li, Z. (2023). Real-time detection of abnormal driving behavior based on long short-term memory network and regression residuals. *Transportation Research Part C: Emerging Technologies*, 146, 103983.

2) Huang, X., Yun, P., Wu, S., & Hu, Z. (2023). Abnormal driving behavior detection based on an improved ant colony algorithm. *Applied Artificial Intelligence*, 37(1).

3) Zhang, Y., He, Y., & Zhang, L. (2023). Recognition method of abnormal driving behavior using the bidirectional gated recurrent unit and convolutional neural network. *Physica A: Statistical Mechanics and Its Applications*, 609, 128317.

4) Abosaq, H. A., Ramzan, M., Althobiani, F., Abid, A., Aamir, K. M., Abdushkour, H., Irfan, M., Gommosani, M. E., Ghonaim, S. M., Shamji, V. R., & Rahman, S. (2022). Unusual driver behavior detection in videos using deep learning models. *Sensors*, 23(1), 311.