DYNAMIC TRAFFIC TOLLING PREDICTION USING SOFT COMPUTING

Dr. Chandra Naik[1], Monisha M[2], Nanda C B[3], Priyanka R[4], Rasi K S[5]
[1,2,3,4,5] Department of computer Science, AIET, Mijar, Dakshina Kannada, India

drchandranaik@aiet.org.in[1], monishamanjunath10@gmail.com[2], bangernanda@gmail.com[3], priyacs099@gmail.com[4], rasimulemane2002@gmail.com[5]

Abstract--With the rapid urbanization and increasing vehicular traffic, smart city initiatives are becoming imperative for sustainable urban development. Among the critical challenges faced by urban planners is the management of traffic congestion, which not only affects the efficiency of transportation systems but also leads to environmental degradation and economic losses. Dynamic traffic tolling emerges as a promising solution to alleviate congestion by dynamically adjusting toll prices based on real-time traffic conditions. However, effective implementation of dynamic tolling requires accurate prediction models to anticipate traffic patterns and optimize tolling strategies.

This research project proposed a new method to predict traffic flow in smart cities using computational software. Computational software techniques, including neural networks, fuzzy logic, and genetic algorithms, provide the flexibility to model nonlinear relationships in traffic. The project aims to create powerful predictive models that will predict traffic and congestion levels with high accuracy and efficiency by leveraging the power of software calculations.

Keywords—Deepsort, Pychart, numpy

I. INTRODUCTION

This research involves collecting real-time traffic data from various sources such as sensors, GPS devices and traffic cameras to create comprehensive data for model, training and practicality. Using the data collected, fuzzy electronics will be used to capture the time and location of traffic patterns, while fuzzy logic can incorporate inaccurate and uncertain information into forecasting methods. Additionally, genetic algorithms will be used to optimize models and improve predictions.

The proposed method for estimating traffic congestion will be evaluated through simulations and case studies in an urban area. Performance metrics such as prediction accuracy, computational efficiency, and activation capacity will be analyzed to verify the effectiveness of the proposed method. There is also the possibility that economic activity will be affected by transportation's electric charging strategy, travel behavior and the Environmental sustainability will be analyzed to provide insight to policy makers and urban planners.

Finally, this research focuses on improving transportation systems in smart cities and supporting more efficient and effective solutions. By integrating software technology into traffic forecasting, the project provides a useful tool to improve urban traffic management and create a better city than before.

Fast traffic cost estimation using OpenCV represents a promising approach for smart city traffic management. Integrating OpenCV computer technology, this new solution allows real-time analysis of traffic incidents captured by camera images sent throughout the city. Cameras provide a continuous stream of data that is processed using image processing techniques to identify vehicles, track their movements and determine vehicle speed. Using machine learning models that learn traffic history, the system can predict future traffic levels and dynamically adjust tolls accordingly. This approach not only improves traffic flow by encouraging drivers to change routes or travel times, but also optimizes fares to increase revenue for transportation authorities. Additionally, dynamic fare forecasting contributes to the long-term and resilience of urban transport by promoting sustainable transport and reducing carbon emissions. As cities grapple with the complexities of urbanization and transportation, traffic cost estimation becomes an important tool to improve efficiency, sustainability, and an overall good life for people.

II. MODELS AND LIBRARIES USED

A. YOLOV8

A deep learning algorithm called YOLOv8 was developed for real-time computer vision applications. With its innovative design and advanced technology, YOLOv8 revolutionizes the way products are searched by performing accurate and

efficient product search in real time. Deep learning models like YOLOv8 have become important in many industries, including robotics, driving, and video surveillance. The ability to instantly capture objects impacts security and decision-making processes. The YOLOv8 architecture uses computer vision technology and machine learning algorithms to quickly and accurately identify and locate objects in images and videos. YOLO, short for We Only See Each Other, began in 2015 with the publication of a new research paper called We Only See Each Other: An Exploration of the World of Time. The authors of the study are Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi. YOLO represents a significant advance in real-time object detection, providing an integrated approach that is transforming the field of computer vision. YOLO has grown and changed a lot since its founding. Each continues to improve upon the previous one. The first version, YOLOv1, introduced the concept of real-world object detection by segmenting the input image into a mesh and predicting bounding boxes and category values. This method allows you to capture multiple objects in an image simultaneously. Building on the success of YOLOv1, extensions such as YOLOv2 and YOLOv3 continue to improve the capabilities of the model. These iterations provide ideas such as anchor fields, pyramid objects, and more efficient estimators to improve search results, improving accuracy and speed. Currently, the latest version of the YOLO series is YOLOv8. This release represents the biggest breakthrough in instant asset discovery. YOLOv8 allows researchers and developers to achieve high accuracy and speed in object detection tasks. It has become the first choice for robotics, autonomous driving and video surveillance.

Version	Year	Advancement
YOLOV1	2015	Introduction to grid-based discovery method
YOLOV2	2016	Multi-scale prediction, feature pyramid networks, and anchor boxes are included.
YOLOV3	2018	increased speed and accuracy following the deployment of Darknet-53 and numerous detection scales
YOLOV8	2021	A cutting-edge deep learning model called YOLOv8 was created for computer vision applications requiring real-time object recognition

B. DEEP SORT

DeepSORT (Deep Simple Online Real-Time Tracking with Deep Relevance Metrics) is a tracking algorithm primarily aimed at computer vision and object detection tasks, especially in the field of multi-vehicle inspection (MOT). Although DeepSORT itself is not used directly for traffic prediction, it can be integrated into general systems for traffic prediction and management. Here's how DeepSORT is used to predict traffic: Searching and tracking objects: DeepSORT is good at tracking multiple objects in a real-time video stream. Within the scope of traffic forecasting, it can be used to track vehicles in traffic cameras or recorded images. DeepSORT accurately tracks vehicles, providing valuable information about vehicle movement, alignment, speed and interaction necessary to identify vehicle patterns and take action. Data collection and analysis: DeepSORT creates a continuous stream of data about movement and interaction. car character. This data can be collected and analyzed to identify incidents, hot spots, and traffic patterns. By processing time tracking data, traffic engineers and planners can gain insight into traffic behavior, rush hours, accidents and other important factors affecting traffic forecasting. Integration with prediction models: Although DeepSORT itself does not make predictions, its output (traffic data) can be fed into prediction models. These models can use a variety of techniques such as machine learning, statistical analysis, or mathematics to

predict future traffic conditions. The traffic monitoring provided by DeepSORT works as a predictive model, allowing them to make predictions about traffic volumes, congestion levels, travel times, and other inaccuracies. Real-time traffic management: DeepSORT can provide instant feedback about real-time traffic. By integrating with traffic forecast models, it can contribute to real-time traffic management by providing up-to-date information on vehicle movements and congestion levels. Transportation authorities can use this information to correct traffic problems, reroute traffic, or take other measures to reduce congestion and improve traffic flow. In summary, although DeepSORT itself is not a traffic predictor, it plays an important role in providing traffic predictions. Accurate and reliable tracking data can be used through traffic prediction models and systems. By integrating DeepSORT into traffic forecasting projects, cities and transportation authorities can improve their ability to effectively predict and manage traffic, ultimately improving urban mobility and reducing constraints.

C. OPEN CV

OpenCV (the open source computer vision library) is a powerful tool that can be used for many computer vision applications, including traffic prediction in smart cities. Applications of OpenCV for vehicle prediction include: Vehicle Detection and Tracking: OpenCV provides features and methods for detection and tracking that are important for emergency vehicle maintenance. Methods such as Haar stage, Histogram of Directed Gradients (HOG), or deep learning methods such as Convolutional Neural Networks (CNN) can be used to detect vehicles in image or video streams. Once discovered, OpenCV can be used to monitor vehicle power over time and provide valuable information for vehicle prediction. Vehicle analysis: Using OpenCV, you can extract and analyze vehicle models by analyzing the movement and behavior of detected and tracked vehicles. OpenCV can help determine parameters such as vehicle speed, acceleration, velocity, lane availability, and traffic levels. This information is needed to understand current traffic conditions and predict future traffic conditions. Traditional detection: OpenCV allows you to detect suspicious or unusual situations such as traffic accidents, traffic jams, and sudden changes in the road. Using OpenCV, crashes can trigger alerts or notify transportation authorities so they can take quick action to reduce disruption and improve traffic flow. Data collection and prioritization: OpenCV helps you collect, prioritize, and optimize the traffic data needed to train predictive models. Provides the ability to read and process video, video, and real-time data from the vehicle's CCTV camera. OpenCV can also be used in data augmentation techniques to increase the diversity and performance of predictive modeling data. Integration with machine learning platforms. Develop and deploy predictive analytics through seamless integration with machine learning platforms such as OpenCV, TensorFlow, and PyTorch. By combining OpenCV's detection and tracking capabilities with machine learning algorithms, you can train a predictive model to predict future traffic conditions based on historical data and analysis time. Visualization and User Interface: OpenCV provides data visualization and traffic prediction tools to help users better understand traffic management. OpenCV's rendering capabilities can be used to visualize real-time traffic data, including traffic, traffic maps, and forecasts, allowing drivers and operators to effectively monitor and control traffic. Overall, OpenCV can be used as a multi-functional tool for smart city traffic prediction, traffic data visualization, monitoring, analysis and visualization, which is very important for the development. Improve city and transportation information. Motion control. Important advice.

D. FUZZY LOGIC

Fuzzy logic can be used for traffic prediction in smart cities to improve accuracy and efficiency. Here is how fuzzy logic can be used in this case: Data Fusion: Smart cities collect data from various sources such as traffic cameras, road-embedded sensors, GPS devices and mobile applications. Fuzzy logic can be used to combine these different data, often in different formats and from different sources, into a coherent model. Fuzzy logic allows the integration of disparate data and provides a framework for managing uncertainty and ambiguity in data. Rule-Based Traffic Forecasting: Fuzzy logic allows the creation of rule-based methods that capture the relationship between variables affecting traffic, such as time of day, weather, special conditions, roads and history of car models. These rules can be

created based on expert knowledge or inferred from data using machine learning. Occupational Uncertainty: Traffic forecasting involves uncertainty due to factors such as traffic flow, unexpected events, and data. Fuzzy logic is good for dealing with uncertainty because it allows the use of different expressions and fuzzy sets to represent fuzzy or uncertain information. By incorporating fuzzy logic techniques, the traffic prediction process can be evaluated based on uncertain or incomplete information. Adaptive Control: Fuzzy logic can be used to develop a traffic control system that instantly adjusts the signal and configuration algorithms based on traffic predictions. These systems can improve traffic efficiency by allocating resources and shifting traffic to reduce congestion and reduce travel time. Feedback mechanism: Fuzzy logic can be combined with feedback mechanisms to monitor and update the traffic prediction model based on the recommendation. By comparing traffic prediction models to actual traffic data, the system can adjust its predictions over time and improve its accuracy. Integration with other smart cities: Fuzzy logic-based traffic prediction can be integrated with other smart cities such as public transportation, nature services emergency and urban planning. This integration allows decision-making and coordinated response to transportation issues, resulting in greater use and better quality of life for city residents. Overall, fuzzy logic provides a flexible and robust power for traffic prediction in smart cities, helping to increase flexibility, accuracy and efficiency in management.

E. PYTROCH

PyTorch is a deep learning framework for predicting traffic congestion in smart cities to use the power of neural networks more accurately and efficiently. Here's how PyTorch is used: Deep learning models: PyTorch, convolutional neural networks (CNN), recurrent neural networks (RNN), Long Short Term Memory Network (LSTM) and Transformer. These models can be trained on traffic history to understand complex patterns and relationships between variables affecting traffic. Temporary data processing: Traffic forecasting involves analysis of real-time data such as traffic flow, congestion levels and travel over time. PyTorch's integrated modeling support makes it ideal for efficient processing of physical data. For example, RNNs and LSTMs are often used in sequential data processing and can capture the temporal patterns expected in traffic patterns. Learning by Feature Representation: PyTorch allows training deep learning models to learn features from raw objects. This feature is particularly useful in traffic forecasts, where the relationship between the input (such as traffic volume, weather, time of day) and the forecast (comparison, i.e. future traffic) can be unpredictable and difficult. Model customization and experimentation: PyTorch's dynamic computational graph and flexible architecture allow researchers and practitioners to customize models and experiment with different network architectures, power loss, and optimization algorithms. This change allows the development of models suitable for specific needs and data sets in dynamic traffic prediction studies. Parallel and distributed computing: PyTorch supports parallel and distributed computing, allowing deep learning models to be trained on large data sets. Scale the traffic dataset. This capability is important to solve the challenge of training complex neural networks on big data, often encountered in car prediction applications in smart cities. Integration with other Python libraries: PyTorch integrates with other Python libraries commonly used in data science and machine learning, such as NumPy, pandas, scikit-learn and Matplotlib. This integration supports data preprocessing, infrastructure development, evaluation, and operational visualization, which are key components of the predictive pipeline. Overall, PyTorch provides a powerful and flexible tool for building traffic prediction models in smart cities, allowing researchers and practitioners to use deep learning to improve the accuracy and efficiency of traffic prediction.

F. NUMPY :

NumPy is a simple Python computational package that can be used to predict traffic in smart cities in a variety of ways: Data representation and management: NumPy provides powerful data such as arrays and matrices suitable for representing traffic, including speed. and volume. Such information. This data model simplifies operations such as data preprocessing, extraction, and modeling, allowing more data to be stored and managed. Mathematics: NumPy provides many mathematical functions that can be used to analyze and process traffic data and create

predictive models. Functions such as mean, median, standard deviation, and percentage are used to record traffic characteristics and identify patterns. Additionally, NumPy's linear algebra functions support matrix operations, eigenvalue decomposition, and ratio value decomposition, which are commonly used in estimation algorithms. Statistical Analysis: NumPy provides statistical functions to analyze the distribution, correlation and patterns of traffic data. Features such as histogram calculation, probability density estimation, and regression analysis can be used to examine the relationship between different features and identify predictive features. Statistical analysis using NumPy can help understand underlying patterns in traffic data and guide the development of predictive models. Time series analysis: Traffic data is usually presented over a period of time; This makes time analysis crucial for predicting future traffic. NumPy provides a full set of scheduling tools, including sliding window operations, delay operations, and automatic analysis. This tool can extract physical patterns and patterns from historical traffic data that can be used to train predictive models. Evaluation and Validation: NumPy supports evaluation and validation by providing functions to calculate performance metrics such as squared error, mean error, and R-squared score. These metrics can measure the accuracy and reliability of predictions and help determine the most effective prediction models for smart city traffic management. Integration with machine learning libraries: Seamlessly integrates with popular machine learning libraries such as NumPy, scikit-learn, and TensorFlow to ensure compatibility with various prediction algorithms. NumPy arrays can be used as input data for machine learning models; It supports the development of prediction models using techniques such as regression, classification, time estimation and grouping. In summary, NumPy plays an important role in predictive modeling by providing simple data representation, operations, arithmetic, statistical analysis, time analysis, models and integration with machine learning libraries. for smart cities. By leveraging the power of NumPy, transportation engineers and data scientists can create accurate and cost-effective predictive models to improve traffic management and improve mobility in the city.

III. DATASETS AND RESULT

Utilizing OpenCV's computer vision capabilities, our dynamic traffic tolling prediction system effectively analyses real-time traffic patterns. Through continuous monitoring of traffic cameras, the system processes live video feeds to extract relevant data for predicting congestion levels. For predicting the traffic, we use videos figure 3.1 shows one of the frame from dataset video.

Figure 3.1 image taken from input video

The yolo algorithm is used to recognising the objects in the video. First the video is converted into number of frames. There are totally 5396 frames are generated from the video. Figure 3.2 will shows the one of the frame from the video.

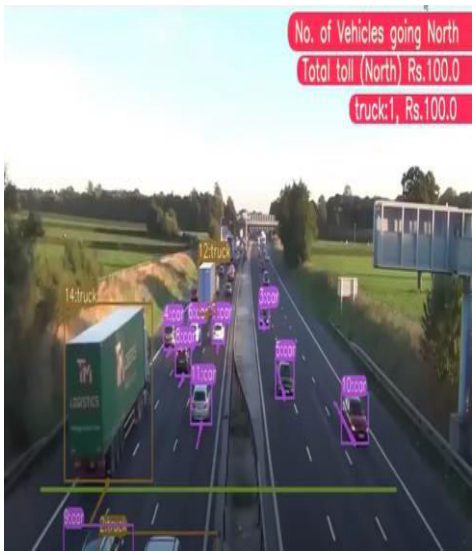

Figure 3.2 frame generated from the video

We will provide one user interface so that user can get information about the traffic then they will decide in which route they can take. The Figure 4.3 will show that the

toll rate for different toll rates for different type of vehicles. Toll rate will vary according to the number of vehicles.

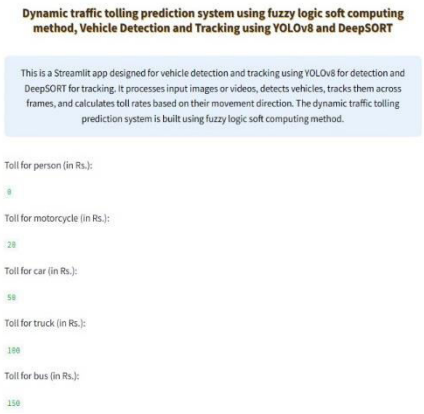

Figure 3.3 user interface image

IV. CONCLUSION

In summary, dynamic traffic toll estimation using software calculation methods provides a good approach for controlling traffic congestion and optimizing toll strategy in urban environment. Through the integration of machine learning, fuzzy logic, neural networks, and other software computing technologies, accurate predictions of traffic patterns and demand can be made; This allows authorities to adjust call rates to achieve desired results, such as reducing congestion and improving air quality. and increase revenue. . One of the main benefits of using software in traffic forecasting is the ability to handle existing relationships, uncertainties and non-linearities in the traffic system. Using historical traffic data, real-time monitoring data, and other factors such as weather and conditions, the calculation model can be modified and learn from change; this leads to robust and reliable predictions. In addition, the use of electronic transmission systems in accordance with the standards used in the sector will benefit passengers and transportation convenience. Passengers can experience fewer trips and more travel time, while transportation organizations can optimize infrastructure use, reduce environmental impact and generate revenue. However, there are some challenges and considerations that need to be addressed when implementing dynamic phone number prediction. These include the need for accurate and complete data collection, improved design standards, consideration of equity and fairness issues in pricing, and integration with transportation and existing rights. In summary, dynamic traffic using software calculation methods for fare estimation has great potential to transform urban transportation into greater efficiency, sustainability and resilience. Continued research, innovation and collaboration between academic, business and government stakeholders are crucial to reap the full benefits of this approach and solve complex transportation problems in today's cities.

REFERENCES

[1] V. Kovalev, A. Kalinovsky, and V. Liauchuk, ‘‘Deep learning in big image data: Histology image classification for breast cancer diagnosis,’’ in Proc. Int. Conf. BIG DATA Adv. Anal. (BSUIR), Jun. 2016, pp. 44–53.

[2] E. Strubell, A. Ganesh, and A. McCallum, ‘‘Energy and policy considerations for deep learning in NLP,’’ in Proc. 57th Annu. Meeting Assoc. Comput. Linguistics, 2019, pp. 3645–3650.

[3] H. Mao, S. Yao, T. Tang, B. Li, J. Yao, and Y. Wang, ‘‘Towards realtime object detection on embedded systems,’’ IEEE Trans. Emerg. Topics Comput., vol. 6, no. 3, pp. 417–431, Jul. 2018, doi: 10.1109/TETC. 2016.2593643.

[4] J. A. Carballo, J. Bonilla, M. Berenguel, J. Fernández-Reche, and G. García, ‘‘New approach for solar tracking systems based on computer vision, low cost hardware and deep learning,’’ Renew. Energy, vol. 133, pp. 1158–1166, Apr. 2019.

[5] B. Moons, D. Bankman, and M. Verhelst, ‘‘Embedded deep learning,’’ in Algorithms, Architectures and Circuits for Always-on Neural Network Processing. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-319- 99223-5.

[6] K. Rungsuptaweekoon, V. Visoottiviseth, and R. Takano, ‘‘Evaluating the power efficiency of deep learning inference on embedded GPU systems,’’ in Proc. 2nd Int. Conf. Inf. Technol. (INCIT), Nov. 2017, pp. 1–5, doi: 10.1109/INCIT.2017.8257866.

[7] G. Plastiras, C. Kyrkou, and T. Theocharides, ‘‘Efficient ConvNet-based object detection for unmanned aerial vehicles by selective tile processing,’’ in

Proc. 12th Int. Conf. Distrib. Smart Cameras, Sep. 2018, pp. 1–6.
[8] O. Rukundo, ‘‘Effects of image size on deep learning,’’ 2021, arXiv:2101.11508.