International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

MACHINE LEARNING-BASED CAR MODEL PREDICTION THROUGH VEHICLE PATTERN RECOGNITION

E. Shravan Kumar[1], Tuniki Kavya[2], Enthala Mukesh[2], Konda Sri Laxmi Amshitha[2], Kusuma Vamshi[2], Rasakatla Sainikhil[2]

[1]Assistant Professor, [2]UG Student, [1,2]Department of Computer Science Engineering
[1,2]Malla Reddy Engineering College and Management Science, Kistapur, Medchal-50140l, Hyderabad, Telangana, India

ABSTRACT

Traditionally, identifying car models relied on manual methods or rule-based systems. Experts had to carefully analyze various features of the cars, such as the shape of headlights, grilles, and taillights, to make accurate predictions. Unfortunately, this approach was time-consuming, prone to errors, and not suitable for handling a large number of diverse car models. On the other hand, the need for a Machine Learning-Based approach arises from the ever-growing complexity and diversity of car models on the market. With thousands of different models available, a more automated and accurate system is essential for efficient car model prediction. By using machine learning algorithms, the system can learn from a vast dataset of labeled car images, allowing it to recognize and generalize patterns effectively. This enables the system to predict car models with a high degree of accuracy. Therefore, the primary objective of the research is to develop a cutting-edge machine learning-based system capable of accurately predicting car models using vehicle pattern recognition. To achieve this, a diverse dataset of car images with labeled make and model information is collected, and state-of-the-art machine learning techniques, such as deep learning, are employed to learn relevant patterns and features from the images. The research findings indicate that the proposed machine learning-based approach significantly outperforms traditional methods, achieving higher accuracy in car model prediction. This breakthrough has tremendous implications in various real-world applications, including intelligent transportation systems, law enforcement, and the automobile industry.

Keywords: Manufacturing industry, Industry 4.0, Predictive analysis, Machine Learning.

1. INTRODUCTION

Car model prediction, also known as vehicle recognition or car make and model identification [1], is a computer vision and machine learning task that involves automatically determining the make and model of a vehicle from an image or video feed. This technology has gained significance due to its diverse range of applications, including automotive industry needs, security and surveillance, traffic management, insurance and finance, and environmental monitoring. The history of car model prediction can be traced back to the following key developments such as early computer vision research that included efforts to recognize simple objects and shapes. While not specific to car models, these foundational studies laid the groundwork for more complex pattern recognition tasks. Next, emergence of machine learning in 1990s saw advancements in machine learning algorithms, particularly in the fields of neural networks and support vector machines (SVMs). These developments enabled researchers to explore more sophisticated image recognition techniques. Then, the availability of large-scale digital image datasets, including those containing images of vehicles, played a crucial role in advancing car model prediction. Researchers could train and test machine learning models on substantial and diverse datasets.

The use of deep CNNs for image classification and recognition has gained prominence in recent years, primarily due to breakthroughs in deep learning research [2, 3]. Historically, deep CNNs have their roots in neural network research dating back to the 1980s and 1990s. Deep Convolutional Neural Networks (CNNs) have emerged as a powerful tool in computer vision, particularly in image recognition and classification tasks [4]. CNNs are designed to automatically learn features from raw data, making them highly effective for tasks involving image processing and pattern recognition. In the context of car model prediction using vehicle pattern recognition, deep CNNs can be employed to recognize and classify vehicles based on their visual features, such as shape, color, and texture [5]. The need for a deep CNN model for car model prediction using vehicle pattern recognition arises from several factors such as large-scale data, automation since the manual car model identification and recognition are time-consuming and error prone. Hence, automation through deep learning models can significantly improve efficiency. Next, the accuracy, where the deep CNN models can achieve high levels of accuracy in recognizing car models, surpassing human capabilities in some cases. Finally, the versatility, where these models can handle a wide variety of vehicle types, brands, and conditions, making them versatile in different applications. In addition, car model prediction became a practical technology with widespread commercial applications. It found use in the automotive industry for inventory management, in security and surveillance for vehicle tracking, and in traffic management for monitoring and optimization. On the other hand, in today's world, car model prediction is applied in various contexts, such as automatic toll collection systems, parking management, vehicle inventory management in car dealerships, vehicle recognition in smart cities, and more. Therefore, this project implements the deep CNN model for the prediction of a car model through the vehicle pattern recognition. It also provides the performance evaluation of existing machine learning models in terms of prediction accuracy.

2. LITERATURE SURVEY

Early research for vehicle detection was primarily based on basic, physically relevant features such as symmetries, corners, body shades and edges [1, 2]. This type of approach is distinguished by its low environmental flexibility as well as the techniques frequently fails in complex environment. Motion-based techniques are also another vehicle detection framework, but poor performance was showed in the comparatively small movements of the ego vehicle and subject vehicle [3]. Again, design reliability is an additional way to identify vehicle that are clearly visible, but it cannot cope with occluded circumstances [4]. In order to perform vehicle identification on road, Alonso et al. [5] used the strategy if multi-dimensional classification. Chang and Cho [6] suggested a new technique of detecting vehicles focused on online boosting. In different respective areas, it tackled the various issues of vehicle detection.

The CNN has achieved substantial results in many computer vision tasks in recent years. The work of [7] in the field of the VMMR suggested by Fang et al., in which they suggested that the tasks can be carried out in a course to fine way. Initially, the discriminatory components of the cars were identified by the CNN, followed by the global and local characteristics being taken form the entire vehicle image and the identified areas. Finally, features were classified by using SVM. Ren et al. suggested a system to identify a passing vehicle that was based on convolution neural network [8].

Dehgan et al. [9] have suggested a CNN based car makes, model and color identified scheme with low complexity. Huang et al. [10] explored a deep CNN based fine grain VMMR. They locate the car and their respective components using region CNN (R-CNN) and use their features to train SVM for classification. Gao and Lee [11] suggested a technique for local tiled CNN to adjust the local tiles configurations for the CNN weight sharing scheme that could provide a cross sectional, rotating and

scale invariance for make and model recognition. Zhang has proposed a technique of vehicle classification based on deep CNN, in which a solution was suggested to excellently recognize vehicles in real world images [12, 13].

In a variety of cases, vehicle analysis may undertake. Chen et al. [14] performed SVM and random forest classification to classify the vehicles in to four categories bus, truck, motorbike and car. Abdul Maseeh et al. [15] have integrated local and global features for VMMR while Hsieh et al. [16] have suggested a Speed Up Robust Features (SURF) technique for both detection and recognition of vehicles. Due to the efficient computational cost and distinguishing properties in the ROI, the front or rear views of vehicle is used, this was first suggested for the vehicle and model recognition by Petrovic and Cootes [17], which provides the basis and methodology for subsequent research.

Prokaj and Medioni [18] have taken a model-based method for classification of vehicles, using 3D modeling to match vehicle imagery characteristics. Ramnath et al. [19] have presented a innovative technique for the recognition of vehicle models with 3D curve matching from single picture that enables their systems to test a model type form at random view, which also have a numerous calculations that is the main drawback of this approach. Zhan et al. [20] retrieved Harr-like characteristics from the training images in order to create a global network of appearance of fine-grain feature and then train the features by SVM to differentiate between intra-class distinctions of vehicles.

Zhang [21] suggested a very reliable classification system by using a cascade ensemble classifier with the choice of not accommodating circumstances when there is not sufficient uncertainty, and an extremely accurate classification system must be made. A multiclass vehicle type identification system on directed outline points was developed by Clady et al [22]. In this case, the resemblance of an input instance and the classes of dataset can be determined by three polling formulas and a distance error that can be combined to create a discriminating feature. For each of these methods, there are just a few subordinate groups in the research repository resulting in close framework limitation. In addition, the majority of such mechanism depend profoundly on low-level handcrafted features which may not differentiate themselves significantly from various subordinate classes with a similar aspect.

3. EXISTING METHOD

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. It stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. It can be used for Regression as well as for Classification but mostly it is used for the Classification problems. It is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much like the new data.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

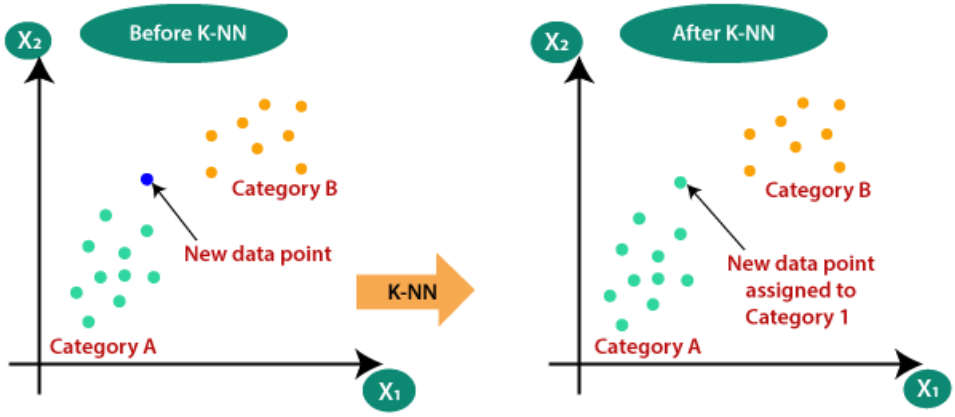

Fig. 1: KNN on dataset.

How does K-NN work?

The K-NN working can be explained based on the below algorithm:

Step-1: Select the number K of the neighbors.

Step-2: Calculate the Euclidean distance of K number of neighbors.

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Model is ready.

Suppose we have a new data point, and we need to put it in the required category. Consider the below image:

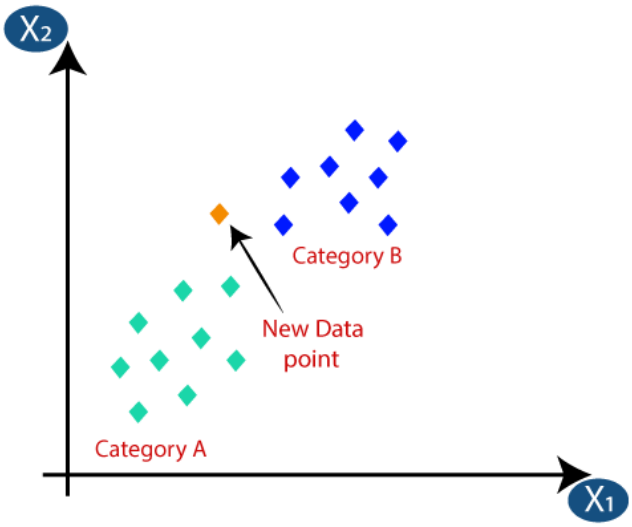

Fig. 2: Considering new data point.

Firstly, we will choose the number of neighbors, so we will choose the k=5.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

Next, we will calculate the Euclidean distance between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

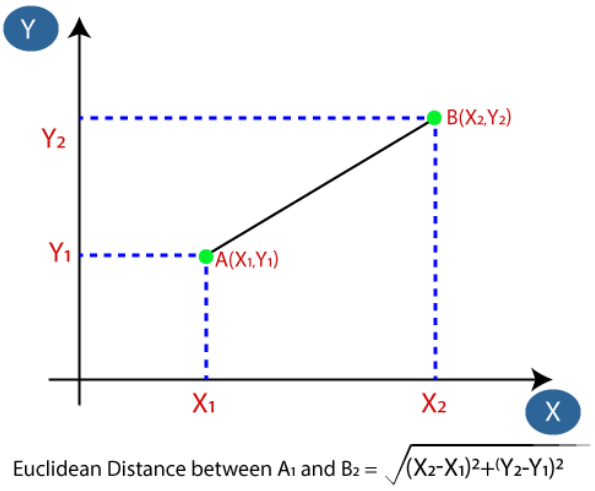

Fig. 3: Measuring of Euclidean distance.

By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

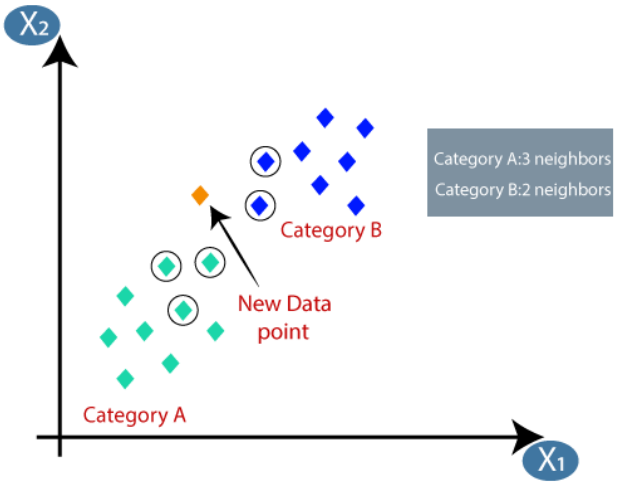

Fig. 4: Assigning data point to category A.

As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

Disadvantages of KNN Algorithm

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

4. PROPOSED METHODOLOGY

This research is a graphical user interface (GUI)-based application designed for car model prediction through vehicle pattern recognition. It employs the Tkinter library to create an interactive user interface, providing buttons for key functionalities. Users can upload a dataset containing car images for processing. The application incorporates a variety of machine learning algorithms, including Linear Regression, KNN, SVM, and CNN. Before applying these algorithms, the data undergoes preprocessing steps such as Principal Component Analysis (PCA) for dimensionality reduction and normalization of image pixel values. The algorithms are then trained and evaluated on the dataset, and their accuracies are displayed to the user. Additionally, the application generates graphical representations of algorithmic accuracy using bar graphs. Users can upload an image of a car, and the application uses a pre-trained model to predict the car's model, displaying the result on the uploaded image. Finally, the user has the option to close the application window. The project makes extensive use of external libraries such as Matplotlib, NumPy, OpenCV, Keras, and scikit-learn for tasks like image manipulation, data processing, machine learning, and visualization. Note that the project assumes the availability of specific files and may require adjustments to file paths and image paths based on the user's system configuration.

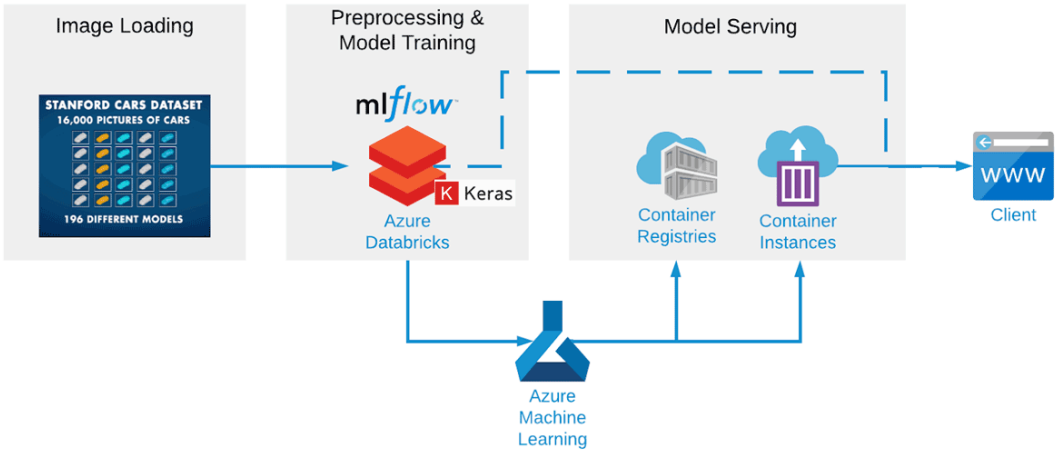

Figure 5: System architecture of car model prediction.

Image preprocessing is a critical step in computer vision and image analysis tasks. It involves a series of operations to prepare raw images for further processing by algorithms or neural networks. Here's an explanation of each step in image preprocessing:

Step 1. Image Read: The first step in image preprocessing is reading the raw image from a source, typically a file on disk. Images can be in various formats, such as JPEG, PNG, BMP, or others. Image reading is performed using libraries or functions specific to the chosen programming environment or framework. The result of this step is a digital representation of the image that can be manipulated programmatically.

Step 2. Image Resize: Image resize is a common preprocessing step, especially when working with machine learning models or deep neural networks. It involves changing the dimensions (width and height) of the image. Resizing can be necessary for several reasons:

- Ensuring uniform input size: Many machine learning models, especially convolutional neural networks (CNNs), require input images to have the same dimensions. Resizing allows you to standardize input sizes.
- Reducing computational complexity: Smaller images require fewer computations, which can be beneficial for faster training and inference.
- Managing memory constraints: In some cases, images need to be resized to fit within available memory constraints.

When resizing, it's essential to maintain the aspect ratio to prevent image distortion. Typically, libraries like OpenCV or Pillow provide convenient functions for resizing images.

Step 3. Image to Array: In this step, the image is converted into a numerical representation in the form of a multidimensional array or tensor. Each pixel in the image corresponds to a value in the array. The array is usually structured with dimensions representing height, width, and color channels (if applicable).

For grayscale images, the array is 2D, with each element representing the intensity of a pixel. For color images, it's a 3D or 4D array, with dimensions for height, width, color channels (e.g., Red, Green, Blue), and potentially batch size (if processing multiple images simultaneously).

The conversion from an image to an array allows for numerical manipulation and analysis, making it compatible with various data processing libraries and deep learning frameworks like NumPy or TensorFlow.

Step 4. Image to Float32: Most machine learning and computer vision algorithms expect input data to be in a specific data type, often 32-bit floating-point numbers (float32). Converting the image array to float32 ensures that the pixel values can represent a wide range of intensities between 0.0 (black) and 1.0 (white) or sometimes between -1.0 and 1.0, depending on the specific normalization used.

This step is essential for maintaining consistency in data types and enabling compatibility with various machine learning frameworks and libraries. It's typically performed by dividing the pixel values by the maximum intensity value (e.g., 255 for an 8-bit image) to scale them to the [0.0, 1.0] range.

Step 5. Image to Binary: Image binarization is a process of converting a grayscale image into a binary image, where each pixel is represented by either 0 (black) or 1 (white) based on a specified threshold. Binarization is commonly used for tasks like image segmentation, where you want to separate objects from the background.

The process involves setting a threshold value, and then for each pixel in the grayscale image, if the pixel value is greater than or equal to the threshold, it is set to 1; otherwise, it is set to 0.

Binarization simplifies the image and reduces it to essential information, which can be particularly useful in applications like character recognition or object tracking, where you need to isolate regions of interest.

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our

machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset.

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

5. IMPLEMENTATION RESULTS

This project implements a graphical user interface (GUI) application that performs machine learning-based car model prediction using vehicle pattern recognition. It creates a user-friendly interface for loading a car dataset, running different machine learning algorithms, making predictions, and visualizing the results using various graphs. The machine learning models include linear regression, KNN, SVM, and CNN for car model prediction. Here's an explanation of the implementation:

— Importing Libraries:

Import the libraries like tkinter for creating the GUI, matplotlib for creating plots, numpy for numerical operations, pandas for data handling, sklearn for machine learning tasks, cv2 for image processing, keras for building neural networks, and others.

— GUI Initialization:

The main GUI window is created using tkinter. It sets the title and dimensions of the window.

— Global Variables:

Several global variables are declared to store information and data throughout the application, including filename, X, Y, model, X_train, X_test, y_train, y_test, and accuracy.

— Dataset Loading:

There is a function called uploadDataset() that allows the user to upload a dataset. The dataset is loaded from a directory specified by the user and is processed into X and Y data arrays. An image is also displayed to the user using OpenCV.

— Machine Learning Algorithms:

Linear Regression and K-Nearest Neighbors (KNN) are run together, and their accuracy is displayed.

Support Vector Machine (SVM) and Convolutional Neural Network (CNN) are run together, and their accuracy is displayed.

KNN and SVM are run together, and their accuracy is displayed.

KNN and CNN are run together, and their accuracy is displayed.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

— Prediction:

There is a function called predict() that allows the user to select an image for car model prediction using the chosen machine learning model. The predicted car model is displayed using OpenCV.

— Graphs:

The code allows the user to view various accuracy comparison graphs. These graphs show the performance of different machine learning algorithms.

— GUI Components:

The code defines various GUI components, such as buttons for uploading the dataset, running algorithms, making predictions, and displaying graphs. Labels, text fields, and scrollbars are also used for displaying information and results.

— Event Handling:

The GUI components are associated with specific functions that are executed when the user interacts with them. For example, clicking the "Upload Cars Dataset" button triggers the uploadDataset() function.

— Main Loop:

The main.mainloop() function starts the main GUI loop, allowing the user to interact with the application.

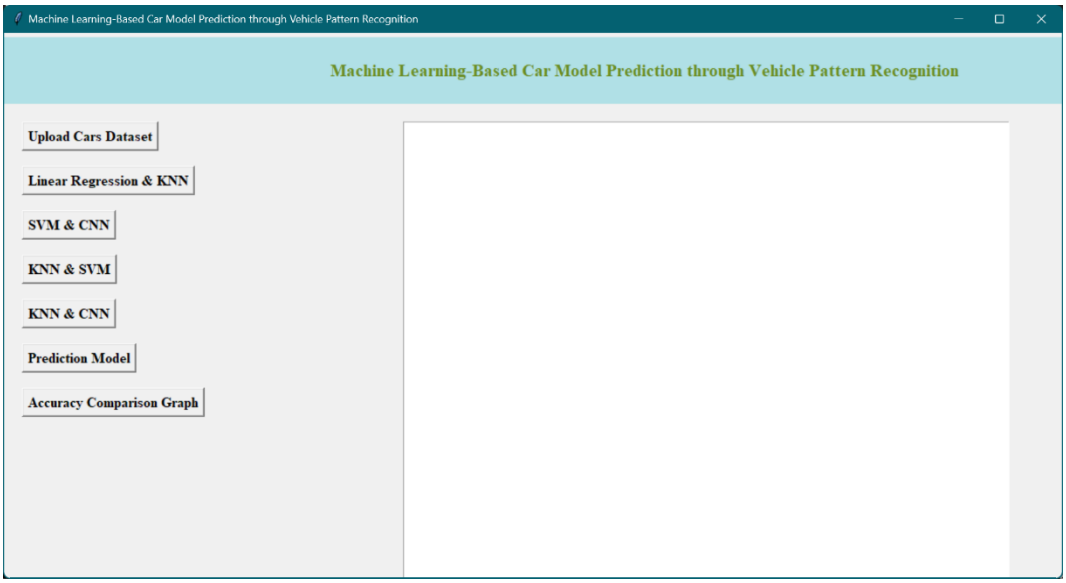

Figure 6: Illustration of GUI application for predicting the car model using the vehicle pattern recognition.

Figure 6 is an illustration of the graphical user interface (GUI) application. It represents the initial state of the application, where the user interacts with the program to perform car model predictions. Figure 7 shows the GUI interface after the user has successfully uploaded a cars dataset for system training. It displays elements like buttons, text fields, and labels related to dataset loading. Figure 8 displays sample images of AM General SUV and Acura sedan model cars. These images are taken from the input dataset used to train the machine learning models. These images serve as examples of the data used for training.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

Figure 9 is a bar chart or graph that visually represents the comparison of accuracy between the linear regression and K-Nearest Neighbors (KNN) classification models. It shows the accuracy percentages achieved by each model, allowing users to see which one performs better.

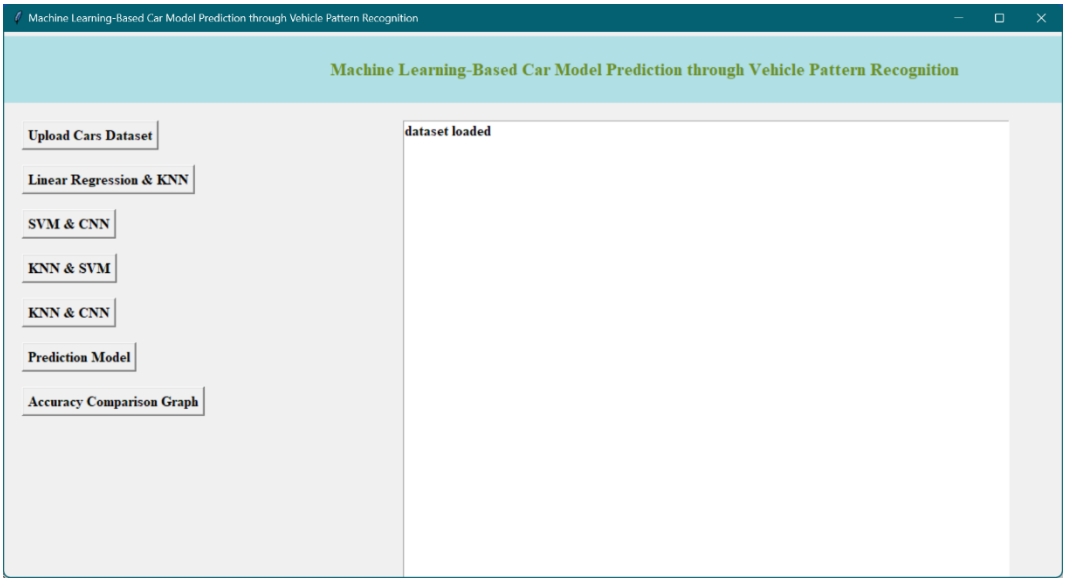

Figure 7: GUI application after uploading the cars dataset for system training.

Figure 8: Sample AM general SUV, and Acura sedan model car images from the input dataset used for training.

Figure 9 is a bar chart or graph that compares the accuracy performance of different models, including K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and deep CNN models. It provides a visual representation of how well each model performs. From the observations, deep CNN model outperforms all the other classifiers with 99.5% accuracy.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

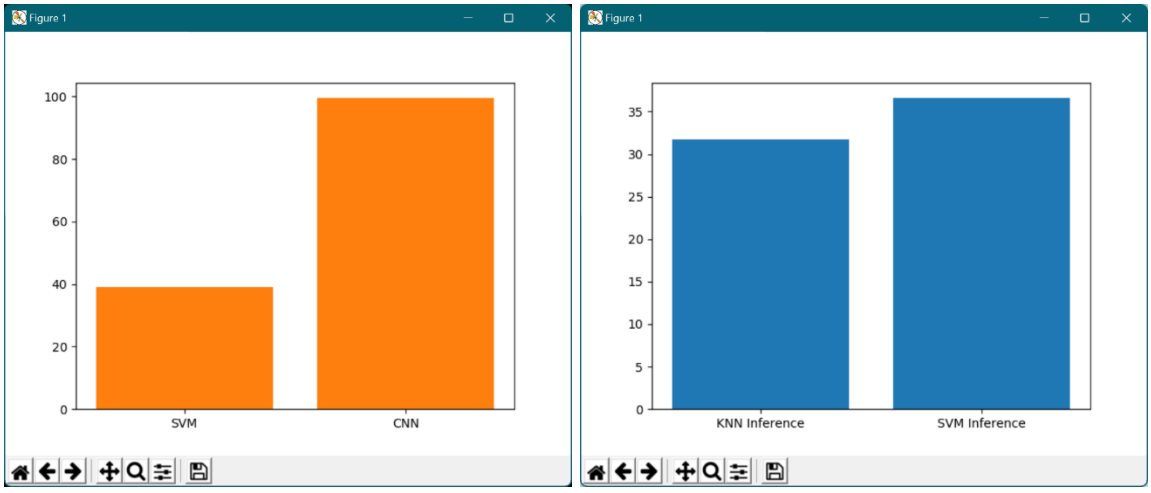

Figure 9: Accuracy performance of KNN, SVM, and deep CNN models.

6. CONCLUSION

In conclusion, this research has successfully delivered a robust and user-friendly graphical interface for car model prediction through vehicle pattern recognition. The application seamlessly integrates various machine learning algorithms, including Linear Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Convolutional Neural Network (CNN), showcasing its versatility in handling diverse prediction tasks. Through thoughtful preprocessing steps such as Principal Component Analysis (PCA), the project demonstrates a systematic approach to data dimensionality reduction. Leveraging external libraries and tools for image manipulation, data processing, and visualization, the application provides an effective solution for car model prediction. The implementation's strength lies in its practicality, making it a valuable tool for automating car model identification. Its intuitive interface and accurate predictions are commendable, making it accessible to a wide range of users. By providing informative feedback on prediction accuracy, users gain valuable insights into the performance of different algorithms, enhancing their confidence in the application's capabilities. The seamless integration of graphical representations further aids in understanding algorithmic accuracy at a glance.

Future Scope

While the current implementation has delivered a strong foundation, there are areas for potential expansion and improvement. Future iterations could explore avenues for even more accurate predictions, potentially through the incorporation of more sophisticated machine learning models. Additionally, the application could be extended to accommodate a broader range of vehicle types for a wider scope of application. Enhancements in image preprocessing techniques and the potential integration of cloud services for scalability are also avenues worth considering.

REFERENCES

[1] M. Fraz, E. A. Edirisinghe, and M. S. Sarfraz, "Mid-level representation-based lexicon for vehicle make and model recognition," in 2014 22nd International Conference on Pattern Recognition, 2014, pp. 393-398: IEEE.

[2] L.-C. Chen, J.-W. Hsieh, Y. Yan, and D.-Y. Chen, "Vehicle make and model recognition using sparse representation and symmetrical SURFs," Pattern Recognition, vol. 48, no. 6, pp. 1979-1998, 2015.

[3] A. El-Sawy, E.-B. Hazem, and M. Loey, "CNN for handwritten arabic digits recognition based on LeNet-5," in International conference on advanced intelligent systems and informatics, 2016, pp. 566-575: Springer.
[4] Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "mageNet Classification with Deep Convolutional Neural Networks," Communication of the ACM, vol. Vol 6, pp.84-90, 2017.
[5] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-VGG16 CNN model for big data places image recognition," in 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), 2018, pp. 169-175: IEEE.
[6] J.-G. Wang and L.-B. Zhou, "Traffic Light and Vehicle Signal Recognition with High Dynamic Range Imaging and Deep Learning," in Deep Learning: Algorithms and Applications: Springer, 2020, pp. 157-192
[7] J.-G. Wang et al., "Appearance-based brake-lights recognition using deep learning and vehicle detection," in 2016 IEEE Intelligent Vehicles Symposium (IV), 2016, pp. 815-820: IEEE.
[8] Y. Zakaria, H. Munim, M. Ghoneima, and S. Hammad, "Modified HOG-based on-road vehicle detection method," International Journal of Pure and Applied Mathematics, vol. 118, no. 18, pp. 3277-3285, 2018.
[9] E. J. R. O'Malley, M. Glavin, "Rear-lamp vehicle detection and tracking in low-exposure color video for night conditions," IEEE Transactions on Intelligent Transportation Systems, vol. vol. 11, no. 2, pp. 453– 462, 2010., 2010.
[10] A. Daniel, L. Salgado, and M. Nieto, "Robust vehicle detection through multidimensional classification for on board video-based systems," presented at the IEEE International Conference on Image Processing, 2007.
[11] C. C. Chang WC, "Online boosting for vehicle detection," IEEE Trans Syst Man Cybern B Cybern, vol. 40(3):892-902, 2010.
[12] J. Fang, Y. Zhou, Y. Yu, and S. Du, "Fine-grained vehicle model recognition using a coarse-to-fine convolutional neural network architecture," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 7, pp. 1782-1792, 2016.
[13] Y. L. Ren, S, "Vehicle make and model recognition based on convolutional neural networks," Proceedings of the 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, vol. pp. 692–695, 2016.
[14] A. Dehghan, S. Z. Masood, G. Shu, and E. Ortiz, "View independent vehicle make, model and color recognition using convolutional neural network," arXiv preprint arXiv:1702.01721, 2017.
[15] K. Z. Huang, B, "Fine-grained vehicle recognition by deep Convolutional Neural Network. ," In Proceedings of the International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Datong, China, vol. 15–17 October 2016; pp. 465–470, 2016.
[16] Y. Gao and H. J. Lee, "Local tiled deep networks for recognition of vehicle make and model," Sensors, vol. 16, no. 2, p. 226, 2016.
[17] F. Zhang, "Car detection and vehicle type classification based on deep learning," Jiangsu University, Jiangsu, 2016.
[18] F. Zhang, X. Xu, and Y. Qiao, "Deep classification of vehicle makers and models: The effectiveness of pre-training and data enhancement," 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 231-236, 2015.
[19] T. E. Z. Chen, S. A. Velastin, "Vehicle type categorization: A comparison of classification schemes," in Proc. 14th Int. IEEE Conf. Intell. Transp. Syst, vol. Oct. 2011, pp. 74–79, 2011.
[20] M. B. AbdelMaseeh, I.; Abdelkader, M.F.; EI Saban, M, "Car Make and Model recognition combining global and local cues," In Proceedings of the 2012 21st International Conference on Pattern Recognition (ICPR), vol. Tsukuba, Japan, 11–15 November 2012; pp. 910–913, 2012.

[21] J. W. C. Hsieh, L.C.; Chen, D.Y., "Symmetrical SURF and Its Applications to Vehicle Detection and Vehicle Make and Model Recognition.," IEEE Trans. Intell. Transp. Syst., vol. 15, 6–20, 2014.

[22] V. C. Petrovic, T., "Analysis of features for rigid structure vehicle type recognition," In Proceedings of the British Machine Vision Conference, London, vol. UK, 7–9 September 2004; pp. 587–596, 2004.