CAPSTONE PROJECT REPORT

(Project Term January-May 2023)

VEHICLE CLASSIFICATION USING MACHINE LEARNING

Submitted by

Priyanshu Ranjan **Registration Number : 11903954**

Tirtha Sanyal **Registration Number : 11900167**

Project Group Number CSERGC0455

Course Code CSE 445/INT 445

Under the Guidance of

HIMANSHU

Assistant Professor, Department of Computer Science & Engineering

School of Computer Science and Engineering

INDIA'S LARGEST UNIVERSITY

LOVELY PROFESSIONAL UNIVERSITY

Transforming Education, Transforming India

TOPIC APPROVAL PERFORMA

School of Computer Science and Engineering (SCSE)

Program : P133::B.Tech. (Information Technology)

COURSE CODE : INT445 **REGULAR/BACKLOG :** Regular **GROUP NUMBER :** CSERGC0455

Supervisor Name : Himanshu **UID :** 29610 **Designation :** Assistant Professor

Qualification : _____ **Research Experience :** _____

SR.NO.	NAME OF STUDENT	Prov. Regd. No.	BATCH	SECTION	CONTACT NUMBER
1	Priyanshu Ranjan	11903954	2019	K19LC	7488732243
2	Vishal Bhardwaj	11901836	2019	K19KP	8427070752
3	Tirtha Sanyal	11900167	2019	K19HZ	8437140599

SPECIALIZATION AREA : Networking and Security-I **Supervisor Signature:** _____

PROPOSED TOPIC : Vehicle Classification: Using Machine Learning

Qualitative Assessment of Proposed Topic by PAC		
Sr.No.	**Parameter**	**Rating (out of 10)**
1	Project Novelty: Potential of the project to create new knowledge	6.07
2	Project Feasibility: Project can be timely carried out in-house with low-cost and available resources in the University by the students.	6.93
3	Project Academic Inputs: Project topic is relevant and makes extensive use of academic inputs in UG program and serves as a culminating effort for core study area of the degree program.	6.00
4	Project Supervision: Project supervisor's is technically competent to guide students, resolve any issues, and impart necessary skills.	7.60
5	Social Applicability: Project work intends to solve a practical problem.	6.00
6	Future Scope: Project has potential to become basis of future research work, publication or patent.	7.60

PAC Committee Members		
PAC Member (HOD/Chairperson) Name: Dr. Harwant Singh Arri	UID: 12975	Recommended (Y/N): Yes
PAC Member (Allied) Name: Ravishanker	UID: 12412	Recommended (Y/N): Yes
PAC Member 3 Name: Dr. Manjit Kaur	UID: 12438	Recommended (Y/N): Yes

Final Topic Approved by PAC: **Vehicle Classification: Using Machine Learning**

Overall Remarks: Approved

PAC CHAIRPERSON Name: 13897::Dr. Deepak Prashar **Approval Date:** 15 Mar 2023

DECLARATION

We hereby declare that the project work entitled ("Vehicle Classification Using Machine Learning") is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Himanshu, during January to May 2023. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Project Group Number: **CSERGC0455**

Name of Student 1: Priyanshu Ranjan

Registration Number: 11903954

Name of Student 2: Tirtha Sanyal

Registration Number: 11900167

(Signature of Student 1)
Date:

(Signature of Student 2)
Date:

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort, and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfilment of the conditions for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

Himanshu : 29610

Signature and Name of the Mentor

Assistant Professor

Designation

School of Computer Science and Engineering,

Lovely Professional University,

Phagwara, Punjab.

Date :

ACKNOWLEDGEMENT

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we give to our capstone project mentor, Mr. Himanshu, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report. Furthermore, we would also like to acknowledge with much appreciation the crucial role of the team who helped each other design, code and develop this project and gave suggestions about the same. Without great support of LOVELY PROFESSIONAL UNIVERSITY, it would be not possible to complete this project. The University has always been very responsive in providing necessary information, and without their generous support plan would lack in accurate information on current developments. We perceive this opportunity as a big milestone in our career development. We will strive to use gained skills and knowledge in the best possible way, and we will continue to work on their improvement, to attain desired career objective.

TABLE OF CONTENTS

1. INTRODUCTION

In Vehicle detection and classification are crucial tasks in different applications, including traffic control, monitoring, and self-driving cars. In recent years, advancements in “machine learning(ML)” and “deep learning” algorithms have made it possible to automatically identify and categorize different types of vehicles based on their visual features. This process involves collecting images of vehicles from cameras installed in various locations and using ML algorithms to analyse the features of the vehicles, such as their shape and size. To train the ML model accurately, a vast dataset of vehicles from various categories is required. Once trained, the model can classify new images or video footage of vehicles in real-time.

1.1 Background of Deep Learning

“Deep learning” is a sub-field of “machine learning” that has experienced significant growth in recent years and is being applied to solve complex problems across a wide range of domains. “Deep learning” involves training “artificial neural networks” with multiple layers to learn from data and make predictions or decisions. The inspiration for “deep learning” models comes from the structure and function like of a human brain's neural networks, which are made up of interconnected neurons that process and transmit information. Deep learning models, like the human brain, can automatically learn features and patterns from large datasets, making it possible to build highly accurate and complex models that can generalize well to new data. Deep learning has been made possible by the availability of large amounts of labeled data, powerful computing resources, and advances in “deep learning algorithms” such as “Convolutional Neural Network(CNN)” and “Recurrent Neural Network(RNN)”. These algorithms can model complex relationships between input and output data and make predictions with high accuracy. CNNs have been particularly successful in image and speech recognition tasks. They work by scanning the input image with a set of filters that can detect specific features such as edges, shapes, and textures. These filters are then combined to form higher-level

features that can represent the input image more abstractly. RNN is typically employed in “natural language processing(NLP)” and speech recognition techniques, whereas deep learning has brought about significant changes in numerous industries, including finance, health, and transportation. In healthcare, deep learning models have been used to analyze medical images, diagnose diseases, and predict patient outcomes. In finance, deep learning models have been used for fraud detection, risk assessment, and investment analysis. In transportation, deep learning models have been used for autonomous driving, vehicle classification, and intelligent traffic management. It can automate complex tasks and enable faster and more accurate decision-making. Deep learning is also paving the way for further innovations in artificial intelligence. It is expected to play a significant role in shaping the future of computing and intelligent systems, particularly in the areas of autonomous systems, robotics, and natural language processing. However, deep learning also presents several challenges, including the need for large amounts of labeled data, computing power, and expertise in building and training deep learning models.

In conclusion, deep learning is a rapidly evolving field that has brought about significant advancements in various domains. It has enabled the development of highly accurate and complex models. Deep learning is expected to continue to drive innovation in artificial intelligence and shape the future of computing and intelligent systems.

1.2 Convolutional Neural Network

“Convolutional Neural Network(CNN)” is a kind of neural network that has been widely used in computer visionary applications. CNN are designed to recognize visual patterns directly from pixels with minimal amount of pre-processing. CNNs consist of multiple layers of small-sized filters (also known as kernels) that can identify visual features such as edges, corners, and textures in an image. Each filter performs a convolution operation on the input image to produce an output feature map. The output feature maps are then fed into the next layer of the network, where the process is repeated with a new set of filters to extract more complex visual features. This process of convolution and pooling

continues through the layers until the final layer, which produces the classification output. The weights of the filters in each layer of the network are learned during the training process, where the network is optimized to minimize the loss function between the predicted output and the actual output. The ability of CNNs to learn features automatically from raw pixel data has made them the go-to choice.

However, traditional vehicle detection and classification techniques may not be suitable for complex environments where the background is dynamic, and lighting conditions change frequently. In such scenarios, it becomes challenging to differentiate between moving objects and background clutter. To overcome these limitations, Convolutional Neural Networks(CNNs) have been developed. CNNs can automatically learn features from images and classify them accurately, making them useful for vehicle classification.

The final layer of a CNN produces a classification output. The vehicle detection and classification pipeline involve background subtraction for detection and CNNs for classification. The approach involves pre-processing the video stream to detect moving objects using background subtraction. The resulting foreground mask is then passed to the CNN for classification, where it analyses the features of the detected vehicles and classifies them into different categories. However, this project is limited to classification of images only. Background subtraction is a technique used to identify the foreground objects in an image by subtracting the background. It works by creating a background model of the scene by averaging the pixel values of the image frames over time. Any pixels that differ significantly from the background model are classified as foreground pixels.

CNNs have shown excellent performance in vehicle classification, even in complex environments where the background is dynamic. They can recognize patterns and features that traditional methods may lack. CNNs can also handle variations in lighting conditions, which is a significant challenge in traditional methods. The accuracy and efficiency of vehicle detection and classification using background subtraction and CNN

algorithms depend on the quality of the data and the complexity of the environment. High-quality data is essential to overcome the limitations and errors in the classification process. The dataset used for training the CNN should be diverse, containing images of vehicles in various categories, sizes, and shapes. Additionally, the dataset should include images captured under different lighting conditions and in different environments, such as urban and rural areas.

In complex environments where the background is dynamic, CNNs can provide better performance than traditional methods. In such scenarios, background subtraction alone may not be sufficient to accurately detect moving vehicles. CNNs can learn to differentiate between moving objects and background clutter, reducing false positives and increasing detection accuracy.

The use of CNNs for vehicle classification has become increasingly popular in the past few years. The accuracy and efficiency of vehicle classification using CNNs are expected to improve further. These advancements could lead to significant improvements in traffic control, monitoring, and self-driving cars, ultimately leading to safer roads and more efficient transportation systems.

In conclusion, vehicle detection and classification are critical tasks in various applications, and ML and deep learning algorithms have made it possible to automate these tasks. CNNs have shown excellent performance in vehicle classification, especially in complex environments where the background is dynamic. The accuracy and efficiency of vehicle detection and classification using CNNs depend on the quality of the data and the complexity of the environment.

2. PROBLEM STATEMENT

2.1. Profile of the Problem

The problem being addressed is the need for accurate and efficient vehicle detection and classification in various applications such as traffic control, monitoring, and self-driving cars. Traditional techniques for vehicle detection and classification may not work well in complex environments where the background is dynamic, and lighting conditions change frequently. In such scenarios, it is challenging to differentiate between moving objects and background clutter, leading to errors in detection and classification. To address this issue, Convolutional Neural Networks(CNNs) have been developed, which can automatically learn features from images and classify them accurately. However, the accuracy and efficiency of vehicle detection and classification using CNNs depend on the quality of the data and the complexity of the environment. A large and diverse dataset of vehicles is required to train the CNN accurately, containing images of vehicles in various categories, sizes, shapes, and lighting conditions.

The successful implementation of accurate and efficient vehicle detection and classification using CNNs has several potential benefits, including safer roads, more efficient transportation systems, and improved monitoring capabilities. Therefore, the profile of the problem includes the need to overcome the limitations of traditional methods for vehicle detection and classification in complex environments and the development and optimization of CNN-based algorithms to achieve accurate and efficient results. messages.

2.2. Rationale/Scope of the Study

The rationale of this study is to develop a machine learning approach for efficient and accurate vehicle detection and classification in traffic surveillance systems. The proposed

method will build upon the existing literature on the use of deep learning techniques, particularly “convolutional neural networks(CNN)”, for these tasks. The reviewed studies highlight the limitations of traditional vehicle detection methods that rely on ad-hoc features and are computationally expensive. The proposed approach will employ a CNN architecture optimized for deployment on embedded systems with limited resources. The study will use a comprehensive dataset of vehicle images for training and testing the accuracy of the proposed CNN model. The model will aim to classify various types of vehicles and accurately count the number of vehicles in non-laned road traffic scenarios. The study will provide insights into the performance of different CNN architectures and their potential applications in traffic surveillance and management systems. The study is significant as it addresses the limitations of existing vehicle detection methods and contributes to the development of efficient and accurate methods for traffic surveillance. The proposed approach has potential applications in traffic management, accident prevention, and law enforcement. The study's scope is limited to non-laned road traffic scenarios, and further research is required to evaluate its effectiveness in other traffic scenarios.

3. EXISTING SYSTEM

3.1. Introduction

The existing system for vehicle classification includes various machine learning techniques, such as genetic algorithms, decision trees, random forests, and convolutional neural networks (CNNs). The traditional methods of vehicle detection rely on the use of ad-hoc features, which can be inaccurate and computationally expensive. Sun, Bebis, and Miller proposed a technique that utilizes an evolutionary Gabor-filter optimization approach for detecting vehicles on roads. Their proposed method uses a genetic algorithm to optimize the parameters of Gabor-filters, which are then used to detect vehicles in road scenes.

Yu and colleagues presented a model for fine-grained vehicle classification that employs a CNN to extract distinctive features from images of vehicles. The authors trained their model on a large dataset of vehicle images and achieved high accuracy in classifying different types of vehicles, including cars, trucks, and buses. They also conducted experiments to evaluate the effectiveness of different CNN architectures and found that a modified VGG16 network performed the best.

Cai, Deng, Khokhar, and Aftab proposed a CNN model whose purpose is to classify various types of vehicles in traffic surveillance systems by analysing their images. To develop and assess the model's accuracy, it was trained and tested on a comprehensive dataset. Chauhan and colleagues proposed an approach for classifying and counting vehicles in non-laned road traffic scenarios using an embedded CNN. The proposed model employs a small CNN architecture optimized for deployment on embedded systems with limited resources.

Ghosh et al. (2021) conducted a study on Bangla handwritten character recognition using state-of-the-art convolutional neural network architectures. They used the BanglaLekha-Isolated dataset and reported the recognition rate in terms of accuracy. The study achieved the best performance with the Xception model, which had an accuracy of 99.84%.

Cai et al. (2018) proposed a vehicle classification method based on deep convolutional neural networks for traffic surveillance systems. The study used a self-collected traffic surveillance dataset and reported on the proposed method's performance without specific accuracy results. Pak and Kim (2017) reviewed deep learning models for image recognition. The study did not use a specific dataset or report on specific accuracy results as it was a review paper.

Choudhary and Gianey (2017) conducted a comprehensive review on supervised machine learning algorithms for classification and regression tasks. The study did not use a specific dataset or report on specific accuracy results as it was a review paper.

Chauhan et al. developed an embedded convolutional neural network (CNN) to classify and count vehicles in non-laned road traffic videos captured from an overhead camera. The authors achieved an accuracy of 94.3% for vehicle classification and 98.6% for vehicle counting.

Zhao et al. (2019) used a dataset of vehicle images captured from different viewpoints and under various environmental conditions. They employed a deep reinforcement learning algorithm with visual attention to classify the vehicles in the images. The authors achieved an accuracy of 93.3%.

Zhang et al. (2019) used a dataset of vehicle trajectories captured from a traffic camera. They employed a reinforcement learning algorithm with partial vehicle detection to control the traffic signal timings. The authors achieved an average travel time reduction of 14.6% compared to fixed-time signal control.

Won (2021) conducted a survey on various datasets and models used in intelligent traffic monitoring systems for vehicle classification. The datasets included the KITTI Vision Benchmark Suite, UA-DETRAC, and CityFlow, and the models included CNN, deep CNN, and support vector classifier(SVM). The survey did not report on specific accuracy results.

In summary, the studies discussed in the provided texts focused on the development and evaluation of different computer vision algorithms for image recognition and vehicle classification. Some studies used specific datasets, while others did not report specific accuracy results. Overall, the studies highlight the importance of machine learning algorithms and computer vision techniques in addressing real-world problems, such as traffic surveillance and intelligent traffic management. Overall, the existing system for vehicle classification involves the use of deep learning techniques, particularly CNNs, for image classification tasks. These models are trained on large datasets of vehicle images and can accurately classify different types of vehicles. The proposed models have potential applications in traffic surveillance and management systems. However, there is still room for improvement in terms of accuracy, computational complexity, and memory footprint.

3.2. Existing Software

There are several software options available for vehicle classification. Here are some examples:

1. Q-Free Intrada: This is a software platform that provides real-time vehicle detection, classification, and tracking.

2. VITRONIC PoliScan: This is a software suite that provides vehicle classification, speed measurement, and enforcement capabilities.

3. Tattile Vega: This is a software platform that provides vehicle classification, license plate recognition, and video analysis capabilities.

4. TrafficVision: This is a software platform that provides vehicle detection, classification, and tracking capabilities.

5. Neurala Brain Builder: This is an AI-powered software platform that can be trained to recognize and classify vehicles based on video footage.

These are just a few examples, and there are many more software options available for vehicle classification, depending on your specific needs and use case.

3.3. DFD for Present System

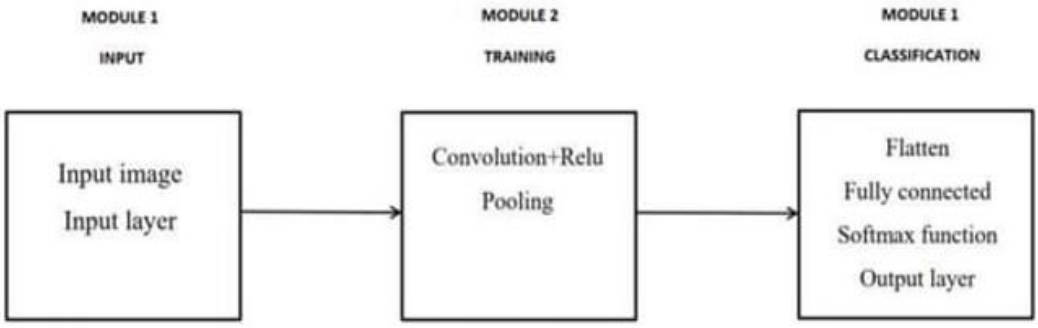

FIGURE 1: DFD of the Model

3.4. What's New in the System to be Developed

There are various options to consider when it comes to adding new features to a vehicle classification system. One potential feature is incorporating color and make/model recognition to provide additional information beyond just the vehicle's size and type. Pedestrian and bicycle recognition could be another useful addition to improve safety and

traffic data accuracy. Real-time traffic monitoring can help identify areas of congestion and provide valuable data for traffic management and city planning. An automated toll collection system that identifies and classifies vehicles passing through toll booths could help reduce congestion and increase efficiency. A weather and environmental conditions recognition feature that adjusts classification algorithms based on weather patterns can improve accuracy and reliability. Integrating the system with other smart city infrastructure, such as traffic lights and public transportation systems, can also help improve traffic flow and reduce congestion. Lastly, predictive analytics that use historical traffic data to make predictions about future traffic patterns can help inform traffic management and infrastructure investment decisions. Ultimately, the features added to a vehicle classification system will depend on the specific needs and use case of the system.

4. PROBLEM ANALYSIS

4.1. Product Definition

The Vehicle Classification System is a powerful software platform that accurately detects, classifies, and tracks vehicles on roadways in real-time. With its advanced algorithms and machine learning capabilities, the system provides valuable data on vehicle type, size, and speed, which can be used to improve traffic management and safety. The system offers a range of key features, including real-time data collection, customizable reporting, integration with existing infrastructure, scalability, and a user-friendly interface. In addition to being cost-effective, the Vehicle Classification System provides transportation officials and city planners with accurate, up-to-date data that can help inform their decisions and improve overall traffic flow and safety. Whether used in small or large urban areas, the system is a valuable tool for those looking to improve traffic management and infrastructure investment decisions.

4.2. Feasibility Analysis

For the proposed vehicle classification project that involves the integration of machine learning and deep learning algorithms, it is necessary to assemble a team of experts in these fields. In addition to the expertise, the team will require access to a significant dataset to train the model, robust hardware to support the training, and a strong server infrastructure to host the app. A comprehensive analysis of technical requirements, resource allocation, and project timeline will be necessary to ensure the project's feasibility. Furthermore, evaluating market demand for the app and potential competitors will also be a critical step. Through the feasibility study, the team can assess the project's viability and anticipate any possible roadblocks that may arise during development. The feasibility study is an integral part of the vehicle classification project, providing valuable guidance for the team's decision-making process throughout the development cycle.

4.3. Project Plan

The project plan will consist of several phases, including planning and research, design and development, testing, deployment and maintenance and support. The team will develop machine learning and deep learning algorithms and conduct extensive testing. The project plan will also include a detailed timeline, resource allocation, and risk management plan to ensure the project's success.

4.3.1. Planning and research

Vehicle classification research and planning involve various aspects, including gathering data on the types of vehicles on the roads, determining the appropriate technology for classification, and planning for the implementation of the system. The research and planning process also involves identifying the challenges associated with vehicle classification, such as environmental factors and data privacy concerns. In addition to technical aspects, the planning process must also consider the cost, scalability, and maintenance of the system. The research phase involves examining the available technologies for vehicle classification, such as machine learning and computer vision algorithms. Planning includes identifying the necessary resources, such as data collection infrastructure, hardware, and software. The implementation plan must also consider regulatory requirements and potential user adoption.

4.3.2. Design and development

The design phase includes creating an architecture that outlines the system's various components and how they will work together to achieve the desired outcome. The development phase involves coding the system and testing it to ensure that it meets the requirements and specifications. During development, the team may use various technologies, such as machine learning algorithms and computer vision systems, to build the classification model.

4.3.3. Testing

In the testing phase, the team will conduct extensive testing to ensure the model's performance. The team will perform tests to evaluate the model's accuracy.

4.3.4. Deployment

In the deployment of a vehicle classification system involves several critical steps to ensure its successful implementation. The first step is to test the system in a controlled environment to ensure that it is functioning as intended. Once the system has been thoroughly tested and validated, the team can begin to deploy it in the intended environment. This may involve installing hardware and software components, configuring the system, and integrating it with other technologies. To minimize disruption, deployment may be phased or conducted outside of peak traffic hours. Once deployed, the system must be monitored and tested regularly to ensure that it is functioning correctly and providing accurate results. The team should also have a plan in place for addressing any issues that may arise during deployment, such as hardware failures or software bugs. By carefully planning and executing the deployment phase, the team can ensure that the vehicle classification system is operating as intended and delivering value to its users.

4.3.5. Maintenance and support

Finally, the maintenance and support are crucial aspects of a vehicle classification system's lifecycle. Regular maintenance ensures that the system functions correctly, while support involves assisting users and addressing issues. Maintenance may involve updating software and upgrading hardware, while support may involve providing training and troubleshooting. Monitoring system performance and conducting periodic audits can help identify and address issues. Robust maintenance and support services ensure that the system delivers value over its entire lifecycle.

5. SOFTWARE REQUIREMENT ANALYSIS

Software Requirement Analysis is an essential phase in the software development life cycle that helps to identify the system requirements and specifications to build a software product. It involves analysing the vehicle image, defining the problem statement, and identifying the software's scope and objectives. In this section, we will discuss the software requirement analysis for our project.

5.1. Introduction

The introduction provides a brief overview of the project and outlines its purpose, scope, and objectives. The goal of our project is to develop a model that can classify the vehicle images into their classes. The software's primary target is to classify vehicles and labelled the unlabelled data.

5.2. General Description

Software requirements analysis for a vehicle classification model involves a systematic approach to defining and documenting the functional and non-functional requirements of the system. In the case of a vehicle classification model that can classify road transport vehicles such as bikes, cars, bicycles, and buses, the requirements analysis would involve identifying the features and specifications needed to meet the system's objectives. This includes identifying the input data sources, defining the classification algorithms, specifying the hardware and software requirements, and determining the system's performance metrics. The requirements analysis also involves considering factors such as system scalability, reliability, and maintainability. The ultimate goal of the software requirements analysis is to ensure that the resulting system meets the stakeholders' needs and requirements, is feasible to develop and implement, and meets the specified performance standards.

6. DESIGN

6.1. System Design

The system design for our vehicle classification model can be broken down into four main components: data collection, data processing, model training and deployment.

6.1.1. Data collection

Data collection is a critical aspect of developing a vehicle classification model based on deep learning. To train and evaluate the model, a high-quality dataset with diverse examples of vehicles is required. One source of such datasets is Kaggle, a platform that hosts a variety of open datasets for machine learning tasks. Kaggle offers a dataset containing four classes of road transport vehicles, including cars, bikes, bicycles, and buses. The dataset consists of thousands of images of different sizes and shapes, captured from various viewpoints and under different environmental conditions. The images in the dataset are labeled with their corresponding vehicle types, allowing for supervised learning of the model. Using this dataset, we can train CNN to recognize and classify the four types of vehicles accurately. However, it is crucial to ensure that the dataset is well-balanced and representative of the target population to avoid bias in the model's predictions. Therefore, it is essential to perform data exploration and pre-processing before training the model to ensure the dataset's quality and relevance.

Data Pre-processing:

Data collection is a critical aspect of developing a vehicle classification model based on deep learning. To train and evaluate the model, a high-quality dataset with diverse examples of vehicles is required. One source of such datasets is Kaggle, a platform that hosts a variety of open datasets for machine learning tasks. Kaggle offers a dataset containing four classes of road transport vehicles, including cars, bikes, bicycles, and buses. The dataset consists of thousands of images of different sizes and shapes, captured

from various viewpoints and under different environmental conditions. The images in the dataset are labelled with their corresponding vehicle types, allowing for supervised learning of the model. Using this dataset, we can train a “convolutional neural network” to recognize and classify the four types of vehicles accurately. However, it is crucial to ensure that the dataset is well-balanced and representative of the target population to avoid bias in the model's predictions. Therefore, it is essential to perform data exploration and pre-processing before training the model to ensure the dataset's quality and relevance.

6.1.2. Data processing

Data processing is the series of activities involved in transforming raw data into useful information for decision-making, analysis, and other purposes. This process involves several stages, from data preparation and cleaning to analysis and interpretation. In this article, we will explore the key aspects of data processing and their importance.

1. Data preparation and cleaning

Data preparation is an essential step in the data processing pipeline. It involves collecting, cleaning, and formatting data to ensure its accuracy and completeness. Cleaning the data entails identifying and correcting error, inconsistencies, and empty values, ensuring that the data is reliable and accurate for further processing. Several techniques can be employed in data cleaning, such as outlier detection, imputation, and data normalization. Outlier detection aims to identify and remove data points that are remarkably different from the rest of the data. Imputation helps in filling in the missing values, while data normalization scales the data to a common range to ensure comparability.

2. Data integration and transformation

Data integration is a crucial step in the data processing pipeline that involves consolidating data from multiple sources into a single, unified dataset. This process ensures that all relevant data is available for analysis and decision-making. Similarly, data transformation is another critical step that involves converting data from one format to another to make it compatible with the analysis tools and techniques being used. Data integration and transformation techniques may include data mapping, data merging, and data aggregation. Data mapping involves identifying common elements in different datasets and aligning them. Data merging involves combining data from different datasets into a single, unified dataset. Data aggregation summarizes data at different levels of granularity to provide insights into patterns and trends.

3. Data analysis and modeling

Data analysis refers to the use of statistical and “machine learning techniques” to extract details from the data. The process involves identifying patterns, trends, and relationships in the data that can inform decision-making and other activities. Furthermore, data modeling involves creating mathematical or statistical models of the data to predict future outcomes or estimate the impact of different scenarios. There are different techniques available for data analysis and modeling, such as regression analysis, clustering, and neural networks. Regression analysis involves identifying the relation between one or more independent variables and a dependent variable. Clustering, on the other hand, involves grouping similar data points into clusters based on their similarity. Finally, neural networks refer to creating artificial neural networks that can learn from the data to make decisions or classifications.

6.1.3. Model training

Model training is an essential step in vehicle classification as it involves teaching the machine learning algorithm to recognize patterns in the insights and make decisions

based on those patterns. There are several different approaches to model training, including “supervised” and “unsupervised” learning.

In supervised learning, the labeled data is provided to the algorithm, which means that each data is already assigned to a particular vehicle category (bike, car, bicycle, or bus). That algorithm then learns to recognize patterns in the data and make decisions based on those patterns. This approach is highly effective but requires a large amount of labeled data, which can be time-consuming and expensive to obtain.

In un-supervised learning, the labeled data is not provided and instead must identify patterns and structure in the data on its own. This approach is less effective than supervised learning but can be useful in situations where labeled data is not available. Once the approach to model training has been chosen, the next step is to select an appropriate algorithm. There are several different algorithms that can be used for vehicle classification, including Convolutional Neural Network(CNNs), Random Forest, and K-Nearest Neighbors(KNN).

After selecting an algorithm, the next step is splitting the data. The training set is determined to teach the algorithm to recognize patterns in the data, while the testing set is used for evaluation purposes.

Finally, the model is trained using the train set, and its performance is evaluated using the test set. If the performance is not satisfactory, the algorithm and parameters may need to be adjusted, and the model retrained. Overall, model training is a critical step in vehicle classification as it is responsible for teaching the algorithm to recognize patterns in the data and make accurate predictions. With the right approach and algorithm, model training can be highly effective in accurately classifying road transport vehicles based on their type.

6.1.4. Deployment

Once the vehicle classification model has been developed and trained, it can be deployed for practical use. Deployment refers to the process of making the model available for use by others, such as through a website or mobile app.

There are several ways to deploy a machine learning model for vehicle classification. One approach is to deploy the model on a website, where users can input an image of a vehicle and receive the classification output. This can be achieved by integrating the model with a web application framework such as Flask or Django, which allows for the creation of web-based user interfaces.

Another approach is to deploy the model on a mobile app. This requires creating a mobile app that can capture an image of a vehicle and send it to the model for classification. This approach can provide a more convenient and accessible user experience, as users can access the vehicle classification functionality directly from their mobile devices.

In both cases, deployment requires setting up a server or cloud-based infrastructure to host the model and its associated code. This involves configuring the necessary software dependencies, as well as ensuring that the infrastructure is scalable and reliable enough to handle multiple requests from users. Additionally, security measures such as encryption and access control must be put in place to protect the model and its output from unauthorized access or tampering. Overall, deploying a vehicle classification model on a website or mobile app can greatly increase its accessibility and usefulness to end-users. However, it requires careful planning and execution to ensure that the model is deployed securely, reliably, and with a user-friendly interface.

6.2. Design Notation

Design notations are graphical or symbolic representations utilized to describe the structure, behavior, and relationships of components in a system. These notations enable

designers to communicate design ideas and concepts in a standardized and consistent manner. There are various design notations that can be used for this project, including UML diagrams, flowcharts, ER diagrams, Gantt charts, and state diagrams. UML diagrams are popular and are used to describe the structure and behaviour of systems. Flowcharts are graphical representations of processes or algorithms, while ER diagrams represent the relationships between entities in a system. Gantt charts are used to illustrate project schedules and timelines, while state diagrams are used to represent the various states and transitions of a system. The specific notations used will depend on the requirements of the project, as well as the preferences and expertise of the design team. The designed notations can be used for the deployment of the vehicle classification model, including the integration of the model into a website or a mobile app. However, a flowchart is a particularly favorable design notation for this project, as it can effectively illustrate the sequence of steps involved in the data processing and analysis workflows.

6.3. Flowcharts

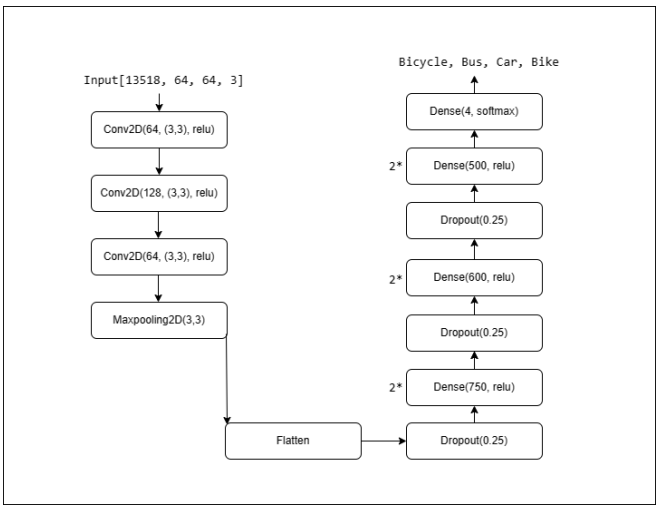

FIGURE 2: FLOWCHART OF MODEL

7. TESTING

Testing is an essential process that is conducted to evaluate the quality, reliability, and performance of software or applications. It involves identifying and analyzing errors, defects, or bugs in the system or application to assure that it meets the specified requirements and is error-free. Testing is an iterative process that starts at the beginning of the software development life cycle and continues until the software or application is ready for release. It helps to identify and eliminate potential issues that can cause errors or failures, improve the of the system, and enhance the user experience. It can be performed manually or using automated tools and techniques, and it is an integral part of software development and quality assurance.

7.1. Functional Testing

Functional testing is a software testing technique that ensures the functionality of applications or systems by evaluating them against the requirements or specifications provided. It involves verifying that the software or system performs the intended functions as per the user's needs. Functional testing can be conducted manually or using automated tools and comprises various types of tests such as unit, integration, system, and acceptance tests. The primary goal of “functional testing” is to ascertain that the system functions appropriately and satisfies the user's requirements in terms of functionality, usability, and performance.

7.2. Structural Testing

Structural testing is a kind of testing that focuses on the internal structure of software applications, such as code or system architecture. This type of testing aims to ensure that the software functions correctly at a code level and that it adheres to design specifications. Structural testing can involve techniques such as code coverage analysis, which assesses how much of the code is being executed during testing, and static analysis, which involves examining code without executing it.

7.3. Levels of Testing

It aims to verify that the software product's performance meets the specified requirements. Its main goal is to identify any defects, errors, or gaps in the system, and ensure that it satisfies the requirement of its intended users. To achieve this objective, software testing is typically divided into various levels, each with a distinct scope, focus, and purpose.

1) Unit Testing

The initial level of "software testing" is known as unit testing, which involves testing the individual components or units of the software in isolation from the rest of the system. The primary objective of unit testing is to validate that each unit of the software performs as intended and meets its design specifications. Typically, developers conduct unit testing by testing individual functions, methods, or procedures. The tests are automated and usually executed each time there is a modification or update to the code.

2) Integration Testing

The following stage of software testing is integration testing, which involves testing the interactions between various components or units of the software. The purpose of integration testing is to ensure that the units can work together correctly and communicate with each other as intended. By conducting integration testing, developers can detect defects that may arise due to interface issues or other interactions between components. Developers usually perform integration testing, and they may use different techniques, such as top-down or bottom-up integration testing.

3) System Testing

System testing is the subsequent level of "software testing", where the entire system is tested. The primary goal of "system testing" is to verify that the software meets the specified requirements and performs correctly in the intended environment. Typically, a

dedicated testing team conducts system testing and may use various techniques, such as black-box or white-box testing. “Black-box testing” focuses on testing the functionality and behavior of the software without accessing the source code. On the other hand, “white-box testing” involves accessing the source code and testing the internal workings of the software.

4) Acceptance Testing

Acceptance testing is the ultimate level of “software testing”, where the software is tested in the production environment to ensure that it fulfills the requirements of its intended users. The purpose of “acceptance testing” is to confirm that the software meets the specified requirements and operates correctly in the target environment. Typically, end-users or a dedicated acceptance testing team conducts acceptance testing.

7.4. Testing the Project

In the vehicle classification model, the system testing level of testing was used to test the entire system to ensure that the software satisfies the specified requirements and operates correctly in the target environment. The testing procedure included dividing the dataset using a 75:25 ratio. The training set was utilized to train the model, while the testing set was utilized to assess the model's performance. To evaluate the model, new images that were not part of the training process were inputted, and the output classification results were analyzed. This testing process aided in detecting any faults or discrepancies in the model and guaranteeing that it conforms to the requirements and operates correctly.

8. IMPLEMENTATION

8.1. Implementation of the Project

These are some process involve in the implementation:

1. Data collection: Collection of data is very crucial part, it is done by rigorously finding the best available dataset in repositories like Kaggle.

2. Data pre-processing: Cleaning the collected data to remove unwanted images, small images, blackout images or any other irrelevant information.

3. Data Labelling: Labelling the data with their respective class such as Bus, Bicycle, Car and Motorcycle.

4. Model training: Training a “machine learning” or “deep learning” model on labelled data and extracted features can enable the prediction of the class of vehicle images.

5. Model evaluation: To evaluate the performance of the model, various metrics such as precision, accuracy, and F1-score are employed.

6. Deployment: Deploying the model in the production environment for classification of vehicle.

Vehicle classification models are designed to classify vehicles into different categories based on their type, make, model, and other relevant features. These models are widely used in the automotive industry, transportation, and security applications. One of the most effective approaches to building a vehicle classification model is by using convolutional neural networks(CNN).

CNN is well-suited for vehicle classification models because it can automatically learn and extract complex features from images. This makes it ideal for identifying patterns in

images that are indicative of specific vehicle types or models. Additionally, CNN can handle large datasets, making it possible to train the model on a vast number of images of different vehicle types. The use of CNN in vehicle classification models has led to high accuracy rates, making it a popular choice in the automotive industry.

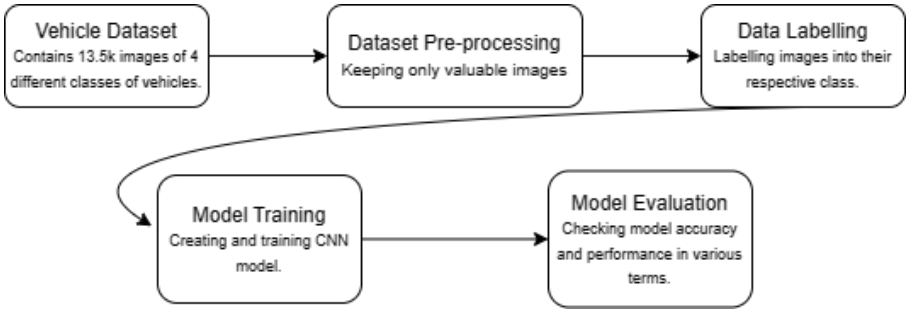

FIGURE 3: Implementation of model

Dataset

The Vehicle dataset is utilized for this study and contains 13518 images of all the 4 classes including Bus, Bicycle, Car and Motorcycle. The images are stored in their respective folders.

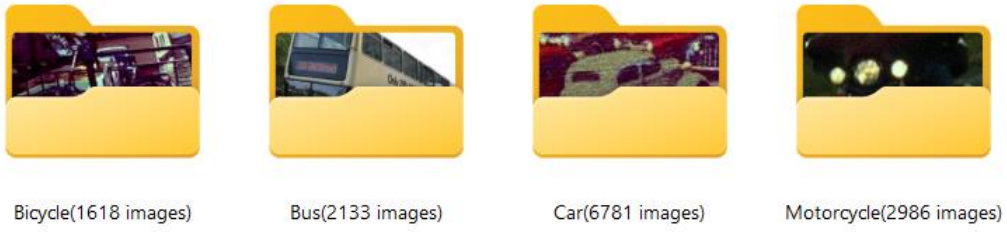

FIGURE 4: Classes present in dataset.

Data Pre-processing

It is a critical step in building an image-based vehicle classification model. The following are some of the steps involved in data pre-processing for vehicle images:

1. Data cleaning: Remove any corrupt or low-quality images from the dataset to ensure that the model is not trained on irrelevant data.

2. Image resizing: Resize all the images to a consistent size to ensure that the model can process them efficiently. This step helps in reducing computational costs and improving model performance.

3. Data augmentation: Data augmentation is a process of creating new images by performing various image transformations such as rotation, flipping, cropping, and brightness adjustment. This step reduces overfitting.

4. Normalization: Normalising the pixel values of images to a standard range to ensure that the model is not biased towards any specific range of pixel values.

5. Splitting the data: Split the dataset into train and test sets in the ratio of 75:25 to evaluate the performance of the model.

6. Labelling: Label each image with the corresponding class, i.e., car, bus, bicycle, or motorcycle, to enable the model to learn from the labelled data.

7. Pre-processing the data: Pre-process the data before feeding it to the model by converting images to arrays, one-hot encoding the labels, and scaling the pixel values.

By following these steps, we can ensure that the data is cleaned, transformed, and formatted.

Splitting of Data

For this system, the splittin of the data set into training and testing set is 75:25 ratio.

The model will be trained using the training_set. whereas for evaluating the performance test_set will be utilize.

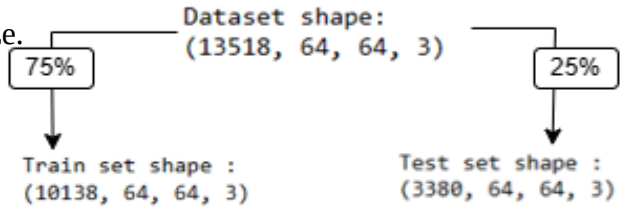

FIGURE 5: Splitting of dataset into train and test sets.

Preparing Model

The proposed study uses the CNN algorithm for preparing model.

To prepare the model the code creates a sequential model for classifying vehicles from images. The model consists of several layers of convolutional and dense layers. The Conv2D layers perform convolutions on the input image to extract important features, while the MaxPooling2D layer decreases the dimensionality of the feature maps, while the Flatten layer changes the 3D tensor output of the convolutional layers into a 1D tensor that is suitable for the dense layers. To avoid overfitting, Dropout layers are employed. The final dense layer, with a softmax activation function, generates the predicted probabilities for each class. The model uses the “categorical cross-entropy loss function” and “stochastic gradient descent optimizer” with a learning rate of 0.001. The model is trained for 30 epochs, and its performance is monitored during training using the validation data.

Model Evaluation

The training accuracy for the model is 94% while the test accuracy is 80.7%.

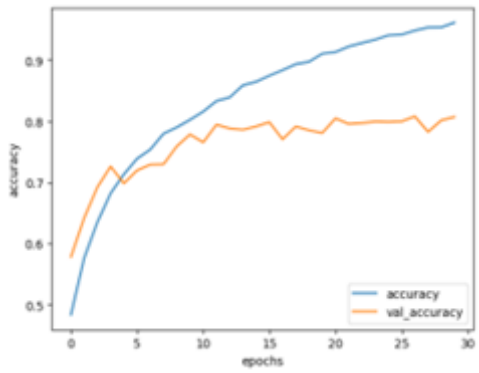

FIGURE 6: Accuracy v/s Validation Accuracy

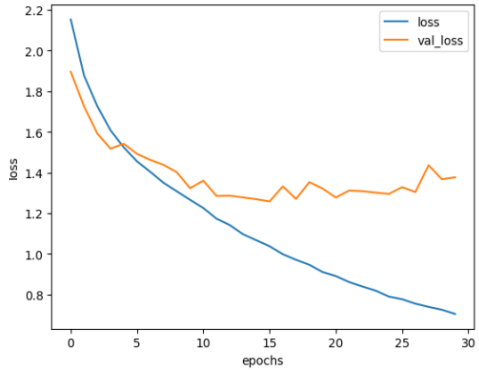

FIGURE 7: Loss v/s Validation Loss

Table 1 is the classification report of this model, consisting of various metrics and support of all the classes as well. Labels 0,1,2 and 3 denote Bicycle, Bus, Car, and Motorcycle, respectively. The classification report is derived from the dataset and is useful in evaluating the model's performance on all the data it was trained on, which includes both the training and validation sets.

Table 1: Classification Report of Training Set

	Precision	Recall	F1-Score	Support
Bicycle	0.86	0.91	0.89	1618
Bus	0.92	0.98	0.95	2133
Car	0.98	0.95	0.96	6781
Motorcycle	0.92	0.92	0.92	2986
Accuracy			0.94	13518
Macro avg.	0.92	0.94	0.93	13518
Weighted average	0.94	0.94	0.94	13518

Table 2 is the classification report for the test_set provides an evaluation of the model's performance on new, unseen data that it has not been trained on. Additionally, the model provides metrics for each class present in the test_set, enabling us to evaluate the model's performance in specific classes of interest.

Table 2: Classification Report of Test Set

	Precision	Recall	F1-Score	Support
Bicycle	0.61	0.71	0.66	409
Bus	0.78	0.92	0.84	529
Car	0.92	0.83	0.87	1686
Motorcycle	0.72	0.73	0.73	756
Accuracy			0.81	3380
Macro avg.	0.76	0.80	0.78	3380
Weighted average	0.82	0.81	0.81	3380

9. PROJECT LEGACY

9.1. Current Status of the Project

The status of the project can be evaluated based on its success in achieving its objectives. The primary goal of the project was to reach the highest accuracy possible and able to predict output by successfully classifying the vehicle image into its respective category.

9.2. Remaining Areas of Concern

Although the model training and testing has been completed, there may be several areas of concern that require further attention to ensure the complete user experience. These areas may include:

9.2.1. Technical Issues

One possible technical issue for vehicle classification is “overfitting”. Overfitting happens when a model is overly complicated and fits the training data too closely, leading to inferior performance when it encounters fresh, unseen data. This may occur if there is inadequate training data, or if the model is excessively complex with an excessive number of parameters compared to the amount of training data. Techniques such as regularization can help mitigate overfitting or reducing the complexity of the model. Another technical issue that can arise in vehicle classification is data imbalance, where one or more classes have significantly fewer samples than others. This can lead to poor performance on the underrepresented classes and can be addressed by using data augmentation or adjusting class weights during training.

9.2.2. User feedback

Typical feedback for a machine learning model may include comments on its accuracy, ease of use, speed, and overall performance. Users may also provide

suggestions for improvements, such as adding new features or improving the user interface. It's important for developers to gather feedback from users to continually improve their models and ensure they are meeting the needs of their intended audience.

9.2.3. Competition

There are various platforms and websites where you can find ongoing and upcoming data science and machine learning competitions, such as Kaggle, Analytics Vidhya, and many others. These competitions often provide datasets and problem statements for participants to solve and develop their skills in data science and machine learning. The competitions can range from beginner level to expert level, with varying levels of difficulty and complexity.

9.2.4. Scalability

Scalability refers to the ability of a system to handle increasing amounts of data or traffic without compromising its performance or functionality. In the case of the vehicle classification model, scalability can be achieved through various techniques. One approach is to use distributed computing frameworks such as Apache Spark or Hadoop to distribute the computation of the model across multiple machines, allowing it to handle larger datasets. Another approach to improve scalability is to use cloud-based services to deploy the model. Cloud platforms provide flexible and scalable computing resources that can be easily scaled up or down depending on the demand. Additionally, optimizing the code of the model to run faster and more efficiently can also improve its scalability.

10. SOURCE CODE

#Data Collection

In [11]:
```
from imutils import paths
import os

imagePaths = list(paths.list_images("D:\\train\\train"))

for imagePath in imagePaths:
    print(imagePath)
    name = imagePath.split(os.path.sep)[-2]
    print(name)
```

#Pre-Processing

[10]:
```
import random
import cv2
import numpy as np
random.seed(21)
random.shuffle(imagePaths)
labels = []
data = []

for imagePath in imagePaths:
    image = cv2.imread(imagePath)
    image = cv2.resize(image, (64, 64))
    data.append(image)
    name = imagePath.split(os.path.sep)[-2]

    if  name == "Bicycle":
        label = np.array([1,0, 0, 0])
    elif name == "Bus":
        label = np.array([0,1, 0, 0])
    elif name == "Car":
        label = np.array([0,0, 1, 0])
    elif name == "Motorcycle":
        label = np.array([0,0, 0, 1])

    labels.append(label)
```

In [10]:
```
from numpy import save

print(data.shape)
```

```
(13520, 64, 64, 3)
```

In [11]:
```
save("vehicle_data.npy",data)
save("vehicle_labels.npy",labels)
```

In [12]:
```
import random
import cv2
import numpy as np

data = np.load("vehicle_data.npy")
labels = np.load("vehicle_labels.npy")
print("Dataset shape: ")
print(data.shape)
```

```
Dataset shape:
(13520, 64, 64, 3)
```

#Splitting of data in train and test set

In [13]:
```
from sklearn.model_selection import train_test_split
train_set, test_set, train_labels, test_labels  = train_test_split(data,labels, test_size=0.25, random_state=21)
```

In [6]:
```
print("Train set shape : ")
print( train_set.shape)
```

```
Train set shape :
(10138, 64, 64, 3)
```

In [7]:
```
print("Test set shape : ")
print(test_set.shape)
```

```
Test set shape :
(3380, 64, 64, 3)
```

#Model Training

In [15]:
```
from keras.utils import np_utils
from keras.models  import Sequential
from  keras.layers import Dense, Activation, Flatten, Dropout
from keras.optimizers import SGD, Adam, RMSprop
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras import regularizers
```

In [21]:

```
model = Sequential()
model.add(Conv2D(64,(3,3), padding="same", input_shape = (64,64,3), activation="relu"))
model.add(Conv2D(128,(3,3), padding="same", activation="relu"))
model.add(Conv2D(64,(3,3), padding="same", activation="relu",kernel_regularizer=regularizers.l2()))
# model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
# model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
# model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
# model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
# model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
# model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
# model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
# model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(3,3)))
model.add(Flatten())

#model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(750, activation='relu'))
model.add(Dense(750, activation='relu'))
#model.add(Dense(750, activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(600, activation='relu'))
model.add(Dense(600, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(500, activation='relu'))
model.add(Dense(500, activation='relu'))
model.add(Dense(4, activation='softmax'))
model.compile(loss="categorical_crossentropy", optimizer=SGD(learning_rate=0.001), metrics = ["accuracy"])
H= model.fit(train_set,train_labels, epochs=30, validation_data=(test_set, test_labels))
```

```
7855
Epoch 20/30
317/317 [==============================] - 588s 2s/step - loss: 0.9102 - accuracy: 0.9109 - val_loss: 1.3209 - val_accuracy: 0.
7811
Epoch 21/30
317/317 [==============================] - 593s 2s/step - loss: 0.8903 - accuracy: 0.9136 - val_loss: 1.2770 - val_accuracy: 0.
8050
Epoch 22/30
317/317 [==============================] - 595s 2s/step - loss: 0.8609 - accuracy: 0.9227 - val_loss: 1.3113 - val_accuracy: 0.
7959
Epoch 23/30
317/317 [==============================] - 584s 2s/step - loss: 0.8390 - accuracy: 0.9284 - val_loss: 1.3082 - val_accuracy: 0.
7973
Epoch 24/30
317/317 [==============================] - 655s 2s/step - loss: 0.8187 - accuracy: 0.9338 - val_loss: 1.3010 - val_accuracy: 0.
7997
Epoch 25/30
317/317 [==============================] - 853s 3s/step - loss: 0.7894 - accuracy: 0.9410 - val_loss: 1.2945 - val_accuracy: 0.
7991
Epoch 26/30
317/317 [==============================] - 780s 2s/step - loss: 0.7763 - accuracy: 0.9421 - val_loss: 1.3278 - val_accuracy: 0.
8000
Epoch 27/30
317/317 [==============================] - 832s 3s/step - loss: 0.7547 - accuracy: 0.9488 - val_loss: 1.3037 - val_accuracy: 0.
8083
Epoch 28/30
317/317 [==============================] - 766s 2s/step - loss: 0.7386 - accuracy: 0.9538 - val_loss: 1.4359 - val_accuracy: 0.
7828
Epoch 29/30
317/317 [==============================] - 849s 3s/step - loss: 0.7247 - accuracy: 0.9541 - val_loss: 1.3669 - val_accuracy: 0.
8018
Epoch 30/30
317/317 [==============================] - 775s 2s/step - loss: 0.7038 - accuracy: 0.9619 - val_loss: 1.3764 - val_accuracy: 0.
8074
```

FIGURE 8: Model training

```
In [25]: #np.save('my_history.npy',H.history)
```

```
In [3]: h=np.load('my_history.npy',allow_pickle='TRUE').item()
```

```
In [22]: import h5py
         #model.save("test9619.h5")
         model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 64)        1792

 conv2d_1 (Conv2D)           (None, 64, 64, 128)       73856

 conv2d_2 (Conv2D)           (None, 64, 64, 64)        73792

 max_pooling2d (MaxPooling2D  (None, 21, 21, 64)       0
 )

 flatten (Flatten)           (None, 28224)             0

 dropout (Dropout)           (None, 28224)             0

 dense (Dense)               (None, 750)               21168750

 dense_1 (Dense)             (None, 750)               563250

 dropout_1 (Dropout)         (None, 750)               0

 dense_2 (Dense)             (None, 600)               450600

 dense_3 (Dense)             (None, 600)               360600

 dropout_2 (Dropout)         (None, 600)               0

 dense_4 (Dense)             (None, 500)               300500

 dense_5 (Dense)             (None, 500)               250500

 dense_6 (Dense)             (None, 4)                 2004

=================================================================
```

FIGURE 9: Model Summary

#Model Evaluation

In [5]:
```
#test0004 96%
from matplotlib import pyplot as plt
plt.plot(h["loss"])
plt.plot(h["val_loss"])

plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend(['loss','val_loss'],loc = 'upper right')

plt.show()
```

In [6]:
```
#test0004

from matplotlib import pyplot as plt
plt.plot(h["accuracy"])
plt.plot(h["val_accuracy"])
plt.ylabel("accuracy")
plt.xlabel("epochs")
plt.legend(['accuracy','val_accuracy'],loc = 'lower right')
plt.show()
```

In [2]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras

model = keras.models.load_model("test9619.h5")
```

In [3]:
```
predicted = []
```

In [9]:
```
pre = model.predict(test_set)
print(pre)
predicted = np.argmax(pre,axis=1)
print(predicted)
```

 # confusion

In [10]:
```
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [11]:
```
actual = np.argmax(test_labels,axis=1)
print(actual)
```

```
[2 0 2 ... 3 2 3]
```

In [12]:
```
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[0,1,2,3])
print('Confusion matrix : \n',matrix)
```

```
Confusion matrix :
 [[1184    1    7   17]
 [   1 1598    5    0]
 [  21   42 5019   13]
 [  25    3    8 2194]]
```

FIGURE 10: Confusion Matrix

```
In [13]: # outcome values order in sklearn
         tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[0,1,2,3])
         print("Outcome values : \n", tp, fn, fp, tn)

         Outcome values :
          [1184    1    7   17] [   1 1598    5    0] [  21   42 5019   13] [  25    3    8 2194]
```

#classification report

```
In [14]: report = classification_report(actual,predicted,labels=[0,1,2,3])
         print("Classification Report : \n", report)  #test classification report

         Classification Report :
                       precision    recall  f1-score   support

                    0       0.86      0.91      0.89      1618
                    1       0.92      0.98      0.95      2133
                    2       0.98      0.95      0.96      6781
                    3       0.92      0.92      0.92      2986

             accuracy                           0.94     13518
            macro avg       0.92      0.94      0.93     13518
         weighted avg       0.94      0.94      0.94     13518
```

FIGURE 11: Classification Report

11. BIBLIOGRAPHY

1. https://en.wikipedia.org/wiki/Convolutional_neural_network

2. https://www.tensorflow.org/

3. https://pytorch.org/

4. https://keras.io/

5. https://www.nvidia.com/en-us/deep-learning-ai/

6. https://deepmind.com/

7. https://machinelearningmastery.com/

8. https://www.technologyreview.com/topic/deep-learning/

9. http://cs231n.stanford.edu/

10. https://ai.googleblog.com/search/label/Image%20Classification

11. https://www.kaggle.com/

12. https://towardsdatascience.com/

13. https://azure.microsoft.com/en-us/services/machine-learning/