A PROJECT REPORT

on

Car Tyre Evaluation based on image processing

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN

INFORMATION TECHNOLOGY

BY

Amlan Prasad	21052136
Harshil Gautam	21052155
Mayank Bhardwaj	21052161
Nikhil Choudhary	21052166
Tanmay Padhi	21052205

UNDER THE GUIDANCE OF
Pradeep Kumar Malik

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

"*Car Tyre Evaluation based on image processing*"

submitted by

Amlan Prasad	21052136
Harshil Gautam	21052155
Mayank Bhardwaj	21052161
Nikhil Choudhary	21052166
Tanmay Padhi	21052205

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during the year 2022-2023, under our guidance.

Date: 29/03

Pradeep Kumar Malik
Project Guide

Acknowledgements

We are profoundly grateful to **Pradeep Kumar Malik** of **Affiliation** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. .....

Amlan Prasad
Harshil Gautam
Mayank Bhardwaj
Nikhil Choudhary
Tanmay Padhi

Contents

List of Figures

Chapter 1

Introduction

In today's automotive industry, ensuring the safety and performance of vehicles is paramount. One crucial aspect of vehicle safety is the condition of its tires. Tires are the only point of contact between a vehicle and the road, making their integrity and condition essential for safe driving. However, manual inspection of tires can be time-consuming, subjective, and prone to errors.

To address these challenges, we propose the development of an Automated Car Tire Inspection System utilizing Image Processing techniques. This project aims to leverage advancements in computer vision and machine learning to automate the process of tire inspection, enhancing efficiency, accuracy, and reliability.

Key Objectives:

1. Automated Inspection: The primary goal is to automate the tire inspection process, eliminating the need for manual intervention. By employing image processing algorithms, the system will analyze digital images of tires to detect various defects and anomalies.
2. Defect Detection: The system will be trained to identify common tire defects such as cuts, punctures, bulges, uneven wear, and tread depth. Through the analysis of tire images, the system will accurately pinpoint any abnormalities that could compromise tire performance or safety.
3. Real-time Processing: Efficiency is crucial, especially in industrial settings. The project aims to achieve real-time processing capabilities, allowing for swift inspection of tires without causing delays in production or vehicle maintenance processes.
4. User-friendly Interface: To ensure usability across different environments, the system will feature a user-friendly interface. Operators will be able to easily initiate inspections, view results, and take necessary actions based on the system's recommendations.
5. Integration Potential: The proposed system will be designed with integration in mind, allowing seamless incorporation into existing automotive manufacturing processes, service centers, or inspection facilities.

Methodology:

- Image Acquisition: High-resolution images of car tires will be captured using digital cameras or specialized imaging devices.
- Preprocessing: Image preprocessing techniques such as noise reduction, contrast enhancement, and image normalization will be applied to optimize image quality.
- Feature Extraction: Relevant features of the tires, including tread patterns, sidewall condition, and any visible defects, will be extracted using image processing algorithms.
- Machine Learning: Supervised machine learning algorithms such as convolutional neural networks (CNNs) will be trained on labeled tire images to recognize and classify various defects.
- Deployment: The trained model will be deployed within the automated inspection system, capable of analyzing incoming tire images and providing inspection reports in real-time.

Expected Outcomes:

- Increased efficiency and accuracy in tire inspection processes.
- Reduction in manual labor costs and human error.
- Enhanced safety and performance of vehicles through timely

Chapter 2

Some Basic Concepts

What is image processing?
Image processing is a multidisciplinary field that encompasses various techniques and methodologies aimed at analyzing, manipulating, and enhancing digital images. It involves the utilization of algorithms and computational methods to extract meaningful information, improve visual quality, and perform various tasks on images captured by cameras or generated by other means.

At its core, image processing involves the following fundamental steps:

Image Acquisition: This step involves capturing or obtaining digital images through sensors, cameras, or other devices. These images may be in various formats such as grayscale, RGB (Red, Green, Blue), or other color models.

Preprocessing: In this step, the acquired images undergo initial processing to enhance their quality and prepare them for subsequent analysis. This may include operations such as noise reduction, contrast enhancement, and image normalization.

Image Enhancement: Image enhancement techniques aim to improve the visual quality of images by emphasizing certain features or reducing unwanted artifacts. This may involve operations such as sharpening, blurring, or adjusting brightness and contrast.

Image Restoration: Image restoration techniques are used to recover or reconstruct degraded or damaged images caused by factors such as noise, blur, or compression. These techniques aim to restore the original appearance of the image as much as possible.

Feature Extraction: Feature extraction involves identifying and extracting relevant information or features from images. This may include detecting edges, corners, shapes, textures, or other objects of interest within the image.

Image Segmentation: Image segmentation divides an image into meaningful regions or segments based on characteristics such as color, intensity, or texture. This facilitates further analysis and understanding of the content within the image.

Object Detection and Recognition: Object detection and recognition techniques involve identifying and classifying specific objects or patterns within images. This may include tasks such as face detection, object tracking, or character recognition.

Image Compression: Image compression techniques reduce the storage space required for images by eliminating redundant or irrelevant information while preserving essential visual quality. This is particularly important for efficient storage and transmission of images in various applications.

Image Understanding and Interpretation: Image understanding aims to interpret the content and context of images, enabling computers to extract semantic meaning or make decisions based on visual information. This may involve higher-level tasks such as scene understanding, image classification, or image-based reasoning.

Overall, image processing plays a crucial role in a wide range of applications, including medical imaging, satellite imagery analysis, surveillance, remote sensing, digital photography, and multimedia systems. Its continuous advancement and integration with other fields such as computer vision and artificial intelligence contribute significantly to technological innovation and development in various domains.

What is Tensor flow?

TensorFlow is an open-source machine learning framework developed by Google Brain, initially released in 2015. It is designed to facilitate the creation, deployment, and scaling of machine learning models, particularly deep learning models. TensorFlow provides a comprehensive ecosystem of tools and libraries for building and training various types of neural networks, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more advanced architectures.

At its core, TensorFlow represents computations as data flow graphs, where nodes represent mathematical operations, and edges represent the flow of data (tensors) between these operations. This computational graph allows for efficient execution on both CPUs and GPUs, making TensorFlow suitable for a wide range of hardware platforms, including desktops, servers, and mobile devices.

Key features of TensorFlow include:

Flexibility: TensorFlow offers flexibility in building and customizing machine learning models, allowing developers to define their neural network architectures and experiment with different configurations easily.

Scalability: TensorFlow is designed for scalability, enabling the training and deployment of machine learning models across distributed computing environments, including clusters of servers and cloud platforms.

High-level APIs: TensorFlow provides high-level APIs, such as Keras (integrated into TensorFlow since version 2.0), which simplifies the process of building and training neural networks, making it more accessible to beginners and experienced practitioners alike.

TensorBoard: TensorFlow includes TensorBoard, a visualization toolkit for visualizing and analyzing the training process and model performance. TensorBoard allows users to monitor metrics, visualize computational graphs, and debug machine learning models effectively.

Support for different languages: While TensorFlow is primarily developed in Python, it offers support for other programming languages, including C++, Java, and JavaScript, through language bindings and APIs, allowing developers to integrate TensorFlow into their existing software stack.

Community and ecosystem: TensorFlow has a large and active community of developers, researchers, and practitioners who contribute to its development, share best practices, and provide support through forums, documentation, and open-source contributions. This vibrant ecosystem enables rapid innovation and adoption of TensorFlow in various domains, including research, industry, and academia.

Overall, TensorFlow has become one of the most popular and widely used frameworks for machine learning and deep learning, empowering developers and organizations to build and deploy state-of-the-art machine learning models for a diverse range of applications, including computer vision, natural language processing, speech recognition, and more.

What is Keras?
Keras is an open-source neural network library written in Python. It was developed with a focus on enabling fast experimentation with deep neural networks and providing a user-friendly interface for building and training various types of neural networks. Keras was designed to be modular, extensible, and easy to use, making it suitable for both beginners and experienced practitioners in the field of machine learning and artificial intelligence.

Key features of Keras include:

User-friendly API: Keras provides a simple and intuitive API that abstracts away the complexities of building and training neural networks. It allows developers to define neural network models using high-level building blocks, such as layers and models, without needing to understand the underlying implementation details.

Modularity: Keras follows a modular design, allowing users to easily assemble neural network models from reusable building blocks. It provides a wide range of pre-defined layers for common

neural network architectures, as well as the flexibility to create custom layers tailored to specific requirements.

Flexibility: Keras supports a variety of backends for tensor computation, including TensorFlow, Microsoft Cognitive Toolkit (CNTK), and Theano. This enables users to choose the backend that best suits their needs while leveraging the high-level API provided by Keras.

Ease of extension: Keras is designed to be easily extendable, allowing developers to add custom functionality or integrate with other libraries seamlessly. It provides a clean and well-documented codebase, making it straightforward to contribute new features or extensions to the Keras ecosystem.

Integration with TensorFlow: Since TensorFlow version 2.0, Keras has been integrated as the official high-level API for building and training neural networks within the TensorFlow framework. This integration provides users with the combined benefits of both TensorFlow's powerful backend and Keras' user-friendly interface.

Community and ecosystem: Keras has a large and active community of developers, researchers, and practitioners who contribute to its development, share best practices, and provide support through forums, documentation, and open-source contributions. This vibrant ecosystem has led to widespread adoption of Keras in various domains, including research, industry, and academia.

Overall, Keras is widely regarded as one of the most accessible and versatile libraries for deep learning, enabling rapid prototyping, experimentation, and deployment of neural network models for a wide range of applications, including computer vision, natural language processing, and reinforcement learning.

Chapter 3

Problem Statement / Requirement Specifications

The project aims to develop an automated system for car tire evaluation using image processing techniques. With the increasing demand for vehicle safety and performance, assessing the condition of car tires is crucial for ensuring roadworthiness and preventing accidents. Traditional methods of tire inspection often rely on manual visual inspection, which can be time-consuming, subjective, and prone to human error. Therefore, there is a need for a reliable and efficient automated solution to evaluate car tires accurately.

The key objectives of the project include:

Tire Tread Analysis: Develop algorithms to analyze the tread pattern of car tires captured in digital images. The system should be able to detect and quantify tread wear, irregularities, and damage such as cuts, cracks, or bulges.

User Interface: Design a user-friendly interface for interacting with the system, allowing users to input images of car tires and view the evaluation results. The interface should provide informative visual feedback and actionable insights to guide users in assessing tire condition and making informed maintenance decisions.

Performance Evaluation: Evaluate the accuracy, reliability, and efficiency of the proposed image processing algorithms through comprehensive testing and validation on a diverse dataset of car tire images. Assess the system's performance in real-world scenarios and compare it against existing manual inspection methods.

By addressing these objectives, the project aims to provide a robust and scalable solution for car tire evaluation using image processing, offering benefits such as improved safety, reduced maintenance costs, and enhanced vehicle performance. This automated system has the potential to revolutionize the way car tires are inspected and maintained, contributing to safer roads and greater peace of mind for vehicle owners and operators.

3.1 Project Planning

Define Scope and Objectives: Clearly outline the goals and scope of the project, focusing on developing an automated system for evaluating car tires using image processing techniques.

Gather Requirements: Collaborate with stakeholders to gather requirements and identify necessary resources for the project, including hardware and software tools.

Research and Literature Review: Conduct a literature review to explore existing methodologies and algorithms related to image processing for car tire evaluation.

Design System Architecture: Design the architecture of the system, defining components for image processing, feature extraction, classification, and user interface.

Develop Image Processing Algorithms: Implement algorithms for analyzing tire tread patterns, detecting damage, and estimating tire pressure.

User Interface Design: Design a user-friendly interface for interacting with the system, incorporating visualizations and feedback mechanisms.

Data Collection and Preparation: Collect a diverse dataset of car tire images and preprocess the data for analysis.

System Integration and Testing: Integrate components into a cohesive system and conduct testing to ensure accuracy and reliability.

Documentation: Prepare comprehensive documentation covering requirements, design, implementation, testing, and user guides.

Deployment and Evaluation: Deploy the system, evaluate performance, and gather feedback for improvement.

Maintenance and Support: Establish mechanisms for ongoing maintenance, updates, and user support.

Project Management: Define a schedule, allocate resources, track progress, and manage risks throughout the project lifecycle.

3.2 Project Analysis

Market Analysis:
Investigate the market demand for automated car tire evaluation systems.
Identify potential competitors and existing solutions in the market.
Assess the current trends and technological advancements in image processing for automotive applications.

User Needs Analysis:
Conduct surveys or interviews with potential users, such as vehicle owners, mechanics, and automotive professionals, to understand their needs and pain points regarding tire evaluation.

Identify specific requirements and preferences for an automated tire evaluation system, including accuracy, ease of use, and integration with existing workflows.

Technical Feasibility:

Evaluate the technical feasibility of implementing image processing algorithms for car tire evaluation.

Assess the availability of relevant datasets and tools for developing and testing the algorithms.

Consider hardware requirements for image acquisition and processing, such as cameras and computing resources.

Regulatory and Safety Considerations:

Investigate regulatory requirements and safety standards related to automotive maintenance and inspection.

Ensure compliance with relevant regulations and standards for deploying the automated tire evaluation system.

Cost-Benefit Analysis:

Estimate the costs associated with developing, deploying, and maintaining the system, including personnel, equipment, and software.

Conduct a cost-benefit analysis to determine the potential return on investment and economic viability of the project.

Risk Analysis:

Identify potential risks and challenges associated with the project, such as technical complexity, data quality issues, or regulatory hurdles.

Develop risk mitigation strategies to address and minimize the impact of identified risks throughout the project lifecycle.

3.3 System Design

Overall Architecture:

Design a modular architecture that encompasses all components of the system, including image acquisition, preprocessing, feature extraction, classification, and user interface.

Ensure scalability and flexibility to accommodate future enhancements and modifications.

Image Acquisition:

Select appropriate hardware for capturing high-quality images of car tires, such as digital cameras or mobile devices.

Determine the optimal camera placement and lighting conditions for consistent and accurate image acquisition.

Develop mechanisms for automatically capturing and importing images into the system.

Preprocessing:

Implement preprocessing techniques to enhance the quality of acquired images, including noise reduction, color correction, and resizing.

Normalize images to ensure consistent features and eliminate variations due to factors such as lighting and perspective.

Feature Extraction:

Develop algorithms to extract relevant features from car tire images, such as tread patterns, texture, and structural characteristics.

Utilize techniques such as edge detection, histogram analysis, and convolutional neural networks (CNNs) for feature extraction.

Classification:

Design classifiers to analyze extracted features and classify car tire images into different categories based on tire condition, such as tread wear, damage, and pressure.

Implement machine learning models, such as support vector machines (SVMs) or deep neural networks, for accurate classification.

User Interface:

Design an intuitive and user-friendly interface for interacting with the system.

Include features for uploading images, displaying evaluation results, and providing actionable insights to users.

Incorporate visualization tools, such as graphs and charts, to present analysis findings effectively.

Integration and Communication:

Establish communication channels between system components to facilitate data flow and interaction.

Ensure seamless integration with external systems or databases for data storage, retrieval, and management.

Scalability and Performance:

Optimize algorithms and data processing pipelines for efficiency and scalability, particularly when dealing with large datasets or real-time processing requirements.

Consider parallelization and distributed computing techniques to improve system performance and responsiveness.

Security and Privacy:

Implement security measures to protect sensitive data and prevent unauthorized access to the system.

Incorporate encryption, authentication, and access control mechanisms to ensure data integrity and confidentiality.

Testing and Validation:

Develop comprehensive testing strategies to validate the functionality, accuracy, and robustness of the system.

Conduct unit tests, integration tests, and system tests to identify and address any issues or inconsistencies.

Documentation and Maintenance:

Document the system design, architecture, and implementation details for reference and future maintenance.

Establish procedures for ongoing maintenance, updates, and support to ensure the system remains functional and up-to-date.

3.3.1 Design Constraints

Hardware Limitations:

Availability of suitable cameras or imaging devices for capturing high-quality images of car tires.

Processing power and memory constraints of hardware devices for performing image processing tasks efficiently.

Data Quality:

Variability in lighting conditions, angles, and perspectives may affect the quality and consistency of captured images.

Limited availability of diverse and representative datasets for training and testing machine learning models.

Real-Time Processing:

Requirements for real-time or near-real-time processing impose constraints on the speed and efficiency of image processing algorithms.

Processing latency must be minimized to provide timely evaluation results to users.

Accuracy and Reliability:

The system must achieve high levels of accuracy and reliability in evaluating tire condition to ensure user trust and confidence.

Constraints on algorithm complexity and computational resources may impact the system's ability to accurately classify tire images.

User Interface:

Constraints on screen size, resolution, and input modalities (e.g., touchscreens, keyboards) for designing the user interface.

Consideration of user preferences, accessibility requirements, and ergonomic principles in interface design.

Regulatory Compliance:

Compliance with regulatory standards and safety requirements for automotive maintenance and inspection.

Ensure that the system meets legal and regulatory obligations related to data privacy, security, and consumer protection.

Integration with Existing Systems:

Compatibility and interoperability constraints when integrating the system with existing automotive diagnostic tools, databases, or management systems.
Ensure seamless data exchange and communication between the car tire evaluation system and other software or hardware components.

Environmental Factors:
Environmental conditions such as temperature, humidity, and dust may impact the performance and reliability of hardware components.
Consideration of environmental factors when selecting equipment and designing the system for durability and robustness.

3.3.2 System Architecture **OR** Block Diagram

Image Acquisition Module:
Responsible for capturing images of car tires using cameras or imaging devices.
Converts analog signals to digital images and transfers them to the preprocessing module.
Ensures consistent image quality and resolution across different lighting conditions and environments.

Preprocessing Module:
Performs initial processing on captured images to enhance quality and prepare them for feature extraction.
Includes operations such as noise reduction, color correction, resizing, and normalization.
Ensures uniformity and consistency in image characteristics to facilitate accurate analysis.

Feature Extraction Module:
Extracts relevant features from preprocessed images that are indicative of tire condition.
Utilizes techniques such as edge detection, texture analysis, and convolutional neural networks (CNNs) to identify key patterns and structures.
Generates feature vectors or descriptors representing the extracted features for input to the classification module.

Classification Module:
Classifies car tire images based on extracted features into different categories, such as tread wear, damage, and pressure level.
Employs machine learning algorithms, such as support vector machines (SVMs), random forests, or deep neural networks, for classification.
Trained on labeled datasets to learn patterns and relationships between features and tire conditions.

User Interface Module:
Provides an intuitive and user-friendly interface for interacting with the system.
Allows users to upload images of car tires, view evaluation results, and receive actionable insights.

Incorporates visualizations, feedback mechanisms, and interactive controls to enhance user experience.

Integration and Communication Layer:

Facilitates communication and data exchange between different modules within the system.

Ensures seamless integration of hardware and software components, such as cameras, processors, and user interfaces.

Enables interoperability with external systems or databases for data storage, retrieval, and management.

Data Management Module:

Manages the storage, retrieval, and processing of image data and associated metadata.

Stores labeled datasets for training and validation purposes.

Implements mechanisms for data backup, versioning, and security to ensure data integrity and confidentiality.

Performance Optimization Layer:

Optimizes system performance and efficiency to meet real-time or near-real-time processing requirements.

Utilizes techniques such as parallelization, multithreading, and distributed computing to enhance computational speed and scalability.

Monitors system metrics and resource utilization to identify and address performance bottlenecks.

Security and Privacy Layer:

Implements security measures to protect sensitive data and prevent unauthorized access to the system.

Encrypts data transmission and storage, implements authentication and access control mechanisms, and enforces privacy-preserving practices.

Complies with regulatory requirements and industry standards for data security and privacy.

By adopting this system architecture, we can design and develop a robust and scalable solution for car tire evaluation using image processing techniques, ensuring accurate and efficient assessment of tire condition for improved safety and performance.

Chapter 4

Implementation

4.1 Methodology OR Proposal

Introduction:

The objective of this project is to create a reliable model that can distinguish between "cracked" and "normal" tire textures. For automated inspection and quality control procedures in the tire manufacturing industries, this classification duty is crucial. In order to successfully complete this classification task, we present the methods for creating and refining a convolutional neural network (CNN) model in this proposal.

Data Collection and Preprocessing:

In the beginning, a sufficient amount of labeled data including pictures of both normal and cracked tire textures must be gathered. To guarantee the durability of the model, these pictures must to show a variety of textures, lighting scenarios, and orientations. To improve dataset diversity and avoid overfitting, data augmentation methods including rotation, flipping, and scaling might be used.

Model Architecture Design:

The classification performance is significantly influenced by the architecture of the CNN model. To extract features from the input photos, we suggest using a CNN architecture that consists of convolutional layers followed by max-pooling layers. Based on testing and performance analysis, the number of convolutional layers, filter sizes, and neurons in each layer can be changed. We'll use dropout layers and batch normalization to speed up convergence and avoid overfitting.

4.2 Testing/Verification Plan

1. Data Testing:
   - Verify the integrity and quality of the collected dataset by inspecting sample images and labels.
   - Randomly select subsets of data for manual inspection to ensure that it represents a diverse range of tire textures.
   - Perform statistical analysis to identify any biases or anomalies in the dataset.

2. Preprocessing Testing:
   - Validate the preprocessing steps such as resizing, normalization, and augmentation by visually inspecting the transformed images.
   - Verify that data augmentation techniques preserve the semantic content of the images while introducing variability.

3. Model Architecture Testing:
- Test the proposed CNN architecture on a small subset of the dataset to ensure proper implementation and functioning.
- Validate the model's architecture by inspecting the summary of layers and parameters.
- Conduct exploratory experiments to evaluate the model's capacity to learn relevant features from the input images.

4. Training Testing:
- Train the model on a subset of the dataset and monitor training progress using metrics such as loss and accuracy.
- Verify that the model converges to a stable state and exhibits reasonable performance on the training data.
- Perform hyperparameter tuning experiments to identify optimal values for learning rate, dropout rate, etc.

5. Evaluation Testing:
- Evaluate the trained model on a separate validation set to assess its generalization performance.
- Calculate standard classification metrics such as accuracy, precision, recall, and F1-score to quantify the model's performance.
- Generate visualizations such as confusion matrices and ROC curves to analyze the model's behavior across different classes.

6. Deployment Testing:
- Deploy the trained model in a test environment to evaluate its performance in real-world scenarios.
- Verify that the deployed model meets the specified performance criteria and provides accurate predictions.
- Conduct stress testing by simulating high loads or unexpected inputs to assess the model's robustness and scalability.

7. Documentation and Reporting:
- Document the testing process, including test cases, methodologies, and outcomes.
- Provide clear and comprehensive reports summarizing the testing results, highlighting any issues or areas for improvement.
- Communicate findings to stakeholders and obtain feedback for further refinement of the model.

8. Iterative Testing:
- Conduct iterative testing and validation cycles to address any identified issues or shortcomings.
- Incorporate feedback from stakeholders and domain experts to continuously improve the model's performance and reliability.

By following this testing/verification plan, the project team can ensure that the developed CNN model for tire texture classification undergoes thorough evaluation and validation, leading to a successful and satisfactory outcome.

4.3 Training Procedure:

To assess the performance of the model, the dataset will be divided into training, validation, and testing sets. Using backpropagation and an appropriate optimization technique, such Adam, RMSprop, or SGD with momentum, the model parameters will be tuned during training. In order to guarantee convergence and avoid overfitting, we shall keep an eye on the training loss and accuracy measures. Techniques for learning rate scheduling and early stopping will be used to optimize the training process and enhance generalization.

Evaluation Metrics:

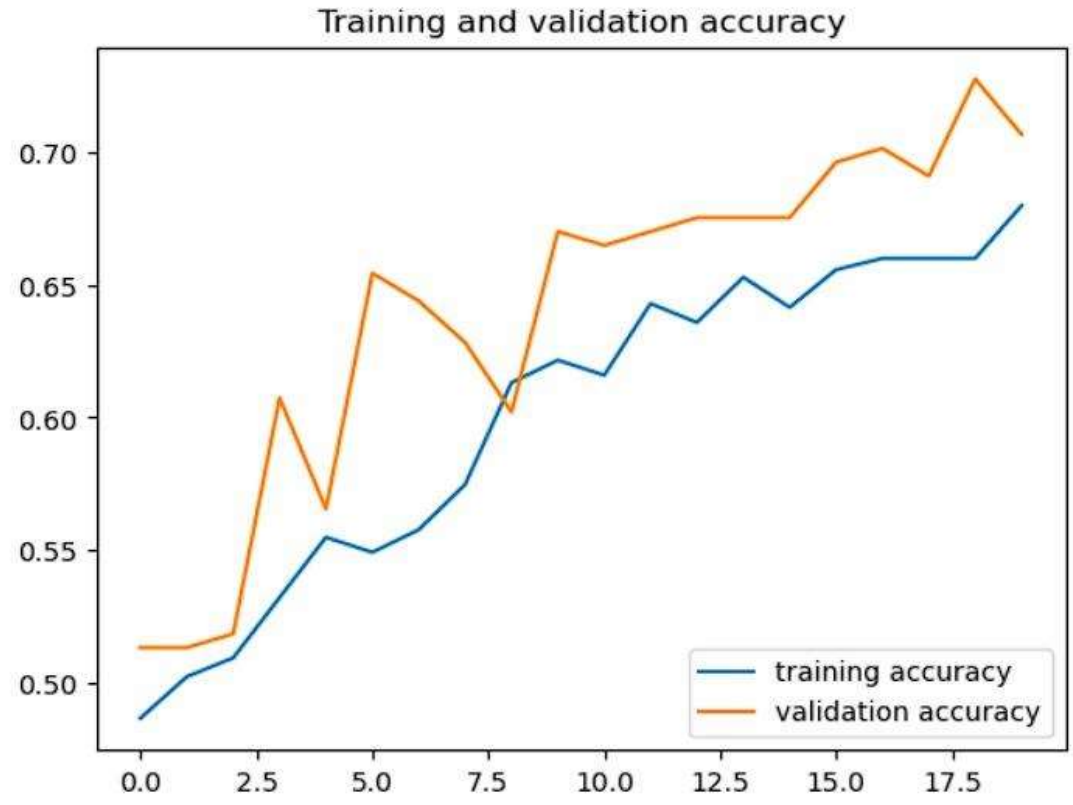
Standard classification metrics, including recall, accuracy, precision, and F1-score, will be used to assess the model's function. In addition, ROC curves and confusion matrices will be produced to evaluate the model's capability to distinguish between tires with normal and cracked surfaces. To verify the model's performance across several dataset subsets, cross-validation techniques can be used.

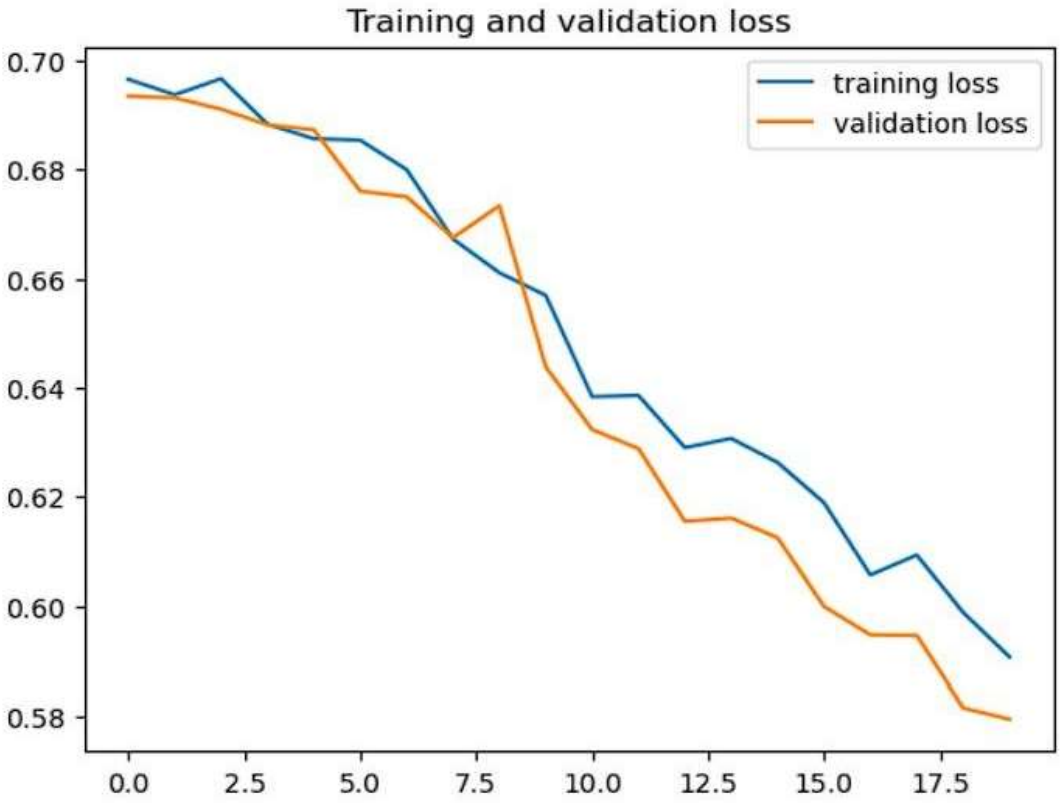
Hyperparameter Tuning:

Using grid search or random search approaches, hyperparameters like learning rate, dropout rate, and the number of neurons in each layer will be adjusted. To determine which hyperparameter setup best maximizes the model's performance on the validation set, we will experiment with various combinations.

4.4 Model Deployment:

The model will be implemented in production contexts for real-time inference after training is finished and performance is adequate. The trained model may need to be transformed into a lightweight format that is appropriate for edge devices or cloud-based services as part of the deployment procedure. The model's performance will be sustained over time through the application of retraining techniques and constant monitoring.


Training and validation accuracy
0.70
0.65
0.60
0.55
0.50
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
training accuracy
validation accuracy


Training and validation loss
0.70
0.68
0.66
0.64
0.62
0.60
0.58
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
training loss
validation loss

Guidelines for Quality Assurance

Assurance of Data Quality:

Make that the dataset is well labeled, representative, and diversified.
To preserve data quality, carry out preprocessing operations on the data, such as normalization, augmentation, and validation.

Model Design and Architecture:

Check if the suggested CNN architecture is appropriate for the purpose of classifying tire textures.
Make sure that best practices like batch normalization, dropout layers, and suitable activation functions are incorporated into the model architecture.

Method of Instruction:

Throughout training, keep an eye on the convergence, loss, and accuracy metrics to validate the process.
Put early halting procedures in place to guarantee model generalization and avoid overfitting.
Make sure the hyperparameters are set correctly to maximize the performance of the model.

Metrics for Evaluation:

Use common classification metrics, like accuracy, precision, recall, and F1-score, to validate the model's performance.
Confusion matrices and ROC curves should be created in order to evaluate the model's capacity to distinguish between tires with normal and cracked surfaces.

Adjusting Hyperparameters:

Keep a record of the hyperparameter tuning procedure, along with the values that were chosen and how they affected the performance of the model.
Make sure that, when performing hyperparameter tuning, methods like grid search or random search are used with diligence.

Model Implementation:

Check to see if the deployed model satisfies the necessary performance requirements in practical settings.
Maintain constant observation over the performance of the deployed model, and use retraining techniques as needed.
Verify that the deployed model complies with all applicable laws and regulations.

Record-keeping and Reporting:

Keep thorough records of every step of the model development process, including data gathering, model architecture, preprocessing, training, assessment, and deployment.
Provide reports that are easy to read and understand, outlining the model's performance, important discoveries, and suggestions for improvement.

Working Together and Communicating:

Encourage candid dialogue and teamwork amongst team members working on the model creation process.
Hold frequent reviews and meetings to go over the status, deal with issues, and make sure the project is still on track.
Comments and Ongoing Development:

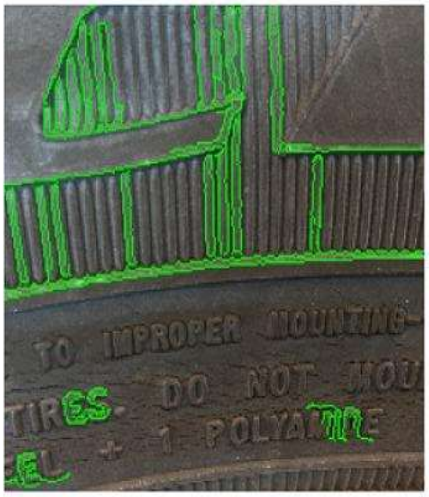
Get input from subject matter experts and stakeholders to determine what needs to be improved.
Establish a feedback loop to take into account lessons discovered and enhance the model's and related processes' quality over time.

Ethical and Compliance Considerations:

Make sure that the rules and ethical standards pertaining to data protection, equity, and transparency are followed during the model building process.
Verify compliance with pertinent standards and regulations by conducting comprehensive audits.

RESULTS:’


```
Predictions-> [[0.4261339]]  Predicted: Good Quality
1/1 [==============================] - 0s 56ms/step
```




```
Predictions-> [[0.56738216]]  Predicted: Bad Quality
1/1 [==============================] - 0s 67ms/step
```

Chapter 5

Standards Adopted

5.1 Design Standards

For car tyre evaluation through image processing techniques, adherence to established design standards is essential to ensure process accuracy and reliability. Principles from engineering standards and best practices in image processing serve as guiding factors.

Practices for Car Tyre Evaluation Design:

1) Adherence to Engineering Standards: Ensure compliance with relevant engineering standards pertaining to automotive safety, tyre performance, and image processing methodologies.
2) Utilization of Image Processing Techniques: Employ suitable image processing algorithms and techniques, such as edge detection, segmentation, and feature extraction, for effective tyre image analysis.
3) Accuracy and Precision: Aim for high levels of accuracy and precision in image processing algorithms to ensure dependable evaluation outcomes.
4) Data Integrity: Implement measures to uphold data integrity throughout the image processing pipeline, encompassing preprocessing stages and post-processing analysis.
5) Validation and Verification: Validate image processing algorithms against ground truth data and verify their performance using appropriate metrics like accuracy, precision, and recall.

5.2 Coding Standards

Adherence to best practices in programming and software development remains crucial for maintaining code quality and readability. Here are coding standards for implementing image processing algorithms:

Coding Standards:

1) Modularity: Design image processing algorithms in a modular manner to enable code reuse and enhance maintainability.
2) Descriptive Naming: Utilize descriptive variable and function names to improve code comprehension and readability.
3) Comments and Documentation: Incorporate comments to elucidate the purpose and functionality of key algorithmic steps, and document the code for future reference.
4) Optimization: Optimize code for efficiency and performance, taking into account computational complexity and memory usage.
5) Error Handling: Implement robust error handling mechanisms to address unexpected scenarios and ensure the reliability of the image processing pipeline.

5.3 Testing Standards

Testing standards play a pivotal role in validating the accuracy and reliability of image processing algorithms utilized in car tyre evaluation. Adherence to recognized testing standards is crucial to ensure comprehensive validation and verification of the image processing pipeline.

Testing Standards:

1) Accuracy Testing: Validate the accuracy of image processing algorithms by comparing results against ground truth data or manual annotations.
2) Robustness Testing: Assess the robustness of image processing algorithms under varying lighting conditions, image quality fluctuations, and tyre type diversities.

3) Performance Testing: Evaluate the performance of image processing algorithms concerning computational efficiency and processing speed.
4) Validation Metrics: Define suitable validation metrics such as precision, recall, and F1-score to gauge the performance of the image processing pipeline.
5) Regression Testing: Perform regression testing to ascertain that alterations to the image processing algorithms do not introduce unintended errors or degrade performance.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion:

In this report, we explored the development of a convolutional neural network (CNN) model aimed at classifying tire textures into two categories: "normal" and "cracked". The CNN model was constructed using the Keras framework with TensorFlow backend.

The process began with data preprocessing, where we implemented custom image preprocessing functions to enhance the discriminative features in the tire texture images. These preprocessing steps included contour detection and filtering based on certain criteria such as area, aspect ratio, and standard deviation of pixel values. Additionally, we utilized data augmentation techniques such as shear, zoom, and horizontal flip to increase the diversity of the training dataset.

The CNN architecture consisted of convolutional, pooling, and fully connected layers. We experimented with different configurations of convolutional layers, dropout rates, learning rates, and optimizers to optimize the model's performance. Regularization techniques such as dropout were employed to mitigate overfitting, and the model was compiled with appropriate loss functions and evaluation metrics.

Training of the CNN model was conducted using a combination of training and validation data generators. We utilized callback functions such as ModelCheckpoint, EarlyStopping, and ReduceLROnPlateau to monitor the training process and prevent overfitting. The training progress was visualized through plots showcasing the accuracy and loss metrics over epochs.

Upon completion of training, the trained CNN model was evaluated on a separate test dataset consisting of unseen tire texture images. Predictions were made on individual images using the trained model, and the results were visually inspected. The model demonstrated promising performance in distinguishing between "normal" and "cracked" tire textures, showcasing its potential utility in automated quality inspection processes.

Overall, this report highlights the successful development and evaluation of a CNN model for tire texture classification. Future work may involve further fine-tuning of model hyperparameters, exploration of alternative architectures, and deployment of the model in real-world scenarios to assess its practical effectiveness.

6.2 Future Scope:

The development of a convolutional neural network (CNN) model for tire texture classification presents numerous opportunities for further research and application. Here are several avenues for future exploration and enhancement:

Fine-tuning and Optimization: Continual refinement of model architecture and hyperparameters can lead to improved classification accuracy and robustness. Exploration of advanced CNN architectures such as DenseNet, Inception, or EfficientNet, along with techniques like transfer learning, could enhance model performance.

Dataset Expansion and Diversity: Increasing the size and diversity of the dataset can help generalize the model better to various tire textures encountered in real-world scenarios. Collecting data from diverse sources, including different tire types, conditions, and

lighting conditions, can enrich the dataset and improve the model's ability to generalize.
Anomaly Detection and Localization: Beyond binary classification, extending the model to detect and localize specific types of tire defects or anomalies (e.g., cracks, bulges, punctures) within the tire texture images could provide valuable insights for quality inspection and maintenance processes in industries such as automotive manufacturing and maintenance.

Real-time Implementation: Implementing the trained model in real-time systems for on-the-fly tire texture classification presents practical applications in automated quality control systems. Integration with industrial cameras and edge computing devices can enable real-time inspection and decision-making, reducing manual intervention and improving efficiency.

Domain Adaptation and Transfer Learning: Adapting the trained model to new domains or environments with minimal labeled data through techniques like domain adaptation and semi-supervised learning can extend its applicability to different contexts and industries, such as rubber manufacturing, material science, and non-destructive testing.

Uncertainty Estimation and Model Explainability: Incorporating uncertainty estimation methods and model explainability techniques can enhance the interpretability and trustworthiness of the model predictions. This is particularly important in safety-critical applications where understanding the model's confidence and decision-making process is essential.

Integration with IoT and Predictive Maintenance Systems: Integrating the tire texture classification model with Internet of Things (IoT) devices and predictive maintenance systems can enable proactive maintenance strategies based on early detection of tire defects. This integration can potentially reduce downtime, prevent accidents, and optimize asset utilization in various industries.

Collaborative Research and Industry Partnerships: Collaboration with industry partners and stakeholders, including tire manufacturers, automotive companies, and research institutions, can facilitate access to domain expertise, data resources, and real-world validation opportunities. Collaborative research endeavors can accelerate the development and deployment of practical solutions addressing industry challenges.

In conclusion, the future scope for tire texture classification using CNN models is vast and multi-faceted, encompassing advancements in model architecture, dataset quality, real-time implementation, domain adaptation, interpretability, and industry collaboration. By pursuing these avenues, researchers and practitioners can unlock the full potential of CNN-based tire texture classification systems and pave the way for transformative applications in quality control, safety assurance, and predictive maintenance across diverse industrial sectors

References

- https://www.kaggle.com/datasets/devsubhash/car-tyres-dataset
- https://www.tensorflow.org/tutorials
- https://keras.io/examples/vision/image_classification_from_scratch/

INDIVIDUAL CONTRIBUTION REPORT:

Car Tyre Evaluation based on image processing

AMLAN PRASAD-21052136
HARSHIL GAUTAM-21052155

Abstract: . Ensuring vehicle safety and performance is critical in today's automotive industry, with tire condition playing a pivotal role. Manual tire inspection is time-consuming, subjective, and error-prone. To address this, we propose an Automated Car Tire Inspection System using Image Processing. This project aims to automate tire inspection through computer vision and machine learning, improving efficiency and accuracy.

Individual contribution and findings: The report's authors include Harshil Gautam (21052155) for Chapter 1, Tanmay Padhi (21052205) for Chapter 2 and 3, Mayank Prasoon Bhardwaj (21052161) for Chapter 4,Nikhil Choudhary (21052166) for Chapter 5 and Amlan Prasad (21052136) for Chapter 6.

Individual contribution to project report preparation: The report was collaboratively authored by five individuals, each contributing to different chapters. Harshil Gautam (21052155) provided the project's introduction in Chapter 1. Tanmay Padhi (21052205) detailed Basic Concepts and Literature Review in Chapter 2, covering Image Processing, TensorFlow, and Keras. In the Problem Statement and Requirement Specifications section, Project Planning and Project Analysis (SRS) were addressed. Tanmay Padhi (21052205) led the latter, defining essential project requirements. Mayank Prasoon Bhardwaj (21052161) authored Chapter 4, detailing the Implementation phase. Nikhil Choudhary (21052166) covered the Standards Adopted in Chapter 5. Chapter 6, contributed by Amlan Prasad (21052136), concludes the report and outlines Future Scope.

Individual contribution for project presentation and demonstration:
The chapters will be presented and demonstrated by each of us in the following order:

Harshil Gautam (21052155): Chapter 1
Tanmay Padhi (21052205): Chapter 2 and 3
Mayank Prasoon Bhardwaj (21052161): Chapter 4
Nikhil Choudhary (21052166): Chapter 5
Amlan Prasad (21052136): Chapter 6

Full Signature of Supervisor: ……

Full signature of the student: ……

TURNITIN PLAGIARISM REPORT

(This report is mandatory for all the projects and plagiarism must be below 25%)

PLAGIARISM SCAN REPORT

Date	April 05, 2024
Exclude URL:	NO

Unique Content	88	Word Count	171
Plagiarized Content	12	Records Found	3

MATCHED SOURCES:

cv2.drawContours(roi_mask, [contour], 0, 255, -1) - CSDN??

https://wenku.csdn.net/answer/c11006f0b2454f068427acdcb6a7b0....

???——???? DF????

https://mc.dfrobot.com.cn/thread-314835-1-1.html

python - Create Coloring book page from image - Stack Overflow

https://stackoverflow.com/questions/76413283/create-coloring....