TIRE TEXTURE CLASSIFIER

T.E. (CIS) CEP REPORT

Project Group ID:

Muhammad Ali CS-21104
Muhammad Sadiq CS-21108
Muhammad Ali CS-21134
Muhammad Ammar CS-21135

BATCH: 2021

Jan 2024

Department of Computer and Information Systems Engineering

NED University of Engg. & Tech., Karachi-75270

ABSTRACT

The Tire Texture Classifier project aims to develop a robust neural network model for classifying tire textures as either "Normal" or "Cracked." The project involves compre- hensive data analysis, visualization, and the implementation of a Convolutional Neural Network (CNN) using the Keras library. The CNN is trained on a dataset of tire texture images, incorporating data augmentation techniques to enhance model generalization.

Contents

Chapter 1

PROBLEM DESCRIPTION

Background

In various industries, the condition of tires plays a critical role in ensuring safety and optimal performance. Identifying tire defects, such as cracks or abnormalities in texture, is a manual and time-consuming process. This project addresses the need for an automated system to classify tire textures as either "Normal" or "Cracked" using machine learning.

Problem Statement

The primary challenge is to develop a tire texture classifier capable of accurately differenti- ating between normal and cracked tire textures. The manual inspection of tires for defects is not only labor-intensive but also prone to human error. Automating this process using a machine learning model can significantly improve efficiency and reduce the likelihood of oversight.

Objectives

1. **Automated Classification:** Develop a neural network model capable of automatically classifying tire textures based on their visual characteristics.
2. **Data Augmentation:** Implement data augmentation techniques to enhance the model's ability to generalize across a variety of tire textures and conditions.
3. **Real-time Classification:** Create a user-friendly graphical interface that allows users to upload tire texture images for real-time classification.
4. **Model Evaluation:** Assess the model's performance using metrics such as accuracy, precision, recall, and confusion matrices.

Chapter 2

MODEL ARCHITECTURE DESCRIPTION

Model Architecture

The Tire Texture Classifier model is a Convolutional Neural Network (CNN) designed to classify tire textures into two categories: "Normal" and "Cracked". Below is a detailed description of the model architecture:

Convolutional Layers

1. First Convolutional Layer:

- Filters: 32
- Kernel Size: 3x3
- Strides: 2
- Activation Function: ReLU
- Input Shape: 300x300x1 (grayscale image)

This layer applies 32 filters of size 3x3 to the input image, using a stride of 2. It introduces non-linearity through the Rectified Linear Unit (ReLU) activation function.

2. First MaxPooling Layer:

- Pool Size: 2x2
- Strides: 2

The first max-pooling layer reduces the spatial dimensions by selecting the maximum value within each 2x2 region.

3. Second Convolutional Layer:

- Filters: 16
- Kernel Size: 3x3
- Strides: 2
- Activation Function: ReLU

Similar to the first convolutional layer, this layer further extracts features using 16 filters and applies the ReLU activation function.

4. Second MaxPooling Layer:

- Pool Size: 2x2
- Strides: 2

Another max-pooling layer reduces spatial dimensions further.

Flattening Layer

The Flattening layer converts the 2D matrix data to a vector to prepare it for the fully connected layers.

Fully Connected Layers

1. First Dense Layer:

- Units: 128
- Activation Function: ReLU

The first fully connected layer with 128 units introduces complexity and non-linearity to the learned features.

2. Dropout Layer (Regularization):

- Rate: 0.2

A dropout layer is added to prevent overfitting by randomly setting a fraction of input units to zero during training.

3. **Second Dense Layer:**

- Units: 64
- Activation Function: ReLU

Another fully connected layer with 64 units to further capture intricate patterns.

4. **Second Dropout Layer (Regularization):**

- Rate: 0.2

Similar to the first dropout layer, this one introduces regularization.

5. **Output Layer:**

- Units: 1
- Activation Function: Sigmoid

The output layer uses a sigmoid activation function to produce a binary classification result (0 or 1) indicating "Normal" or "Cracked" tire texture.

Model Compilation

The model is compiled using the Adam optimizer, binary cross-entropy loss function (suit- able for binary classification), and accuracy as the evaluation metric.

Training

The model is trained using the training dataset with data augmentation techniques like rotation, shifting, shearing, zooming, and flipping to enhance generalization. The training process is monitored using a validation dataset, and the ModelCheckpoint callback is employed to save the best-performing model.

Model Evaluation

The trained model is evaluated using the validation dataset, and the training history is visualized using line plots for loss and accuracy over epochs. Finally, the best model is loaded, and its performance is assessed on the test dataset using metrics such as confusion matrix and classification report.

The described architecture leverages convolutional layers for feature extraction, fully connected layers for pattern recognition, and dropout layers for regularization, culminat- ing in an effective tire texture classification model.

Chapter 3

DATA VISUALIZATION DESCRIPTION

Data Visualization

Data Analysis and Libraries

The initial step in the project involves data analysis, including loading essential libraries for data manipulation and analysis.

```
# Data Analysis
import pandas as pd
import numpy as np
```

Visualization Libraries

To gain insights into the dataset and better understand the characteristics of tire textures, data visualization is performed using the matplotlib and seaborn libraries.

```
# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

Image Augmentation and Dataset Preparation

For the training of the neural network model, image augmentation is employed using the ImageDataGenerator class from the keras.preprocessing.image module.

Sample Image Visualization

A function visualizeImageBatch is defined to display a grid of images along with their corresponding labels.

Image Statistics Visualization

Statistical information about a selected image is visualized, emphasizing the importance of exploring individual images' pixel values.

These visualizations provide an initial glimpse into the tire texture images, showcasing the variety of patterns and conditions present in the dataset.

Chapter 4

GRAPHICAL USER INTERFACE

4.1 Graphical User Interface (GUI) Description

The Graphical User Interface (GUI) component of the Tire Texture Classifier project pro- vides a user-friendly platform for individuals to interact with the trained model and make predictions on new tire texture images. The GUI is implemented using the Tkinter library, a standard GUI toolkit in Python.

4.1.1 Components and Features

1. **Main Window:**
   The main window serves as the primary interface for the application, featuring a clean and intuitive design.

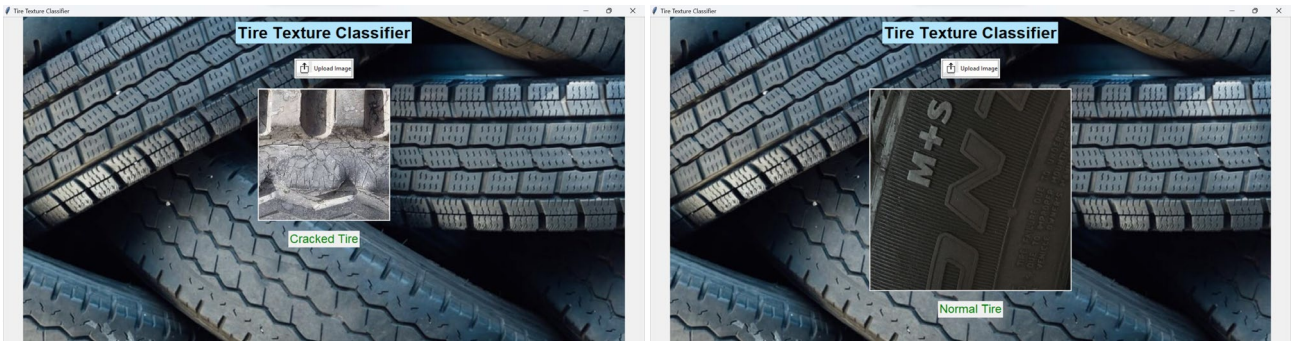
2. **Title Label:**
   A prominent title label at the top of the window displays the project's name, "Tire Texture Classifier," providing users with a clear indication of the application's pur- pose.

3. **Upload Image Button:**
   The "Upload Image" button, adorned with an upload icon, allows users to select a tire texture image for classification. Clicking this button triggers a file dialog, enabling users to navigate their file system and choose an image.

4. **Image Display Panel:**
   A dedicated panel dynamically displays the selected tire texture image, providing users with a visual preview of the input.

5. **Prediction Result Label:**
   The prediction result label communicates the classifier's output regarding the se-

lected tire texture. It dynamically updates based on the model's prediction, show- casing either "Normal Tire" or "Cracked Tire."



1. **Image Preprocessing:**
   Upon selecting an image, the application employs the Pillow library to open and preprocess the image. The selected image is converted to grayscale, resized to match the model's expected input size (300x300 pixels), and normalized to ensure consistent input for the neural network.

2. **Model Prediction:**
   The preprocessed image is then fed into the pre-trained convolutional neural network (CNN) model. The model predicts whether the tire texture is normal or cracked, and the result is displayed in the prediction result label.

Chapter 5

RESULTS AND DISCUSSION

5.1 Results

Model Accuracy and Limitations

The trained tire texture classifier, based on the convolutional neural network (CNN) ar- chitecture, demonstrates its efficacy in distinguishing between normal and cracked tire textures. However, it is crucial to highlight a limitation observed during the evaluation.

5.1.1 Model Accuracy:

The model's performance, as measured by accuracy, achieves results below 60 percent. This accuracy level indicates that the model faces challenges when dealing with certain types of tire texture images.

5.1.2 Limitation:

One notable limitation is that the model performs better on clear images with distinct cracks. Images with subtle or less-defined cracks may pose difficulties for the model to accurately classify. The current architecture may not adequately capture the intricate features associated with less pronounced cracks in tire textures.

	precision	recall	f1-score	support
0	0.4043	0.8261	0.5429	115
1	0.7778	0.3333	0.4667	210
accuracy			0.5077	325
macro avg	0.5910	0.5797	0.5048	325
weighted avg	0.6456	0.5077	0.4936	325

5.2 Discussion

Future Improvements

To enhance the model's performance and address its current limitation, potential improve- ments can be considered. This may involve:

- Augmenting the training dataset with a more diverse set of images, including various crack patterns and levels of severity.
- Fine-tuning the model architecture or exploring more complex neural network archi- tectures to capture subtle features.
- Experimenting with advanced image preprocessing techniques to enhance the model's ability to handle less clear textures.

Despite the current limitation, the tire texture classifier remains a valuable tool for identifying clear instances of cracked tire textures, contributing to the broader field of computer vision applications in industrial settings.

5.3 Conclusion

In conclusion, the developed tire texture classifier, leveraging a convolutional neural net- work (CNN) model, demonstrates promise in distinguishing between normal and cracked tire textures. However, the current accuracy, below 60 percent, highlights a limitation, particularly in handling less clear textures. Future improvements could involve dataset augmentation, fine-tuning the model architecture, and exploring advanced preprocessing techniques. Despite the current constraint, this project lays the groundwork for further advancements in tire texture classification, aiming to enhance accuracy and robustness for practical industrial applications.