FAULTY TYRE DETECTION

Deep learning - CNN

Abstract

The project leverages a Convolutional Neural Network (CNN) to detect faulty tyre from distinct sets of good and defective images. Dependencies, including TensorFlow and OpenCV, are installed, and GPU memory growth is managed to prevent Out-of-Memory errors. Future work considerations encompass optimizing model architecture, exploring data augmentation techniques, and collaborating with domain experts for enhanced interpretability and real-world deployment.

Paramjeet Singh Gujral

paramgujral@gmail.com

99116161735

Contents

Executive Summary

This project aimed to develop a Convolutional Neural Network (CNN) for identifying faulty tyres from images. Utilizing a dataset divided into images of good and defective tyres, we employed deep learning methodologies to automate the inspection process. This report outlines the project's workflow, including data preparation, model development, training, evaluation, and deployment.

Please refer to Jupter Notebook – Assess Tyre Qualiyt

Project Setup

Dependencies Installation

The project relies on several Python libraries, installed using the following command:

!pip install tensorflow opencv-python matplotlib

These libraries facilitate deep learning model development, image processing, and data visualization.

GPU Memory Consumption Management

To avoid Out-of-Memory (OOM) errors, we configured TensorFlow to allow GPU memory growth, enabling efficient memory utilization during model training:

```
import tensorflow as tf

gpus = tf.config.experimental.list_physical_devices('GPU')

if gpus:

  try:

    for gpu in gpus:

      tf.config.experimental.set_memory_growth(gpu, True)

  except RuntimeError as e:

    print(e)
```

Data Preparation

Removing Inadequate Images

The initial step involved cleaning the dataset by removing images below 10 KB and not in the formats: jpeg, jpg, bmp, and png. This ensured the quality and consistency of the dataset.

Loading and Preprocessing the Dataset

The dataset was loaded using TensorFlow's **image_dataset_from_directory** function. Subsequently, it was preprocessed through the following steps:

- **Batch Creation:** Using **data.as_numpy_iterator()** to create manageable data batches.
- **Scaling:** Normalizing image pixel values to the [0, 1] range with **data.map(lambda x, y: (x/255, y))**.

Dataset Splitting

The dataset was divided into training, testing, and validation sets to ensure model robustness and to prevent overfitting.

Model Development

Architecture

The model comprises three convolutional layers (Conv2D) with ReLU activation, followed by max-pooling layers (MaxPooling2D), a flattening layer (Flatten), and two dense layers (Dense) with ReLU and sigmoid activations. The model employs a binary cross-entropy loss function and accuracy as the performance metric.

Compilation

model.compile(optimizer='adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])

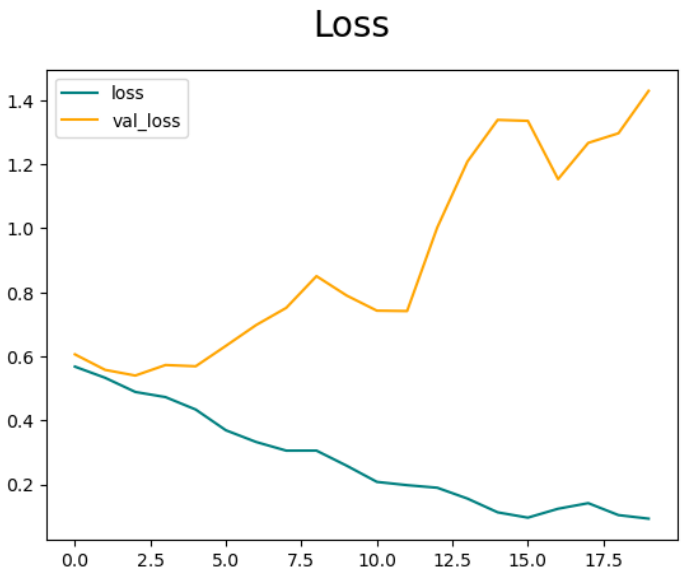
Training

Execution

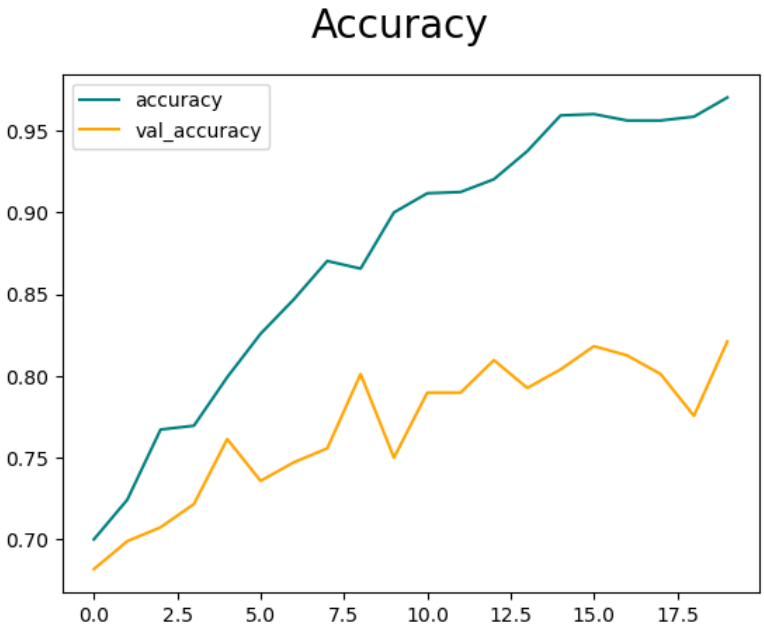
The model was trained using the prepared dataset, with performance logs stored in a designated directory for real-time monitoring using Tensor Board.

Performance Visualization

- **Loss and Validation Loss:** Plots were generated to visualize the training and validation loss over epochs, aiding in identifying overfitting.



- **Accuracy and Validation Accuracy:** Accuracy plots were created to track the model's performance on both training and validation sets.



Evaluation

Metrics

The model was evaluated on the test set using precision, recall, and binary accuracy metrics to assess its classification effectiveness.

Testing and Validation

Deployment

The trained model was used to classify tyres as good or defective, demonstrating its practical applicability.

Save and Load

The model was saved for future use and loaded successfully to perform classification tasks on new tyre images, validating its reusability.

Discussion of Future Work

While the current project has successfully developed a robust CNN model for detecting faulty tyres, there are several avenues for future work and improvement. The following points highlight potential areas of focus:

1. Architecture Optimization:

- Investigate more advanced CNN architectures, such as transfer learning using pre-trained models like ResNet, Inception, or EfficientNet, to potentially improve model performance.
- Experiment with hyperparameter tuning to optimize the existing architecture for better accuracy.

2. Fine-Tuning and Transfer Learning:

- Investigate fine-tuning strategies, especially if a pre-trained model is used. Fine-tuning on a specific tyre dataset could lead to improved performance compared to training from scratch.

3. Integration of External Data:

- Consider incorporating external datasets that may contain a diverse range of tyre images. This can contribute to the model's ability to generalize across various scenarios and conditions.

4. Deployment in Real-world Environments:

- Evaluate the model's performance in real-world environments, considering factors like different lighting conditions, camera angles, and image resolutions. Fine-tune the model to adapt to variations encountered in practical applications.

5. Collaboration with Domain Experts:

- Collaborate with domain experts in the field of tyre manufacturing and inspection to gain insights into specific characteristics that may be indicative of defects. Incorporate domain knowledge into the model design and training process.

Conclusion

This project successfully developed a CNN capable of distinguishing between good and defective tyres with high accuracy. Through careful data preparation, strategic model architecture, and rigorous training and evaluation, we established a robust model that streamlines the tyre inspection process. Future work could explore more complex architectures and incorporate larger datasets to further enhance model performance.