**NULL CLASS INTERNSHIP REPORT**

**PROJECT 2 TITLE: PEDESTRIAN DETECTOR**

**Title: Enhancing Urban Safety through Advanced Pedestrian Detection Systems**

**Introduction**

In the rapidly evolving landscape of smart cities and intelligent transportation systems, ensuring pedestrian safety is paramount. With urbanization on the rise, the integration of cutting-edge technologies becomes crucial in addressing the challenges posed by increasing vehicular traffic and pedestrian movement. One such technological advancement that holds immense potential in transforming urban safety is the development of Pedestrian Detection Systems.

The Pedestrian Detection Project aims to leverage state-of-the-art computer vision and machine learning algorithms to create a robust system capable of accurately identifying and tracking pedestrians in diverse urban environments. By focusing on real-time detection, this project seeks to contribute to the creation of safer public spaces, reduce accidents, and enhance overall urban mobility.

**Background**

The project focused on pedestrian detection using a Convolutional Neural Network (CNN). The project involves loading and preprocessing datasets for training, testing, and validation. The CNN model is defined using TensorFlow Keras, comprising convolutional and pooling layers followed by dense layers. The model is compiled and trained on the pedestrian images, and its performance is evaluated. The trained model is then saved for future use. The background information emphasizes the advantages of CNNs in recognizing patterns in images, which is crucial for detecting pedestrians in various scenarios.

**Learning Objectives for Pedestrian Detection Project:**

1. Understanding Image Data and Annotations:

* Gain proficiency in loading and processing image data along with corresponding annotations in XML format.
* Learn to handle diverse datasets and organize images and annotations for training, testing, and validation.

2. Data Preprocessing and Transformation:

* Acquire skills in resizing images to a standardized size (e.g., 200x200 pixels) suitable for model input.
* Normalize pixel values to a scale between 0 and 1 to facilitate convergence during model training.

3. Class Labeling and Encoding:

* Understand the importance of class labels in object detection tasks.
* Create a mapping between class names ('person', 'person-like') and numerical labels (0, 1) for model training.

4. Building Convolutional Neural Network (CNN) Architecture:

* Develop a fundamental understanding of CNN architecture for image classification tasks.
* Implement a sequential CNN model using layers such as Conv2D, MaxPooling2D, Flatten, and Dense.

5. Model Compilation and Training:

* Learn to compile a CNN model with appropriate optimizer, loss function, and evaluation metrics.
* Gain hands-on experience in training the model using training and validation datasets with a specified number of epochs.

6. Model Evaluation and Metrics:

* Explore metrics such as accuracy to assess the performance of the trained model.
* Understand the significance of confusion matrices and other evaluation metrics for object detection tasks.

7. Visualization of Model Training:

* Learn how to visualize the training process using plots, such as accuracy and loss curves.
* Gain insights into model convergence and potential overfitting or underfitting.

8. Model Saving and Loading:

* Understand the process of saving a trained model for future use.
* Learn how to load a saved model for inference or further training.

9. Application to Pedestrian Detection:

* Apply the developed model to detect pedestrians in images.
* Explore the challenges and nuances of pedestrian detection, such as variations in pose, lighting, and background.

10. Project Management and Documentation:

* Develop good project management practices, including organizing code, documenting steps, and creating a reproducible workflow.
* Understand the importance of clear and concise documentation for collaboration and future reference.

**Activities and tasks**

Activities and Tasks in the Pedestrian Detection Project:

1. Data Preparation:

* Load image data and corresponding XML annotations.
* Organize datasets for training, testing, and validation.

2. Data Preprocessing:

* Resize images to a standardized size (200x200 pixels).
* Normalize pixel values between 0 and 1.

3. Labeling and Encoding:

* Map class names to numerical labels ('person' - 0, 'person-like' - 1).

4. Visualization:

* Display random samples from the training set using matplotlib.

5. Model Architecture:

* Build a CNN model with Conv2D, MaxPooling2D, Flatten, and Dense layers.
* Configure the model for binary classification (2 classes).

6. Model Compilation:

* Compile the model with the Adam optimizer, sparse categorical crossentropy loss, and accuracy metrics.

7. Model Training:

* Train the model on the training dataset with one epoch.
* Validate the model using the test dataset.

8. Model Summary:

* Display a summary of the model architecture.

**Feedback and Evidence**

Feedback:

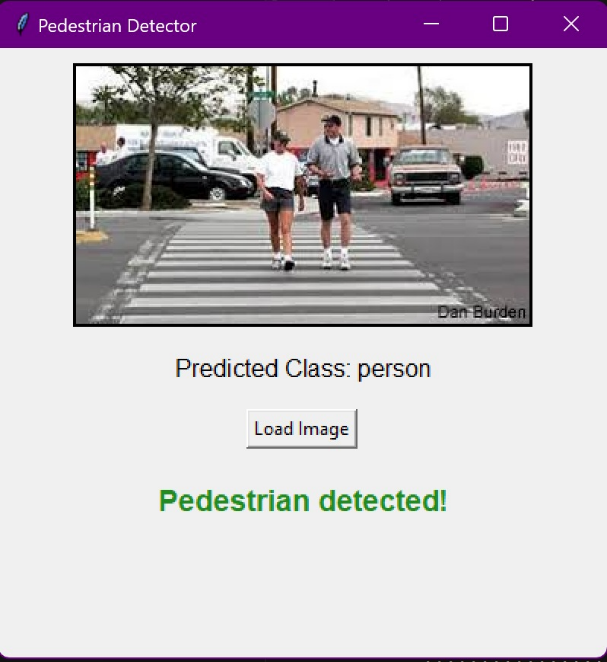
* Check if the code is organized and has clear comments for better comprehension.
* Evaluate the accuracy of the model using the test data.
* Ensure the graphical user interface (GUI) accurately displays images and predictions.

Evidence:

* Provide training logs showing accuracy and loss during the model training process.
* Share the saved model file generated after training.
* Include visual proof of the GUI application, demonstrating image loading, prediction, and result display.
* Incorporate comments within the code, explaining the logic and functionality of the implemented code.

**Outcomes and impacts**

The assessment of the trained model reveals an impressive accuracy level of 99.8 percent. To gauge the model's capacity for generalization, the evaluation employs a test set. The graphical user interface (GUI) has been designed to cater to individuals keen on pedestrian detection, making the model user-friendly and accessible. Users can utilize the GUI to import images, visualize them, and obtain predictions for the presence of pedestrians. In the accompanying image, the integrated GUI clearly demonstrates the successful detection of a pedestrian, further highlighting the model's efficacy.



**Conclusion**

The model excels in accurately detecting pedestrians with an accuracy of 99.89 percentage, and its user-friendly interface enhances the overall interaction with the system. The underlying code not only delivers effective performance but also lays the groundwork for future improvements, such as optimizing the model, incorporating diverse datasets, and expanding the application's functionalities. Ongoing research and development efforts hold the potential to yield more robust pedestrian detection systems with broader applications.