**IMPLEMENTATION**

**MODULES:**

* Data Collection
* Dataset
* Data Preparation
* Feature Extraction
* Splitting the dataset
* Model Selection
* **Training the Model**
* Analyze and Prediction
* Accuracy on test set
* Saving the Trained Model
* Prediction Module
* Model Evaluation Module

**MODULES DESCSRIPTION:**

**Data Collection:**

* In the first module of A System for Automated Vehicle Damage Localization and Severity Estimation Using Deep Learning, we make the data collection process. This is the first real step towards the real development of a deep learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get; the better our model will perform.
* There are several techniques to collect the data, like web scraping, manual interventions. The dataset is located in the model folder. The dataset is referred from the popular dataset repository called kaggle. The following is the link of the dataset:
* Kaggle Dataset Link:

https://www.kaggle.com/datasets/jayaprakashpondy/car-dataset

**Dataset:**

* Organizing the data into train, val sets and Converting annotations to the format required by YOLOv8.
* Annotations typically include bounding boxes around the objects of interest (Car Damage) and their corresponding classes. YOLOv8 requires annotations in a specific format, such as YOLO format (class, x\_center, y\_center, width, height) and Total dataset size is 778 images 7

0: damaged door

1: damaged window

2: damaged headlight

3: damaged mirror

4: dent

5: damaged hood

6: damaged bumper

7: damaged wind shield

**Data Preparation:**

* Ensure that each set contains a representative sample of images to avoid overfitting. Convert annotations into the format required by YOLOv8, ensuring consistency between image paths and annotation paths.
* Resizing: Resize the images to a consistent size to ensure the model can process them efficiently.
* Normalization: Normalize the pixel values of the images to be within a specific range, usually between 0 and 1, to improve model performance.

**Feature Extraction:**

* In YOLOv8, feature extraction is handled by the model architecture itself, specifically by its convolutional layers. These layers extract relevant features from the input images, enabling the model to detect objects such as damaged door, damaged window, damaged headlight, damaged mirror, dent, damaged hood, damaged bumper, damaged wind shield.
* Use the YOLOv8 architecture, which is a state-of-the-art object detection model. YOLOv8 extracts features from the images using a backbone network and then predicts bounding boxes and class probabilities.

**Splitting the dataset:**

* Divide your dataset into training and validation to evaluate your model's performance. Typically, you might use an 80-20 split, but this can vary based on your dataset size and specific requirements.

**Model Selection:**

* The training module is responsible for training the deep learning models using the preprocessed data. It implements YOLOv8 architectures

***YOLOv8:***

* Selecting YOLOv8 as the object detection model for tasks like damaged door, damaged window, damaged headlight, damaged mirror, dent, damaged hood, damaged bumper, damaged wind shield detection is a strategic choice owing to its high accuracy and efficiency. YOLOv8, short for "You Only Look Once version 8," represents the latest iteration of the YOLO (You Only Look Once) family of object detection models, known for their real-time performance and strong detection capabilities.
* Here's a deeper exploration of why YOLOv8 is a compelling choice:
* State-of-the-art Performance: YOLOv8 builds upon the success of its predecessors and incorporates advancements in deep learning techniques and model architectures. It achieves state-of-the-art performance in terms of detection accuracy and speed, making it suitable for various real-world applications.
* Efficiency: YOLOv8 is designed to be highly efficient, striking a balance between accuracy and computational resources. Its architecture optimizes the use of hardware resources, allowing for fast inference speeds even on devices with limited computational power. This efficiency makes YOLOv8 particularly appealing for deployment in resource-constrained environments or for applications requiring real-time processing.
* Single-stage Detection: YOLOv8 follows the principle of single-stage detection, meaning it processes the entire image in a single forward pass through the network. This design choice eliminates the need for complex post-processing steps and significantly reduces inference time compared to multi-stage detection approaches.
* Multi-scale Feature Fusion: YOLOv8 incorporates multi-scale feature fusion techniques, enabling it to capture context and spatial information at different scales within the image. This enhances the model's ability to detect objects of varying sizes and aspect ratios with high accuracy.
* Flexibility and Customization: YOLOv8 offers flexibility and customization options to adapt the model to specific use cases and datasets. Users can choose from different model variants (e.g., YOLOv8-s, YOLOv8-m, YOLOv8-l) based on their requirements for speed and accuracy. Additionally, the model can be fine-tuned on custom datasets to further improve performance on specific tasks.
* Open-source Implementation: YOLOv8 is often available as an open-source implementation, making it accessible to a wide range of developers and researchers. This fosters collaboration, experimentation, and innovation within the computer vision community.
* Overall, the selection of YOLOv8 as the object detection model for tasks like damaged door, damaged window, damaged headlight, damaged mirror, dent, damaged hood, damaged bumper, damaged wind shield detection reflects its reputation for achieving high accuracy and efficiency. By leveraging the strengths of YOLOv8, developers can build robust and effective detection systems capable of meeting the demands of various real-world applications.

**Training the Model:**

* Training loop: Train the YOLOv8 model using the training set. The model learns to predict bounding boxes and class probabilities for damaged door, damaged window, damaged headlight, damaged mirror, dent, damaged hood, damaged bumper, damaged wind shield.
* Loss function: Use a loss function such as mean average precision (MAP) to measure the model's performance during training.
* Optimizer: Use an optimizer such as stochastic gradient descent (SGD) or Adam to update the model's weights based on the loss.

**Analyze and Prediction:**

* Model evaluation: Evaluate the model's performance on the validation set during training to monitor its progress.
* Prediction: Use the trained model to predict bounding boxes and class probabilities for new, unseen images.

**Accuracy on test set:**

* Evaluate the final accuracy of the model on the test set to ensure its effectiveness in real-world scenarios. Calculate metrics like mAP, precision, and recall on the test set to quantify the model's performance objectively.

**Saving the Trained Model:**

* Save the trained YOLOv8 model for future use. YOLOv8 models can be saved in a format that allows easy reloading for inference and deployment in production environments.

**Prediction Module:**

Develop a prediction module that loads the saved YOLOv8 model and takes an input image or video stream. The module should be capable of processing the input data and outputting the detected

0: damaged door

1: damaged window

2: damaged headlight

3: damaged mirror

4: dent

5: damaged hood

6: damaged bumper

7: damaged wind shield

with bounding boxes, facilitating real-time detection.

**Model Evaluation Module**

* This module evaluates the performance of the trained models using the testing dataset. It calculates accuracy metrics and other performance indicators to assess model effectiveness.
* Evaluate model accuracy, precision, recall, and F1-score.
* Generate confusion matrices for both models.
* Accuracy, precision, recall, and F1-score are used to evaluate model performance.