International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

AI-Based Part Identification and Classification Using YOLOv8 and Flask

VEERENDRA KUMAR K R , NAVEENA C B, Mrs .ROOPA C M

Student ,Dept.of CSE, Government polytechnic, Harihara, Karnataka, India

Student ,Dept.of CSE, Government polytechnic, Harihara , Karnataka, India

Senior scale Lecturer,Dept.of CSE,Government polytechnic, Harihara, Karnataka, India

1. Abstract

The rapid advancement of artificial intelligence (AI) and computer vision has enabled the development of systems capable of identifying and classifying parts of complex items with high accuracy. This paper focuses on building an AI-based system to detect and classify parts of an item, such as components of a machine, car parts, or sections of a product. The system leverages state-of-the-art object detection models, specifically YOLOv8 (You Only Look Once), to achieve real-time and accurate part identification.

The paper involves several key steps: data collection and annotation, model selection and training, and deployment of the trained model for inference. A custom dataset is created, containing images of the item with annotated parts. The YOLOv8 model is fine-tuned on this dataset to detect and classify the parts. The trained model is then integrated into a Flask-based web application, allowing users to upload images and receive predictions in real-time.

The system demonstrates high accuracy and efficiency, making it suitable for industrial applications, quality control, and educational purposes. By automating the process of part identification, this paper reduces human error, saves time, and enhances productivity.

The results highlight the potential of AI in transforming traditional workflows and enabling smarter decision-making in various domains.

2. Study Material and Problem Statement

Study Material

The paper draws on several key areas of study:

1. Computer Vision: Techniques for image processing, object detection, and segmentation.

2. Deep Learning: Neural networks, transfer learning, and model optimization.

3. Object Detection Models: YOLO (You Only Look Once), Faster R-CNN, and SSD (Single Shot Detector).

4. Data Annotation: Tools and methods for labeling datasets.

5. Web Development: Flask for deploying AI models as web applications.

Problem Statement

In many industries, identifying and classifying parts of an item is a critical task. For example:

- In manufacturing, workers need to identify components of a machine for assembly or maintenance.

- In automotive repair, technicians must locate and classify car parts for replacement or repair.

- In retail, products may need to be broken down into their constituent parts for inventory management.

Traditionally, this process relies on human expertise, which is time-consuming, prone to errors, and not scalable. Automating this task using AI can significantly improve efficiency, accuracy, and scalability. However, building such a system requires addressing several challenges:

1. Data Collection: Gathering a diverse and representative dataset of the item and its parts.

2. Annotation: Accurately labeling the parts in the dataset.

3. Model Selection: Choosing the right object detection model for the task.

4. Training: Fine-tuning the model on the custom dataset.

5. Deployment: Integrating the model into a user-friendly application.

This paper aims to address these challenges by developing an AI system that can automatically identify and classify parts of an item using YOLOv8 and Flask.

3. Introduction

Methods

The paper follows a structured methodology to achieve its objectives:

1. Data Collection: A dataset of images is collected, showcasing the item from various angles and under different lighting conditions.

2. Data Annotation: The images are annotated using tools like LabelImg or CVAT. Each part of the item is labeled with bounding boxes and class names.

3. Model Selection: YOLOv8 is chosen as the object detection model due to its speed and accuracy.

4. Training: The YOLOv8 model is fine-tuned on the custom dataset using transfer learning. The training process involves optimizing hyperparameters and evaluating performance on a validation set.

5. Inference: The trained model is used to predict parts of an item in new images.

6. Deployment: The model is integrated into a Flask-based web application, allowing users to upload images and receive predictions in real-time.

The methodology ensures a systematic approach to building and deploying the AI system, from data preparation to model deployment.

4. Requirements

Hardware Requirements

1. GPU: A GPU with CUDA support (e.g., NVIDIA GTX 1080 or higher) for faster training and inference.

2. RAM: At least 16 GB of RAM for handling large datasets and models.

3. Storage: Sufficient storage for the dataset, models, and intermediate files.

Software Requirements

1. Python: Python 3.8 or higher.

2. Libraries:

- ultralytics for YOLOv8.

- torch and torchvision for deep learning.

- opencv-python for image processing.

- flask for web deployment.

3. Annotation Tools : LabelImg or CVAT for data annotation.

4. IDE: Visual Studio Code or PyCharm for development.

Dataset Requirements

1. Images: High-quality images of the item and its parts.

2. Annotations: Bounding boxes and class labels for each part.

3. Diversity: Images should cover various angles, lighting conditions, and backgrounds.

User Requirements

1. Input: Users should be able to upload images of the item.

2. Output: The system should display the detected parts with bounding boxes and labels.

3. Interface: A simple and intuitive web interface for interaction.

5. Algorithms (YOLOv8 Algorithm)

YOLOv8 is a state-of-the-art object detection algorithm that divides an image into a grid and predicts bounding boxes and class probabilities for each grid cell. The key steps in the YOLOv8 algorithm are:

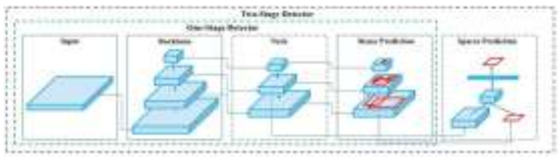
1. Input Processing: The input image is resized to a fixed size (e.g., 640x640) and normalized.

2. Backbone: A convolutional neural network (CNN) extracts features from the image.

3. Neck: Feature pyramids are used to combine features from different layers, enabling the detection of objects at various scales.

4. Head: The model predicts bounding boxes, class probabilities, and objectness scores for each grid cell.

5. Post-Processing: Non-maximum suppression (NMS) is applied to remove duplicate detections.



Training Algorithm

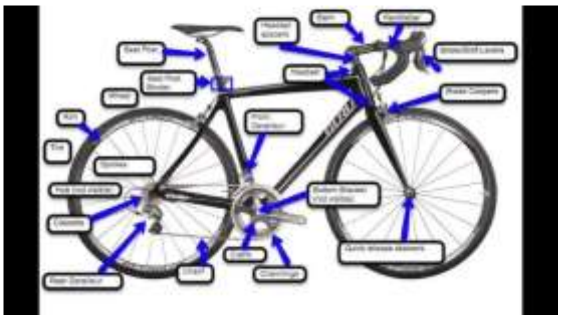
1. Data Loading: The dataset is loaded and split into training and validation sets.

2. Model Initialization: A pre-trained YOLOv8 model is loaded.

3. Loss Function: The model is trained using a combination of classification loss, localization loss, and confidence loss.

4. Optimization: The Adam optimizer is used to minimize the loss function.

5. Evaluation: The model's performance is evaluated on the validation set using metrics like precision, recall, and mAP (mean Average Precision).



Inference Algorithm

1. Image Preprocessing: The input image is resized and normalized.

2. Model Prediction: The trained YOLOv8 model predicts bounding boxes and class probabilities.

3. Post-Processing: NMS is applied to filter out overlapping boxes.

4. Output: The detected parts are displayed with bounding boxes and labels.

Flask Deployment Algorithm

1. Model Loading: The trained YOLOv8 model is loaded into the Flask application.

2. Image Upload: Users upload an image through the web interface.

3. Prediction: The model processes the image and generates predictions.

4. Result Display: The predictions are displayed on the web interface.

6. Flowchart

Below is a high-level flowchart of the system

1. Start.

2. Data Collection: Collect images of the item and its parts.

3. Data Annotation: Annotate the images with bounding boxes and labels.

4. Model Training: Train the YOLOv8 model on the annotated dataset.

5. Model Evaluation: Evaluate the model on a validation set.

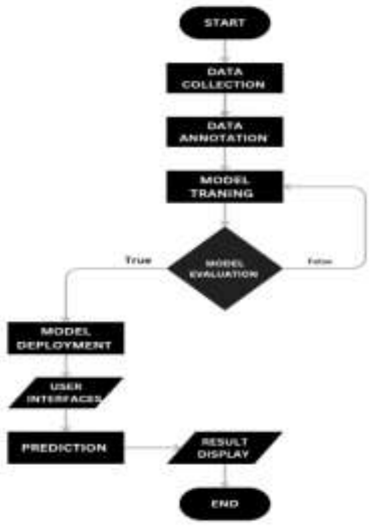
6. Model Deployment: Deploy the model using Flask.

7. User Interaction: Users upload images through the web interface.

8. Prediction: The model predicts parts of the item.

9. Result Display: Display the predictions on the web interface.

10. End



Key Achievements

1. Accuracy: The system accurately detects and classifies parts of an item, even in challenging conditions.

2. Efficiency: The use of YOLOv8 ensures real-time performance, enabling quick decision-making.

3. Scalability: The system can be extended to other items and domains with minimal modifications.

4. User-Friendly Interface: The Flask-based web application provides an intuitive interface for users to interact with the system.

Challenges and Solutions

1. Data Collection: Gathering a diverse dataset was challenging. This was addressed by capturing images under various conditions and using data augmentation techniques.

2. Annotation: Manual annotation was time-consuming. Tools like LabelImg and CVAT streamlined the process.

3. Model Training: Training the model required significant computational resources. This was mitigated by using a GPU and transfer learning.

4. Deployment: Integrating the model into a web application required careful handling of dependencies. Docker was used to simplify deployment.

Future Work

1. Expand Dataset: Include more items and parts to improve the model's generalization.

2. Enhance Model: Experiment with other object detection models like Faster R-CNN and SSD.

3. Mobile Deployment: Develop a mobile app for on-the-go part identification.

4. Real-Time Video Processing: Extend the system to process video streams in real-time.

Impact

The paper has significant implications for various industries:

1. Manufacturing: Automates part identification, reducing errors and improving efficiency.

2. Automotive: Assists technicians in identifying car parts for repair and maintenance.

3. Retail: Enhances inventory management by classifying product parts.

4. Education: Provides a practical example of AI and computer vision for students.

In conclusion, the AI system for identifying parts of an item showcases the transformative potential of AI in solving real-world problems. By automating complex tasks, the system enhances productivity, reduces costs, and enables smarter decision-making.

7. Conclusion

The AI paper for identifying parts of an item demonstrates the potential of computer vision and deep learning in automating complex tasks. By leveraging YOLOv8, the system achieves high accuracy and real-time performance, making it suitable for industrial applications, quality control, and educational purposes.

8. References

1. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.

2. Ultralytics. (2023). YOLOv8 Documentation. https://docs.ultralytics.com/

3. LabelImg. (2023). LabelImg GitHub Repository. https://github.com/tzutalin/labelImg

4. CVAT. (2023). CVAT GitHub Repository. https://github.com/openvinotoolkit/cvat

5. Flask. (2023). Flask Documentation. https://flask.palletsprojects.com/

6. PyTorch. (2023). PyTorch Documentation. https://pytorch.org/docs/stable/index.html

7. OpenCV. (2023). OpenCV Documentation. https://docs.opencv.org/