Developing an Item Recognition System using Deep Learning

Project-I (CS47005) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Aseem Anand

(19CS10013)

Under the supervision of

Professor Pabitra Mitra

Department of Computer Engineering

Indian Institute of Technology Kharagpur

Autumn Semester, 2024-25

November 07, 2024

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: November 07, 2024 (Aseem Anand)

Place: Kharagpur (19CS10013)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR-721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Developing an Item Recognition System using Deep Learning**" submitted by **Aseem Anand** (Roll No. 19CS10013) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2024-25.

Date: November 07, 2024
Place: Kharagpur

Professor Pabitra Mitra
Department of Computer Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Aseem Anand** Roll No: **19CS10013**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Computer Science and Engineering**

Thesis title: **Developing an Item Recognition System using Deep Learning**

Thesis supervisor: **Professor Pabitra Mitra**

Month and year of thesis submission: **November 07, 2024**

This report outlines the development of an item recognition system capable of identifying specific items within images containing multiple objects. The project involves training a deep learning model to recognize items while ignoring irrelevant features, such as stands or backgrounds. The report discussed the application's relevance, challenges encountered within the dataset, the coding and implementation of mode, and provides a placeholder for the results and outputs of the trained model

Contents

Declaration i

Certificate ii

Abstract iii

Contents iv

1 Introduction 2

2 Application of the Project 3

3 Challenges of Dataset 5

3.1 Coding and Implementation 7

4.1 Environment Setup 7
4.2 Data Preparation 7
4.3 Model Training 8
4.4 Feature Extraction 9
4.5 Main Program Implementation 11
4.6 Testing and Validation 12

4 Results and Output 14

5 Conclusion 15

6 References 16

Introduction

In the realm of computer vision, object detection and recognition play pivotal roles in numerous applications, ranging from retail inventory management to autonomous navigation. This project focuses on developing a system that can identify and localize specific items within images containing multiple objects. The primary challenge is ensuring that the model recognizes only the items of interest while ignoring irrelevant features, such as the stands on which items are placed.

The project utilizes a deep learning approach, combining object detection and feature matching techniques. By leveraging a large dataset with annotated images and employing state-of-the-art models like YOLOv5 and ResNet50, the system aims to achieve high accuracy in item recognition tasks.

Applications of the Project

The item recognition system developed in this project has several practical applications across various industries:

1. Retail and Inventory Management

- **Automated Checkout Systems:** Enhancing self-service checkout kiosks by accurately identifying products placed by customers, reducing the need for manual barcode scanning.
- **Inventory Tracking:** Monitoring stock levels on shelves in real-time by recognizing and counting items, leading to efficient inventory management and restocking processes.
- **Loss Prevention:** Detecting misplaced or stolen items by comparing shelf contents with expected inventory.

2. E-Commerce

- **Visual Search:** Allowing customers to search for products by uploading images, improving user experience and engagement.

- **Product Recommendation:** Identifying items in user-uploaded photos to suggest similar or complementary products.

3. Robotics and Automation

- **Robotic Grasping:** Enabling robots to identify and pick up specific items in cluttered environments, essential for automation in warehouses and manufacturing.
- **Autonomous Navigation:** Assisting robots in navigating spaces by recognizing objects and obstacles.

4. Quality Control in Manufacturing

- **Defect Detection:** Recognizing items and inspecting them for defects or irregularities during production lines.
- **Sorting Systems:** Automating the sorting of items based on recognition, increasing efficiency and reducing manual labour.

5. Healthcare

- **Medical Inventory:** Tracking medical supplies and equipment in hospitals to ensure availability and proper management.
- **Assistive Technologies:** Aiding visually impaired individuals by recognizing items and providing auditory descriptions.

Challenges of the Dataset

The dataset used in this project presents several challenges that needed to be addressed:

1. **Diversity of Items**

- Large Number of Classes: With over a thousand items, the dataset has a high number of classes, increasing the complexity of the classification task.
- Similar Appearance: Many items may look similar, making it difficult for the model to distinguish between them solely based on visual features.

2. **Presence of Stands**

- Dominant Background Features: The stands on which items are placed are often larger than the items themselves, which can mislead the model to focus on the stand rather than the item.
- Consistent Stand Appearance: Uniformity in the stands across images can cause the model to associate features of the stand with certain items, leading to incorrect classifications.

3. **Variations in Image Quality**

- Lighting Conditions: Differences in lighting can affect the visibility of item features, impacting model performance.
- Image Resolution: Variations in image resolution and quality can pose challenges in feature extraction.

4. **Annotation Inconsistencies**

- Bounding Box Accuracy: Inaccurate or inconsistent bounding boxes in annotations can lead to incorrect training signals.
- Missing Annotations: Some items may lack proper annotations, reducing the amount of usable training data.

5. **Imbalanced Data**

- Uneven Class Distribution: Some items have significantly more images than others, causing the model to be biased toward classes with more training examples.

Coding and Implementation

The implementation of the item recognition system involves several key steps, each encapsulated in separate Python scripts for modularity and clarity.

1. Environment Setup

Purpose: Establish a Python environment with all necessary dependencies to ensure consistent and reproducible results.

Steps:

Python Version: Ensure Python 3.7 or higher is installed.

IDE: Use PyCharm for development due to its robust features and virtual environment support.

Virtual Environment: Create a new virtual environment within PyCharm to isolate project dependencies.

Dependencies: Install required packages using pip:

pip install torch torchvision numpy opencv-python pillow ultralytics

2. Data Preparation

Script: data_preparation.py

Purpose:

Parse JSON annotations to extract bounding boxes and class labels.

Preprocess images to focus on items and exclude stands.

Organize data into training and validation sets.

Key Functions:

- Parsing Annotations: Load the JSON file and extract necessary information, such as image IDs, file names, bounding boxes, and category IDs.
- Processing Images: Read each image, apply any necessary preprocessing (e.g., cropping to exclude stands), and save the processed images.
- Converting Annotations: Transform annotations into the YOLO format, which requires normalized coordinates for bounding boxes.
- Dataset Splitting: Organize images and labels into train and val directories for training and validation purposes.

Challenges Addressed:

- Excluding Stands: By carefully adjusting bounding boxes or applying image segmentation, the script ensures that the model learns features from the items rather than the stands.
- Handling Large Data: Efficiently processing over 50,000 images by optimizing the script for speed and memory usage.

3. Model Training

Script: train_model.py

Purpose:

Train an object detection model using YOLOv5 on the prepared dataset.

Validate the model's performance using a validation set.

Key Components:

- Model Selection: Utilize YOLOv5 for its balance of speed and accuracy in object detection tasks.
- Training Parameters:
- Epochs: Number of passes through the entire training dataset.
- Batch Size: Number of samples processed before the model's internal parameters are updated.
- Image Size: Dimensions to which input images are resized.
- Data Configuration: Specify paths to training and validation data, number of classes, and class names in a *data.yaml* file.
- Training Execution: Use the *model.train()* method to start the training process, which includes automatic validation and checkpointing.

Challenges Addressed:

- Overfitting: Mitigated by using a separate validation set and monitoring performance metrics to prevent the model from memorizing training data.
- Class Imbalance: Addressed through techniques like weighted loss functions or data augmentation to provide more samples of underrepresented classes.

4. Feature Extraction

Script: feature_extraction.py

Purpose:

Extract feature vectors from images using a pre-trained Convolutional Neural Network (CNN), specifically ResNet50.

Store extracted features for use in the item matching process.

Key Functions:

- Model Loading: Load ResNet50 without its final classification layer to use it as a feature extractor.
- Image Preprocessing: Apply necessary transformations such as resizing and normalization to match the input requirements of ResNet50.
- Feature Extraction: Pass images through the model to obtain high-level feature representations.
- Feature Storage: Save the extracted features to disk in a structured format (e.g., NumPy arrays).

Challenges Addressed:

- Computational Efficiency: Optimize the extraction process to handle large

numbers of images without excessive computation time.

- Consistency: Ensure that features are extracted consistently across all images by using the same preprocessing steps.

5. Main Program Implementation

Script: main.py

Purpose:

Integrate object detection and feature matching to identify and localize the query item within a target image containing multiple objects.

Workflow:

1. Load Models:
    - Object Detection Model: Load the trained YOLOv5 model for detecting items in the target image.
    - Feature Extraction Model: Load the pre-trained ResNet50 model for extracting features.
2. Process Input Images:
    - Target Image: Read the image containing multiple objects (Image 1).
    - Query Item: Read the image of the specific item to search for (Image 2).
3. Extract Query Features:
    - Use the feature extractor to obtain a feature vector for the query item.
4. Detect Objects in Target Image:

- Apply the object detection model to locate items in the target image.
- Obtain bounding boxes and class predictions for detected items.

5. Extract Features from Detected Objects:
   - For each detected item, extract its feature vector.

6. Feature Matching:
   - Compute the similarity between the query item's features and each detected item's features using cosine similarity.
   - Determine matches based on a predefined similarity threshold.

7. Visualization:
   - Draw bounding boxes around matched items in the target image.
   - Display similarity scores for matched items.

Challenges Addressed:

- Threshold Selection: Determining an appropriate similarity threshold to balance between false positives and false negatives.
- Efficiency: Optimizing the detection and matching processes for real-time or near-real-time performance.

6. Validation

Purpose:

- Evaluate the model's performance using validation and datasets.
- Adjust model parameters based on validation results to improve accuracy.

Approach:

Validation Set: Utilize the val2019 dataset, which contains images with multiple items, to assess the model during training.

Metrics:

- Precision and Recall: Measure the model's ability to correctly identify items without producing excessive false positives.
- Mean Average Precision (mAP): Evaluate the overall detection performance across all classes.
- Confusion Matrix: Analyze misclassifications to identify patterns and areas for improvement.

Challenges Addressed:

- Real-World Scenarios: Ensuring the model performs well on images that reflect actual use cases, not just on the training data.
- Generalization: Verifying that the model can accurately detect and recognize items it has not seen before or that appear in different contexts.

Results and Output

Validation output

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95
all	6000	73602	0.651	0.728	0.746	0.628
1_puffed_food	1991	4763	0.634	0.707	0.738	0.605
2_puffed_food	1610	3408	0.645	0.712	0.754	0.626
3_puffed_food	1540	3168	0.629	0.686	0.701	0.585
4_puffed_food	1597	3667	0.614	0.698	0.729	0.594
5_puffed_food	1978	4538	0.646	0.704	0.743	0.611
6_puffed_food	2630	6642	0.672	0.743	0.762	0.647
7_puffed_food	2493	6041	0.668	0.713	0.738	0.613
8_puffed_food	1864	3965	0.647	0.67	0.711	0.606
9_puffed_food	1737	3998	0.656	0.704	0.727	0.611
10_puffed_food	1868	4408	0.67	0.759	0.776	0.649
11_puffed_food	1945	4494	0.68	0.771	0.788	0.665
12_puffed_food	1390	3087	0.639	0.728	0.742	0.626
13_dried_fruit	1709	3776	0.654	0.712	0.748	0.622
14_dried_fruit	2107	4910	0.611	0.678	0.704	0.588
15_dried_fruit	1640	3728	0.662	0.746	0.766	0.646
16_dried_fruit	2756	6952	0.686	0.764	0.785	0.663
17_dried_fruit	1034	2057	0.605	0.676	0.693	0.574

Speed: 1.8ms preprocess, 85.2ms inference, 0.0ms loss, 1.7ms postprocess per image
Results saved to runs\detect\val3
Validation Metrics:
Precision (mean): 0.651
Recall (mean): 0.728
mAP@0.5 (mean): 0.746
mAP@0.5:0.95 (mean): 0.628

Key Performance Highlights:

1. Mean Average Precision (mAP):
   - mAP@0.5: 0.746 – Strong performance at detecting objects with looser localization thresholds.
   - mAP@0.5:0.95: 0.628 – Decent performance across varying IoU thresholds but indicates room for improvement in precise localization.

2. Precision and Recall:
   - Mean precision: 0.651 – Reflects the model's ability to minimize false positives.
   - Mean recall: 0.728 – Indicates a good capacity to detect true positives, though some objects are still missed.

3. Class Balance:
   - Performance metrics suggest consistent results across classes, with no significant bias towards or against particular object types.

Strengths

1. Balanced Performance:
   - The model achieves a good balance between precision and recall, making it suitable for applications where both false positives and false negatives must be minimized.

2. Generalization:
   - The consistent performance across multiple classes shows that the model generalizes well, even in diverse datasets with varying object characteristics.

3. High mAP@0.5:
   - The high score at IoU 0.5 demonstrates robust object detection capabilities, particularly for tasks where loose localization is acceptable.

   - Real-World Suitability:
   - The metrics suggest the model is deployable in real-world scenarios like retail, inventory management, or surveillance.

Weaknesses:

1. Localization Precision:
   - The gap between mAP@0.5 (0.746) and mAP@0.5:0.95 (0.628) suggests the model struggles with tight bounding box placement, particularly for smaller or occluded objects.

2. Missed Objects:
   - A recall of 0.728 indicates that while the model detects most objects, it may still miss some, especially in challenging scenarios like cluttered environments or poor lighting.

3. Limited Performance in Stringent Applications:
   - For use cases that demand perfect precision (e.g., medical diagnostics or safety-critical systems), further refinement is needed.

Practical Implications:

1. Deployment Scenarios:
   - Suitable for tasks requiring reliable object detection, such as retail surveillance, inventory management, or robotics.
   - The model is less ideal for applications needing highly precise bounding boxes or near-perfect detection rates.

2. Areas for Improvement
   - Enhance bounding box accuracy for stricter IoU thresholds.
   - Address challenging scenarios like occlusion, poor lighting, or small object detection.
   - Augment the dataset and explore advanced training techniques to reduce missed detections.

The model demonstrates strong and reliable performance with balanced precision and recall, making it suitable for many practical applications.
While good enough for general deployment, there is room for improvement in precise localization and handling edge cases. Future iterations should focus on tighter bounding box placement and addressing hard-to-detect scenarios.

Conclusion

The development of an item recognition system presents both opportunities and challenges. By leveraging deep learning techniques and carefully addressing dataset complexities, the project aims to create a robust model capable of identifying specific items within cluttered environments. The modular approach to coding facilitates understanding and future enhancements.

The model demonstrates strong and reliable performance with balanced precision and recall, making it suitable for many practical applications.

While good enough for general deployment, there is room for improvement in precise localization and handling edge cases. Future iterations should focus on tighter bounding box placement and addressing hard-to-detect scenarios.

The successful implementation of this system has the potential to significantly impact various industries by automating processes, improving efficiency, and enabling new technologies. Future work may involve refining the model's accuracy, expanding its applicability to more diverse datasets, and optimizing performance for deployment in real-world applications.

References

1. YOLOv5 Documentation: Ultralytics. YOLOv5

2. PyTorch Documentation: PyTorch

3. ResNet Paper: He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778. Deep Residual Learning for Image Recognition

4. COCO Dataset: Lin, T.-Y., et al. (2014). Microsoft COCO: Common Objects in Context. arXiv preprint arXiv:1405.0312. COCO Dataset

5. OpenCV Documentation: OpenCV

6. Kaggle Retail Product Checkout Dataset: Kaggle Dataset