Amazon ML Challenge Code Explanation

Introduction

The key objective is to build a machine learning model capable of extracting entity values, such as weight, volume, or dimensions, directly from product images. This is essential in fields like e-commerce and healthcare, where such values are often missing in textual descriptions but critical for understanding product details. Our goal is to predict values for these entities in the required format from a test dataset of product images.

Our approach leverages manual annotation, a YOLO object detection model, and post-processing steps to generate accurate predictions.

Tools & Technologies Used

- **Programming Language**: Python
- **Libraries**:
    - **Pandas**: For data manipulation.
    - **Ultralytics:** For training & using YOLO model.
    - **OpenCV / PIL**: For image processing and manipulation.
    - **OCR (easyocr)**: For extracting text directly from images.
    - **Regular Expressions (Regex)**: For parsing the predicted text to match the required format.

Our Approach & Code Explanation

Step 1: Manual Image Annotation

We manually annotated 2000 images from the training set, identifying the bounding boxes around the required entities (e.g., weight, height and other entities) using `labelme` annotation tool. Each bounding box was associated with the correct entity class.

Step 2: Training YOLO for Object Detection

We trained a **YOLOv10x** model on the annotated dataset. The model was designed to detect bounding boxes around the entities of interest, such as "item_weight" or "height." etc. YOLO is well-suited for detecting entities due to its real-time object detection capabilities.

Step 3: Bounding Box Detection and Cropping

For each test image, we used the trained YOLO model to generate bounding boxes around potential entity values. We focused only on the bounding box related to the requested `entity_name`. The detected bounding boxes were sorted based on confidence scores from the YOLO model, and the one with the highest confidence was selected.

Step 4: Cropping the Entity

The bounding box for all the possibilities were then cropped out of the image. This isolated the part of the image that contained the entity value, reducing noise from the rest of the image. Why all the possibilities even after sorting? to decrease the possibility of discarding a good prediction just because it's confidence is not good. All the possibilities are passed through OCR and then the one giving the best result and having better confidence is chosen.

Step 5: Unit Conversion and Formatting

The cropped entity value was passed through a function that parses the value and converts it into the required format. We ensured that the predictions adhered to the allowed units from `constants.py`, converting them as necessary.

Step 6: Generating the Final Prediction

The most confident entity value, after unit conversion, was saved as the final prediction for each test sample. The predictions were formatted and written to the output CSV file.

Challenges & Optimizations

Annotation

- Ensuring accurate bounding boxes detection around the required images
- Converting all extracted values to the acceptable format with appropriate units.

Optimizations

- Fine-tuning the YOLO model for specific entity detection by increasing the number of annotated images. The dataset is long-tailed (i.e. it contains more entries for particular classes than other), we tackled that by taking only equal entries of images to train the dataset

Model Evaluation

The performance of our model was evaluated using the F1 score. Our focus was on ensuring high precision by selecting the most confident predictions from the YOLO model and accurate unit conversions to reduce false positives.

Conclusion

This approach combines manual annotation, YOLO object detection, and a robust post-processing pipeline to extract and format entity values from product images. The model prioritizes confidence in predictions and adherence to acceptable unit formats, providing accurate results for the challenge.