MAGPANTAY, NINO JANDEL C.

BSCS-4B-IS

CCST 106 PERCEPTION AND COMPUTER VISION

MR BERNARDINO

Mid-term Project: Implementing Object Detection on a Dataset

Project Documentation: Object Detection Using YOLOv5 and HOG-SVM

Dataset Description

For this project, we used a dataset containing images of food items, specifically for an object detection task. The dataset comprises various food categories, such as fruits, vegetables, snacks, and packaged goods, providing a broad range of images to train the model. This dataset was chosen because of its diverse and realistic food-related objects, which is particularly useful in the domain of object detection, where recognizing common objects in daily life is critical.

The images in this dataset are relatively varied in terms of lighting, background, and object positioning, making it suitable for a robust training process. Each image contains multiple objects, and the bounding boxes for these objects were manually annotated for training the model. The dataset allows us to work with real-world challenges such as cluttered backgrounds, occlusions, and small objects.

Preprocessing Steps

Image Resizing and Normalization

The dataset images were initially resized to a fixed size of 640x640 pixels to ensure uniformity for model training. Image resizing is a critical preprocessing step, as object detection models generally require input images of fixed dimensions. Resizing the images also ensures that they fit into the memory of the model without issues.

In terms of normalization, although YOLOv5 handles its normalization internally during training, for other models such as HOG-SVM, we converted the images to grayscale. This reduces the complexity of the feature extraction process. No explicit normalization (such as scaling pixel values) was required because the chosen models already handle this within their pipelines.

Labeling and Bounding Boxes

For the object detection task, the images were labeled with bounding boxes that denote the position of the food objects. The bounding boxes were manually annotated using tools like LabelImg, and each object within the image was assigned a class label corresponding to its category (e.g., apple, banana, sandwich). These annotations were saved in a format compatible with YOLOv5 (such as .txt files), where each line in the file corresponds to a bounding box with the class ID, along with the normalized coordinates of the box.

Selecting the Algorithm

We chose two different algorithms for object detection: YOLOv5 and HOG-SVM. YOLOv5 was selected due to its state-of-the-art performance in real-time object detection tasks. YOLO (You Only Look Once) models are known for their speed and efficiency, which is vital when working with larger datasets or real-time applications. The model can predict multiple objects within an image simultaneously, providing a powerful tool for real-time food object detection.

On the other hand, HOG (Histogram of Oriented Gradients) - SVM (Support Vector Machine) is a more classical method of object detection. HOG is used to extract feature descriptors from images, while SVM is a classifier that is trained on these features. We chose HOG-SVM as a comparison to YOLOv5 because it provides insight into how classical computer vision methods perform against modern deep learning approaches.

Building the Model

YOLOv5 Model

We leveraged YOLOv5, a pre-trained model from the Ultralytics library, for the deep learning approach. The model was fine-tuned on our custom dataset. Here's how the model was built:

Model Loading: The YOLOv5 model was loaded using the following code:

python

```
model = torch.hub.load('ultralytics/yolov5', 'custom', path='/content/foods/weights/best.pt')
```

This loads a pre-trained YOLOv5 model, and the custom weights (best.pt) were used to fine-tune the model on our dataset.

Training: During training, we passed the dataset, including image files and annotations, to the model. The model was trained for 10 epochs using the following command:

bash

!python train.py --img 640 --batch 16 --epochs 10 --data /content/food/data.yaml --weights yolov5s.pt

This step involved selecting key parameters such as the number of epochs (10), image size (640x640), and batch size (16).

Evaluation and Testing: After training, we used the model to detect objects in test images. The inference results included bounding boxes around the predicted food items.

HOG-SVM Model

For the HOG-SVM approach, we used the skimage library to extract HOG features from images. The SVM classifier was then trained on these features. The following steps were followed:

Feature Extraction: The HOG features were extracted using the hog function from the skimage library:

python

fd, hog_image = hog(image, pixels_per_cell=(16, 16), cells_per_block=(2, 2), visualize=True)

Training the SVM: We used an SVM classifier from the sklearn library to train the model on the extracted HOG features:

python

clf = svm.SVC(kernel='linear', C=1.0, random_state=42)

clf.fit(features, labels)

Explanation of Key Aspects

Feature Extraction for HOG-SVM

HOG features are used to capture the gradient information of an image. The key aspects of HOG include:

Pixels per Cell: This parameter controls how the image is divided into smaller cells for calculating gradients.

Cells per Block: This parameter is used to normalize the gradients within each block for robust feature extraction.

Visualization: After extracting HOG features, we visualized them to better understand how the model captures object shape and texture.

YOLOv5 Architecture

YOLOv5 uses a deep convolutional neural network (CNN) architecture. It divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell. The architecture consists of several layers, including:

Backbone: EfficientNet or CSPDarkNet as the backbone for feature extraction.

Neck: A PANet or FPN-like structure to aggregate features from different layers.

Head: The output layers predict the class, bounding box coordinates, and confidence score.

Dataset Splitting

The dataset was split into training and testing sets. Typically, 80% of the data was used for training the model, and 20% was reserved for testing. This split ensures that the model has enough data to learn from while leaving out a portion for unbiased evaluation.

Training Process

The training process involved configuring key parameters, such as:

Learning rate: The rate at which the model adjusts its weights. A smaller value typically leads to slower learning but more stable convergence.

Batch size: The number of images processed in each training step. A batch size of 16 was used.

Epochs: The number of times the entire dataset was passed through the model. We trained the model for 10 epochs, but this number can vary depending on performance.

During training, the loss function minimized the error between the predicted and actual bounding boxes and class labels. The model outputted training metrics such as loss and accuracy at the end of each epoch.

Training Output

The training output included metrics such as accuracy, precision, and recall. These metrics were used to monitor the model's performance and adjust hyperparameters accordingly.

Testing and Evaluation Metrics

After testing the model, we used metrics like precision, recall, and accuracy to evaluate its performance. These metrics provide insights into how well the model detects objects:

Precision: Measures how many of the predicted objects were correct.

Recall: Measures how many of the actual objects were detected.

Accuracy: Measures the overall performance of the model.

Visual Results

The model was able to successfully detect objects in test images, drawing bounding boxes around the food items. The visualization process displayed these images with their corresponding bounding boxes, class labels, and confidence scores.

Comparison of YOLOv5 and HOG-SVM

We compared YOLOv5 with HOG-SVM in terms of both performance and speed. YOLOv5 outperformed HOG-SVM in terms of detection accuracy and speed. YOLOv5's ability to detect multiple objects in real-time was a significant advantage over HOG-SVM, which could only detect one object at a time and was slower in terms of inference time.

Challenges

Several challenges were encountered during the project:

Data Quality: The dataset contained noisy images with occlusions and small objects, which made the training process more difficult.

Model Tuning: Fine-tuning the YOLOv5 model on the custom dataset required several iterations to achieve satisfactory results.

These challenges were overcome by using data augmentation techniques and adjusting model hyperparameters to improve training stability.

Learning

Throughout this project, I gained valuable experience in training and evaluating object detection models. The most important lesson was understanding the trade-offs between classical computer vision techniques like HOG-SVM and deep learning models like YOLOv5. YOLOv5 provided superior accuracy and speed, but classical methods like HOG-SVM are still useful for simpler tasks and smaller datasets.

Summary

This project involved training and evaluating two object detection models—YOLOv5 and HOG-SVM—on a food dataset. The YOLOv5 model performed better in terms of detection accuracy and speed, demonstrating the power of modern deep learning approaches. We discussed the dataset preprocessing, model building, training, and evaluation steps, highlighting the differences between the two approaches and concluding that YOLOv5 is a superior solution for real-time object detection in complex environments.