CSE 5095 Final Project

William Brodhead, Brennan Giglio, Siddharth Sinha

Abstract

This project explores the application of advanced machine learning models for the automated classification of pomegranate growth stages. The research focuses on the performance of several state-of-the-art models including the OWL-ViT (Vision Transformer for Open-World Localization), YOLOv5 (You Only Look Once, version 5), and Faster RCNN—to identify and categorize pomegranate growth stages from buds to ripe fruits. These models are evaluated based on their accuracy, processing speed, and usability in practical agricultural settings. The primary objective is to determine the most effective model for real-time monitoring and management in precision agriculture. This comparison not only sheds light on the capabilities and limitations of each model in handling complex agricultural imaging data but also aims to enhance decision-making processes in crop management through improved technological integration. The findings are expected to contribute to the advancement of agricultural technologies by facilitating more accurate and efficient crop monitoring systems.

Introduction

In the realm of agricultural technology, the accurate classification of crop growth stages through automated systems holds significant promise for enhancing crop management and yield prediction. This research leverages the power of advanced machine learning models to classify pomegranate growth stages from visual data, spanning from buds to ripe fruits. The objective is to explore the efficacy and efficiency of three state-of-the-art pretrained models—OWL-ViT, YOLOv5, and Faster R-CNN—in recognizing and categorizing the developmental stages of pomegranates into five distinct classes: bud, flower, early-fruit, mid-growth, and ripe.

Each of these models brings a unique approach to the task: OWL-ViT utilizes a Vision Transformer architecture that processes the image in a series of patches and employs transformers to interpret the entire scene collectively. This approach is advantageous for handling complex scenes with overlapping objects, making it well-suited for dynamic environments like agricultural fields. YOLOv5, known for its speed and accuracy in real-time object detection, applies a more traditional convolutional approach, optimized for performance. Lastly, Faster R-CNN, designed for precise object detection, integrates deep convolutional networks with region proposal networks, allowing for effective detection and classification of different growth stages in complex images.

The dataset employed in this study is

an extensive collection of pomegranate images designed to facilitate the detection and classification of the fruit's various growth stages. Sourced from the publicly available repository hosted by ScienceDirect [1], this dataset comprises approximately 5000 images with resolutions of either 640x480 or 480x640. It is categorized into five distinct classes, representing different developmental phases of the pomegranate. This dataset is particularly valuable for training machine learning models due to its diversity in image backgrounds and stages, providing a robust basis for evaluating the effectiveness of object detection and classification algorithms.

We aim to accurately assess the growth stages of a fruit given an image using these models and provide insight into the practical applicability of each model in agricultural monitoring systems. By integrating these cutting-edge technologies, we seek to advance the frontier of precision agriculture, enabling more informed decisions that could lead to improved crop health management and increased agricultural productivity.

Background and Related Work

The three models we focused on include YOLOv5, Faster R-CNN, and OWL-ViT. We decided to use these three models based on the architecture of the dataset, preliminary research, and recommendations from various individuals.

The article from Analytics Vidhya titled 'Implementation of Faster R-CNN in Python for Object Detection' provides a comprehensive tutorial on setting up and training a Faster R-CNN model for object detection using Python. It outlines the necessary steps and supplies a Github repository with code required to implement the architecture, focusing on utilizing TensorFlow 1.0 as the underlying framework. While the tutorial offers valuable insights into the model construction and training process, the specific use of TensorFlow 1.0 posed compatibility issues with our current software environment, which supports more recent versions. Despite this, the explanations of the Faster R-CNN components and the structured coding examples served as a useful reference. They contributed significantly to our understanding and helped us in crafting our custom implementation of Faster R-CNN tailored to our specific requirements and software infrastructure[2].

"An Image is worth 16x16 words: transformers for image recognition at scale" explores the application of Vision Transformer (ViT) models to image recognition tasks. Originally developed for natural language processing, the transformer architecture is tested here for its efficacy in handling image data directly, without relying on convolutional neural networks (CNNs). The key idea is to treat image patches as tokens similar to words in text processing, applying a transformer directly to these patches. This approach challenges the dominance of CNNs in image processing by demonstrating that a pure transformer model can achieve excellent results when pre-trained on sufficiently large datasets[3].

"You Only Look Once: Unified, Real-Time Object Detection" by Joseph Redmon et al. introduces YOLO, a novel approach

to object detection that frames the task as a single regression problem to spatially separate bounding boxes and their associated class probabilities from full images in one evaluation. This method contrasts with prior works that repurpose classifiers for detection, basically offering a more direct and efficient way to handle object detection tasks[4].

METHODS

In the initial phase of our research, we explored the development of a custom convolutional neural network (CNN) model aimed at classifying various bounding boxes within the images. However, we encountered a significant challenge due to the inconsistent number of bounding boxes present across different images. A subsequent detailed analysis of our dataset revealed that it was ideally suited for YOLO-based architectures. An attempt to construct a bespoke YOLO model was made, but the inherent complexity of the model design proved too formidable, resulting in suboptimal outcomes. Consequently, we shifted our focus to the Ultralytics YOLOv5 model, with which we achieved a remarkable precision of 92% across all classes.

Building on the success of the YOLOv5 model, we expanded our exploration to other prominent models. Despite substantial efforts, our custom-built Faster R-CNN model achieved an accuracy of 57% over three epochs, indicating potential yet unoptimized performance. Additionally, we ventured into developing a DETR model; however, constraints related to time and computational resources hindered our ability to derive meaningful results. Attempts to integrate a pre-trained SSD model were also thwarted by compatibility issues associated with its legacy software framework, underscoring the challenges of adapting older technologies to new datasets.

PREPROCESSING THE DATA

The dataset was initially structured in the VOC2007 format, comprising folders for Annotations, ImageSets, JPEGImages, and labels, making it well-suited for training with YOLO models. Throughout the course of our research, we encountered the need to engage with multiple data formats to accommodate various machine learning models. While some models were compatible with the .xml files of the VOC format, others required data transformation. Notably, the DETR model necessitated the use of the COCO format, which organizes data into a JSON file detailing image names and associated bounding box coordinates. Recognizing the prevalence and utility of the COCO format across various models other than YOLO, we developed a script to efficiently convert our dataset into this more universally applied format. Again, we were unable to use it, though.

YOLOv5 Architecture

YOLOv5 (You Only Look Once, version 5) is one of the latest iterations in the YOLO series of models, which are known for their efficiency and effectiveness in real-time object detection. YOLOv5, like its predecessors, is designed to perform detections at a single network evaluation,

making it extremely fast and suitable for real-time applications.[5]

The architecture of YOLOv5 consists of a backbone, a neck, and a head. The backbone is based on the CSPNet (Cross Stage Partial Network), a variant of the standard convolutional network. It uses a modified version of the Darknet architecture for feature extraction. The neck of YOLOv5 uses PANet (Path Aggregation Network), which helps it learn spatial hierarchies between different scales of the feature maps. This is crucial for improving detection performance, especially for small objects like tiny little buds in the background of these images, by facilitating efficient information flow across different scales. The detection head of YOLOv5 outputs bounding boxes and class probabilities. It uses predefined bounding boxes called "anchor boxes" that aid the network in predicting the location and size of the objects. The network predicts a set number of bounding boxes per grid cell, and each box prediction includes x, y coordinates, width, height, a confidence score, and class probabilities. This is perfect for what we want to output.

Faster R-CNN Architecture

Faster R-CNN is an influential model in object detection that effectively merges Region Proposal Networks (RPN) with the Fast R-CNN into a cohesive architecture. This model utilizes deep convolutional networks as its backbone, in our case, the ResNet-50 with an FPN (Feature Pyramid Network), which provides a rich hierarchical representation of images at multiple scales, crucial for detecting objects at different sizes. [6]

In our adaptation of the Faster R-CNN, the model was tailored to classify different growth stages of pomegranates, involving a modification of the ROI head to accommodate five classes plus background. This was achieved by adjusting the classifier and the bounding box regressor in the model's region of interest heads to match our dataset specifications. The model, pre-trained on a comprehensive dataset, was fine-tuned using our dataset formatted in the COCO style, allowing for effective transfer learning.

The training regimen was executed over three epochs, employing a stochastic gradient descent (SGD) optimizer with a learning rate of 0.005, momentum of 0.9, and a weight decay of 0.0005. Each epoch involved processing the images and their associated annotations through the model by using a custom collate function, optimizing both the localization and classification components.

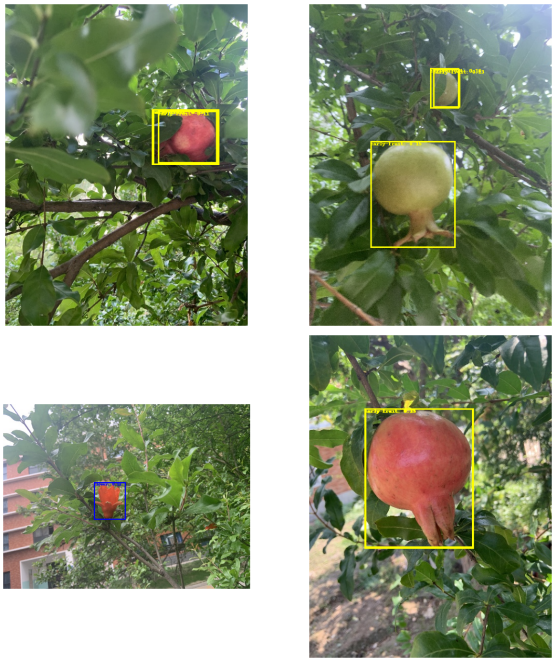
The training process emphasized not just loss reduction but also improvements in the Intersection over Union (IoU) and accuracy of label matching, which were crucial metrics for assessing performance. By the end of training, significant strides had been made in the model's ability to accurately detect and classify the target objects.

Given the complex nature of our model and the limited number of training epochs, there is substantial potential for improvement with enhanced computational resources. Access to high-performance GPUs would enable extended training periods, allowing the model to better generalize and potentially achieve higher accuracy. Such resources would also facilitate more exten-

sive experimentation with hyperparameters and model architectures, further optimizing performance for our specific application in agricultural imaging.

OWL-ViT Architecture

We tried a whole slew of models on this dataset, and though it initially seemed pretty easy to work with, this really wasn't the case. There was, in reality, a good bit of pre-processing involved, and many models just take completely different image formats, and output things that you don't necessarily want or need. The Owl ViT, for example, actually worked decently well.



If we had more time to train this one it also would've been much better. This architecture utilizes a Vision Transformer (ViT) as its backbone, consisting of multiple layers configured for processing image inputs. Initially, the model inputs pass through a patch embedding layer where images are split into fixed-size patches, linearly embedded, and positioned encoded. Subsequent to patch embedding, the architecture employs a series of transformer encoder layers. Each encoder layer comprises multi-head self-attention and feed-forward neural networks, both interspersed with layer normalization, and includes residual connections. For object detection, the final global token pooling layer typical in ViTs is omitted. Instead, each token output from the last transformer layer is fed into separate heads for classification and bounding box regression, enabling the detection and classification of multiple objects within an image. This architecture allows for end-to-end training and is tailored for open-vocabulary tasks by leveraging text embeddings for classifying a broad array of object categories, even those not present in the training dataset. [7]

For our dataset, we configured the OWL-ViT model to detect objects described by specific growth stage labels such as "bud", "flower", "early-fruit", "mid-growth", and "ripe". The model's ability to perform zero-shot learning is particularly advantageous in agricultural applications where training on every possible variation of an object (e.g., fruit at different growth levels) isn't feasible. Our preliminary results indicated that OWL-ViT could maintain robust detection performance despite the variations in fruit appearance and overlapping growth stages within the images.

The post-processing step using the OWL-ViTProcessor enhances the model by translating the model's outputs into action-

able insights, such as the precise locations and confidence levels of the detected growth stages. Once the model predicts the bounding boxes, their coordinates are used to draw rectangles on the original image. Here, we use the ImageDraw module from the PIL library, which allows for drawing over images. Each bounding box is color-coded according to the detected class, enhancing the visual feedback for assessment.

SSD Architecture

SSD (Single Shot MultiBox Detector) is a prominent architecture for object detection that achieves high performance through its ability to detect objects in a single forward pass of the network, making it efficient and fast, particularly suitable for real-time applications.

The architecture of SSD is distinctive because it combines predictions from multiple feature maps at different scales, which allows it to handle objects of various sizes effectively. The SSD model has two main components: the backbone and the head.

The backbone of the SSD model is typically a standard convolutional neural network (CNN) designed for high-quality image classification, such as VGG16 or ResNet, modified to be more suitable for detection tasks. This backbone acts as the feature extractor, where lower-level feature maps are used for detecting small objects and higher-level maps for larger objects.

The head of the SSD, which is integrated into the latter stages of the backbone without additional components like a separate neck, consists of several convolutional layers. These layers are responsible for producing a fixed number of detection predictions using a technique called MultiBox. Each of these predictions includes classifications (class labels) and adjustments to anchor boxes (location, size) for the detected objects. This setup enables SSD to perform detection over multiple categories efficiently.

RESULTS

YOLOv5

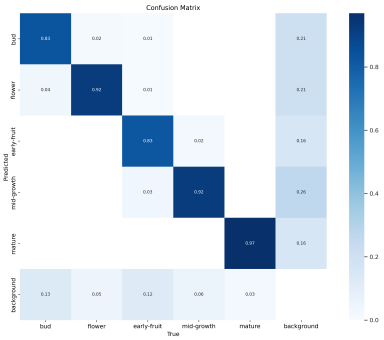


Fig. 1. Confusion Matrix

This figure shows the confusion matrix for the trained YoloV5 (Small) model. These values represent the proportion of correct predictions for each class. Higher values on the diagonal are indicative of better model performance for specific classes. The confusion matrix provided reveals various insights into the classification model's performance across different classes, particularly highlighting its strengths and weaknesses. The model demonstrates very good accuracy in identifying 'flower' (92%), 'mid-growth' (92%) stages and 'mature' (97%), suggesting effective learning for these more visually distinct categories. However, it struggles relatively more with the 'early-fruit' class (83%

accuracy) and 'bud' (83% accuracy). Most background misclassifications occur for the 'bud' category, which is most likely due to its small size . Overall, while the model is adept at distinguishing stages that are visually distinct, it faces some challenges with less distinctive or smaller stages, indicating a need for improved instance segmentation, data augmentation, and possibly enhanced training or re-annotation for under performing classes such as 'bud' and 'background' to improve overall model accuracy and effectiveness.

Fig. 2. Actual Labels

Fig. 3. Predicted Labels

This YOLO model could certainly use some more training, as it's clearly underperforming. As we can see, it struggles to classify the, relatively speaking, "smaller" pomegranates in the images. There are also a couple clear misclassifications, including the top right corner predicting what appears to be a leaf as a mid-growth pomegranate.

Faster R-CNN

The results from training and testing our Faster R-CNN model demonstrate consistent and comparable performance metrics, suggesting effective learning without significant overfitting. Throughout the training process, there was a notable and steady decrease in average loss from 0.4305 to 0.1452, with corresponding improvements in the average Intersection over Union (IoU) from 0.6822 to 0.8600, and accuracy from 0.2298 to 0.5689 over three epochs. Testing on unseen data yielded similar results with an average loss of 0.1497, an IoU of 0.8446, and an accuracy of 0.5727. The close alignment of training and testing metrics suggests that the model maintained its generalization capabilities across different datasets. However, the continued reduction in loss during the final epochs of training indicates that the model might not have fully converged, implying that additional training epochs could potentially yield further improvements.

OWL-ViT

The OWL-ViT model, notably, showed a unique capability to perform decently without extensive training, due to its zero-shot learning features. This model could

recognize different growth stages directly from the raw data, suggesting significant potential if further training and resource allocation were feasible. This characteristic makes OWL-ViT an intriguing candidate for scenarios where rapid deployment on diverse data without prior extensive training is desirable.

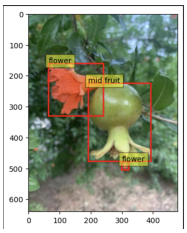


Fig. 4. Prediction from Faster R-CNN model

CONCLUSION

In this study, we investigated the applicability of various advanced machine learning models for the classification of pomegranate growth stages. While the YOLOv5 model demonstrated remarkable precision and is highlighted as the most effective in our trials, the exploration of other models like Faster R-CNN and OWL-ViT provided valuable insights into the challenges and potentials of utilizing sophisticated machine learning frameworks in agricultural technology.

In the future, we aim to extend the application of our machine learning models to real-world physical growing systems. This could revolutionize precision agriculture by providing timely insights into crop growth stages, enabling proactive management and potentially increasing yield efficiency. With access to more resources and advanced GPUs, we would also be able to fully train our models to convergence and accurately compare the models we outlined. By pushing the boundaries of our current methodologies and exploring these future avenues, we anticipate contributing significantly to the advancement of smart farming technologies with the information in this paper.

REFERENCES

[1] B. Pakruddin and R. Hemavathy, "A comprehensive standardized dataset of numerous pomegranate fruit diseases for deep learning," *Data in Brief*, vol. 54, p. 110284, 2024. [Online]. Available: https://doi.org/10.1016/j.dib.2024.110284

[2] P. Sharma, "A practical implementation of the faster r-cnn algorithm for object detection (part 2 – with python codes)," Analytics Vidhya, 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[4] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[5] Ultralytics, "YOLOv5: A state-of-the-art real-time object detection system," https://docs.ultralytics.com, 2021, accessed: Apr 2024.

[6] A. F. Gad, "Faster r-cnn explained for object detection tasks," Paperspace Blog, 2022, accessed: Apr 24. [Online]. Available: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/

[7] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, and N. Houlsby, "Simple open-vocabulary object detection with vision transformers," 2022.