Comparative Analysis of Object Detection Models: Identifying Drinks, Utensils, and Laptops Using Faster R-CNN and YOLO Architectures

Sriramteja Veerisetti

March 15, 2024

Abstract

This study provides a comparative analysis of two object detection frameworks: Detectron2, which uses a Faster R-CNN model architecture, and YOLOv5. Data augmentation and hyper-parameter tuning were utilized to assess the performance of these models for object detection of common items such as drinks, utensils, and laptops. The evaluation criteria included mean Average Precision (mAP) at IoU thresholds of 0.50 (mAP@50) and averaged over IoU thresholds from 0.50 to 0.95 (mAP@50-95), alongside a deep-dive into the differences of the models' inference speed and parameter size. Based on the results, it was concluded that YOLOv5 outperforms Detectron2 in performance on the test set, achieving mAP scores of 0.885 at mAP@50 and 0.656 at mAP@50-95, showing its better accuracy and generalizability across the three classes. Further, the YOLOv5 model demonstrated better efficiency with a shorter training duration and a larger model size, which did not hinder the performance, but rather allowed the model to explore more intricate and nuanced patterns within the images. It is proposed that further inclusion of a broader and more diverse dataset, coupled with more precise, accurate, and automated annotation techniques may enhance the robustness and accuracy of the model.

1 Introduction

The deep learning concept of object detection, a fundamental branch of modern computer vision, has been critical in the shift to technological dependence in many countries around the world. Object detection impacts a wide variety of fields such as medicine, vehicle manufacturing, robotics, natural disaster preparedness, surveillance and security, retail, sports analytics, and more. This unique technology continues to catalyze innovation across many sectors through its ability to transform visual data into actionable insights and product innovation.

For this project, the dataset was curated by gathering 100 everyday images of drinks, utensils, and laptops. The goal of the project was to garner a concrete understanding of how to take raw images and develop meaningful algorithms that can accurately and precisely detect these everyday images. Manual annotation of more than 300 images was conducted via the Computer Vision Annotation Tool (CVAT), which is a popular open-source tool used to annotate images and videos. To understand a more comprehensive understanding of how model selection can impact object detection, the Faster R-CNN Detectron2 model and YOLOv5 model were compared based on mAP@50 and mAP@50-95 metrics, training and inference speed, and model size. The rest of the paper will be formatted as follows:

- Fundamentals of Object Detection in Computer Vision
- Architecture of Faster R-CNN Model
- Architecture of You Only Look Once (YOLO) Model
- Modeling Process and Results of Detectron2
- Modeling Process and Result of YOLOv5
- Discussion of Detectron2 Vs. YOLOv5

- Conclusion
- Citations and Reference List

2 Fundamentals of Object Detection in Computer Vision

2.1 Basic elements of an Object Detection Model

There are a few foundational elements when building the framework of an object detection model. Images annotated with ground truth labels for each class, as well as bounding boxes that encompass objects within these images, are the cornerstone of the process. The responsibility of the model is to identify regional proposals via a selective process or anchor boxes. The key metric within this process is the IoU (Intersection over Union), which assesses the accuracy of the proposed bounding box by calculating its overlap with the ground truth label. An offset variable is calculated to incrementally update the proposals to match the ground truth labels more accurately. The model's ultimate goal is to predict the class of the object as well as the corresponding bounding box coordinates. The mAP (mean Average Precision), which aggregates precision and recall scores across different IoU thresholds and object classes, is the most commonly used metric to evaluate the performance of an object detection model.

2.2 Non-maximum Suppression (NMS) and Region Proposal Generation

Non-maximum Suppression (NMS) is a common algorithm that is utilized to improve both the accuracy and efficiency of the model's object detection capabilities. The primary goal of NMS is to select which bounding box, out of many overlapping boxes, best encapsulates the target object. NMS evaluates each bounding box based on the model's confidence score that an object is actually within the bounding box coordinates. The algorithm also uses a pre-defined threshold of IoU when identifying which bounding boxes should be kept based on the area of overlap. In essence, the bounding box with the highest probability is kept and all other neighboring bounding boxes that do not cross the pre-defined IoU threshold are suppressed, which helps reduce the redundancy of bounding boxes.

Region proposals are critical to identifying the locations of target objects within an image. The region proposal network (RPN), a key component of object detection frameworks such as Faster R-CNN, is tasked to determine bounding box proposals that encapsulate the object of focus within an image. In contrast, a selective-search algorithm could be used to generate proposals based on pixel intensities and segmentations. Today, most Faster R-CNN-based models tend to use the RPN since it offers a more streamlined approach to generating these proposals.

2.3 Intersection of Union (IoU) and mean Average Precision (mAP) Calculations

To measure the accuracy of the predictions made by the object detection model, the IoU is often utilized. In essence, the IoU is a tool that measures the ratio of the overlapping area between two bounding boxes and the total combined region of both bounding boxes. This ratio is bounded between 0 and 1. An IoU value closer to 1 indicates that the proposed bounding box has more alignment with the ground truth, which is preferred. In general, the formula of IoU can be written as:

$$Intersection of Union(IoU) = \frac{Area of Overlap}{Area of Union}$$

When evaluating the performance of a model, the mAP metric is used to determine how accurate the resulting prediction of the model is. Traditionally, precision is used to measure the true positives divided by the sum of true positives and false positives. The model aims to maximize the true positive rate, which looks at the total number of bounding boxes that predicted the correct class of the target object, while also having an IoU greater than a pre-defined threshold.

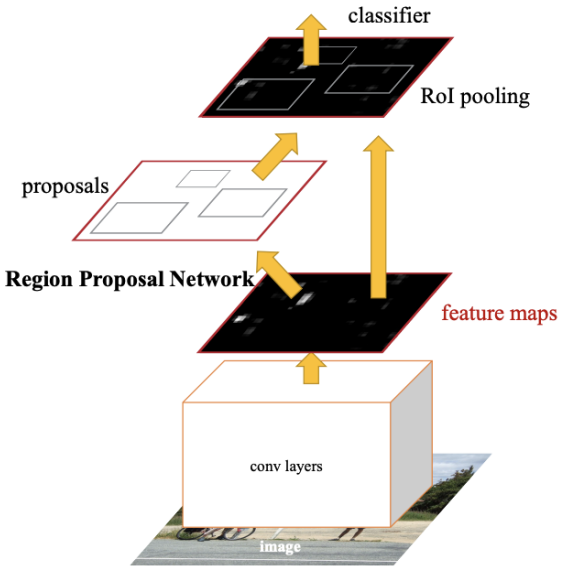

Figure 1: Basic Architecture of Faster R-CNN

3 Architecture of Faster R-CNN Model

3.1 Region Proposal Network

The RPN within the Faster R-CNN architecture identifies region proposals using a series of unique steps. First, the RPN trains a model to slide anchor boxes that vary in size and aspect ratio over the image to capture potential regions. Second, the IoU is calculated between the ground truth and the proposed bounding boxes and then the objectness score for each proposal is also calculated. Third, proposals that cross a pre-defined IoU threshold are kept, while proposals that do not are discarded. Fourth, after training the model, NMS is used to determine the proposal that yields the highest probability of encapsulating the target object to reduce redundancy.

3.2 Feature Extraction Using Feature Map and ROI Pooling

The Faster R-CNN model corrects a major problem in the R-CNN model by creating a single feature map, which allows the model to extract important features once per image. The R-CNN model feature extraction occurs separately for every single proposal, which significantly increases computation time.

The feature maps produced in the previous step are passed through the ROI pooling layer in sequential order. The resulting feature vector has a standardized shape, even though the proposals have variable shapes and sizes. The key point to articulate is that the ROI pooling layer is introduced to replace the warping concept that was introduced within the R-CNN layer.

3.3 Classification and Offset Calculation

The two main problems that arise from the previous steps are that the region proposals may not correspond tightly with the object and there is no class label generated for the proposal that encompasses the object. To tackle both issues, fully connected layers are introduced to predict the class of the object as well as corresponding offsets, which allow the proposal to better fit the target object. For this project, since there were 3 classes, the resulting neural network output would yield 8 outputs - 4 classes when including the background class, and the 4 offset measurements for the width, height, and center coordinates of the proposals.

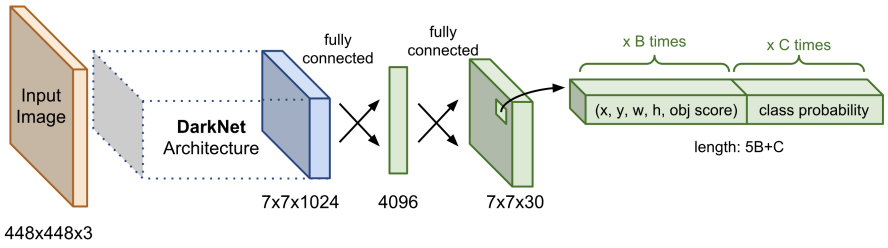

Figure 2: Architecture of YOLO Model

4 Architecture of You Only Look Once (YOLO) Model

4.1 Image Grid Division and Bounding Box Prediction

Contrary to a Faster R-CNN model, YOLO initially uses grid division by dividing the input image into an S x S grid. Each grid cell is responsible for detecting any object that has its center fall within it. Once objects have been detected, bounding boxes and corresponding confidence scores are generated per bounding box. The confidence score is key to understanding the probability of each class being in a cell.

4.2 Class Prediction

If a cell predicts that the center of the object is indeed within its cell, then it is responsible for predicting the class of the object by finding the conditional probability. The YOLO model will predict the probability the object belongs to a certain class out of the total set of classes available. In mathematical terms, the conditional probability can be calculated by:

$$ConditionalProbability = Probability(Class_x|Object)$$

During inference time, the class probabilities and confidence predictions are multiplied by each other to garner the final confidence score for the class for each bounding box. The final confidence score encompasses both the class label as well as the probability an object exists in the cell.

For this particular project, YOLOv5 was chosen. The YOLOv5 model has a CSPDarkent53 backbone and a PANet neck. In terms of loss functions, the YOLOv5 model utilizes both binary cross entropy and the logit loss function. Compared to previous versions of YOLO, this particular model is typically more accurate and efficient.

5 Modeling Process and Results of Detectron2

5.1 Detectron2 Model and Training Set-Up

For the first object detection model, a Faster R-CNN model using a Detectron2 framework was utilized. To increase the training size, generalizability, and robustness of the model, a custom data loading process that uses a mapper function was incorporated to introduce data augmentation for random adjustments such as flipping, rotations, brightness, contrast, saturation, cropping, lighting, and more. The fasterrcnnR50FPN3x.yaml file is integral for defining the architecture of the model as well as introducing the RPN to generate proposals. The configuration setup leverages pre-trained weights that were trained on the COCO dataset and fine-tuned to classify everyday objects such as drinks, utensils, and laptops.

The parameters set in the configuration were specifically tailored for this particular project. The number of working threads used was set to 4 to increase training speed, while also making sure not the overwhelm the GPU being used for training. Further, 2 images were used per batch during the training process since there was limited GPU memory available. A base learning rate of 0.00025 was used when back-propagating and updating the weights of the model. The learning rate was set appropriately so that the global minima of the loss function would not be overshot, but also making

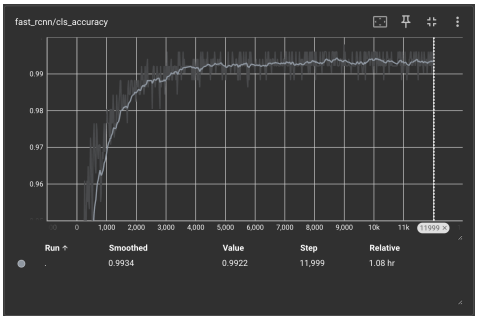

Figure 3: Detectron2 Validation-Set Classification Accuracy Over 12,000 Iterations

sure that the training process does not take too long. 12,000 iterations were used so that the model could thoroughly be trained, even at the expense of longer computational training time. A gamma of 0.1 was implemented so that as the model approached convergence, the learning rate was reduced by one-tenth of the current learning rate value. This gamma value was implemented at the 3000th and 6000th iterations, which allows for a more fine-tuned approach when adjusting the weights for this particular task. When training the ROI heads, 128 ROI proposals were processed for every image since typically more proposals can generally lead to better object detection performance.

After the training process, the COCO evaluator function is used which performs inference on the validation set to calculate desired metrics such as the mAP@50 and mAP@50-95 metrics that were discussed before.

5.2 Detectron2 Results

First, the three key metrics that were used to evaluate the performance of the Detectron2 model on the validation set were: mAP values, speed, and size. To standardize the comparison between Detectron2 and YOLO, the mAP@50 and mAP@50-95 values were used as the focal mAP numerical metrics. On the validation set, the mAP@50 value was 0.893, while the mAP@50-95 was 0.675.

For the three separate categories, the laptop class achieved the highest mean Average Precision at 0.83505 over the IoU range from 0.50 to 0.95. The drink and utensil classes registered lower mAP scores, with 0.68300 and 0.50708 respectively. When comparing how well the model was able to detect objects of different sizes, the model demonstrated superior performance in detecting larger objects, as indicated by a mean Average Precision over the IoU range from 0.50 to 0.95 of 0.675. In contrast, the model's ability to detect small and medium-sized objects was significantly lower, with the mAP value being negligible or not applicable based on the -1.00. Figure 3 shows how the classification accuracy changed throughout the training process. The classification accuracy measures how often the model was able to correctly classify objects that were within the proposals created by the RPN and can indirectly impact the mAP values. As the training process progressed, the classification accuracy grew closer to 1, showing that the model got better over time at classifying objects that were within the proposals.

When observing Detectron2's performance on the test set, the mAP@50 value across the classes yielded 0.806, while the mAP@50-95 was 0.598. When comparing the mAP@50-95 values across all categories, it can be observed that the mAP@50-95 value of 0.63172 for laptops was larger than for utensils and drinks which had corresponding values of 0.61379 and 0.54730. When further dissecting the model's performance on small, medium, and large objects, it was found that the model had a higher mAP@50-95 value for larger objects at 0.614 when compared to medium and small objects which had a mAP@50-95 of -1.000 and 0.191 correspondingly. Table 1 summarizes all information related to mAP values in a compact way.

To further understand how well the model performed on the test set, a confusion matrix was created, which is shown in Figure 4. The model performed perfectly on the drink class because when the ground truth label was indeed a drink, it was able to predict this every single time, which is reflected by a 1.0 on the confusion matrix. The model got confused in two different scenarios, one where the predicted label was a drink and the ground truth was a utensil, and another when the predicted label was a drink and the ground truth was a laptop. 18.5 percent of samples that belonged to the utensil class ended

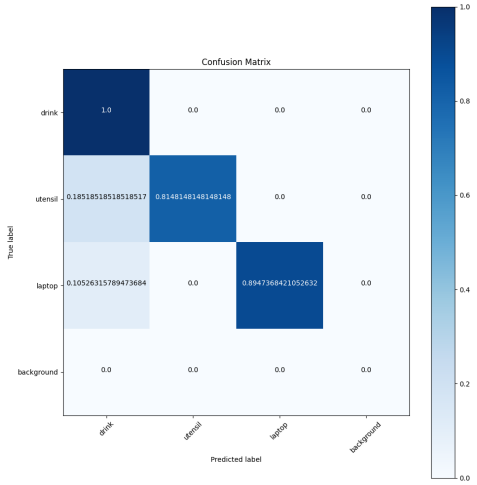

Figure 4: Detectron2 Confusion Matrix Results on Test Set

up incorrectly being predicted as a drink. On the other hand, 10.5 percent of samples that belonged to the laptop class ended up incorrectly being predicted as a drink.

Table 1: Performance Evaluation of Detectron2 Model

Metric	**Validation Set**	**Test Set**
mAP@50	0.893	0.806
mAP@50-95	0.675	0.598
Performance by Category (mAP@50-95)		
Laptop	0.83505	0.63172
Drink	0.68300	0.54730
Utensil	0.50708	0.61379
Performance by Object Size (mAP@50-95)		
Small	-1.000	0.191
Medium	-1.000	-1.000
Large	0.675	0.614

To visually understand how the model performed on sample images from the test set, code was utilized to generate inferences on a few images. Based on Figure 5, it can be seen that the model incorrectly predicts a rice cooker as a laptop and it also mistakenly drew two bounding boxes around a single utensil, which is also an error. On the other hand, there were some images that Detectron2 performed remarkably well on. For example, Figure 6 shows an example of when the Detectron2 model was able to accurately identify utensils and drinks completely correctly in a given image.

In terms of speed, the Detectron2 model training speed was (0.3442 s / it) with 11998 iterations completed in 1 hour, 8 minutes, and 50 seconds. Only 17 seconds were needed for the model to make inferences on the test set, while 16 seconds were needed for the model to make inferences on the validation set.

Further, the number of parameters in the Detectron2 model was 41305311. The disk size of the model file was 330107220 bytes or 314.8147773742676 MB.

6 Modeling Process and Results of YOLOv5

6.1 YOLOv5 Model Set-Up and Training

In the setup and training process for the YOLO model, the workflow can be divided into several key steps, ensuring that both the dataset structure and training parameters are optimally configured for effective model training and evaluation. The structured data is then pooled into a YAML file, which

Figure 5: Misclassification Example of Detectron2

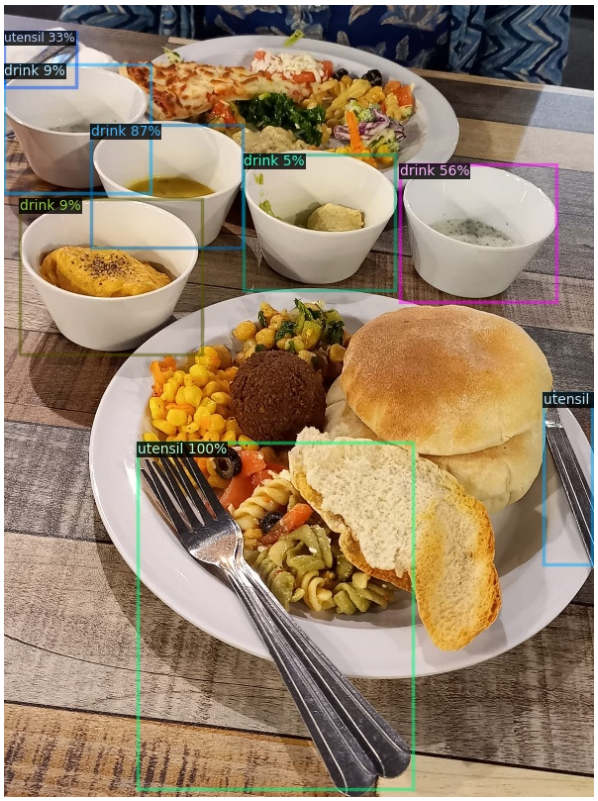

Figure 6: Good Classification Example of Detectron2

contains the pathways to the training, validation, and test image directories. The YAML file also notes the three classes that are involved within the file: drink, utensil, and laptop.

After the YAML file was created, hyperparameter tuning was conducted by manually changing the base hyperparameter file that was cloned from a YOLOv5 GitHub repository. Some of the key modifications that were made were: utilizing an Adam optimizer for more optimal optimization performance, fine-tuning augmentation parameters such as enabling mosaic augmentation, using the Adam optimizer, incorporating jitter, rotating degrees, scaling, shearing, and even translation. These augmentations and modifications were made to enhance the model's ability to generalize to different forms of the training data.

Once hyperparameter tuning was complete, the YOLOv5 model was trained through the command line. The image size was set to 640, batch size to 16, and the number of epochs to 50 to ensure quality training, while not over-fitting the model. The yolov5l.pt file was used to load in pre-trained weights as a baseline for the transfer learning process.

After training was complete, the model was evaluated on the test set to garner a deeper understanding of how well the model did on unseen images. This process mirrors the command line structure that was used for the training process, such as using the same image size, batch size, and epochs.

6.2 YOLOv5 Results

As mentioned before, the three key metrics that were used to evaluate the training performance of the model were: mAP values, speed, and size. When observing the object detection results on the validation set, across all classes, the mAP@50 value was 0.925 and the mAP@50-95 value was 0.725. Upon examining the mAP@50 metric for each class individually, it became evident that the model performed well in identifying laptops, achieving a score of 0.993. The model's performance on the drink and utensil categories was lower, with scores of 0.9 and 0.882, respectively. When comparing the mAP@50-95 for every class, the model once again performed better at identifying laptops, with a mAP@50-95 value of 0.827, compared to the drink and utensil categories, with scores of 0.722 and 0.625, respectively.

Table 2: Model Performance Evaluation on Validation and Test Sets

Metric	**Validation Set**		**Test Set**	
	mAP@50	**mAP@50-95**	**mAP@50**	**mAP@50-95**
Overall	0.925	0.725	0.885	0.656
Laptop	0.993	0.827	0.958	0.784
Drink	0.9	0.722	0.847	0.607
Utensil	0.882	0.625	0.852	0.577

When specifically observing the model's performance on the test set, the model yielded a mAP@50 value of 0.885 and mAP@50-95 0.656 across all classes. More specifically, the model was able to detect laptops the best with a mAP@50 value of 0.958, compared to the drink and utensil classes which had values of 0.847 and 0.852 respectively. Once again, when observing the mAP@50-95 values across all classes, it is clear that the laptop class had the best value of 0.784, compared to the drink and utensil categories which had scores of 0.607 and 0.577 respectively. Table 2 summarizes the mAP value information compactly.

To analyze how well the YOLOv5 model performed on the test set, a confusion matrix was created. Figure 7 shows that the model classified the drink category with the highest true positive rate of 0.91. The model also classified the laptop class with a reasonably high true positive rate of 0.89, but was not able to retain this strong performance when classifying the utensil class, which had a true positive rate of 0.78.

There were some notable errors by the model such as how 22 percent of samples that belonged to the utensil class ended up incorrectly being predicted as the background. Further, the background class had the lowest precision, with considerable confusion with the drink and utensil classes. There was also a scenario where 11 percent of samples that belonged to the laptop class were deemed background images.

To measure the accuracy of the model on the test set, the F1 score was found because it is the harmonic mean of both the recall and precision. Typically, the closer the F1 score is to 1, the better.

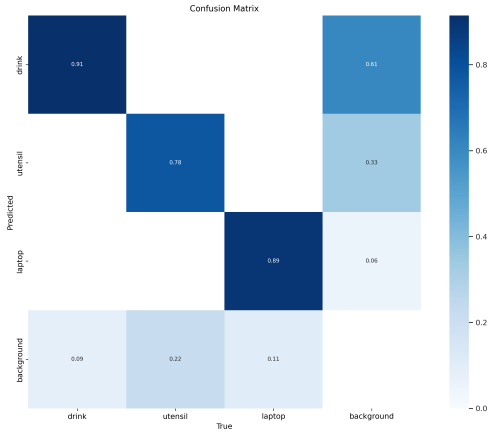

Figure 7: YOLOv5 Confusion Matrix Results on Test Set

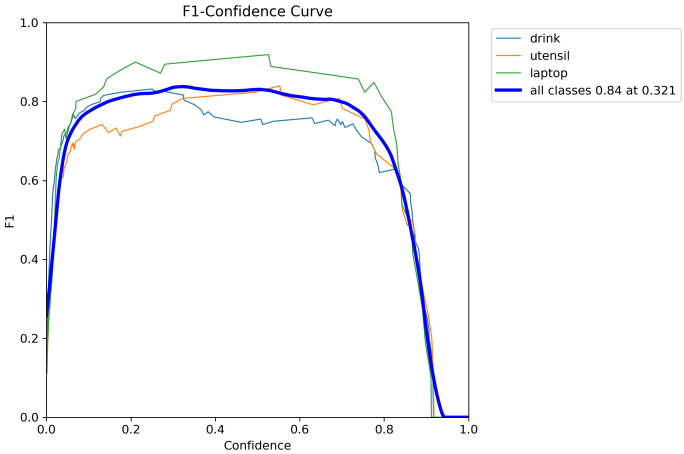

Figure 8: YOLOv5 F1 Score Results on Test Set

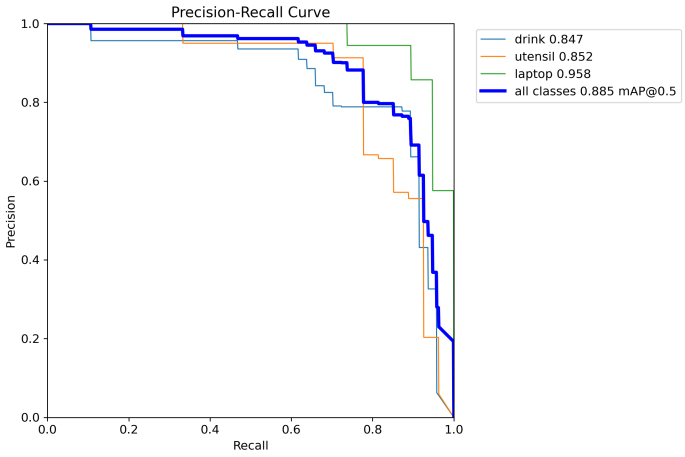

Figure 9: YOLOv5 PR Curve Results on Test Set

Figure 8 shows the relationship between how confident the model was in its prediction as well as how accurate those predictions ended up being. When observing this figure, it is clear that the laptop F1 score is relatively high and stable across the confidence thresholds, which indicates robust performance. The drink and utensil categories have a comparatively lower F1 score as the confidence threshold increases. The optimal F1 score is achieved for all classes at 0.84 at the 0.321 confidence threshold.

To find out which class the model was best at predicting, a precision-recall plot was created. For precision-recall plots, the closer the plot is to the top right, which indicates high precision and recall, the better. According to Figure 9, across all levels of recall, the laptop class had the highest level of precision, peaking at an average precision of 0.958. The average precisions of the utensil and drink classes were not far behind at 0.847 and 0.852 respectively. The mAP reached 0.885 at a 0.5 IoU threshold, which indicates strong performance across all of the recall levels.

To understand how the model performed on a few sample images from the test set, code was developed to utilize the trained model to perform inferences. Figure 10 shows the performance of YOLOv5 on a few test set images. Based on this image, it can be seen that the model was able to identify drinks, utensils, and laptops well. However, there were examples where the model failed to recognize an object that falls into one of these classes, which could be a problem if missing detections are severe such as in the medical community.

The YOLOv5 model that was utilized for the training and test set evaluation had 267 layers, 46119048 parameters, and operates at 107.7 GFLOPs (Giga Floating Point Operations per second). The time needed to train the model was 31.62 minutes. In terms of the speed of the model, YOLOv5 took 1.1ms for pre-processing, 17.0ms for inference per image, and 10.2ms NMS per image at shape (32, 3, 640, 640). In total, the inference speed is roughly 28.3 ms per image.

7 Discussion of Detectron2 Vs. YOLOv5

7.1 mAP@50 and mAP@50-95 Value Comparisons

The mean average precision is a critical metric to cross-compare model performance between Detectron2 and YOLOv5. In general, mAP measures the combination of precision and recall over a variety of different IoU thresholds. In this project, the mAP@50 and mAP@50-95 were particularly used. The mAP@50 measures the average precision of the object detection task where the model's bounding box prediction overlaps with the ground truth label by at least 50 percent. On the contrary, the mAP@50-95 offers a more in-depth measurement of calculating the average precision. This metric calculates the average precision over a range of IoU thresholds from 50 percent to 95 percent, which emphasizes the precision of the bounding box's size/position in comparison to the ground truth.

The results section shows the strengths and weaknesses of both models. As observed from Table 1 and Figure 4, the Detectron2 model resulted in a mAP@50 value of 0.893 and a mAP@50-95 value of

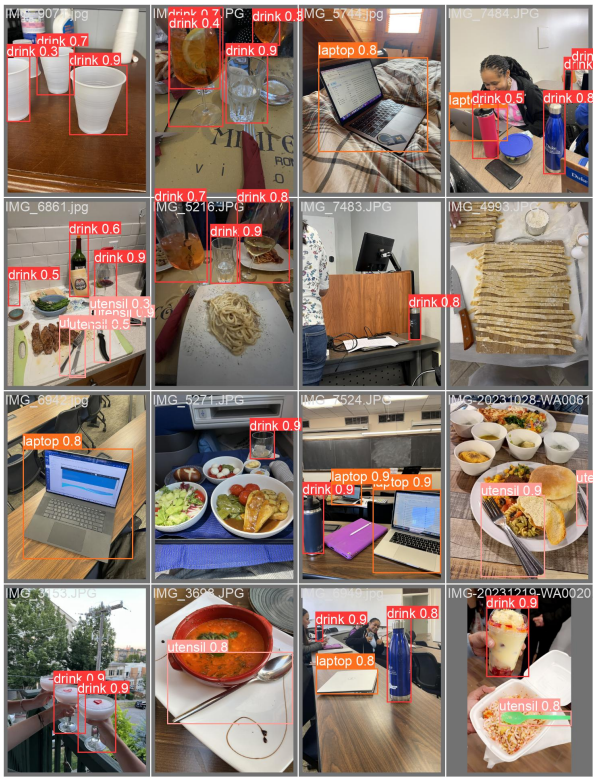

Figure 10: YOLOv5 Inferences on a few Test Set Images

0.675, which shows that the model does reasonably well in correctly detecting objects with different IoU thresholds. The Detectron2 model does well in identifying laptops in particular, which is reinforced by having the highest mAP@50 and mAP@50-95 values across the categories. A key weakness of this model is that the performance significantly dipped on small and medium-sized objects in both the validation and test sets, which could indicate overfitting and a lack of generalizability of the model. The model's performance also dipped on the test set across all categories, which further reinforces the idea of overfitting to the validation set. Introducing more training data, utilizing more data augmentation methods, and implementing more in-depth hyperparameter tuning could potentially help combat this issue in the future and help the model improve performance.

On the contrary, the YOLOv5 model outperformed the Detectron2 model on both the validation and test set. This is indicated in Table 2 with the higher mAP values across all categories: 0.925 at mAP@50 and 0.725 at mAP@50-95 for the validation set and 0.885 at mAP@50 and 0.656 at mAP@50-95 for the test set. Similar to Detectron2, the YOLOv5 model's mAP@50 and mAP@50-95 values were the highest for the laptop category, however, the Detectron2 model was not able to surpass the impressive mAP@50 and mAP@50-95 values that were displayed by the YOLOv5 model for each class in general. These results show the robust and generalizable performance of the YOLOv5 model across different categories.

7.2 Speed Comparison

The second key metric that was observed was speed. The YOLOv5 model and Detectron2 models have similar inference times on each image of the test set with roughly 17.0ms taken per image. However, it should be noted that the overall training time for the YOLOv5 model was roughly 32 minutes, while the Detectron2 model required more than an hour for the full training process.

The difference in training time could be attributed to the architectures of these models. For example, YOLO may be faster than Detectron2 since it uses a singular step to not only identify an object but also classify it. In essence, the YOLO model can predict bounding boxes and the class probabilities all in one go from the image. On the other hand, the Faster R-CNN based Detectron2 model has to first find the region of interest using an RPN and then classify those boxes separately,

which may add computation time. Furthermore, the YOLO model utilizes advanced techniques such as layer fusion, which combines many operations into a single layer, which enhances training speed and efficiency compared to Detectron2.

7.3 Model Size Comparison

When comparing the sizes of the models, the YOLOv5 model was considerably larger at 46119048 parameters than the Detectron2 model, which had 41305311 parameters. Since the YOLOv5 model was larger, it was able to capture more complex and intricate patterns, which potentially explains the higher accuracy and reduction in misclassification as shown in Figure 7. However, there are some drawbacks to a larger model. For example, a larger model may require more computational resources, which may not bode well for other tasks that require larger volumes of training data.

Although the YOLOv5 model is larger than the Detectron2 model, it is clear that it does not impede upon its performance due to the quicker training and more accurate mAP values across the board. This could potentially be due to a more effective model architecture. Detectron2 may be preferred when model size is a significant factor because even though it performed worse than YOLOv5, it still has solid results that me be favored in some circumstances.

7.4 Confusion Matrix Comparison

The confusion matrices for both models, Figures 4 and 7, show contrasting results. The Detectron2 model confused the utensil and laptop classes with the drink class. Figure 4 highlights that for a pure drink classification model, Detectron2 does an excellent job because it showed perfection in predicting drinks when the ground truth label for the object was indeed a drink (true positive). However, since this project deals with detecting multiple classes, the misclassifications that Detectron2 exhibits are not favorable and are alarming. For example, 18.5 percent of samples that belonged to the utensil class were incorrectly predicted as a drink. Also, 10.5 percent of samples that belonged to the laptop class were incorrectly predicted as a drink. On the other hand, Figure 7 shows that YOLOv5 has a much more balanced misclassification spread. Although no class reached perfection like the drink class in Detectron2, YOLOv5 avoids the high misclassification rates that were exhibited in Detectron2. The key weakness of the YOLOv5 model occurs when the model predicts either a drink or utensil when the ground truth is background, with misclassifications of 61 percent for drinks and 33 percent for utensils, which indicates that the model tends to falsely detect objects within the background. These represent false-positive rates, which could be indicative of overfitting of the model. However, for this project which aims to identify the three categories, the YOLOv5 still has the edge.

Misclassifications could have negative impacts in the real world, which is why the true positive rate should be heavily valued for object detection tasks. Although both models have misclassifications, the alarming rate of misclassifications in the Detectron2 model should be noted. Future techniques such as introducing a wider variety of training data and incorporating a larger model may enhance both the Detectron2 and YOLOv5 model performance and reduce potential classification mistakes.

7.5 The Better Model

The F1 score and PR results, highlighted in Figures 8 and 9, further reinforce the strong performance of YOLOv5. The high F1 scores across the confidence threshold show that the YOLOv5 model was able to consistently balance between precision and recall. Since the model minimizes false negatives and false positives, the model's performance can be viewed as quite robust. On the other hand, the PR curve in Figure 9 shows that the model does well in distinguishing between the three classes: laptop, utensil, and drink. The model exhibits high recall and precision for each class, which indicates that the false positives and negative rates are kept to a minimum. Figure 9 also shows a high area under the curve (AUC) which further reinforces that this model is capable of producing a high true positive rate.

In summary, the YOLOv5 model seems to be more favorable when accuracy and speed are prioritized and computational resources are not limited. For object detection across diverse classes and real-time detection, YOLOv5 should be utilized.

8 Conclusion

In conclusion, when comparing various metrics such as the mAP value, speed, and model size, YOLOv5 emerges as the superior model for this particular project. As previously mentioned, it boasts a higher mAP value across the mAP@50 and mAP@50-95 thresholds than the Detectron2 model. The architecture of the YOLOv5 model enables the model to have a faster training and inference time, even though it has more parameters than Detectron2. This shows that a model's larger size does not necessarily imply a slower performance if it is designed more efficiently. Even though Detectron2 performed well in the drink class, YOLOv5's more balanced misclassification rates and higher precision/recall make it more suitable for real-world applications such as the task at hand. In the future, exploring a larger variation of data augmentation and hyperparameter tuning could be used to improve its performance in general. Although YOLOv5 performed better than Detectron2, the model is larger, which typically requires more computational resources. Potentially exploring methodologies that can retain model performance, while decreasing the size of the model would broaden the model's applicability when tailored to a resource-sensitive setting. Although roughly 300 images were used in total for the modeling process, implementing more images could enhance the model's capabilities across various sizes, unseen objects, etc. Overall, YOLOv5 is the recommended model for this particular task and dataset due to its high accuracy and speed.

9 Citations and a References List

Chen, Xu. "Lecture 4" Applied Computer Vision, 02/5/2024, Duke University. Lecture.

Chen, Xu. "Lecture 5: Object Detection: RCNN and Fast-RCNN" Applied Computer Vision, 02/12/2024, Duke University. Lecture.

Chen, Xu. "Lecture 6: Advanced Object Detection" Applied Computer Vision, 02/19/2024, Duke University. Lecture.