Isaac Steiner
CIS 530
15 May 2025

Golf Ball Tracking with YOLO - Term Project

Introduction

Tracking small objects with computer vision presents a great challenge. It involves tracking a fast moving object that takes only a few frames to be out of sight. In this project, we focus on detecting a hit golf ball from the tee. The task is difficult because the small size and motion blur can make it hard to accurately detect the ball's position. Also detecting a white golf ball in an outdoor environment presents a huge challenge due to the lighting and contrast differences against the background. Creating a tool that does such is critical in spaces like sports, so that golfers can analyze their ball's trajectory.

To implement this challenge, we utilize two object detection models in particular: YOLOv7 and YOLOv9-m. These models are from Ultralytics and are well known for object detection performance. YOLOv7 was chosen as it is a widely adopted model used for general detection use cases (Wang et al., 2022), while YOLOv9-m was released with better performance for detection on small objects (Ultralytics, 2024), which proves worthy for a comparison against YOLOv7. Both models were trained under the same dataset and used consistent threshold and inference settings to allow for fair comparison results.

The goal of this project is to evaluate which model is better suited for golf ball detection at tee-off using a custom Roboflow dataset by the user Trungam (2024). This dataset contains annotated video frames, with the "ball" classes being crucial for the model to train on. It also contains the "fmo_ball" class which is frames where the ball is showing but has motion blur. The last class the dataset includes is the "club_head" class, but was opted out of detection due to the task specification of tracking a golf ball's center point. The evaluation of the models involves standard object detection metrics including but not limited to precision, mean average precision (mAP), and recall. Additionally, the project utilizes real-world usability to determine if the detection outputs are usable in actual live environments.

By comparing YOLOv7 and YOLOv9-m with this specific task we aim to show which model offers better performance for object detection on a small, rigid-body golf ball that can be usable in real-world sports analytics.

Background and Related Work

Object detection is a critical task used in computer vision. Applications of such involve autonomous driving, sports analytics, and more. An influential object detector in the vision space is the YOLO (You Only Look Once) models by Ultralytics, which aim to balance speed and accuracy.

Why YOLOv7 vs YOLOv9-m

For YOLOv7, it was recognized for exceptional speed and accuracy (Wang et al., 2022). It is also widely adopted, used in Samuel Sung's medium article for a golf tracking project (Sung, 2024). For YOLOv9-m, it is the mid-sized variant (easier on compute resources, reduced training times compared to YOLOv9-c). It claims improved detection of small and blurred objects, has notable advancements in balancing complexity and detection performance (Ultralytics, 2024). Ultralytics also claims that this model offers significant reductions in parameters and computations against the backdrop of improved accuracy (2024).

Why this work matters

Tracking a golf ball can be a useful tool for golfers who want to track the trajectory of their ball. Monitoring the behavior of the ball is a metric that is also desired by broadcasts for viewers at home to follow the ball easier.

Methodology

The methodology for this project involves evaluating and comparing the performance of two object detection models (YOLOv7 and YOLOv9-m). Using these models, the goal is to detect fast moving golf balls in high-speed video frames. Both models were fine-tuned utilizing a custom dataset using identical annotations and training conditions to ensure fair competition/comparison.

Dataset

The labeled dataset comes from Roboflow (trungam, 2024), and involves using high-resolution golf swing footage in frame by frame analysis. Each frame could contain the following annotations:

1. "Ball": the golf ball
2. "Fmo_ball": the golf ball in motion blur
3. "Club_head": the head of the golf club.

The primary objective of using this dataset was to recognize the first two ball classes, in order to maintain tracking in blurred states. The "club_head" class was eliminated during testing and inference due to the task specification of focusing on the ball itself. The dataset has a Train/Valid/Test split of 70/20/10 respectively. All annotations were downloaded in YOLO format. The dataset was made publicly available under the Roboflow Universe's reuse terms. All credit and thanks go to user Trungam (2024).

Model Training

Both of the YOLO models (v7 and v9-m) were initialized with pre-trained weights. The following settings were used:

- Input Size: 640x640
- Epochs: 150
- Optimizer: SGD
- Confidence threshold: 0.001 validation, 0.05 inference.
- IoU threshold: 0.65 for validation.

The environment for the experiment was a docker container with PyTorch and CUDA. The model weights were trained inside /workspace/yolov*.

Hardware

The model was trained on an NVIDIA RTX 3060 Laptop GPU, 6GB of VRAM. This was done intentionally versus using a cloud resource such as Google Colab or Beocat, due to the nature of the project. Since this is a beginner project, the goal was to test using beginner laptop hardware that a common AI beginner would have.

Baseline/Alternative Model Code Source:

YOLOv7 training was conducted using the official WongKinYiu/Yolov7 repository (Wong, 2022) and YOLOv9-m training was done using the WongKinYiu/Yolov9 repository (Wong, 2024). Both models were trained using the same image augmentation pipeline and data splits to ensure experimental fairness. Minor modification was done to these files, only to fix errors and modify the drawings to draw a center dot on the ball. All code was used under the GNU General Public License v3.0. Credit and thanks goes to the respective author(s) of the repository.

Inference Modifications

For our task of tracking small, fast moving objects, we modified the detect.py script to draw detections as center points instead of the standard bounding boxes with labels that come with the repository.

Additionally, the club_head class was excluded using the –classes flag, specifying the other 2 class names only.

Evaluation Approach

Each model was run on the test set using test.py for YOLOv7 and val.py for YOLOv9-m. The standard detection metrics were used:

- Precision: Proportion of predicted positives that were correct

- Recall: Proportion of actual positives that were correctly predicted
- mAP@0.5: Mean Average Precision at IoU 0.5
- mAP@0.5:0.95: Mean average Precision over IoU thresholds form 0.5 to 0.95.

Experiment Design and Results

Experimental Setup

The Roboflow dataset was divided into training, validation, and test sets at a 70/20/10 split respectively. All models were evaluated on the same test set of consistency and fairness. In order to maintain fairness, both models used the same annotation files and data.yaml for both. Inference was run on a threshold of 0.05.

Quantitative Results

Overall Results (Full Dataset)

Model	Precision	Recall	mAP@0.5	mAP@0.5:0.95
YOLOv7	0.689/0.812	0.905/0.386	0.902/0.553	0.533/0.224
YOLOv9-m	0.933/0.804	0.810/0.702	0.901/0.727	0.568/0.340

Using these quantitative results, we can interpret the following:

- Precision: Alternative model improves detection precision significantly, meaning fewer false positives. Both perform similarly for the fmo_ball class.
- Recall: Baseline is better at recalling slow-moving balls, while Alternative model is better at recalling fast moving balls (fmo_ball). This shows YOLOv9-m's improvements to small object detection.
- mAP@0.5: Alternative has an edge for the fmo_ball class, while performance is identical for ball
- mAP@0.5:0.95: Alternative leads for both classes, performing better across varying IoU thresholds, a sign of more precise detections overall
- Quantitative Conclusions: Alternative is more balanced, Baseline is better in recall for stationary or slow balls but is not as good at fmo_ball recall. Overall the alternative is a better choice for the task.

Paired T-Test

To evaluate, a paired t-test was conducted using mAP@0.5 values across 11 video groups. The resulting p-value was 0.0705. This was calculated using the excel T.TEST formula, with the following parameters:

- Array1: YOLOv7 mAP@0.5 values
- Array2: YOLOv9-m mAP@0.5 values
- Tails: 1 (one-tailed)

- Type: 1 (paired test)

Analyzing this value, it suggests a positive trend in YOLOv9-m's favor. However this is over the standard threshold of 0.05 (Beers, 2024). What that means is that since the value we found is over the standard, we cannot reject the null hypothesis. But due to the close nature of the value to the standard threshold, we can suggest YOLOv9-m is probably better. In order to prove it, we would need more data to prove it with the standard 95% confidence.

Qualitative Results

The models were also run on real swing footage and visualized detections. Instead of standard bounding boxes, dots were drawn at the center of each predicted ball.

Example Image from detect test set, v7(left) and v9-m(right). Img Source: Trungam, 2024

The models did a fairly good job on the test set of drawing a dot on the balls in most frames. However, the baseline model did struggle with some moving ball objects. Balls that appeared motion blurred (fmo_ball class) were sometimes unable to be detected by the baseline, while the alternative was able to.

Baseline(v7, left) vs Alternative(v9-m, right) results for a fmo_ball class. Img Source: Trungam, 2024

In videos not from the test set, the models both often struggle to maintain tracking in blurry, or low frame-rate videos. They also struggle to maintain the tracking as the ball gets further away. Even with training, the models can detect balls in a controlled setting, but lack the consistency in video results to suggest either models are capable of enterprise-level usage. As a beginner computer vision project however, the models can detect quite a bit in extracted frames with the right data context.

Summary and Future Work

Key Findings

In summary, the following are the takeaways from this project:

1. YOLOv9-m outperformed YOLOv7 slightly

Although results from the paired t-test do not allow us to say this for certain, the results from the overall image set found in the quantitative result section and the qualitative results allow us to suggest such. While data suggests its close in some areas, a general eye test can also allow us to remove the scope of data analysis and look at the video results. In general, YOLOv9-m can detect fast ball objects better, see example image in qualitative results. YOLOv9-m was found to produce fewer false negatives and higher consistency in quantitative analysis.

2. YOLOv7 is Competitive

YOLOv7 can produce similar results to v9, with easier setup and faster inference. Given the local limited resource environment this project was conducted in, v7 proved that we can not for certain state v9-m is the better model. This is why we state in the point above that we are suggesting that it is better, however due to the t-test and relative closeness the baseline YOLOv7 model was similar in the scope of usefulness by an enterprise level project.

Lessons Learned

Some lessons I learned throughout this project are as follows:

1. Model Evaluation goes beyond accuracy, speed and stability are impactful.
2. Working with small objects like golf balls requires clean, high frame-rate, high resolution data. Decent lighting is also important to get desired results.
3. Consider using cloud compute resources to train models.

Limitations/Challenges Faced

Some challenges I faced throughout the project are as follows:

1. Model Compatibility Issues

YOLOv9-m required modifications to detect.py and val.py to match project needs and output formats, which was not needed for YOLOv7. The baseline appeared to be a more stable version of the object detection code.

2. Detection of Small/Fast Objects

The motion blur and fast moving nature of the golf balls made it difficult to detect. Even with training, results showed a good start to detection but often missed frames.

3. Computational Constraints

Training YOLOv9-m on the GPU took around 13 hours, similar time for the YOLOv7 training. Batch size and worker tuning were done to help with speed. Overall, using the laptop GPU was a good learning process for how computationally demanding AI computer vision projects can be.

4. Video Processing Failures

Yolov7 failed to output annotated videos from .mp4 files directly. It required frame extraction in order to run. This was done using the FFmpeg tool. Re-assembly was done in the same manner.

Future Work

In future iterations of the project, the following are some things to incorporate:

1. Incorporate the Kalman filters mentioned in Sung's article (Sung, 2024)
2. Train YOLOv9-m on more classes, using things like airborne ball may help improve performance
3. Expand the training set to cover more angles, most images were tee shots, while we could have included putting angles to help with that, as golfers also care about tracking a ball's path on the green.

References

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. ArXiv:2207.02696 [Cs]. https://arxiv.org/abs/2207.02696

Sung, S. (2024, September 2). Track Your Golf Ball With Computer Vision - Towards AI. Medium; Towards AI. https://pub.towardsai.net/track-your-golf-ball-with-computer-vision-511886bd0f12

Ultralytics. (2024). YOLOv9. Ultralytics.com. https://docs.ultralytics.com/models/yolov9/

trungam. (2024). Golf Fmo Dataset. Roboflow. https://universe.roboflow.com/trungam/golf-fmo/dataset/8

Wong, K.-Y. (2022, September 12). Official YOLOv7. GitHub. https://github.com/WongKinYiu/yolov7

Wong, K.-Y. (2024, March 15). WongKinYiu/yolov9. GitHub. https://github.com/WongKinYiu/yolov9

Beers, B. (2024, November 13). What P-Value Tells Us. Investopedia. https://www.investopedia.com/terms/p/p-value.asp