Counting the Number of Punches Thrown in Boxing Matches

Youngjo Choi, Wenxuan Zhu, Ameer Ali, Xinrui Yu and Jafar Saniie
Embedded Computing and Signal Processing Research Laboratory (http://ecasp.ece.iit.edu/)
Department of Electrical and Computer Engineering
Illinoise Institute of Technology, Chicago, IL, U.S.A.

***Abstract*— Our project involves the use of Computer Vision and Artificial Intelligence to identify punches during boxing matches. Briefly, our method involves collecting a custom dataset, which is achieved by extracting images from footage of boxing matches. We employ Roboflow software to manually tag objects within these images. These annotated images are then fed into the YOLO (You Only Look Once) software. YOLO is designed to automatically recognize the tagged objects within unlabeled images. Subsequently, we integrate the analyzed dataset from YOLO into our Python code. We then construct an algorithm within this code to specifically identify punches. (*Abstract*)**

Keywords—Artificial Intelligence, Object detection YOLO, Computer Vision, Boxing matches, Roboflow dataset

I. Introduction

The genesis of our project lies in the contentious history of judging in boxing. Numerous results, even in recent matches, have sparked significant controversy. Consequently, our first course of action was to investigate the current scoring system.

The scoring system consists of three principal elements. Initially, there is a referee who is entrusted with the authority to declare a winner if they believe one boxer can no longer continue with the match. If the match progresses to the point of judgement, three judges positioned at ringside play a crucial role. They evaluate the match based on a multitude of criteria before issuing their final decision. Lastly, the count of punches, an integral part of the scoring process, is manually performed by two operators.

- Referee: They can stop a match if the referee thinks one player cannot continue.
- Three Judges: When a match goes to a judge's scorecard, they decide the winner.
- Two Operators: They count the number of punches of each player.

I perceive the punch counting statistics to hold substantial significance, as while judging scores and refereeing may inherently bear elements of subjectivity and inconsistency, the process of punch counting emerges as the most objective among the three components.

Consequently, I closely scrutinized the method used for counting punches. At present, all major boxing competitions employ a system known as "CompuBox". This involves two operators, with operator 1 assigned to player 1 and operator 2 to player 2. The operators do not exchange their counts; thus cross-validation is absent. Moreover, several matches have sparked debates among fans who argue that the post-match released statistics do not reflect accuracy.

Inevitably, such controversies arise given that the counting process is manually performed by two individuals. Moreover, as these statistics are only occasionally disclosed during the match, fans are unable to access them in real-time.

For these reasons, we opted to create a program capable of automatically counting the number of punches from provided images, thereby enhancing accuracy.

The advantages of our program over the existing Compubox are twofold. Firstly, as our program automatically counts punches, the likelihood of bias is significantly reduced. Secondly, with live punch-counting, audiences can access the statistics in real-time, further augmenting their match-watching experience.

II. System Modules

The implementation process can be segmented into three phases. Initially, we will gather data and label the corresponding images. Next, we'll feed these labeled datasets into YOLO for automated object detection. Finally, we will develop a Python algorithm capable of tallying the number of punches, utilizing the classified objects provided.

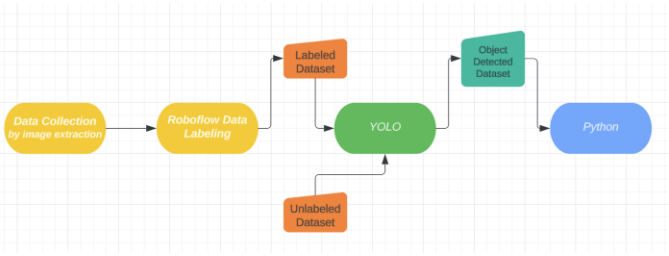

Fig. 1. System Flowchart

III. Hardware

The hardware used was a desktop pc that utilized the following specifications:

- AMD Ryzen 9 5900x
- Nvidia GeForce RTX 3080 12gb
- 1tb NVME SSD

- 64gb DDRY Memory

The specifications of the pc was enough to handle the task at hand while being more accessible than doing multiple Google COLAB purchases for resources. An NVIDIA gpu is highly recommended as PyTorch(and therefore YOLO) work best when using the CUDA cores found in Nvidia gpus. Having 10gb or more of VRAM is also recommended in order to train the model on the images.

IV. Software

The AI algorithm that was utilized in this project was the YOLO algorithm. YOLO is an object detection algorithm that is fast and efficient allowing for excellent real-time object detection. YOLO uses a convolutional neural network(CNN) to put objects in an image in a bounding box[1]. To measure metrics, YOLO uses Intersection over Union (IoU) and mean Average Precision(mAP) to see how accurate the model is for the objects in each image[1].

A. Update scores

The other algorithm used was a custom designed Python algorithm that was used to count the number of punches each boxer had against the other. The punch count algorithm utilized the Ultralytics library(this was done to use the images that were trained with YOLO), the numpy library, math and cv2 libraries. The punch counting algorithm uses a function named *update_score*. This function takes in the YOLO detections (results), previous contacts between gloves and boxers, and current scores of the two boxers. It extracts bounding boxes of the boxers and their gloves. If a glove bounding box intersects with the opposing boxer's bounding box, it is considered a contact, and the score of the attacking boxer is updated.

B. Update hit ratio

To determine a “hit” in the algorithm’s *update_hit_ratio* function, the algorithm calculates the distance between the center of a boxer’s body and their glove. If this distance exceeds a threshold(meaning a punch was thrown) the punch is considered to have been thrown by the algorithm. The hit ratio is then calculated as the number of hits over the number of punches thrown. The distance used in the algorithm is the Euclidean distance. To determine contact in the *contact,* the algorithm will take two bounding boxes as inputs and calculate whether or not they are intersecting. If they are then it would return True else False. The main section of the algorithm will have YOLO process a video frame by frame for object detection. It will then use the *update_score* function to to update the scores for each boxer based on contacts and then use the *update_hit_ratio* function to calculate the hit ratios. Within the processed video the algorithm will render the scores and hit ratios for viewers.

The programming language used in this project was Python. Python allows for easy use of the Ultralytics library in order to utilize YOLO tools in a custom algorithm and Python allows for easy utilization of OpenCV. When compared to other languages such as Java for example, it may also have an OpenCV implementation, but Java cannot utilize the Ultralytics library. The same can be said for the other programming languages which cannot utilize the Ultralytics library making integration with the images trained on YOLO very difficult.

There are currently no real-time constraints as the model was able to track the boxers in real time and the punch counting algorithm was able to track the punches from the boxers in real-time as well. If a delay with the punch counting algorithm occurs the problems that would arise is that there would be an increase in processing time, frame skipping and out-of-sync output. In the custom algorithm the most likely cause of a delay would be the time it takes for YOLO to process each frame. Some measures to reduce this could be to use a gpu for YOLO as the parallel processing power would speed up the algorithm, reduce the frame rate or use batch processing. There are no security issues that were encountered.

The YOLO algorithm was developed by a team of researchers and was not made by our group [2]. The punch counting algorithm was custom developed by us and has no previous references.

V. Related Work

One relevant project is "Soccer Player Recognition using Artificial Intelligence and Computer Vision" by Charles-Alexandre Diop, Baptiste Pelloux, Xinrui Yu, Won-Jae Yi, and Jafar Saniie [3]. Similar to our focus on the role of an operator in a boxing match, this project aims to provide post-match analysis, player tracking, ball tracking, and relevant statistical information about teams and players in soccer games. Both projects utilize artificial intelligence and computer vision techniques to gather information about athletes and visualize game data to enhance interaction with the audience.

A. Task objectives,

The "Soccer Player Recognition using Artificial Intelligence and Computer Vision" project, as well as our boxing project, seek to facilitate post-match analysis. This includes tracking players, tracking the ball, and generating statistics related to teams and players. Both projects leverage AI and computer vision technologies to collect information about athletes and visualize game data more effectively for audience interaction.

B. System design

The "Soccer Player Recognition using Artificial Intelligence and Computer Vision" project involves a detection and recognition system to identify players on the field, their facial features, and their jersey numbers. In our boxing project, we detect and recognize the upper body striking areas of different boxers, as well as their respective boxing gloves.

C. Algorithms

The "Soccer Player Recognition using Artificial Intelligence and Computer Vision" project utilizes YOLOv3 as the algorithm for the detection part, while our project employs YOLOv8. YOLOv3 has a complex architecture, utilizing a Darknet-53 backbone consisting of 53 convolutional layers. It achieves higher accuracy but sacrifices speed, making it suitable for scenarios where accuracy is crucial. Additionally, YOLOv3 incorporates techniques like anchor boxes and feature pyramid networks (FPN) for detection [4].

Furthermore, we have implemented a post-processing algorithm to calculate the number of punches thrown by each athlete and their accuracy. This information is crucial for post-match statistical analysis by the operator and enhances real-time interaction between the audience and the boxing match.

D. Datasets

The aforementioned project collected over 3,000 images from a soccer match between the French team and the Argentinian team during the 2018 FIFA World Cup. These images were obtained from the internet and added to their custom dataset. The project also utilized the Haar Cascades Classifier's face detection algorithm to extract facial information of players. In our project, we extracted approximately 3,000 frames from the Canelo Alvarez vs. Billy Joe Saunders boxing match in 2022 as available images for our dataset.

VI. RESULTS

To begin with, in relation to the dataset, we curated a collection of around 3,000 frames from the highly anticipated match between Canelo Alvarez and Billy Joe Saunders in 2022. Subsequently, a meticulous annotation process was carried out, wherein approximately 200 images were diligently labeled using instance segmentation with a focus on four distinct categories: boxer_1, boxer_2, gloves_1, and gloves_2. These frames were carefully chosen to ensure the availability of diverse and representative images for our dataset.

Fig. 2. Extracted frame for the custom dataset

Fig. 3. Training images with labels

Additionally, we utilized the pre-trained YOLOv8 model from the COCO dataset to train our custom dataset. The validation batch yielded promising results, demonstrating the model's accurate identification and localization of the annotated categories. This approach leveraged the existing knowledge and expertise from the COCO dataset, enhancing the efficiency and reliability of our training process. The successful adaptation of YOLOv8 to our specific dataset establishes a solid foundation for further advancements in object detection and recognition within the realm of boxing.

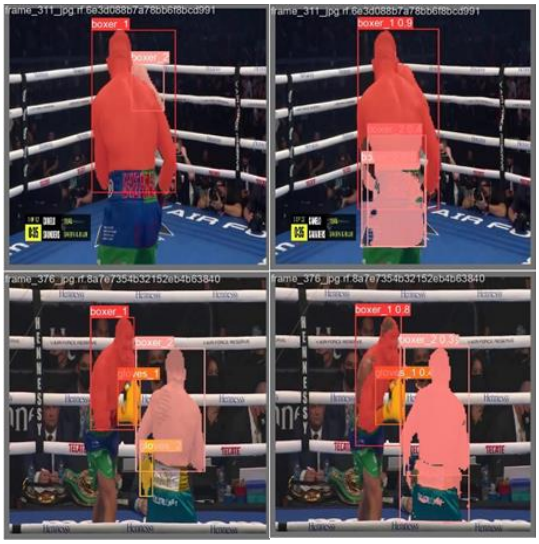

Fig. 4. Original validation data (Left) and Prediction validation data (Right)

The figure below displays the variations of different losses over 120 epochs. It demonstrates that regardless of whether it is box loss, segmentation loss, class loss, or depthwise feature learning loss, the curves tend to converge and stabilize below 1 after a certain number of epochs. This suggests that there is potential for further reduction in loss if the training is extended beyond the current number of epochs. The graph indicates that additional training could lead to continued improvement in loss reduction.

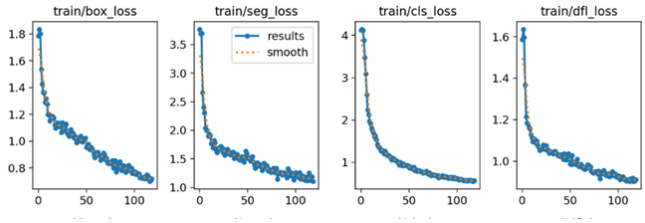

Fig. 5. Loss curve during training process

The following graph illustrates the precision and recall curves for both box and segmentation. From the earlier epochs, it can be observed that recall converges faster than precision. This could be attributed to recall being typically associated with lower confidence threshold, allowing the model to detect objects more easily with lower confidence during the training process, thereby increasing recall. On the other hand, precision is usually linked to a higher confidence threshold, demanding the model to be more confident in object detection, resulting in higher accuracy. However, the final value of recall tends to be slightly lower than precision. This could be because recall is computed

at lower confidence thresholds, requiring the model to detect as many objects as possible, even if it includes some false positives. In contrast, precision is calculated at higher confidence thresholds, emphasizing confident object detection for higher accuracy [5]. Similar results are also demonstrated in the mean Average Precision curve.

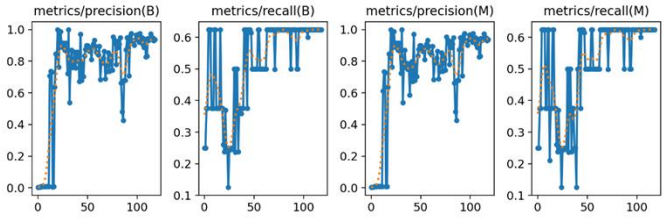

Fig. 6. Precision and recall graph

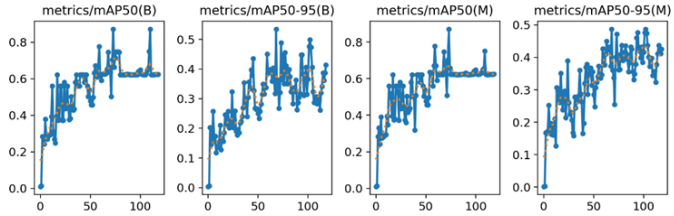

Fig. 7. mean Average Precision graph

To assess the validity of hits, we utilize the predicted results from the YOLO model. Specifically, we determine if the bounding box of boxer1's glove intersects or overlaps with the bounding box of boxer2's body part, while also confirming that there was no contact or overlap in the previous frame. This methodology allows us to identify legitimate hits for boxer1 at a given moment. Similarly, we apply the same approach to calculate the score for boxer2. By evaluating the intersection and temporal continuity, we can accurately determine the validity of hits for both boxers based on their respective bounding boxes and track the scoring in the match.

Fig. 8. Illustration of the counting score algorithm

In order to determine if a boxer has thrown a punch, we assess the distance between the center of the glove's bounding box and the center of the current boxer's body part bounding box. If this distance exceeds a predefined threshold and was below the threshold in the previous frame, it signifies a punch attempt. By analyzing the change in distance and comparing it against the threshold, we can identify instances where boxers have initiated a punching action. This method allows us to accurately detect, and track punch attempts based on the positional relationship between the glove and the boxer's body part.

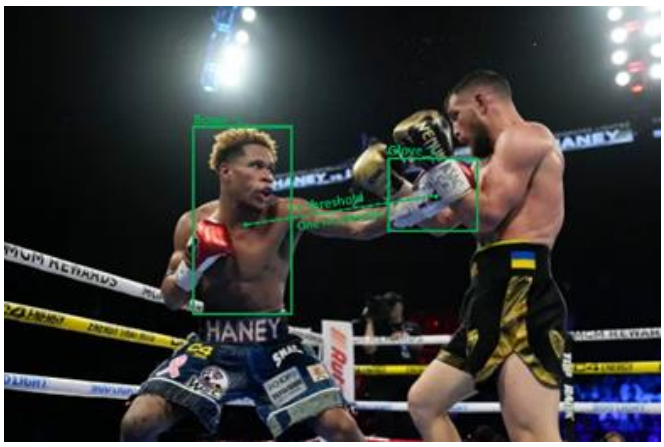

Fig. 9. Illustration of detecting punch thrown

When initially establishing the custom dataset, we considered using bounding box annotations. However, after testing the model with some training data, we found that the results were not satisfactory. As a result, we switched to using instance segmentation annotations, which led to improved model performance. However, during the data labeling process using Roboflow, we faced challenges as we couldn't classify the arm region into the boxer_1 and boxer_2 categories since hitting the opponent's arms doesn't score points in boxing. Thus, creating the custom dataset posed some difficulties for us.

While designing the post-processing algorithm, we initially considered using human pose estimation to calculate each player's score and hit rate. However, considering the complexity of the process, we opted for a geometric method based on assumptions. The establishment and testing of the post-processing algorithm took place after the progress report. The provided screenshot from the final output video demonstrates the players' scores and their hit rates.

Fig. 10. Illustration of showing the score and hit ratios

VII. Conclusion and Future Work

In terms of Algorithm Constraints and Future Work, we encountered some false detections and missed detections during model testing. To address this, adjusting the confidence

threshold for each bounding box proved to be a viable solution, improving the model's performance. Additionally, relying solely on the contact or overlap between the glove and boxer's bounding boxes for determining a valid hit is not a foolproof approach. A potential enhancement could involve extending the predicted mask information slightly and verifying if the glove area overlaps with the body area. This approach may yield more accurate results. Furthermore, an essential aspect of our future work involves developing a web page that enables users to upload local boxing videos and provides the option to download the analyzed videos, thus enhancing accessibility and convenience for users.

In this paper, we explored the application of artificial intelligence and computer vision techniques to detect and recognize the gloves and body parts of boxers, calculating their scores and hit rates in a boxing match. However, the study faced challenges due to limitations imposed by the camera angles of the matches, particularly in adopting instance segmentation methods. We utilized the YOLOv8 algorithm for detection and designed our own assumption-based geometric post-processing algorithm. The obtained results indicate that there is still potential for improvement in the model's performance. If we can obtain multi-angle footage and ensure both players are captured on opposite sides of the frame, the model's performance will likely be enhanced.

REFERENCES

[1] Kundu, R. (n.d.). Yolo algorithm for object detection explained. YOLO Algorithm for Object Detection Explained. https://www.v7labs.com/blog/yolo-object-detection

[2] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (1970, January 1). You only look once: Unified, real-time object detection. Page Redirection. https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/

[3]] Diop, C.-A., Pelloux, B., Yu, X., Yi, W.-J., & Saniie, J. Soccer Player Recognition using Artificial Intelligence and Computer Vision. Embedded Computing and Signal Processing Research Laboratory. Retrieved from http://ecasp.ece.iit.edu/

[4] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. Retrieved from https://arxiv.org/abs/1804.02767

[5] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Information retrieval evaluation. In Introduction to Information Retrieval (Chapter 8). Cambridge University Press.