Darts Detection

Asmaa Shaarawy
Computer Engineering Department
University of Ottawa
Cairo, Egypt
ashaa016@uottawa.ca

Mohamed Elnamoury
Computer Engineering Department
University of Ottawa
Cairo, Egypt
melna086@uottawa.ca

Nora Yehia Abdelhai
Computer Engineering Department
University of Ottawa
Cairo, Egypt
nabdo060@uottawa.ca

Salma Elgendy
Computer Engineering Department
University of Ottawa
Cairo, Egypt
selge047@uottawa.ca

Youtham Boulos
Computer Engineering Department
University of Ottawa
Cairo, Egypt
yboul052@uottawa.ca

***Abstract*—Score calculation in darts is very time consuming and requires some math calculations, also each player is responsible for memorizing his score each round. Motivated to develop an easy automatic solution, we present an approach that uses key point detection to predict the score from a single image taken from any angle with any lighting condition. We trained a model to detect these keypoints as objects representing four calibration points on the corners of the board, the center of the board and the tip of the arrow. We used a dataset that contains 500 images taken with different qualities and with different setings. The model used was " You Only Look Once" (YOLOV4) and we got "mean average precision" (mAP) of 85%. The codes [1] and dataset[2] are available.**

I. Introduction

Darts sports is widely spreading and everyone literally playing it whether it's professional or amateur level with the evolution of deep learning techniques specially in computer vision field it's an attractive field to apply computer vision learning based techniques to save time and effort and keep the flow of the game without interruption.

Our main goal in this work is to exploit the potential of deep learning in computer vision to perform automatic score keeping in steel-tip darts. Dart sport is played professionally recently. In professional dart playing, a lot of effort is required to calculate the score accurately and quickly. The model will be deployed on a camera in a stable position to calculate the score based on the position of the dart on the darts board. In the effort of developing accessible automated scoring system, we investigated the use of deep learning in the area of object detection and applying it in sports field.

Our research paper consists of the following: We searched in the literature for papers attempting to solve this problem, then we decided to go with YOLOv4. Moreover, we manually collected and annotated a 500-image dataset and designed a preprocessing pipeline before training the model on the dataset. At the end, we have implemented YOLOv4, the experiment and the work done will be discussed in details in the following sections.

[1]https://github.com/AsmaaSobhyy/Darts_score
[2]https://github.com/AsmaaSobhyy/Darts_dataset

II. Related Work

William et al. [1] tried to solve a similar problem which is darts detection and score calculation. The dataset Used was collected with a camera at a fixed location from the dart board. The images were then annotated with 4 calibration points and the tip of the arrow as shown in figure 1 to turn it into a key point detection problem. The model used is YOLOv4-tiny, and the validation method was percent correct score (PCS) which represent the number of times the model predicted the score correctly over the total number of trials. Afterwards, they tried different augmentation methods like flipping, rotations and Perspective Warping and tested the model to calculate which method improved the model more. The augmentation method with highest scores was dartboard Rotation with 36° which got PCS of 84%.

To the best of our knowledge, there are no publications other than this paper that tackle this problem. However, publications that have similar problems like [[2], [3]] used non-deep learning approaches for automatic scoring in range shooting.

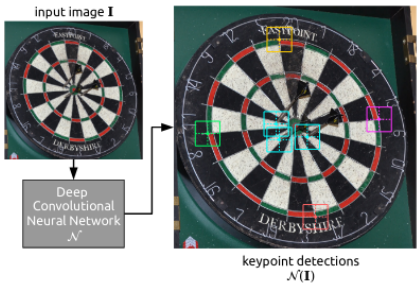


Fig. 1: annotation from Deep Darts [1]

III. Methodology

The approach used in this solution passes through multiple stages starting with the data collection, preprocessing and

annotation where we shot 500 images and annotated them manually and performed a preprocessing algorithm on them to prepare them to the final step which is the model training and validation.

A. Data Collection

The dataset consists of 500 photos captured with five different smartphone cameras in various locations and lighting conditions. To generalise the answer, the photos were acquired from various angles. One or two darts were thrown over the dart board in the photographs on many and haphazard positions.

B. Data Annotation

The images were manually annotated with LabelImg [8] which is an annotation tool written in python and its installation is simple and usually done via a command prompt/terminal. The annotations represents 6 different classes which are the four calibration points identifying the 4 directions of the board, the fifth class is the center of the board and the final class is the landing position of the arrow, each point is represented by it's X and y coordinates. With the knowledge of these points, the relative distance of the dart can then be calculated to calculate the final score.

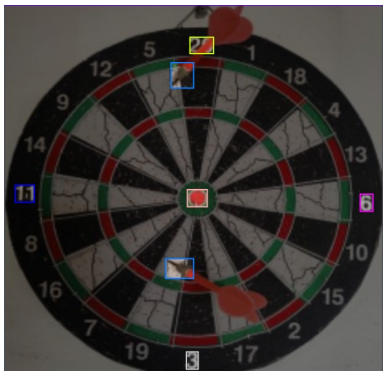


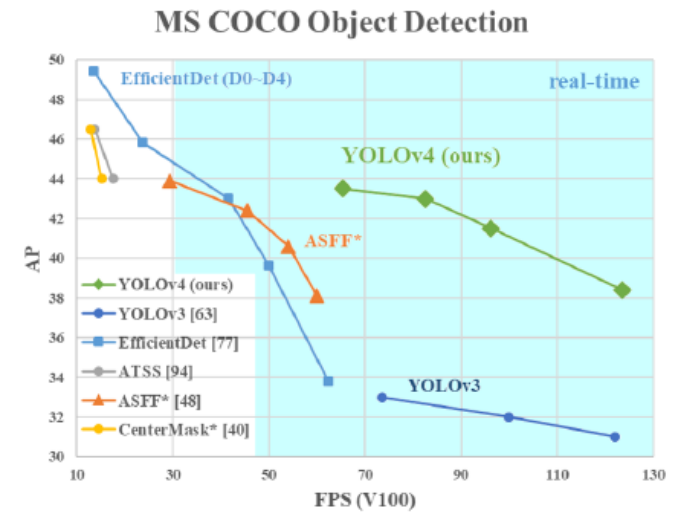
Fig. 2: annotation

C. Preprocessing

The images passed through a preprocessing pipeline which first cropped the image around the board to remove irrelevant features and remove any confusion for the model. This step was performed through an algorithm that used canny edge detection to locate the board then removed all black pixels which represents the padding around the board creating a mask that was then applied to the actual image to isolate the board. A median blur was then applied to remove the noise caused by different lighting angles, Finally, the images were resized into 416*416 to be compatible with the YoloV4 model. A sample output of the preprocessing step is shown in Fig. 3 where the image on the left represents the actual shot image and the image on the right is the preprocessed image.

Fig. 3: Preprocessing

D. Model



The model we adopted for this task is YOLOv4 [4], which is the go-to for object detection problems that require the most accurate and efficient solution. The 4th version of YOLO is significantly better than its predecessors in terms of accuracy, and in its possibility of training it on conventional GPUs. This was done through optimizing for parallel computations rather than the low computation volume theoretical indicator (BFLOP). Moreover, they tried and tested the state-of-the-art Bag-of-Freebies (methods increasing the accuracy by increasing the training cost only) and Bag-of-Specials (methods increasing the accuracy by slightly increasing inference costs) and used the ones that improved the detection accuracy. In addition to this, some methods were streamlined to perform better for single GPU training. In the figure above, a comparison of YOLOv4 with other state-of-the-art detectors. YOLOv4 shows better AP (Average Precision) and much faster performance (2x) on the Tesla V100 GPU.

IV. Experiments

This section describes the training methodology and results while applying different ablation experiments.

A. Implementation Details

Beginning with the preprocessing of the data mentioned before, first by using a canny edge detector to find the board and crop around it then removing noise using median filter

and resize to 416x416 to be applicable with YOLOv4. Finally annotating the data with 6 classes tip, center, cal1,cal2,cal3 and cal4. Because of choosing YOLOv4 as the training model we followed the pipeline described by Roboflow [5] to be able to apply transfer learning and train YOLO on our custom dataset. Building on YOLOv4 while detecting objects not included in YOLO's pretrained weights.

Data. To use the data with the model, we uploaded all the preprocessed images and annotations on Roboflow public datasets [**?**] which helps us in reviewing the data, check for mistakes, fix annotations and get various versions while exporting them in a compatible way with YOLO darknet.

Configuration. Used Google Colab notebooks for training as it has free access to GPUs besides its numerous installed packages especially CuDNN and OpenCV removing the headache of installation and configuration from us and saving a lot of time. To use Darknet framework, we had to install it then do some configurations using a makefile to fix dependencies shifted by colab. Then download YOLO's weights and set up the darts dataset. Finally, configure the parameters for training in a config file.

Training. Training was performed on the Nvidia K80 GPU of Colab using the Darknet framework [7]. Training was run for 1500 epochs based on the pre-trained weights that were trained on the Microsoft COCO dataset, with a base learning rate of 0.001, with a warm start of 1000 iterations, so the learning rate starts at 0 and reaches the base learning rate at epoch number 1000. Moreover, the learning rate was decreased by a factor of 10 at the 1300th and 1450th iteration. This adaptive learning rate dynamic was found to yield the best accuracy scores after some trials and testing with different configurations. Momentum and weight decay are also applied throughout the optimization process. Also, we used some image augmentation features provided by the Darknet framework such as pixel-level augmentations like random jitter, saturation, hue, and exposure and spatial-level augmentations like mosaic which was presented in the YOLOv4 paper.

B. Ablation Study

In this section we evaluate the effectiveness of the model with different options.

Augmentation. To investigate if we used different images with different shapes how the output will be, we used different augmentation strategies each on every image so for our training set of 400 images applied 3 techniques resulting in 1200 images in total. Using Roboflow augmentation first rotated by 90 degrees clockwise and anti clockwise Then retouched the image by changing saturation by decreasing it by 25% on some images and increasing it by 25% for others. Finally applied mosaic technique which works by combining 4 different images using random crop on each image, combining classes if they did not appear together much in the training set and varies the number of classes in the image. Examples in Fig. 4.

By training on 2000 iterations and batch size of 64 to test the augmentation we reached f1-score of 84% and mAP of 84.28%. The average precision for each class is reported in Fig. 5.

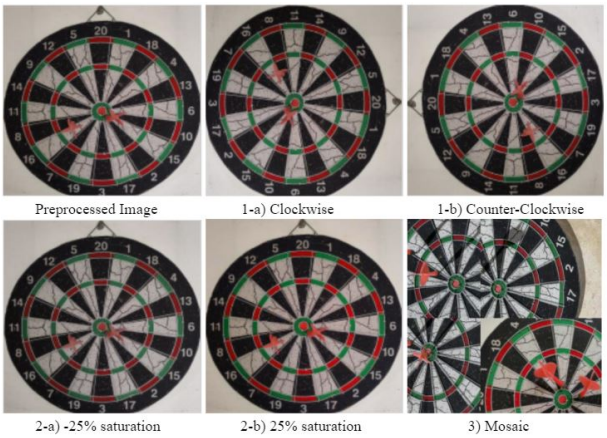


Fig. 4: Augmentation Strategy

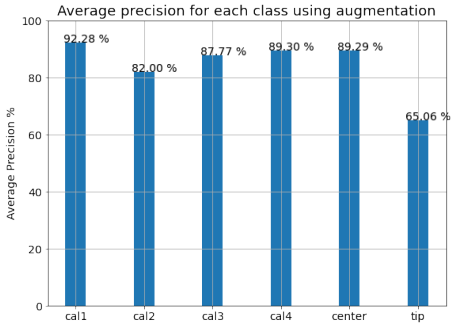
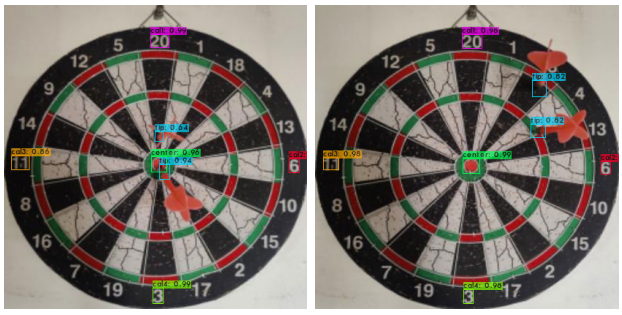


Fig. 5: Effect of augmentation on average precision for each class

Learning Rate. Updated the learning rate to 0.08 but unfortunately the average loss increased highly by time and the model did not converge.

C. Results

Figure 6 shows some predictions from unseen data from the test set after optimizing the model. The results yielded from testing the model on the test set are summarised in



(a) label 1 (b) label 2

Fig. 6: Sample predictions

the following table, where each label is listed against the average precision, number of true positives, and number of false positives.

Label	AP	TP	FP
cal1	96.81%	88	4
cal2	80.27%	76	13
cal3	94.00%	86	5
cal4	94.90%	87	4
center	89.00%	85	13
tip	56.12%	114	65

After training, the model achieved a value of precision of 0.84, a recall of 0.85, and F1-score of 0.84 for confidence threshold equal to 0.25. The overall mean average precision reached 85.18% IOU=0.5. We believe that the scores are acceptable for all the almost all the classes except the tip. The variation of the shape of the tip while varying the angles could be a reason why the model did not manage to capture its pattern. To elaborate, the bounding box of the tip sometimes contains the silver part of the tip, and other times, when the camera captures the top view of the dart while being perpendicular on the board, it only shows the upper coloured part (its flight). This could be addressed by considering this variation while preparing the dataset by adequately representing each shape in a balanced way.

V. Conclusion

In this paper, we presented two main contributions: a manually annotated dataset of a darts' board with bounding boxes of the darts' locations and calibartion points. In addition to this, we optimized a deep learning model to predict the positions of a dart on a traditional darts' board. Our solution takes advantage of the prominent YOLOv4 architecture and pre-trained weights to solve this problem using object detection. The output showed promising results: our model achieved 85.18% average precision on a small dataset. A drawback of our model was found to be the accuracy of the tip class, which could be improved by collecting a bigger and a more balanced dataset, showing the different tip from different angles.

The score calculation has been left for future work due to lack of time. With the knowledge of the four calibration points and the center of the circle we would calculate homogeneous coordinates of the darts into the dart board plane via a direct linear transform algorithm [6]. After score calculation the model needs to be evaluated on the scores predicted correctly to evaluate the whole pipeline. An additional task would be to increase the dataset as the model was trained on only 500 images and the tip of the arrow class was not represented to generalize all the angles for the arrow so we would take this issue into consideration while collecting the new dataset. We would also improve the model performance by training the model on more epoch and better tune the hyperparameters.

References

[1] William McNally, Pascale Walters, Kanav Vats, Alexander Wong and John McPhee. DeepDarts: Modeling Keypoints as Objects for Automatic Scorekeeping in Darts using a Single Camera,2021.

[2] Faizan Ali and Atif Bin Mansoor. Computer vision based automatic scoring of shooting targets. In INMIC, 2008.

[3] Penghua Ding, Xuewu Zhang, Xinnan Fan, and Qianqian Cheng. Design of automatic target-scoring system of shooting game based on computer vision. In ICAL, 2009.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong- Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[5] J. Solawetz, "How to Train YOLOv4 on a Custom Dataset", Roboflow Blog, 2020. [Online]. Available: https://blog.roboflow.com/training-yolov4-on-a-custom-dataset/.

[6] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[7] AlexeyAB, "darknet: YOLOv4 Neural Networks for Object Detection", GitHub, 2021. [Online]. Available: https://github.com/AlexeyAB/darknet.

[8] tzutalin, "labelImg", GitHub, 2021. [Online]. Available: https://github.com/tzutalin/labelImg.