International Journal of Scientific Advances
ISSN: 2708-7972
Volume: 3 | Issue: 5 | Sep - Oct 2022 Available Online: www.ijscia.com
DOI: 10.51542/ijscia.v3i5.22

Eye Pupil Localisation and Labeling Using a Small Size Database and YOLOv4 Object Detection Algorithm

Varuzhan H. Baghdasaryan

National Polytechnic University of Armenia

*Corresponding author details: Varuzhan H. Baghdasaryan; varuzh2014@gmail.com

ABSTRACT
Eye-related research has shown that eye gaze data are very important for applications that are essential to human daily life. Eye gaze data has been used in research and systems for eye movements, eye tracking and eye gaze tracking. Eye pupil localization, labelling and tracking are challenging problems in computer science. This article discusses and explores that problem. The YOLOv4 ("You only look once") object detection algorithm which is an evolution of the YOLOv3 model has been evaluated in a tiny database consisting of 103 eye images. The YOLOv4 algorithm was created by Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao [3]. It is twice as fast as EfficientDet with comparable performance. The main purpose of this article is to test the YOLOv4 algorithm, to find out its effectiveness in process of localization and labelling of eye pupils and find out (determine) the effectiveness of the algorithm when training with a tiny database and with a relatively small number of iterations.

Keywords: eye pupil localization; eye pupil labelling; neural network; YOLOv4; tiny database; a small number of iterations.

INTRODUCTION
Real-time detection and tracking of the eye pupil are an active area of research in computer vision. Localization, labelling and tracking of the eye pupil can be useful in face alignment, gaze tracking, lie detecting and human-computer interaction.

Such systems solve important problems such as:
- detect the drop point of visual attention (known as point of gaze (PoG)),
- detect the direction of visual attention (known as the line of sight (LoS)).

Also, these systems can be categorized as:
- diagnostic, which evaluate visual and attentional processes,
- interactive, which use eye gaze data to interact with the user based on eye movements.

Both categories mentioned above use eye gaze or pupil detection and tracking technologies to extract or generate the necessary data from gaze or pupil location information.

For eye gaze tracking traditional techniques require physical equipment such as contact lenses, electrodes, and head-mounted devices or digital cameras. The above-mentioned physical devices will not be used for this experiment, as the experiment will be performed only to test whether the YOLOv4 algorithm will detect and label eye pupils in the images.

According to Ildar Rakhmatulina's and Andrew T. Duchowskib's paper [1], there are a large number of different neural networks for eye pupil and gaze tracking tasks. However, speed and high accuracy are required in these tasks and because of that reason only a few of these neural networks are suitable but the main problem of these deep neural networks is the need to use a large number of images to train the network. One of that suitable neural networks is the YOLOv4 which is one of the best implementations of the FCNN neural network in the world. The FCNN network of YOLO has 24 convolutional layers, followed by 2 fully connected layers and instead of the original modules used by GoogLeNet, just a (1×1) reduction layer was used, followed by a (3×3) convolution layer [2].

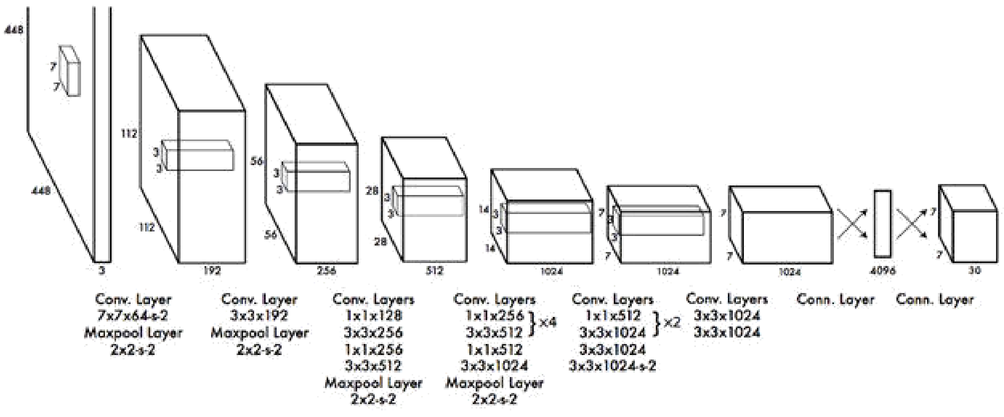

FIGURE 1: main layers of YOLOv4's neural network structure [2].

As reported by YOLO's paper [2], YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers (see figure 1). It divides the image into an (S×S) grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an (S×S×(B∗5+C)) tensor. Compared to other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in an image, YOLO architecture is more like FCNN (fully convolutional neural network) and passes the image (nxn) once through the FCNN and output is (mxm) prediction. This architecture is splitting the input image in mxm grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes. Note that the bounding box is more likely to be larger than the grid itself.

RELATED WORK

There is a lot of research about eye gaze tracking, eye localization and tracking, which are using different approaches and technologies to achieve the destination results. Many of them use YOLO, its modified variants or newer versions.

According to the paper [5] of Jiali Cui, Fuqiang Chen, Duo Shi and Liqiang Liu, YOLO (v1) trained 50000 iterations to detect 3 labels ("left eye", "right eye" and "eyes") and because of a small number of images (1000 images for training and 500 images for testing) in the database, the images are augmented by small angles and translations. After training step YOLO achieved an accuracy of 47.4% for the monocular model and 63.5% for the binocular model [5].

According to the paper [6] of Niharika Kumari, Verena Ruf, Sergey Mukhametov, Albrecht Schmidt, Jochen Kuhn and Stefan Küchemann, YOLOv4 model was fine-tuned up to 34000 iterations (used pre-trained model learned on the "MS COCO" dataset [7] and had a good mAP score of 75%)). The algorithm has trained on a dataset containing 1000 images (each image size is 416 × 416 px) and annotated using the "LabelImg" software (annotated classes are the arrow, the graph, the lens, the light source, the paper, the picture, the power supply, the questions, the rail, the text and others) [6]. The experiment of processing 25 frames per second gave a 0.89 macro-precision score and 0.83 recall score (recall is the ratio of the number of true positives to the total number of actual objects. For example, if the model correctly detects 85 dogs in an image, and there are actually 100 dogs in the image, the recall is 85 per cent.) [6].

Considering the results of the articles mentioned above it's easy to notice that in the first case iteration number is large and the database is augmented to increase the size of the database. In the second case again iterations number and database are relatively large and used fine-tune technique to avoid training from scratch.

DATABASE

The images were collected from the web and annotated using the "LabelImg" tool which is an open-source data annotation tool. During annotation, the eye pupil was annotated with a "pupil" tag and supports YOLO and CreateML formats. The database for the experiment contains annotated images of eyes of different colours. The images selection process was absolutely random. In some images eye pupil is in the centre of the eye and in some images it is in one of the corners of the eye.

The tiny database is consist of 103 RGB (3 layers) jpg, jpeg and png images up to a maximum size of 512x512. The images contain only eyes, where it will be possible to annotate the pupil of the eye. An example of a selected image is shown in Figure 2.

There is no need to do data basic augmentation (image flip, rotate, crop, brightness adjustment or noise adding) to get new images and increase database size because as mentioned in YOLOv4's paper [3], the algorithm already does data augmentation automatically at runtime while training. Mosaic data augmentation [3] combines 4 training images into one in certain ratios. This allows the model to detect objects outside their normal context and learn how to identify objects at a smaller scale than normal. It also is useful in training to significantly reduce the need for a large mini-batch size.

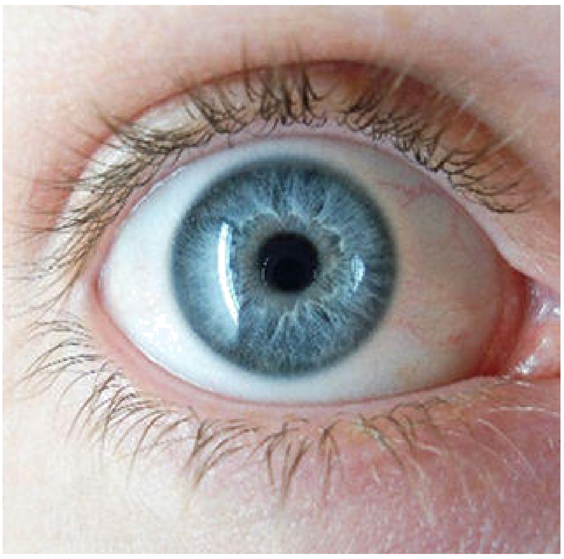

FIGURE 2: example of a collected image.

Annotation parameters are:
- class id = label index of the class to be annotated,
- Xo = X coordinate of the bounding box's centre,
- Yo = Y coordinate of the bounding box's centre,
- W = Width of the bounding box,
- H = Height of the bounding box,
- X = Width of the image,
- Y = Height of the image.

For multiple objects in the same image, this annotation is saved line-by-line for each object.

METHODS

YOLO model has several advantages over classifier-based systems. It looks at the whole image at test time so its predictions are informed by the global context in the image. It also makes predictions with a single network evaluation unlike systems like R-CNN which require thousands for a single image. This makes it extremely fast, more than 1000x faster than R-CNN and 100x faster than Fast R-CNN.

According to paper [3] YOLOv4 achieves state-of-the-art results at a real-time speed on the "MS COCO" [7] database with 43.5 % AP value running at 65 FPS on a Tesla V100, beating the fastest and most accurate detectors in terms of both speed and accuracy. When compared with YOLOv3, the AP and FPS have increased by 10 per cent and 12 per cent, respectively.

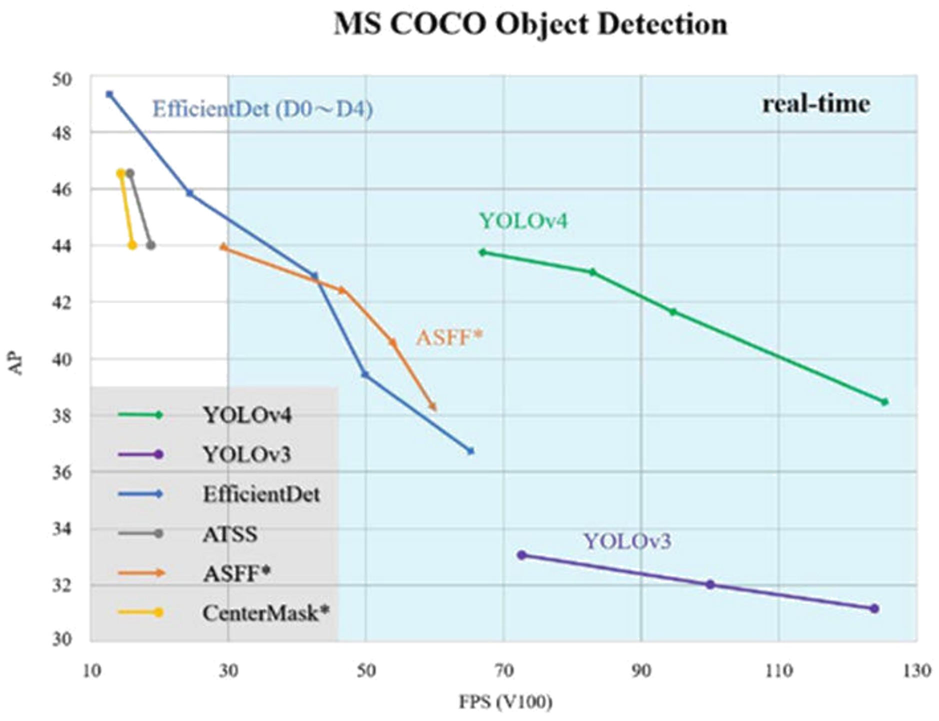

FIGURE 3: comparison of state-of-the-art object detectors
(models that fall in the light-blue area are considered real-time object detectors (+30 FPS)) [3].

Figure 3 shows the dependence of average precision (AP) on the number of frames per second (FPS) for the most popular and state-of-the-art object detectors (neural networks). From Figure 3 it becomes clear that EfficientDet D4-D3 achieves better AP than YOLOv4 models, but they run at speed of smaller than 30 FPS on a V100 GPU. On the other hand, YOLO is able to run at a much higher speed greater than 60 FPS with very good accuracy.

TRAINING

As mentioned above YOLOv4 algorithm will be used for the experiment and the modified configuration of the algorithm are as follows:

- batch=1,
- subdivisions=1,
- network size width=416,
- network size height=416,
- max_batches=6000,
- steps=4800,5400,
- filters=(1(classes) + 5) x3,
- classes=1.

max_batches parameter is 6000 because it is must be not less than the number of training images and not less than 6000 ((classes*2000).

The database division ratio for the training and testing stages is accordingly 90% and 10%. Batch and subdivisions parameters are 1 because it is preferable to process the images one by one to learn all the useful features of the eye pupil by the neural network and get the preferable result. All other configurations have remained the same. It should be noted that the experiment will be performed without using fine-tune or transfer-learning techniques and the model will learn eye pupil features from scratch.

RESULTS

To evaluate training results, the mean average precision (mAP) [4] was used. The mAP compares the ground-truth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections.

The training process took 11.53 hours (because of using only CPU) and after every 600 steps, the mAP was fixed. On 1200 step it reached 1.021573 mAP then 1.474165 and so on. Based on the collected results, diagram was made shown in Figure 4. As can be deduced from the figure 4 diagram, there is a positive increase in learning, but the final result is quite low.

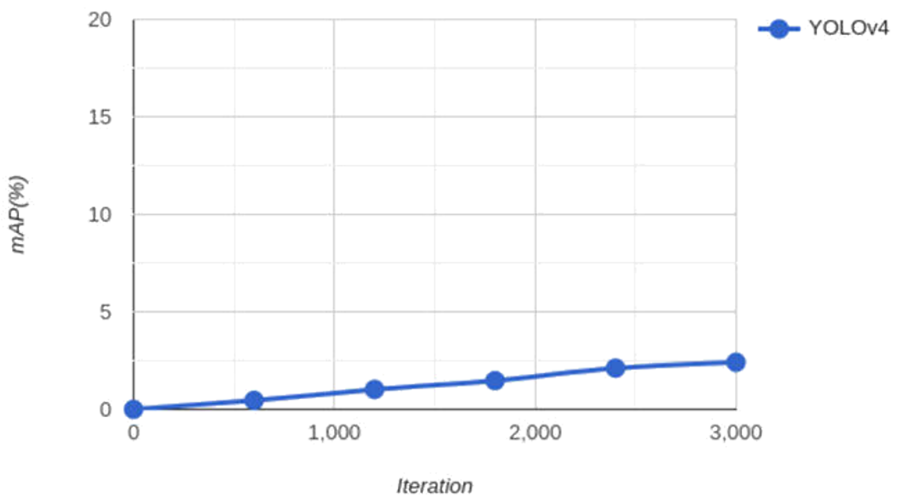

FIGURE 4: experiment mAP chart.

After the 3190 iterations, YOLOv4 result is 2.413619 mean average precision (mAP). Because of runtime Mosaic data augmentation technique [3], 6382 images and augmented sub-images were processed. It is clear from the experiment that YOLOv4 is quite fast both in the training and testing stages, but the expected result was not so good because of some reasons (the tiny database and the small number of training steps (iterations)). For trained model single image test neural network achieved 42% probability for eye pupil but a bounded box of the detected pupil is not very accurate although it completely encloses the eye pupil (see figure 5).

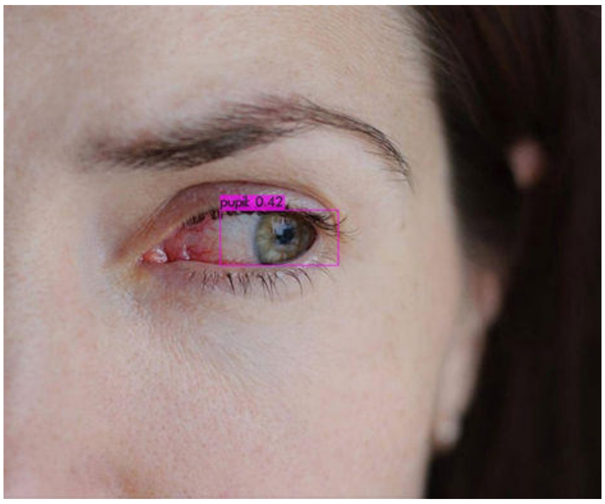

FIGURE 5: test example.

CONCLUSIONS

In this article, we have discussed YOLO's use in gaze tracking, eye pupil localization and labelling problems. The final results can be considered correct for the eye pupil tracking problem too because the YOLO algorithm can perform real-time detections also by detecting objects in the video frames.

In a summary, the YOLOv4 was trained on a small database consisting of 103 images. The database is quite small, and the number of training steps is not enough. Given these circumstances, it can be understood that the algorithm nevertheless gave sufficient results to believe that the algorithm will be quite effective if it is trained with a sufficient amount of data and sufficient training steps.

The conclusion is that the YOLOv4 algorithm is not applicable for the solving of gaze tracking, eye pupil localization, labelling and tracking problems if it trains with small size database and small training steps.

REFERENCES

[1] Ildar Rakhmatulina, Andrew T. Duchowskib. (2020). Deep Neural Networks for Low-Cost Eye Tracking. 24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Procedia Computer Science 176, 685–694.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. (09.05.2016). You Only Look Once: Unified, Real-Time Object Detection. University of Washington, Allen Institute for AI, Facebook AI Research. arXiv preprint arXiv:1506.02640v5.

[3] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. (23.04.2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv: 2004.10934v1.

[4] Paul Henderson, Vittorio Ferrari. (16.03.2017). End-to-end training of object class detectors for mean average precision. University of Edinburgh. arXiv preprint arXiv:1607.03476v2.

[5] Jiali Cui, Fuqiang Chen, Duo Shi, Liqiang Liu. (2019). Eye Detection with Faster R-CNN, International Conference on Advances in Computer Technology, Information Science and Communications - CTISC, 111-116, Xiamen, China.

[6] Niharika Kumari, Verena Ruf, Sergey Mukhametov, Albrecht Schmidt, Jochen Kuhn and Stefan Küchemann. (18.11.2021). Mobile Eye-Tracking Data Analysis Using Object Detection via YOLO v4, Sensors 2021, 21(issue. 22), 7668.

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár. (06.09.14-12.09.2014). Microsoft COCO: Common Objects in Context; Proceedings of the 13th European Conference on Computer Vision; Zurich, Switzerland, pp 740–755.