fastgate

R.J. Bouwmeester
Delft University of Technology, 2629 HS, the Netherlands
https://github.com/gemenerik/fastgate

ABSTRACT

In this paper, a drone race gate detection method using *YOLOv3* and *YOLOv3-tiny* is proposed. Both manage close to perfect classifier accuracy, for an IoU threshold below 0.6. *YOLOv3-tiny* provides satisfactory performance for implementation on powerful racing drones .

1 INTRODUCTION

The goal of this paper is to design an automatic gate detection method, using a data set of images, masks, and gate corner coordinates, from an autonomous drone race, that can ultimately be implemented and run in real-time on a racing drone. Many architectures can be suited to detect these gates. The rectangular-shaped, checkerboard-textured, blue-colored, widely-spaced gates allow implementations of SIFT [1], SURF [2], Haar-like features, color/edge/corner detection, optical flow (although gate displacement between images is relatively large), or more novel, specialized architectures such as the snake-gate algorithm [3]. With better embedded GPU hardware, specifically designed for high performance on tensor operations with low power usage and at low weight [4, 5], and consistent performance improvements in deep learning, neural networks have found their way to drones [6]. These neural networks are superior in accuracy to previously mentioned architectures for multi-class tasks, but can also be implemented for single-class ones. This accuracy is traded off with a decrease in performance. In this paper a lightweight neural network architecture to detect drone racing gates is implemented and evaluated.

Object detection networks detect objects of certain classes within an image. Detected objects are marked by means of a bounding box. Coincidentally, our gates are rectangular, which could make these otherwise quite obnoxious, imprecise bounding boxes describe the location of the gates quite accurately (except for roll or extreme pitch angles). Gate corner coordinates are provided to generate labels from. Several papers cover real-time implementations of such networks [7].

An alternative is semantic segmentation, which segments all pixels of an image into classes. Would require instance segmentation, which labels all instances as separate objects, especially given overlapping gates. Masks are provided to generate training labels from. Could more accurately describe the location of gates when compared to object detection networks, for a significant drop in real-time performance [8].

2 METHOD

In this paper, we propose a gate detection method using the (real-time) object detector network *YOLOv3* [9]. The third version in the series *YOLO* [10], *YOLOv2/YOLO9000* [11]. Managing 45 fps with 28.2 MAP on the COCO dataset [12, 13]. It comes with a 'tiny' version, that trades reduced accuracy for a significant boost in FPS. Many well-documented PyTorch implementations are available on GitHub.

308 images are available to train, validate and test the network. 28 test images (see GitHub), with 64 gates in them, were selected for difficult, important, extreme cases (close/far/overlapping/cut-off gates, high rotation/pitch angles, motion blur). From the remaining images, 32 were randomly picked for validation. The remaining 248 images are used for training (approximate 80-10-10 split). An option to randomly select training/testing/validation data is available, to evaluate the effects of the hand-picked initialization. Note that a lot less data is available than in a 'classical' object detection problem. However, satisfactory performance can be achieved as the network has to identify only one class ('gate') and the *YOLO* network has data augmentation methods built in.

Image labels describe the bounding boxes around the gates. These are not rotated, so unless an image is perfectly head-on to the center of a gate with level heading, the bounding box labels are imperfect. It was decided to run these from the extremities of the gate, and thus potentially include parts of the environment into the box, so that all information of the gate can be learned.

For training, a batch size of 8 is used. The *YOLO* network uses the *Adam* optimizer, so the learning rate is fixed at the default 0.001, and no learning rate scheduling is required. Training is continued until no significant improvement is made in 50 epochs. Validation data is used to prevent overfitting. Hyperparameters are determined using an empirical approach. Validation and test sets are somewhat on the small side (Pareto principle, 80/20 train/val is conventional), to ensure sufficient training data. Further tweaking could improve results (see section 5).

3 RESULTS

In figure 1 and 2 the training loss and validation TPR are shown for training the *YOLOv3-tiny* network. In figure 3 and 4 receiver operating characteristic (ROC) curves for *YOLOv3* and *YOLOv3-tiny* are plotted. The image shows the relation between the false positive rate (FPR) and true positive rate

(TPR), for varying intersection-over-union (IoU) thresholds (0.4 - 0.8, step 0.1), for varying confidence thresholds (0.05 - 0.95, step 0.05). For lower IoU threshold results were the same as for an IoU threshold of 0.4, higher IoU threshold resulted in an extreme shift from true positives to false positives. The closest to optimal performance (TPR = 1, FPR = 0) is at the high confidence threshold around 0.85, 0.9, 0.95, for all IoU threshold values. *YOLOv3-tiny* is only marginally less accurate.

Note that the initialization might affect what parameters are optimal. Using random training/testing/validation images resulted in higher accuracy (perfect accuracy for some IoU and confidence threshold values) for both networks.

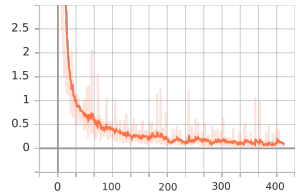

Figure 1: Training loss per epoch (*YOLOv3-tiny*).

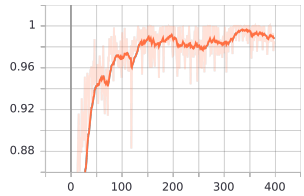

Figure 2: Validation TPR per epoch (*YOLOv3-tiny*)

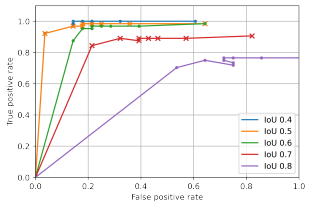

Figure 3: ROC for *YOLOv3*, for varying IoU and confidence threshold.

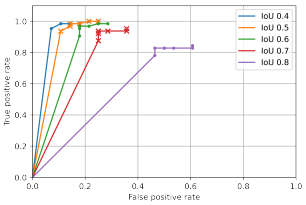

Figure 4: ROC for *YOLOv3-tiny*, for varying IoU and confidence threshold.

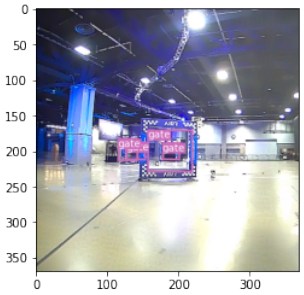

Figure 5: False positive / double detection on leftmost gate

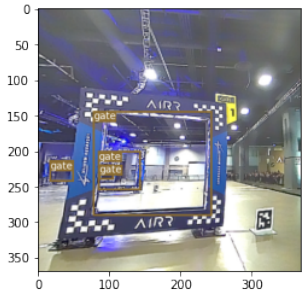

Figure 6: Four gate detection

4 Performance

Inference times are marked per tested image. Inference is done on Google Colab (as is training). Tests have been performed on three virtual machines, running an NVIDIA Tesla K80, NVIDIA Tesla P100, and an NVIDIA Tesla T4, respectively. Resulting inference FPS can be found in table 1. Currently, the algorithm is not suitable for CPU-inference (CPU-training is available).

	Tesla K80 11441MiB	Tesla P100 16280 MiB	Tesla T4 15079 MiB
'Regular'	15 FPS	43 FPS	40 FPS
'Tiny'	42 FPS	57 FPS	58 FPS

Table 1: FPS per GPU per model version

Despite the differences between the Tesla P100 and Tesla T4 chips, their performance is very similar. The difference between the regular and tiny models is clear, but not as significant as for the significantly worse performing Tesla K80. State of the art embedded GPU systems such as the NVIDIA Jetson AGX Xavier have approximately 1.4 times faster inference than the NVIDIA GTX 1070 [14] (and approx. 14 times inference efficiency in FPS/W), which in turn is approx. 2 times faster than the Tesla K80 [15]. These values are based on benchmarks by NVIDIA and GPU users, with little information on the specifics of their benchmarks, and as such should be taken with a pinch of salt. If anything, they show that state of the art embedded systems can perform at least as good as the Tesla K80, and can probably outperform it.

5 Discussion

Both networks show high accuracy, even for extreme cases. *YOLOv3-tiny* shows promising results for implementation on a drone, while sacrificing only a small amount of accuracy with regards to the regular version, which may be because regular *YOLOv3* is overdesigned for detecting just one class. Most inaccuracies were double detections of a single, far away gate. For set IoU, there is little trade-off between true and false positive rate. To get a proper evaluation of the performance achieved, it should be compared with other algorithms, such as image segmentation. It could be interesting to see how well the algorithm performs in a environment different from the train environment.

Accuracy highly depends on set IoU threshold. One could theoretically get away with relatively low IoU in this particular application. The most important is that gate bounding boxes scale from large to small in order of gates close to far away.

Some of the false positives resulted from a single gate being detected as two gates. The occurrence of such double detections can be reduced by tweaking the *non-maximum suppression* (*NMS*) threshold.

Performance could be further improved given more data. Problematic is that it requires human labelling. Should investigate unsupervised object detection learning [16, 17]. Additionally an ensemble of methods should be considered, such as Haar-like features to guide the detection. Currently, there is no use of temporal information; if there was a gate approximately here last frame, it should be in the same area in the next.

During finalization of this report, *YOLOv4* [18] has been released, promising improved speed and accuracy.

REFERENCES

[1] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[3] Shuo Li, Erik van der Horst, Philipp Duernay, Christophe De Wagter, and Guido CHE de Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-gram drone. *arXiv preprint arXiv:1905.10110*, 2019.

[4] Nathan Otterness, Ming Yang, Sarah Rust, Eunbyung Park, James H Anderson, F Donelson Smith, Alex Berg, and Shige Wang. An evaluation of the nvidia tx1 for supporting real-time computer-vision workloads. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 353–364. IEEE, 2017.

[5] Shan Ullah and Deok-Hwan Kim. Benchmarking jetson platform for 3d point-cloud and hyper-spectral image classification. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 477–482. IEEE, 2020.

[6] Karim Amer, Mohamed Samy, Mahmoud Shaker, and Mohamed ElHelw. Deep convolutional neural network-based autonomous drone navigation. *arXiv preprint arXiv:1905.01657*, 2019.

[7] Real-time object detection. https://paperswithcode.com/task/real-time-object-detection.

[8] Real-time semantic segmentation. https://paperswithcode.com/task/real-time-semantic-segmentation.

[9] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[11] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[12] Real-time object detection on pascal voc 2007. https://paperswithcode.com/sota/real-time-object-detection-on-coco.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[14] Jetson agx xavier and the new era of autonomous machines. https://github.com/dusty-nv/jetson-presentations/raw/master/20181004_Jetson_AGX_Xavier_New_Era_Autonomous_Machines.pdf.

[15] Is free kaggle k80 gpu better than gtx-1070 maxq? https://medium.com/@saj1919/is-free-kaggle-k80-gpu-is-better-than-gtx-107

[16] Ioana Croitoru, Simion-Vlad Bogolin, and Marius Leordeanu. Unsupervised learning of foreground object detection. *arXiv preprint arXiv:1808.04593*, 2018.

[17] Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3412–3420, 2019.

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