Detection of Airplanes on the Ground Using YOLO Neural Network

Volodymyr Kharchenko, Iurii Chyrka
Research department
National aviation university
Kyiv, Ukraine
kharch@nau.edu.ua, yurasyk88@gmail.com

Abstract— **The presented paper benchmarks the performance of state-of-the-art methods of objects detection in the particular case of airplanes on the ground identification detection in aerial images taken from unmanned aerial vehicles or satellites. There were tested two popular single-stage neural networks YOLO v.3 and Tiny YOLO v.3 based on the "You Only Look Once" approach. The considered artificial neural network architectures for objects detection has been trained and applied over the specifically created image database. Experimental verification proves their high detection ability, location precision and real-time processing speed using modern graphics processing unit. That approach can be easily applied for detection of many different classes of ground objects.**

Index Terms— **Convolutional neural network; object detection; real-time processing; unmanned aerial vehicle**

I. Introduction

Unmanned aerial vehicles (UAVs) play an important role in many fields of human activity. UAVs can be either remote-controlled aircraft (by a pilot at a ground) or fly autonomously driven by some flight program or controlled by a higher-level control system.

UAV technology has many advantages that include low cost, small size, safety, ecological operation, and most of all, the fast and on-demand acquisition of images [1]. Most of all, they are an effective and powerful method of capturing high-resolution remote sensing (RS) images [1-3]. The recent rapid advances of UAV technology led to many studies proposing many novel ways for UAV applications and image analysis in relation to corresponding areas including infrastructure surveillance, fire detection, vegetation monitoring, marine surface monitoring, nature changes observation, disasters management, traffic monitoring etc [3–7]. Most of them can be generally described as detection, recognition, and tracking of various objects of interest.

In this paper, we consider the usage of modern convolutional neural network (CNN) architectures for the detection and classification of objects during autonomous UAV operations in civil fields. This paper shows the example of successful application of "You Only Look Once" YOLO and Tiny YOLO architectures on real-time airplanes detection on the ground from the video feed during UAV operation test.

II. Review of Existing Methods

Object detection is one of the fundamental tasks in computer vision and refers to the determination of the presence or absence of specific features in the image [8]. When features are detected, an object can be further classified as belonging to one of a pre-defined set of classes and then the bounding box around that object or object central point is predicted.

There are roughly three types of object detectors: classic, two-stage, and one-stage ones. Classic detectors operate in the sliding window, in which a classifier is applied every time over a predefined image grid. The most known of them are CNN for digits recognition, proposed by LeCun et al [9]; Viola and Jones face detector [10] and the histogram of oriented gradients (HOG) method for pedestrian detection [11]. Later with the development of deep learning, they have been outperformed by two-stage detectors, described next.

More recent approaches use region proposal methods to first generate a sparse set of candidate proposals that should contain all objects while filtering out the majority of negative locations in an image and then run a classifier on these proposals in order to separate them into foreground classes/background. Such two-stage detection is the dominant paradigm nowadays. Region-based convolutional neural networks (R-CNN) have grown over last years with several improvements [12-14] and numerous extensions [15-17].

Yet another modern approach to object detection assumes a single detection stage that is closer to the natural way of objects detection by humans. The three main methods can be mentioned here: SSD [18], YOLO [19, 20] and RetinaNet [21]. Roughly, their effectiveness grows in the same order. SSD has about 10-20% less average precision. YOLO (version 3) has almost the same precision as two-stage detectors and RetinaNet so far is the most effective state of the art object detector. However, YOLO v.3 is the fastest among them and still has acceptable precision. That is an important factor for autonomous UAVs as it will be described next.

Coming back to UAVs operations, the self-controlled flying process can be divided into three stages. First, raw data is recorded during flight via sensors which an UAV is equipped with. Then the real-time data processing is performed by the onboard system. The final stage supposes autonomous decision-making based on the processed data. All stages should be conducted in a few milliseconds. The crucial part here is the second stage, where the onboard system is supposed to detect and classify observed objects in real time.

In this situation, the solution comes with usage of single-stage detectors based on CNN. It worth to mention, that one of the superior features of CNNs is their parallel nature that perfectly fits the architecture of a graphical processing unit (GPU), which consists of thousands of cores designed to handle multiple tasks simultaneously. Their combination allows dramatic reduction of computation time while maintaining superb precision.

In view of recent advances in GPU hardware development, price and size of the GPU units have been reduced considerably. This allows designing an integrated software-hardware module capable of real-time processing, which is light and inexpensive enough to be mounted on an UAV. However, before such incorporation, CNN needs to be trained and tested on the more powerful equipment.

III. YOLO Neural Network Architecture

The CNN algorithm considered in this paper has been built on an open-source platform complied from the Darknet framework written in C and CUDA [22] that has an implementation of the YOLO architecture of the 3rd version [20] and it's simplified version from the original paper [19] with the enhanced classifier Tiny YOLO v.3.

The main advantage of YOLO as a single-stage approach is that the single neural network evaluates the whole image. It makes all predictions based on the actual image, not the proposed regions as it goes for two-stage methods. The input image is represented as a tensor of size n × m × 3, where n and m represent width and height in pixel and 3 denotes three color channels). All input images of various sizes are automatically resized to 416 × 416; therefore, we used a 416 × 416 × 3 input tensor every time for training. Actually, that size can vary in certain range but those particular values give the output feature map with an odd number of cells with a single central cell as it will be described later.

The network uses the backbone Darknet-53 that is a 53-layer feature extracting deep neural network. Its structure is shown in Table 1. Faster or Tiny YOLO feature extractor architecture has a simplified structure with much less amount of layers, as it is shown in Table 2.

The procedure supposes that the picture is divided into the grid of equal cells of some size. For 416 × 416 input image, it is a grid of 13 × 13 cells. Each cell is responsible for prediction of some amount of bounding boxes that covers this cell.

Each prediction of a bounding box contains the following information: coordinates of the bounding box, width, and height, that are calculated through predictions from cell coordinates and bounding box prior width and height as

$$b_x = \sigma(t_x) + c_x, \quad b_y = \sigma(t_y) + c_y, \qquad (1)$$
$$b_w = p_w e^{t_w}, \qquad b_h = p_h e^{t_h}.$$

TABLE I. Darknet-53 structure

	Type	Filters	Size	Output
	Convolutional	32	3×3	416×416
	Convolutional	64	3×3/2	208×208
1×	Convolutional	32	1×1	
	Convolutional	64	3×3	
	Residual			208×208
	Convolutional	128	3×3/2	104×104
2×	Convolutional	64	1×1	
	Convolutional	128	3×3	
	Residual			104×104
	Convolutional	256	3×3/2	52×52
8×	Convolutional	128	1×1	
	Convolutional	256	3×3	
	Residual			52×52
	Convolutional	512	3×3/2	26×26
8×	Convolutional	256	1×1	
	Convolutional	512	3×3	
	Residual			26×26
	Convolutional	1024	3×3/2	13×13
4×	Convolutional	512	1×1	
	Convolutional	1024	3×3	
	Residual			13×13

TABLE II. Tiny YOLO Feature Extractor Structure

Type	Filters	Size	Output
Convolutional	16	3×3/2	208×208
Convolutional	32	3×3/2	104×104
Convolutional	64	3×3/2	52×52
Convolutional	128	3×3/2	26×26
Convolutional	256	3×3/2	13×13
Convolutional	512	3×3/1	13×13

Besides coordinates, it predicts the probability that the bounding box contains the object and *C* probabilities of belonging to each of predefined *C* object classes. We suppose that each cell can predict up to 3 bounding boxes. Therefore, the final output tensor is 13 × 13 × [3* (5 + *C*)].

Additionally, YOLO v.3 predicts boxes at 3 different scales for better detection of small objects. Tiny YOLO v.3 does the same, but for two scales only. Finally, duplicate detections are eliminated by non-maximal suppression.

IV. CNN Training

Although there are a few YOLO networks trained on several known datasets, the CNN still needs to be trained for better precision to work with objects specific to our particular tasks. A few task-specific parameters such as batch size, learning rate, decay, iteration number, and detection thresholds should be additionally tuned for the best performance. The number of epochs required for training was determined empirically. "Epoch" means a single run through the entire data set during training of a neural network. For batch training, the input data are fed to the network within batches that includes a fixed number of samples (called as a "batch size") and weights are updated every time it passes through the learning algorithm. "Learning rate" is a constant used to control the rate of learning or gradient descent. "Decay" refers to the ratio for decreasing learning rate at a certain number of epochs.

Our networks (full and tiny) have been trained for only a single object class i.e. "airplane" ($C = 1$). That gives us a size of the first scale output tensor as $13 \times 13 \times 18$.

Preliminary analysis has shown that images with airplanes taken by UAVs differ significantly from the images available at common databases with a lot of objects. As an example, many photos from the PASCAL VOC database are taken from the frontal or side view, while the images from UAV mostly from the top-down view. That promises a significant gain in performance between a network trained on those databases and the one trained on a custom database containing satellite and UAV-acquired images. For the data augmentation during network training, there were used a few transformations: random scaling up to 60% change of the original image size, changes of hue, saturation, and exposure of the image were randomized in the range 0.1, 1.5, 1.5 respectively in the hue saturation value (HSV) model.

The custom database of images with objects of the class "airplanes" was created by capturing satellite images of airplanes grounded on civil and military airfields mostly across Europe from Google Maps. Example pictures from our database are shown in the Fig. 1 (pictures are taken after processing with bounding boxes painted around airplanes). Additionally, there were downloaded several photos were taken from the air (mostly with not right angles, that better suits to actual operating conditions of low-altitude flying UAV) by browsing them on the Internet. Images from Internet were used due to current restrictions on operating UAVs in airfield proximity.

These images consisted of a variety of airplane types, shapes and color schemes, with a wide range of image scales, resolutions, and compositions. For example, images were selected in a way to show airplanes as close as possible (when an airplane covers almost the whole image) and from large distances (when an airplane is a small spot on the image, but still recognizable by its shape). Images quality varies from HD to Full HD and size of airplane shapes on them varies from about ten pixels wide to hundreds of pixels. The most of images contain more than one airplane, sometimes with overlapped bounding boxes for tightly placed airplanes. All airplanes have been fully placed within frame boundaries. Cases, where only a part of the airplane is present on the image have been excluded from the training set or have not marked as airplanes. The rest simply have a single airplane. All images contain photos in daylight conditions. Image quality also varies from high-quality clean images to noisy and distorted by compression artifact images. The training set consists of 204 images containing 1245 airplane objects in total.

The open-source tool known as YOLO mark [23] was used to label all airplane instances in the dataset and specify ground truth bounding boxes. Training has been carried out with the next parameters, batch size 64, momentum = 0.5, decay = 0.0005, base learning rate = 0.001, and maximum iteration number = 50000. Learning rate has been changed during training in a next way: 100% of the base on steps 0-39999, 10% on steps 40000-44999, 1% on steps 45000-49999.

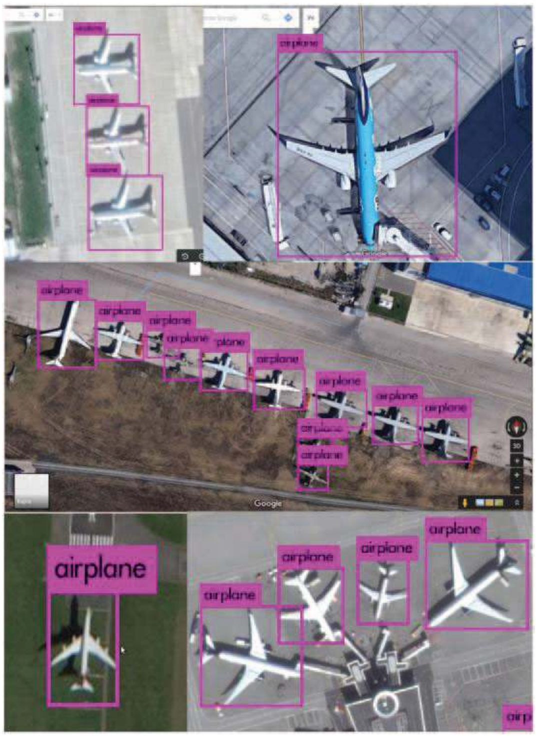

Figure 1. Pictures examples from our database of photos with "airplanes"

V. Experimental results

For validation of the trained neural networks, a new data set of 50 images similar to the ones in the training set and containing a total of 203 airplanes was used. Both trained neural networks have been tested with different sizes of an input image from 224×224 to 608×608 with the step 32 pixels. As far as the network is convolutional, the input tensor for it can be different sizes with a small restriction to be divided by 32. A such variability is an additional tool in finding a reasonable trade-off between desired speed and accuracy. Table 3 shows results for various cases for both YOLO v.3 and Tiny YOLO v.3 networks. There were carried out preliminary tests in order to avoid overfitting of the model. The best results (in terms of average precision) on the validation test have been obtained with the weights after 34000 iterations for YOLO v.3 and after 45300 iterations for Tiny YOLO v.3.

The table contains the next information: a set of sizes of input tensor; mAP is a mean average precision that means an averaged value on precision/recall curve calculated over 11 points [0, 0.1, …, 1]; IoU is an intersection over union that is the ratio of an intersection area of a ground true bounding box and a predicted bounding box to area of union of these boxes; FPS means processing speed frames per second (measured on the computer with the next hardware elements: CPU Intel Core i7-7700K, GPU NVIDIA 1060 6Gb, 16Gb RAM). Those parameters can be interpreted in the next way: mAP is an indicator of detection effectiveness, IoU is an indicator of bounding boxes positioning precision, FPS is a common performance and speed indicator.

TABLE III. YOLO V.3 AND TINY YOLO V.3 PERFORMANCE

Size	**mAP**	**IoU**	**FPS**
YOLO v.3			
608 × 608	90.91%	85.44%	13
416 × 416	90.73%	84.56%	23
320 × 320	90.64%	81.83%	34
224 × 224	81.72%	78.32%	51
Tiny YOLO v.3			
608 × 608	89.41%	80.18%	41
416 × 416	90.30%	79.75%	68
320 × 320	81.62%	78.83%	92
224 × 224	71.94%	78.46%	105

One can see that YOLO v.3 has a quite high precision level that is achievable even for a typical 25fps video. It has a slight dependence on the input tensor size. However, at the smallest size 224 × 224 there is a significant drop in the probability of detection. Tiny YOLO roughly is 2 times faster with slight drop of quality, especially in terms of mAP. But with the biggest input size, it can achieve a detection level comparable with YOLO v.3 at lower input sizes and still is faster.

It should be mentioned, that the trained YOLO v.3 CNN was able to detect an airplane in the image, even if its contours were obscured by another object, for example, a tower on the ground, or in pretty different conditions, for example, photos of airplanes in the air taken from the bottom, but size in pixels of the airplane must be relatively big. If an airplane is not fully shown in the image, the network recognizes it only when most of it is present, that has been expected.

On the other hand, when the size of the airplane image is distorted by compression artifacts, or when it has a livery or a shape not present in the training set it usually is missed by the network, especially at the low size of the airplane in the picture. Obviously, that can be overcome with the more thorough database, carefully selected, graded and cleaned of repetitive examples.

ACKNOWLEDGMENT

The research work reported in the paper was partly supported by the state research program; grant 131-DB17, funded by the Ministry of Education and Science of Ukraine.

REFERENCES

[1] N. Ammour, H. Alhichri, Y. Bazi, B. Benjdira, N. Alajlan, and M. Zuair, "Deep Learning Approach for Car Detection in UAV Imagery," Remote Sensing, Vol. 9(312), pp. 1–15, 2017.

[2] G. Maria, E. Baccaglini, D. Brevi, M. Gavelli, R. Scopigno, "A drone-based image processing system for car detection in a smart transport infrastructure," Proc. 18th Mediterranean Electrotechnical Conf. (MELECON). Limassol, Cyprus, 2016.

[3] C. Castiblanco, J. Rodriguez, I. Mondragon, C. Parra, and J. Colorado, "Air Drones for Explosive Landmines Detection," in ROBOT 2013: First Iberian Robotics Conference (M. A. Armada, A. Sanfeliu, M. Ferre Eds.) Springer International Publishing, pp. 107–114, 2013.

[4] Y. Xu, G. Yu, Y. Wang, X. Wu, and Y. Ma, "Car Detection from Low-Altitude UAV Imagery with the Faster R-CNN," Journal of Advanced Transportation, Vol. 2017, 2017.

[5] J. Lee, J. Wang, D. Crandall, S. Sabanovic, and G. Fox, "Real-Time Object Detection for Unmanned Aerial Vehicles based on Cloud-based Convolutional Neural Networks," Proc. IEEE International Conference on Robotic Computing (IRC). Taichung, Taiwan, 2017.

[6] F. S. Leira, T. A. Johansen, T. I. Fossen, "Automatic Detection, Classification and Tracking of Objects in the Ocean Surface from UAVs Using a Thermal Camera," Proc. IEEE Aerospace Conference. Big Sky, MT, USA, 2015.

[7] J.-N. Lee, K.-C. Kwak, "A Trends Analysis of Image Processing in Unmanned Aerial Vehicle," International Journal of Computer and Information Engineering, Vol. 2(8), pp. 271–274, 2014.

[8] S. R. Dhawad, R. R. Itkarkar, "Car Detecting Method using high Resolution images," International Journal on Recent and Innovation Trends in Computing and Communication, Vol. 2(4), pp. 197–203, 2016.

[9] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Müller, E. Säckinger, P. Simard, and V. Vapnik, "Comparison of learning algorithms for handwritten digits recognition," Proc. 1995 Int. Conf. on Artificial Neural Networks. Paris, France, pp. 53-60, 1995.

[10] P. Viola, and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proc. Computer Vision and Pattern Recognition (CVPR-2001). Kauai, HI, USA, pp. I-511 – I-518, 2001.

[11] N. Dalal, and B. Triggs, "Histograms of oriented gradients for human detection," Proc. Computer Vision and Pattern Recognition (CVPR-2005). San Diego, CA, USA, Vol. 1, pp. 886–891, 2005.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Proc. Computer Vision and Pattern Recognition (CVPR-2014). Columbus, OH, USA, pp. 580–587, 2014.

[13] R. Girshick, "Fast R-CNN," Proc. Computer Vision (ICCV-2015). Santiago, Chile, pp. 1440–1448, 2015.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, No. 6(39), pp. 1137–1149, 2015.

[15] T.-Y. Lin, P.Doll´ar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," Proc. Computer Vision and Pattern Recognition (CVPR-2017). Honolulu, HI, USA, pp. 936–944, 2017.

[16] K. He, G. Gkioxari, P. Doll´ar, and R. Girshick, "Mask R-CNN," Proc. Computer Vision (ICCV-2017). Venice, Italy, pp. 2980–2988, 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proc. Computer Vision and Pattern Recognition (CVPR-2016). Las Vegas, NV, USA, pp. 770–778, 2017.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed, "SSD: Single shot multibox detector," Proc. European Conference on Computer vision Computer Vision and Pattern Recognition (ECCV-2016). Amsterdam, The Netherlands, 2016.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," Proc. Computer Vision and Pattern Recognition (CVPR-2016). Las Vegas, NV, USA, pp. 779–788, 2016.

[20] J. Redmon, and A. Farhadi, "YOLOv3: An Incremental Improvement," Available at: https://pjreddie.com/media/files/papers/YOLOv3.pdf (Accessed 06.04.18)

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll´ar, "Focal Loss for Dense Object Detection," Proc. Computer Vision (ICCV-2017). Venice, Italy, pp. 2999–3007, 2017.

[22] Darknet: Open source neural networks in c. Available at: https://github.com/AlexeyAB/darknet (Accessed 09.05.18)

[23] YoloMark. Available at: https://github.com/AlexeyAB/Yolo_mark (Accessed 05.04.18)