Real-Time Implementation of Modified YOLO Object Detector on FPGA

Timophey Levkin[1], Petr Skonnikov [1] and Dmitriy Trofimov [1]

[1] JSC «S&P Complex «Alpha-M», Sviazy, 25, Ryazan, 390047, Russian Federation

Abstract

Some operating principles of the YOLO object detector are considered. The functions of processing the array elements coming from the last convolutional layer are analyzed. It is shown that three different types of functions are used to process various parameters. For example, the logistic sigmoid is used to predict the positioning accuracy and the bounding boxes vertical and horizontal offsets. The exponent is used to predict the height and width of boxes, and the SoftMax function is used to predict class probabilities. Formulas for the derivatives of these functions, which are required when training the network, are given. The replacement of these functions with simpler ones for implementation in hardware is proposed. The logistic function is replaced by a rational sigmoid, the exponent is replaced by a shifted fourth-order parabolic curve. The modified SoftMax function also uses a shifted parabola. The derivatives of proposed functions are presented to calculate the error backpropagation. Using the obtained formulas, a YOLO detector based on a modified neural network was trained. The modified YOLO detector with the proposed functions is implemented on a specially designed Xilinx FPGA board. Experimental studies of the board showed the high speed of the detector (more than 60 frames per second) and the high quality of object detection and classification.

Keywords

Digital image processing, artificial neural networks, YOLO, backpropagation, output layer.

1. Introduction

In modern computer vision systems, detectors based on artificial neural networks are used to detect and recognize objects [1]. In real-time systems, single-pass detectors are most widely used, since they have high quality characteristics at relatively low computational and time costs. These detectors include various modifications of the YOLO detector [2 – 4].

These detectors are based on neural networks of various architectures. They have different input sizes and number of input image channels. They also differ in the number of classification classes. Despite this, they all have a common principle of interpreting the array coming from the output layer for post-processing.

This three-dimensional array B of size $X \times Y \times N(5+C)$ is an ordered set of predictions for predefined anchor boxes , where X and Y are the number of anchor box initial positions in two coordinates, N is the number of anchor boxes at each initial position, and C is the size of the class alphabet [2]. These boxes are initially placed on the image in an $X \times Y$ grid. In each position of the grid, N boxes with predetermined sizes x_n and y_n are placed. The neural network predicts $5+C$ parameters for each $n = \overline{1,N}$ anchor box: the degree of overlap between the anchor box and the object IoU (Intersection over Union [5]), box displacements Δx and Δy relative to the initial position, scaling factors δx and δy for adjusting the initial height and width of the box, and conditional class probabilities $c = \overline{1,C}$.

GraphiCon 2022: 32nd International Conference on Computer Graphics and Vision, September 19-22, 2022, Ryazan State Radio Engineering University named after V.F. Utkin, Ryazan, Russia
EMAIL: tim-12345@mail.ru (T. Levkin); skonnikovpn@yandex.ru (P. Skonnikov); samael1978@rambler.ru (D. Trofimov)
ORCID: 0000-0003-4286-2816 (T. Levkin), 0000-0002-4989-3757 (P. Skonnikov), 0000-0001-9678-2024 (D. Trofimov)

In order for the elements of the array A coming from the last convolutional layer to receive the meaning described above, the transformation $B = \mathrm{f}(A)$ is carried out, bringing the array elements $a \in \mathbf{A}$ to the required range of values. This transformation contains an exponential function, the calculation of which requires significant computational costs in hardware. In this regard, it seems appropriate to carry out a simpler transformation for implementation on the FPGA.

2. Original transformation of output array elements

The YOLO algorithm uses three kinds of processing for different elements of the output array.

To predict IoU, Δx and Δy, an elementwise nonlinear transformation is performed, which has the form of a logistic function [2-4]:

$$b = \mathrm{f}_1(a) = \frac{1}{1+e^{-a}}, \tag{1}$$

where a and b are elements of the arrays A and B corresponding to the values IoU, Δx and Δy.

The transformation for δx and δy has an exponential form [2-4]:

$$b = \mathrm{f}_2(a) = e^a. \tag{2}$$

The values array element $b \notin B$ for the class probabilities p_c depend not only on the element a, located at the same position in the array A, but also on other elements of the array A corresponding to other classes at the current grid position for the current anchor box. That is, each element b_k, $k = \overline{1,C}$ is determined by SoftMax function [2...4]:

$$b_k = \mathrm{f}_3(a_1, a_2 \ldots a_C) = e^{a_k} \Big/ \sum_{c=1}^{C} e^{a_c}. \tag{3}$$

When the detector is running, $3XYN$ operations (1), $2XYN$ operations (2) and $CXYN$ operations (3) are performed in each frame, which requires significant computational costs on hardware.

When training a network using methods based on stochastic gradient descent and its modifications [6], error backpropagation through the layer that performs transformations (1)-(3) is performed using the chain rule:

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial b}\frac{\partial \mathrm{f}(a)}{\partial a}, \tag{4}$$

where $\partial L / \partial a$ is the component of the desired loss function gradient to be transferred to the previous layer, $\partial L / \partial b$ is the component of the loss function gradient coming from the next layer, and the derivative $\partial \mathrm{f}(a) / \partial a$ can be expressed analytically in terms of the values a, b and $\partial L / \partial b$, known at the stage of operation (4).

Particularly,

$$\frac{\partial \mathrm{f}_2(a)}{\partial a} = e^a = \mathrm{f}_2(a) = b, \tag{5}$$

that is, in practice, the calculation of the exponent is not performed, since it will lead to the previously calculated value b.

For function f_1

$$\frac{\partial \mathrm{f}_1(a)}{\partial a} = \frac{e^{-a}}{(1+e^{-a})^2}. \tag{6}$$

It is easy to see that a simpler calculation of the product $b(1-b)$, where $b = \mathrm{f}_1(a)$, will lead to the same result.

It is known [7] that for the function f_3, its derivative can also be expressed in terms of $b_k = \mathrm{f}_3(a_1, a_2 \ldots a_C)$:

$$\frac{\partial \mathrm{f}_3(a_k)}{\partial a_c} = \begin{cases} b_k(1-b_k), & \text{if } k = c; \\ -b_k b_c, & \text{if } k \neq c. \end{cases} \tag{7}$$

3. Proposed transformation of output array elements

As noted above, the multiple execution of transformations (1)–(3), required for the YOLO detector operation, needs the calculation of an exponential function values. Methods of calculus mathematics make it possible to obtain a fairly accurate approximation of the exponential for a certain number of terms of the corresponding series taken into account. Nevertheless, to ensure the necessary accuracy of the correspondence between transformations (1)–(3) and operations performed on hardware, an unacceptably high computing power is required. In this paper, we propose a different approach that does not require an exact approximation of transformations (1)–(3).

The logistic function (1) is replaced by a rational sigmoid:

$$\mathrm{f}_1(a) = \frac{1}{2} \cdot \frac{a}{1+|a|} + \frac{1}{2}. \tag{8}$$

In this formula, the scale and shift factors $1/2$ are needed to satisfy the conditions $\lim\limits_{a \to -\infty} \mathrm{f}_1(a) = 0$, $\lim\limits_{a \to +\infty} \mathrm{f}_1(a) = 1$ and $\mathrm{f}_1(0) = 1/2$, as well as for function (1).

The exponential function (2) is replaced by the parabola:

$$\mathrm{f}_2(a) = (a + \beta)^\alpha / \beta^\alpha. \tag{9}$$

The parabola shift value $\beta = 16$ is chosen so that all possible values of fixed-point signed numbers, which are used in the implementation of the FPGA detector described below, fall on the monotonically increasing branch of the parabola. The index of power is taken equal to $\alpha = 4$, since such a value converts the input values a to the range of values b of the original transformation (2) more accurately than other integer indices.

A similar replacement of the exponent is also performed in transformation (3), with the difference that normalization with respect to β^α is not required:

$$b_k = \mathrm{f}_3(a_1, a_2 \ldots a_C) = (a_k + \beta)^\alpha / \Sigma. \tag{10}$$

where $\Sigma = \sum\limits_{c=1}^{C} (a_c + \beta)^\alpha$.

As noted above, expressions (8)–(10) are not exact approximations of transformations (1)–(3). In this regard, the work of the ANN with transformations (8)–(10), trained using formulas (5)–(7), will lead to incorrect results. This means that the replacements made must be taken into account when training the network.

For the backpropagation computing according to the rule (4), it is necessary to calculate the derivatives of the functions (8)–(10):

$$\frac{\partial \mathrm{f}_1(a)}{\partial a} = \frac{1}{2} \cdot \frac{1}{(1+|a|)^2}. \tag{11}$$

$$\frac{\partial \mathrm{f}_2(a)}{\partial a} = \frac{\alpha}{\beta^\alpha} (a + \beta)^{\alpha-1}. \tag{12}$$

$$\frac{\partial \mathrm{f}_3(a_k)}{\partial a_c} = \begin{cases} \dfrac{\alpha (a_k + \beta)^{\alpha-1}}{\Sigma} - \dfrac{\alpha (a_k + \beta)^{2\alpha-1}}{\Sigma^2}, & \text{if } k = c; \\ -\dfrac{\alpha (a_c + \beta)^{\alpha-1} (a_k + \beta)^\alpha}{\Sigma^2}, & \text{if } k \neq c. \end{cases} \tag{13}$$

After the partial derivatives (13) have been calculated, the backward propagation of errors through transformation (10) can be calculated as follows [8]:

$$\frac{\partial L}{\partial a_k} = \sum_{c=1}^{C} \frac{\partial L}{\partial b_c} \frac{\partial \mathrm{f}(a_c)}{\partial a_k} \tag{14}$$

However, the assumption

$$\frac{\partial L}{\partial a_k} = \frac{\partial L}{\partial b_k} \frac{\partial \mathrm{f}(a_k)}{\partial a_k} \tag{15}$$

accepted during the training of the network, also leads to the correct operation of the detector.

Network training by formulas (11)–(13) requires more computational costs than calculation by formulas (5)–(7). However, it should be taken into account that training is performed on a high-performance video card within an acceptable time frame (from several hours to several days), and real-time training of ANN on FPGA is not required.

4. Experimental studies of the detector hardware implementation

The YOLO detector with the proposed modifications is implemented on a specially designed board. The appearance of this board is shown in Figure 1.

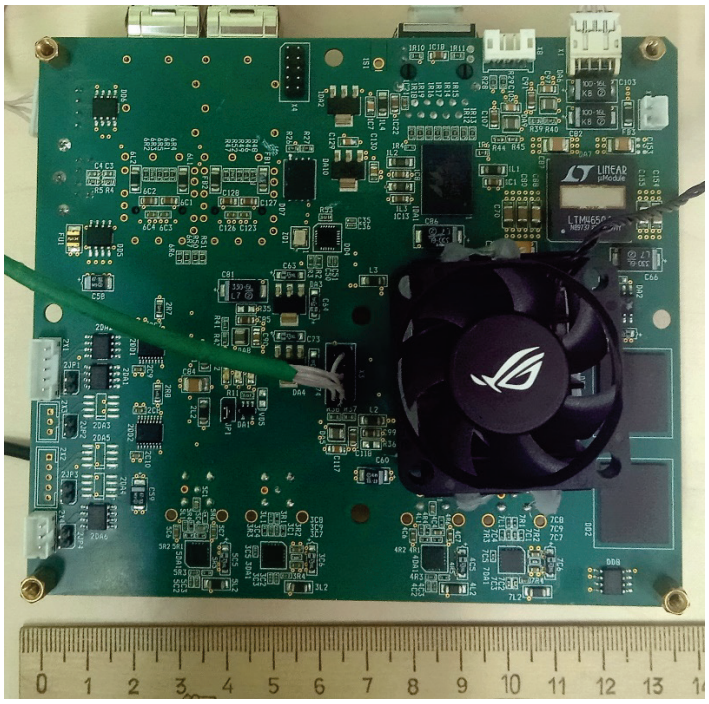

Figure 1: The appearance of the online video tracking and classification board

Calculations are made on Xilinx FPGA. The board is equipped with four SDI digital video interfaces, as well as Ethernet and two SFPs for connecting other interfaces. The dimensions of the board do not exceed $13\times11{,}5$ cm, and the average power consumption during detector operation is no more than 20 W.

Color frames up to 1920×1080 pixels in size are processed, following at a frequency of 60 Hz. The size of the search area for objects in the ongoing experimental studies was 522×522 pixel, and this size, if necessary, can be increased.

It takes 8 ms to pass the neural network in the forward direction. Further post-processing, including filtering, non-maximum suppression, rendering of character information and transferring the output frame for display, takes from 3 to 6 ms, depending on the number of detected targets.

When processing typical video recordings, the values of the output array of floating-point numbers obtained on the video card of a personal computer coincided with the corresponding values calculated by hardware on the board with an error of less than 1%.

Recognition was carried out using two modified YOLO detectors of the same architecture based on the ResNet-18 network: one for recognizing ground objects and one for airborne objects.

Based on the validation results, it was obtained that the area under the Recall-Precision curve [9] is at least 0.9 for each class, and the area under the receiver operating characteristic [10] is at least 0.97. The same quality indicators were obtained when training the neural network using formulas (5)–(7) and detecting objects using formulas (1)–(3).

Examples of the obtained results for recognizing ground and airborne objects are shown in Figure 2.

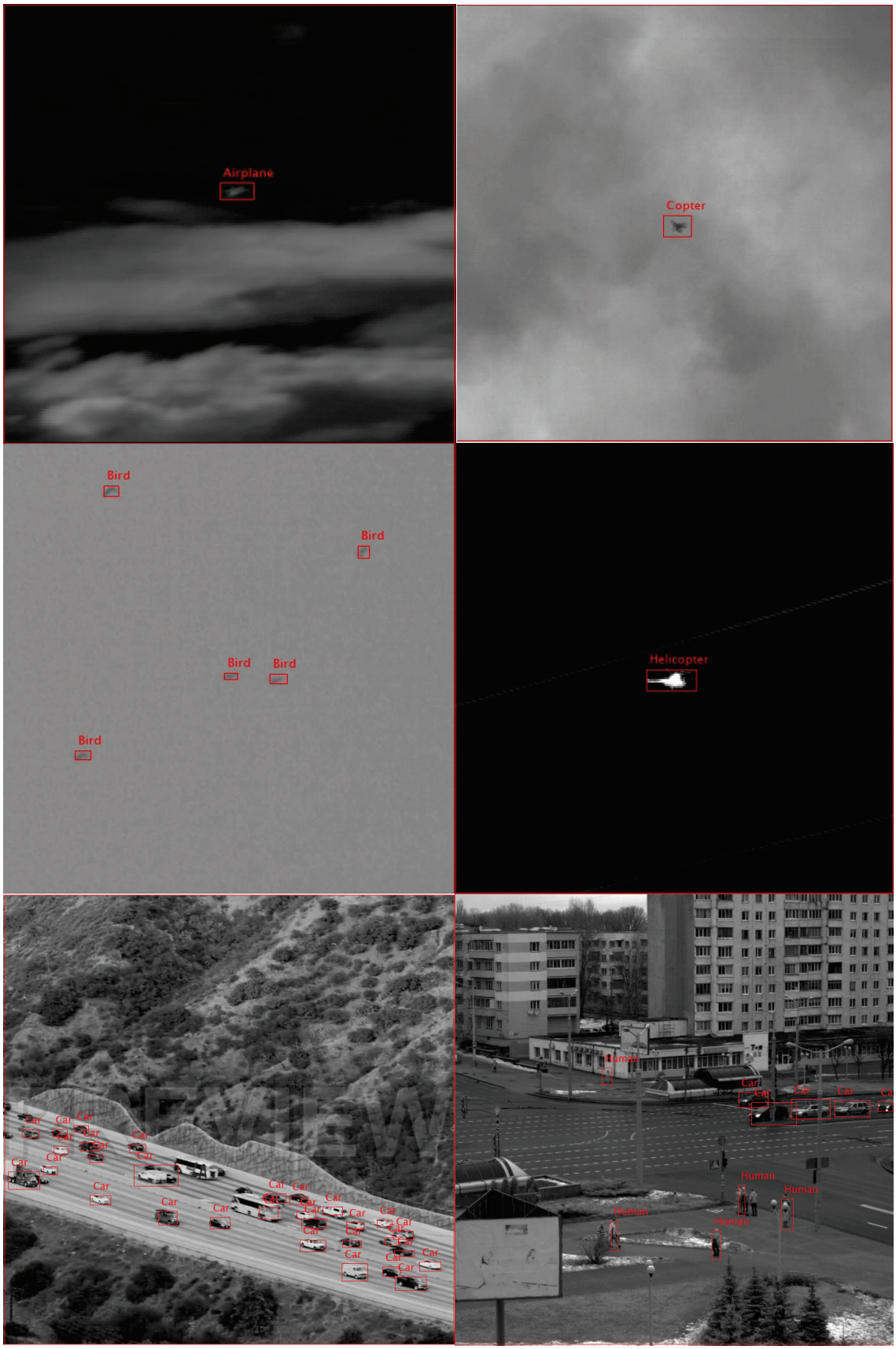

Figure 2: Examples of the obtained results for recognizing ground and airborne objects

The examples given show that the modified detector recognizes small-sized, low-contrast objects on the complex background.

5. Conclusion

The proposed transformations (8) – (10) made it possible to create a hardware implementation of the YOLO object detector on the board.

The created prototype of a vision system with object detection and classification works with low latency in real time.

The analytically derived formulas (11) – (13) are used in the program that implements the training of the modified neural network.

The effectiveness of the proposed transformations is confirmed by the qualitative characteristics of object recognition, which are not inferior to the characteristics of the original detector.

6. References

[1] Aziz L. et al. Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review. IEEE Access. 2020. Vol. 8. pp. 170461-170495. doi: 10.1109/ACCESS.2020.3021508.

[2] Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. pp. 779-788. doi: 10.1109/CVPR.2016.91.

[3] Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. pp. 7263-7271. doi:10.1109/CVPR.2017.690.

[4] Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. ArXiv. 2018. URL: https://arxiv.org/pdf/1804.02767.pdf. doi: 10.48550/arXiv.1804.02767.

[5] Rezatofighi H. et al. Generalized intersection over union: A metric and a loss for bounding box regression. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. pp. 658-666. doi:10.48550/arXiv.1902.09630.

[6] Shalev-Shwartz S., Ben-David S. Understanding machine learning: From theory to algorithms. Cambridge university press. 2014. 449 pp. doi: 10.1017/CBO9781107298019.

[7] Backpropagation with Cross-Entropy and Softmax. ML Dawn. 2021. URL: https://www.mldawn.com/back-propagation-with-cross-entropy-and-softmax/.

[8] The Derivative of Softmax Function. ML Dawn. 2021. URL: https://www.mldawn.com/the-derivative-of-softmaxz-function-w-r-t-z/.

[9] Buckland M., Gey F. The relationship between recall and precision // Journal of the American society for information science. 1994. Vol. 45. No. 1. pp. 12-19. doi: 10.1002/(SICI)1097-4571(199401)45:1<12::AID-ASI2>3.0.CO;2-L.

[10] Van Trees H. L. Detection, estimation, and modulation theory, part I: detection, estimation, and linear modulation theory. – John Wiley & Sons. 2004. 716 pp.