Gradual Labeling of the Training Set to Improve the Efficiency of Image Detection by a Neural Network on the Example of License Plate Recognition

Yaroslav Schegolihin, Maksim Mitrohin, Valeriya Sazykina
Penza State University
Penza, Russian Federation
Yaroslav Schegolihin, yaroslav.schegolikhin@yandex.ru

Maksim Semenkin
CodeInside
Penza, Russian Federation
maxim.semenkin@codeinside.ru

***Abstract*—Currently, the automation of various tasks is very important in modern society. One of these tasks is the recognition of the car license plate in the video stream. Our project is aimed at this problem. This task is not new and there are many open source solutions available. All solutions can be two types of implementation: with using neural networks and without using it. Our implementation is a neural network for car license plate recognition on an image. We use the YOLOv3 detector for recognition. Training of neural networks is often complicated due to the small number of training samples. When you train a neural network using an insufficiently amount of data, it is difficult to achieve the required effectiveness. This problem can be solved in two ways: by expanding the training dataset, or by selecting the optimal training parameters and transforming the existing dataset. This article describes the solution to this problem by selecting the optimal training parameters and transforming the training dataset. The gradual labeling of the training set is discussed. The tuning of the anchor boxes parameter is discussed in detail. The presentation of the test results provides a visualization of the work performed. Research results show an increase in the efficiency of the neural network detector by 30.5% from the initial value. Submitted method helps to save a lot of time, because automate expanding the training dataset.**

I. Introduction

Currently, neural networks are used in many areas. One of the most popular areas is image processing for the finding and classifying various objects. Depending on the goal, the following tasks can be solved using neural networks (see Fig. 1) [1]:

- Semantic Segmentation is the division of the image into different types of areas [2];
- Classification + Localization is the classification of the object on the image and the determination of its position;
- Object Detection is a finding objects of predefined classes in the image;
- Instance Segmentation is a method for finding objects on the image and applying a mask to it.

There are many open source tools for implementing this task [3-6]. But firstly, you need to create a dataset that will contain labeled images in order to train image detectors based on neural networks. There are a large number of already labeled datasets for solving different tasks [7-9]. However, in some specific tasks, the issue of collecting and labeling the training dataset stays quite acute. One of these tasks is the task of detecting car registration plates. Manually marking a large data set is a time-consuming process, because in addition to marking objects you need to control the correctness of the produced labels.

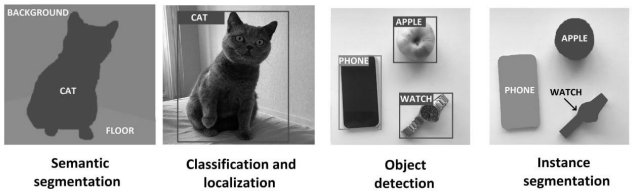

Fig. 1. Types of tasks in image processing.

II. Main part

A. The current state

The task of detecting a car license plate on an image is not new, [10-14] contains information about its solution. To summarize, the existing methods can be divided into two groups:

1) Using neural networks;
2) Without using neural networks.

Detecting a car license plate can be solved without using neural networks, by statistical analysis [15], [16] or machine learning methods, for example, with Haar cascades [17]. These types of algorithms work quite effectively in real time. The image is submitted to the input of the algorithm and the Haar features are calculated for each part of it. The convolution of a part of the image with Haar primitives is calculated to obtain the features:

$$I_t(x,y) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} I(x-u, y-v) \cdot g_t(u,v), \qquad (1)$$

where I(x, y) is a part of the original image, $k = (p - 1)/2$ and $l = (q - 1)/2$, $g_t(u, v)$ – *t-th* a Haar primitive of size $p \cdot q$.

Each primitive (see Fig. 2) represents several contiguous rectangular zones, each of which can contain one of two values - plus 1 (light) and minus 1 (dark). Primitives vary by the location of light and dark zones, and their orientation.

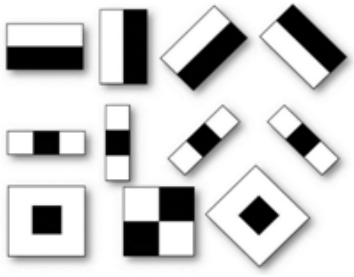

Fig. 2. Haar primitives

The feature is calculated for each convolution:

$$f^t = \sum_{(x,y)\in A} I_t(x,y) - \sum_{(x,y)\in B} I_t(x,y) \quad (2)$$

where A is a set of coordinates corresponding to the light part of the Haar primitive involved in the convolution operation, B is a set of coordinates corresponding to the dark part of the Haar primitive.

The features obtained from expression (1) are submitted to the classification algorithm, which makes a decision in compliance with the expression:

$$h(I) = \begin{cases} 1, if \sum_{t=1}^{T} a_t \cdot h_t(f^t) \geq \frac{1}{2}\sum_{t=1}^{T} a_t, \\ 0, else, \end{cases} \quad (3)$$

where T is the number of Haar primitives, and a_t are the weights determined as a result of the training algorithm.

Using neural networks for image processing is a stable trend of last time. Neural networks show higher effectiveness in solving many image processing problems than other algorithms [18].

Most modern neural network architectures for image processing are based on convolution layers, which are the main element of feature extraction from the image.

Convolutional layers implement the operation of convolution of the input image with the kernel, which is described by the expression:

$$I_{out}(x_{out}, y_{out}) = \sum_{i=1}^{x_{knl}} \sum_{j=1}^{y_{nrl}} I_{in}(x_{out} - i + 1,\ y_{out} - j + 1) \cdot K(i,j), \quad (4)$$

where I_{in} и I_{out} – input and output images with sizes (x_{in}, y_{in}) and (x_{out}, y_{out}), K – the core of the filter with sizes (x_{knl}, y_{knl}).

In this case, the weight coefficients of the kernel are determined in the process of training the neural network.

There are several neural network architectures that are based on convolutional layers: AlexNet [19], VGG [20], GoogleNet [21], ResNet [22], DarkNet [23]. The Architectures is distinguishes by the number and parameters of convolutional layers and the presence of some special layers, such as Inception or Residual.

Convolutional networks solve the problem of detecting objects only for images with a fixed size that is equal to the size of the input layer of the neural network. Therefore, on the basis of convolutional neural networks image detectors are built.

The main tasks of the detector is to select fragment of the image that have size that corresponding to the size of the input layer of the neural network and to combine the solutions of the neural network obtained for each fragment into one solution on an image that potentially contains objects and other elements of the scene (region of interest).

Detectors differ by the approach of the regions of interest selection and the way of forming the detection area. Currently, the most popular are the family of R-CNN detectors [24-26], YOLO detectors [23] and SSD detectors [27].

The R-CNN (Regions with CNNs) approach relies on image preprocessing. Pre-selected regions (region proposal) submitted to the input CNN, where the required objects are presumably located. The method that makes such assumptions is the unsupervised image segmentation algorithm - Selective Search [28].

CNN is used to extract features from the fragments of the image, which was selected by Selective Search, and then objects is classified by N (by the number of classes) linear support vector machines (SVM). Each SVM performs a binary classification, based on its own object class. Combining solutions for multiple regions is performed using the IOU metric (Intersection over Union). The bounding-box regression method [29] is used for specifying the position of the bounding box, which is performed during the error analysis procedure. The parameters dx, dy, dw, dh of the offset of the predicted bounding box relative to ground truth are determined after classifying the content of the candidate region based on features from CNN using linear regression.

In the Fast R-CNN modification the entire image is submitted to the CNN instead of individual regions, the last max-pool layer is replaced by the RoI pooling layer. Also, binary SVMs are replaced with a fully connected layer and softmax with N+1 outputs.

Faster R-CNN implements the idea of calculating regions of interest from the feature map obtained from CNN instead of initial image. The Region Proposal Network (RPN) module was added to do this. The RPN layers select the regions of interest, which are passed to the object detection and correction for the covered area.

You Only Look Once (YOLO) is a one of CNN-based image detectors. Full image is submitted to CNN at the same time, and a grid is superimposed on the resulting feature map, which nodes are associated with the true bounding boxes and calculates the probability that the desired object is located there for each section of the image. Thus, the neural network is trained to predict the presence of individual parts of the

desired object inside the grid cells instead of the entire object as a whole. YOLO also uses the concept of anchor boxes, which defines the average size of bounding boxes for detection objects. Therefore, YOLO tries to predict the position of the anchor boxes relative to the actual bounding boxes, rather than the coordinates of the object's position in the image.

Single Shot MultiBox Detector (SSD) is similar to YOLO by its ideology. It also processes the entire image without selecting regions and tries to predict the presence of a part of an object of a certain class at the same time in the pre-generated default boxes.

The source image is fed to the input of the convolutional network. The size of the feature map decreases as the convolutional layers pass through, but its depth increases. The featuremap is submitted to the Detector & Classifier block from several convolutional layers. The default boxes generator generates limits covering the original image inside this block. The auxiliary convolutional layers correct the default boxes to detect objects and classify it. The results are corrected, classified, and filtered at the final stage of this block, at the output, we get the detected object. The resulting solutions on default boxes are fed to Fast Non-Maximum Suppression, which combines them into the final result.

There are a lot of open source projects [30-32] that make it possible to easily realize the recognition of any objects in a video stream or image. The main obstacle for this is the presence of a labeled image dataset. It is often impossible to find a labeled training base in free access. Manual labeling of large volumes of images is quite labor-intensive.

B. Gradual labeling of the training set

The choice of detector. We need to choose a detector that will be used for the license plate detection. The most suitable for this task are the one-stage detectors YOLO and SSD. They don't use a separate algorithm to determine the regions of interest, this increases the rate of the detector.

We chose the YOLOv3 detector from the specified pair of detectors, because it has more flexible parameters. In particular, the ability to adjust the size of anchor-boxes for the size of objects of interest.

The first training model. To create dataset we have collected 7749 images with cars that have car license plates. Images of cars are represented by various scenes: in the traffic flow, in outdoor and indoor parking lots, in the courtyards. The images were captured from different angles, at different camera mounting heights, and distances (see Fig. 3). 1489 images from this set were manually labeled (training set 1), the first iteration of the YOLO detector training was performed on them. During training, the weights of the neural network were set randomly, the training parameters are presented below.

Parameters of the [net] section:

- batch = 16 – the number of sample items that are being processed during one iteration before the weights change;
- subdivisions = 1 – the number of mini batches (this parameter indicates the number of training examples used in a single loop). Calculated as mini_batch = batch/subvisions;
- width = 416 – each image is applied to the input of the network will be changed in width in accordance with this parameter;
- height = 416 – the image changes in height similar to the width parameter;
- channels = 3 – it determines the size of the network that submits to the input, as the width and height parameters. Each image is converted to 3 channels (RGB);
- angle = 0 – a parameter that indicates the number of degrees by which the image is randomly rotated during training;
- saturation = 1.5 – a random change in the saturation of images during training;
- exposure = 1.5 – a random change in the brightness during a training;
- hue = .1 – random color change during training.

Fig. 3. Training set examples

Optimizers:

- momentum = 0.9 – characterizes how much the history affects the further change in the weights;
- decay = 0.0005 – characterizes a weaker update of weights for typical features, eliminates an imbalance in the dataset;
- learning_rate = 0.001 – the value of the learning rate;
- burn_in = 1000 – the initial burn_in will be calculated in the first 1000 iterations that used in the formula current_learning_rate = learning_rate * $(\text{iterations}/\text{burn\_in})^{\text{power}}$ = 0.001 * $(\text{iterations}/1000)^4$, where power = 4 by default;
- max_batches = 500200 – maximum number of selection items;

- policy = steps – rules for changing learning-rate. Possible parameters: constant (by default), sgdr, steps, step, sig, exp, poly, random. The effective learning rate will change according to the formula: current_learning_rate = learning_rate * rand_uniform(0,1)^{power}, if policy = random;
- steps = 400000, 450000 – this parameter is indicated when policy = steps. The parameter specifies at which iterations the learning_rate will be changed according to the scales parameter;
- scales = 0.1, 0.1 – this parameter is indicated when policy = steps. If steps = 400000, 450000, the number of the current iteration is more than 400000, but less than 4500000, then the formula for calculating: current_learning_rate = learning_rate * scales[0] = 0.001 * 0.1 = 0.0001. If the current iteration is more than 450000, the multiplication will be performed by scales [0] * scales.

Section [yolo]:

- anchors – a set of predefined bounding boxes of a certain height and width;
- mask = 3, 4, 5 – the numbers of the anchor boxes used on the layer;
- classes = 1 – number of classes to train;
- jitter = .3 – randomly resizing the image from x * (1 – 2 * jitter) to x * (1 + 2 * jitter);
- random = 1 – random change in network size after each 10th iteration from x/1.4 to x * 1.4 of the initial network size;
- truth_thresh = 1 – IOU threshold value (metric of the degree of intersection between two bounding boxes);
- ignore_thresh = .7 – threshold value that determines whether the error should be taken into account if the truth_thresh threshold is not exceeded;
- num = 9 – the number of anchor boxes.

The collected number of training samples was not enough for high-quality training of the YOLO model. The results of testing the model after the first training are shown in Table I.

TABLE I. THE RESULTS OF THE FIRST YOLO TEST.

The correct results	False positives	Nothing was found	Total
4289(55,4%)	102(1,3%)	3358(43,3%)	7749(100%)

Then, we trained the Haar cascade from the OpenCV [33] library to increase the number of labeled images on the initial set. Contrary instances are required to train the cascade. That is images, that don't contain detectable objects. The sample of contrary instances contained 1268 images of various inscriptions, signboards, road signs, advertising objects, etc. After training the Haar cascade, all 7749 source images were processed by the resulting detector. The results of the Haar cascade are shown in Table II.

TABLE II. RESULTS OF TESTING THE HAAR CASCADE.

The correct results	False positives	Nothing was found	Total
5491(70,9%)	26(0,3%)	2232(28,8%)	7749(100%)

Therefore, the number of labeled dataset increased by more than 3.6 times. The sample formed in this way was split into training and test set. The training set included 3233 labeled images (training set 2), the test set included 2258. The weights of the neural network were set randomly during training, the training parameters are similar to those specified earlier.

The YOLO detector was trained on the training set 2. The test showed a correct recognition rate of 73.4% (Table III).

TABLE III. TEST RESULTS AFTER TRAINING ON SAMPLE 2.

The correct results	False positives	Nothing was found	Total
1657(73,4%)	21(0,9%)	581(25,7%)	2258(100%)

Such characteristics still do not allow us to use the system in real environment, despite the improvement of the result. Another way to improve the quality of the model is to adjust the parameters.

Selection of anchor boxes. The usage of anchor boxes in the YOLO detector allows adapting the grid superimposed on the feature map to the geometric parameters of objects of interest with different classes. This parameter defines the set and size of predefined boxes. The YOLO detector initially uses anchor boxes that are optimized for the characteristics of COCO Dataset objects.

The method of calculating anchor boxes [34] involves the analysis and clustering of bounding boxes of placed objects by area and the ratio of width and height. Large areas on the chart define the size of the anchor boxes. The desired number of anchor boxes sets the number of clusters into the divided geometric space (see Fig. 4), where each example of the training sample is represented by the coordinates (x, y), where x is the bounding box area of training sample, y is the aspect ratio width/height of training sample.

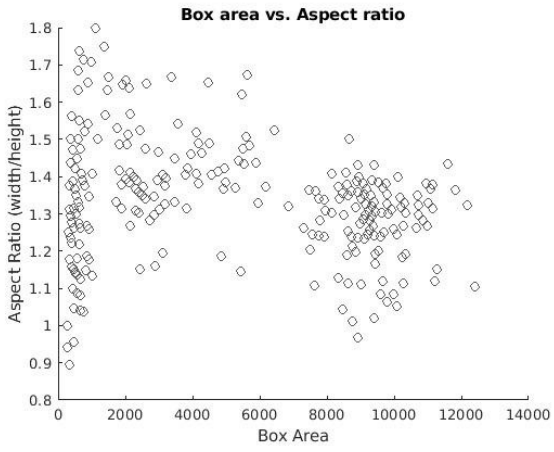

Fig. 4. Box area vs. Aspect ratio

Standard sizes: 10,13; 16,30; 33,23; 30,61; 62,45; 59,119; 116,90; 156,198; 373,326.

The dimensions obtained as a result of selection on the training sample: 120,40; 139,47; 155,52; 172,58; 187,63; 203,68; 227,76; 286,96; 437,146.

Training the detector with the new anchor boxes improved the result of correctly recognized numbers by 7%, but at the same time the result of false positives increased to 2.7%. The allocated boxes of car plates began to be localized more accurately at the same time (see Fig. 4).

TABLE IV. RESULTS OF TRAINING ON SAMPLE 2 WITH NEW ANCHOR BOXES.

The correct results	False positives	Nothing was found	Total
1816(80,4%)	61(2,7%)	381(16,9%)	2258(100%)

Fig. 5. Visualization of the result

The transformation of a training dataset. At the next stage, the training was retrained on the training set 2, but with the pre-trained weights in the previous iteration. This attempt didn't help to improve the result.

There was a set of images that the cascade could not recognize during the labeling with the Haar cascade. This set of images was processed by the detector and the recognized images were added to the training set. This is how the training set 3 was formed. Training the detector on a training set 3 allowed us to increase the result of the detected license plate. However, the result of false positives has also increased (Table IV).

TABLE V. TEST RESULTS AFTER TRAINING ON THE TRAINING SAMPLE 3.

The correct results	False positives	Nothing was found	Total
1882(83,3%)	88(3,9%)	288(12,7%)	2258(100%)

Analysis of the remaining images without labels showed that the dataset mostly contains images with license plates that are located at an angle of more than 30 degrees to the horizontal, or polluted, or poorly lit, or obscured by vegetation or other objects of the scene. These images were labeled manually. Some of them were added to the training set (training set 4 was formed), and some were made up for a "difficult" test sample, with a size of 1217 images. We calculated anchor boxes for the updated training set and retrained the detector.

1000 images were randomly selected from the initial test set, which were removed from it and formed a "simple" test sample. 1000 images from the "difficult" test set were added to the initial test set instead of the removed images. This is how we formed the "averaged" test set.

So we got 3 test sets. One test set contained images that were initially recognized by the OpenCV cascade (a "simple" set). The second set contained images with dirty license plates, license plates at an angle to the horizontal, and other images whose recognition was complicated (a "difficult" set). The third was a "mixed" set, which consisted of "simple" and "difficult" images. None of the images from the test sets were included in the training set 4. The results of testing the detector trained on the training set 4 are shown in Table VI.

TABLE VI. TEST RESULTS AFTER LEARNING ON TRAINING SAMPLE 4 AND UPDATED ANCHOR BOXES.

Test base	The correct results	False positives	Nothing was found	Total
Simple	986(98,6%)	10(1%)	4(0,4%)	1000(100%)
Difficult	919(75,5%)	86(7%)	212(17,4%)	1217(100%)
Mixed	1946(86,2%)	96(4,2%)	216(9,6%)	2258(100%)

This transformation allowed us to achieve 98.6% efficiency on images of the "simple" test set, 85.9% correct detections on the "mixed" test set and 75.5% correct detections on "difficult" images.

III. CONCLUSION

Thus, we improved the model accuracy from the initial 55.4% to 85.9% by transforming the input data and training parameters. Using automatically labeled images as training data allowed us to improve the quality of recognition. The quality of the YOLO detector improves with an increase of number of samples in the training set. But there is an increase in the false detections at the same time. One of the parameters that can significantly influence the results of the YOLO detector are the anchor boxes. It is recommended to configure this parameter for a specific training dataset during training the detector. The further research is to solve the problem of improving the accuracy by augmenting automatically labeled data. For example, you can randomly change the position of the image during training, apply the transformation of stretching and compression of images by horizontal and vertical, rotation by an arbitrary angle. Also we plan to find out the reason for the increase of false positives with increasing training dataset and try to solve this problem.

We have shown that the creation of a training set for neural networks can be performed with automatically labeling by less efficient algorithms. This saves time and improves quality when combined with manual labeling.

REFERENCES

[1] Medium website, Image Classification vs. Object Detection vs. Image Segmentation, Web: https://medium.com/analytics-vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81.
[2] Nanonets website, A 2021 guide to Semantic Segmentation, Web: https://nanonets.com/blog/semantic-image-segmentation-2020/.
[3] PyTorch official website, Web: https://pytorch.org/.
[4] TensorFlow official website, Web: https://www.tensorflow.org/.
[5] Kera official websites, Web: https://keras.io/.
[6] Darknet official website, Web: https://pjreddie.com/darknet/.
[7] ImageNet official website, Web: http://www.image-net.org/.
[8] COCO official website, Web: https://cocodataset.org/#home.
[9] CIFAR-10 official website, Web: https://www.cs.toronto.edu/~kriz/cifar.html.

[10] M.T. Qadri, M. Asif. “Automatic Number Plate Recognition System for Vehicle Identification Using Optical Character Recognition”, *International Conference on Education Technology and Computer. Singapore*, May 2009, pp. 335-338.
[11] N. Arora, S. Jain, P. Gour. “Survey of Number Plate Recognition for Use in Different. Countries using an Improved Segmentation.” *International Journal of Computer Applications*, vol. 102, Wardha. March 2014, pp. 47-50.
[12] Y. Zhao, X. Gu. “Vehicle License Plate Localization and License Number Recognition Using Unit-Linking Pulse Coupled Neural Network”, *in International Conference on Neural Information Processing*, vol. Part V, Bangkok, Thailand, Nov. 2012, pp. 100-108.
[13] L.G.C. Hamey, C. Priest. “Automatic Number Plate Recognition for Australian Conditions”, *Digital Image Computing: Techniques and Applications*, Dec. 2005. Queensland, Australia.
[14] J. Wang, W.Q. Yan, “BP-Neural Network for Plate Number Recognition”, *Deep Learning and Neural Networks*, Jan 2020, pp 1189-1199.
[15] K. Aboura, R. Al-Hmouz. “An Overview of Image Analysis Algorithms for License Plate Recognition”, *Organizacija*, Vol 50 (3), 2017, pp. 285-295.
[16] V.H. Deepthi, B.B. Singh, V.S. Rao. “Automatic Vehicle Number Plate Localization Using Symmetric Wavelets”, *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India*, vol 1, Jan. 2014. pp 69-76.
[17] P. Viola, M.J. Jones “Rapid Object Detection using a Boosted Cascade of Simple Features”, *Proceedings Conference of Computer Vision and Pattern Recognition*, vol. 1, Kauai. USA, Feb. 2001. pp. 511-518.
[18] M.R. Aguila, I. Requena, J.L. Bernier, E. Ros, S. Mota. “Neural Networks and Statistics: A Review of the Literature”, *Soft Methodology and Random Information Systems*, Jan. 2004, pp. 597-604.
[19] A. Krizhevsky, I. Sutskever, G.E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, *Neural Information Processing Systems*, vol. 60, Jan. 2012.
[20] K. Simonyan, A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *International Conference on Learning Representations*, Sep. 2014
[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke. A. Rabinovich. "Going deeper with convolutions", *In Proceedings of the IEEE conference on computer vision and pattern recognition*, June 2015, pp. 1-9.
[22] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition" *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, June 2016, pp. 770-778.
[23] J. Redmon, A. Farhadi, “YOLOv3: An Incremental Improvement”, *ArXiv*, Apr. 2018
[24] R. Girshick, J. Donahue, T. Darrell, J. Malik, “Region-Based Convolutional Networks for Accurate Object Detection and Segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, Dec. 2015, pp. 142-158.
[25] R. Girshick, "Fast R-CNN" *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, April 2015, pp. 1440-1448.
[26] S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, June 2015, pp. 1137 - 1149
[27] W. Liu, D. Anguelov, D. Erhan,C. Szegedy, S. Reed, C. Fu, A. Berg, “SSD: Single Shot MultiBox Detector”, *European Conference on Computer Vision*, vol. 9905, Sep. 2016, pp. 21-37.
[28] J. Uijlings, K. Sande, T. Gevers, A.W.M. Smeulders, “Selective Search for Object Recognition”, *International Journal of Computer Vision*, vol.104, April 2013, pp. 154-171.
[29] S. Lee, S. Kwak, M. Cho, “Universal Bounding Box Regression and Its Applications”, *Computer Vision – ACCV 2018*, vol. 11366, May 2019, pp. 373-387.
[30] P. Mishra. “CNN and RNN Using PyTorch*”, CNN and RNN Using PyTorch*, Jan. 2019, pp. 49-109.
[31] P. Kumar, U. Dugal. “Tensorflow Based Image Classification using Advanced Convolutional Neural Network”, *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, June 2020, pp. 994-998.
[32] A. Mishra. “Building a Deep Convolutional Neural Network with Keras”, *Machine Learning for iOS Developers*, pp. 235-286.
[33] M.A. Mitrohin, Y.P. Schegolihin, D.O. Neshko, M.V. Semenkin, “An analysis of the OpenCV library’s solutions efficiency in the problem of license plate recognition”, *Engineering sciences. Computer science, computer engineering and control*, vol. 2(50), pp. 39-46. April 2019.
[34] MathWorks official website, Estimate Anchor Boxes From Training Data, Web: https://www.mathworks.com/help/vision/ug/estimate-anchor-boxes-from-training-data.html.