Organizational and Legal Aspects of Managing the Process of Recognition of Objects in the Image

Nataliya Boyko[0000-0002-6962-9363], Lesia Mochurad[0000-0002-4957-1512],

Iryna Andrusiak[0000-0001-6887-0510], Yurii Drevnytskyi[0000-0001-6481-380X]

Lviv Polytechnic National University, Lviv79013, Ukraine
nataliya.i.boyko@lpnu.ua, lesia.i.mochurad@lpnu.ua,
airyna2016@gmail.com, yuriytom81@gmail.com

Abstract. The issue of object recognition using ANN models is considered. The object of training is the YOLO approach for recognizing objects in an image. The subject of training is Keras, TensorFlow, and the ability to create and explore ANN models using them. The purpose of the work is to write a program that recognizes certain objects in the images and learn how it works according to the YOLO approach. The analysis of the YOLO approach of object recognition on images is carried out. An example of its use for recognizing objects on a student card: a barcode, a seal and a signature is given. The high-level Keras API and the TensorFlow library were used to build the ANN architecture to build and work with the computation graph. An analysis of LeNet-5, AlexNet, GoogleNet architectures was performed, while building ANN's own architecture and analysis of the YOLO approach for object recognition, a program for object recognition in an image was written using Python, TensorFlow, Keras, and TensorBoard for visualization of training and architecture artificial neural network. YOLO's approach to image recognition is explored. I have better studied TensorFlow, Keras for constructing and exploring ANN and TensorBoard models to visualize the training process of ANN, the graph of calculations. Gained practical skills in writing ANN, and their practical application. Has deepened his knowledge in the field of machine learning. The hardest part of the network was learning to recognize object sizes.

Keywords: object recognition, ANN, YOLO, TensorFlow, Keras.

1 Introduction

There are many problems that have different arithmetic saturation. Some of them are easier to tell to computers: performing arithmetic operations, while others can be solved mentally by language recognition, image analysis, object classification and more. The advantage of solving tasks using computers is that they can perform known arithmetic and trigonometric operations sequentially and without fail. In addition to the usual processing of algebraic tasks, object recognition tasks are added. Known object recognition algorithms for an image usually have two parts: localization - de-

 CybHyg-2019: International Workshop on Cyber Hygiene, Kyiv, Ukraine, November 30, 2019.

termining the location of the object in the image, and classification - determining what it is for the object, ie to which class it belongs. This paper describes how to use a CNN convolutional neural network and the YOLO algorithm to identify individual objects in a static image. To analyze the task at hand, it is first scanned for computer vision.

2 Setting the Task

The task is to explore the YOLO approach for recognizing objects in images and writing implementation using API TensorFlow and Keras in Python. The main task is to train the model from scratch, using the YOLO approach to identify individual objects, for example: a barcode, a seal and a signature on a student card. Achieve at least 80% average accuracy on test data. To solve it, you need to solve the following tasks:

- explore the YOLO approach;
- learn the basics of Keras, TensorFlow;
- write your own implementation of the YOLO approach from scratch and the ANN model architecture;
- create a dataset of different student card photos;
- mark the dataset: barcode, seal and signature on the images;
- write an algorithm for automatic generation of augmented data in YOLO format;
- train the model to an accuracy of at least 80% on the test data.

3 Methods of Solving

Conditionally, the recognition algorithm can be divided into 2 components:
1) Localization – determine the coordinates and dimensions of the object:

Input is an image, first there are important features of the image, then there is a function of dependence between the image features and the coordinates of the center, the height and width of the object.
2) Classification – the definite on to which class an object be longs:
Input is a localized object, first the features of the object are found, then there is a function of dependency between the features of the object and the class to which it belongs.

We chose the YOLO (you only look once) architecture because it combines 2 steps of recognition – localization and classification. Due to the fact that all recognition is performed by one net work, it can be optimized specifically for recognition efficiency: “A single neural network predicts bounding boxes and class probabilities directly from full image simonies valuation. Since the whole detection pipe line is a single network, it can be optimized end-to-end directly on detection performance” [8-13].

This approach optimizes the speed of the algorithm, because the attributes of object classes are determined by one network with the attributes of its location and size:

“Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors” [1-4].

“Finally, YOLO learns very general representations of objects. It outer - forms all other detection methods, including DPM and R-CNN, by a wide margin when generalizing from natural images to artwork on both the Picasso Dataset and the People-Art Dataset” [6].

We changed some parts of the algorithm to accomplish this task: since there is only one copy of each student card per class, there is no need to use non max suppression to determine one final prediction among others. It is enough to take 1 bbox for each class with the maximum level of confidence for that class.

Our task is easier than recognizing any objects - the objects are in static positions relative to each other, only the position of the student card changes, 1 image of the student card has only 1 copy of each class. So we decided to use a model with far fewer parameters than the YOLO model. In addition, it is my task to train the network from scratch, and training such a large architecture as the YOLO model is a very complex process that requires a lot of data, time and computing resources [10-14].

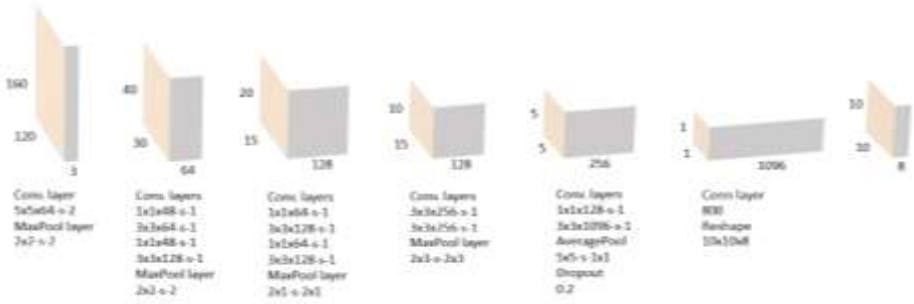

Fig. 1. ANN architecture

First two digits - kernel size, third digit –amount of filters, s - strides (horizontal, vertical).

Typically, three modes are used for ANN research:

- Training - to increase the accuracy of object recognition.
- Predictions - for object recognition.
- Calculation - to calculate certain functions that evaluate the operation of the algorithm.

This is the input to the algorithm for further processing. Input data vary depending on the mode of operation. Training: The training input has two components (Features, Labels).

Features are the images that you want to recognize. Used to calculate network output - location, size, and feature class predictions.

Filed in [N x W x H x channels], where: N is the number of images; W, H is the length and height of the image in pixels; channels - number of image channels (3 for RGB images) [18-21].

Labels are the true coordinates of the object center, their size, and the class indexes to which they belong. They are used by the error, accuracy, and other functions that are intended to evaluate the operation of the algorithm by determining the difference between the correct data and those predicted by the algorithm.

According to the YOLO approach, the input image is divided by a grid into a grid of [S x S] cells, each cell having the same size, and is responsible for recognizing a specific area of the input image:

Fig. 2. Divide the input image into cells

Each cell consists of [B] bounding boxes, to locate the object and [C] the probabilities of the object belonging to a particular class Pr ($Class_i$ | Object) for its classification (C is the number of classes). Only one object in a class can recognize a cell - the probability of which among the C probabilities of all classes is the highest. Bouldering box is a vector of format numbers (x, y, w, h, confidence) where: (x, y) - the coordinates of the bbox center are calculated as the offset from the coordinates of the lower left corner of a particular cell of the image, so they take values between 0 and 1; (w, h) are the length and width of the bbox, normalized to the size of the input image, so they take values between 0 and 1.

Fig. 3. Bbox one of the cells of the image (bbox border is red, cell grid is yellow)

Confidence - the level of confidence that the bbox really recognized the true object.

The confidence value reflects how confidently you can say that a given bbox has an object, and how true the output of that bbox is. The goal of dream faience is to show how true the results of the prediction are, with no ready answers like in training mode. If there are no objects in this bbox, confidence = 0, if any, confidence = IoU(intersection over union) between this bbox (predicted) and true bbox. confidence = probability (obj) * iou (pred, label) - formal definition of confidence [4-6].

Labels are fed to the algorithm's input in format [N x Sw x Sh x (B*5+C)] , where: N - number of images; Sw - the number of cells in the width of the grid; Sh - the number of cells in the height of the grid; B - the number of bboxes per cell; C - number of object classes.

To identify the barcode, stamp and signature on the student card I used: Sw = Sh =10, B = 1, C = 3. Foresight: features. Calculations: features, labels.

Example of normalization of coordinates and size of bbox (Fig. 4):

Fig. 4. Provided by the bbox object in the image (bbox borders are marked in red)

Coordinates of two predicted bboxes for 1 object (barcode) in an image size 128x128px (img_w, img_h = 128px, S=4):

[x, y, w, h]p1 = [47, 38, 66, 30] (px)

[x, y, w, h]p2 = [51, 39, 60, 29] (px)

Each cell size [cell_w x cell_h]: you should adjust the size of the image so that (img_w, img_h) is divided exactly by the number of cells (S) so that the cells cover the entire image.

cell_w = img_w//S – weight of cell

cell_h = img_h/S – high of cell

cell_w = 128/4 = 32(px)

cell_h = 128/4 = 32(px)

Coordinate normalization:

Xnorm = (x – ((x// cell_w) * cell_w)) / cell_w
Xp1_norm = (47 – ((47 // 32) * 32)) / 32 = (47-32) / 32 = 15 / 32 = 0.46875
Xp2_norm = (51 – ((51 // 32) * 32)) / 32 = (51-32) / 32 = 19 / 32 = 0.59375
Ynorm = (y – (y// S) * cell_h) / cell_h
Yp1_norm = (38 – ((38 // 32) * 32)) / 32 = (38-32) / 32 = 6 / 32 = 0.1875
Yp2_norm = (39 – ((39 // 32) * 32)) / 32 = (39-32) / 32 = 7 / 32 = 0.21875

Size normalization:

wnorm = w / img_w
wp1_norm = 66 / 128 = 0.515625
wp2_norm = 60 / 128 = 0.46875
hnorm = h / img_h
hp1_norm = 30/128 = 0.234375
hp2_norm = 29/128 = 0.2265625

Normalized bboxes:

[x, y, w, h]p1_norm = [0.46875, 0.1875, 0.515625, 0.234375]
[x, y, w, h]p2_norm = [0.59375, 0.21875, 0.46875, 0.2265625]

This is the data that results from processing the input algorithm. The output also differs depending on the mode of operation. Training: no initial data

Prediction: Output is the end product of the algorithm provided for each cell of an input image of an object's location and its class. The output dimension has the same format as the Labels input:

$$\left\lfloor N \times S_w \times S_h \times (B * 5 + C) \right\rfloor \quad (1)$$

Due to the fact that only one copy of each class is guaranteed to recognize objects on one student card, there is no need to use non max suppression. For each asset class, a bbox with a maximum confidence level greater than a certain confidence level of all bboxes in that class is selected. A class is defined as the index of the maximum value among the class predictions for a given grid cell, all the bboxes in the cell belong to that class (Fig. 5).

Calculation: Output is the value of certain metrics designed to evaluate the accuracy, object recognition error algorithm (IoU, max IoU, mean IoU, probability). Probability is a metric that combines the prediction of an object's coordinates with its class's predictions. This can be done by the following formula:

$$\text{Propability} = \Pr(\text{Class}_i \mid \text{Object}) * \Pr(\text{Object}) * \text{IoU}_{\text{tpred}}^{\text{ruth}} = \Pr(\text{Class}_i) * \text{IoU}_{\text{tpred}}^{\text{ruth}} \quad (2)$$

$\Pr(\text{Class}_i \mid \text{Object})$ - the probability of a particular class for an object in bbox, provided that it has that object.

Pr (Object) - the probability that a particular bbox has an object.

$\text{IoU}_{\text{tpred}}^{\text{ruth}}$ - the ratio of intersection to combining true bbox and predicted.

$\Pr(\text{Object}) * \text{IoU}_{\text{tpred}}^{\text{ruth}}$ - calculated as the confidence of a particular bbox.

This gives class-dependent confidence points for each bbox. Shows the full probability that an object of a particular class is in a particular bbox.

IoU is a feature that reflects the intersection of true and predicted bbox to merge. It is used to estimate the accuracy of the object algorithm localization. I use this feature because I need to evaluate the accuracy of object localization by an algorithm to understand how well it works.

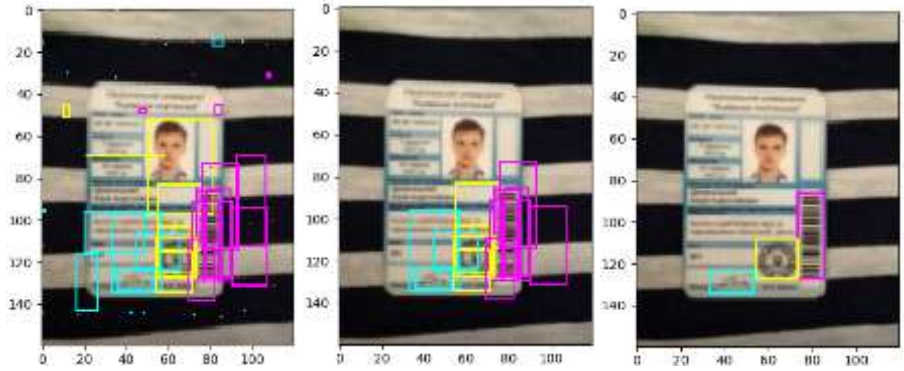

Fig. 5. Choosing the best predictions From left to right: all predictions, bbox with confidence> 0.5, max bbox with confidence> 0.5

Fig. 6. Visualization of IoU function

IoU = I/U, where: I - is theater section of true bbox and predicted network; U - is the combining true bbox and predicted network. “At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall”.

Example of calculating IoU:

Fig. 7. Provided and true bbox facility Green rectangle - true, red - provided.

IOU calculations for one of the predicted bbox and true bbox:
[x, y, w, h]p1 = [47, 38, 66, 30](px) - The coordinates of a true rectangle
[x, y, w, h]t = [48, 38, 67, 34](px) - The coordinates of the predicted rectangle
x1 = x-w/2; y1 = y-h/2 - bottom left bbox point
x2 = x+w/2; y2 = y+h/2 – upper right bbox point
x1p1 = 47-66//2 = 14
y1p1 = 38-30//2 = 23
x2p1 = 47+66//2 = 80
y2p1 = 38+30//2 = 53
x1t = 48-67//2 = 15
y1t = 38-34//2 = 21
x2t = 48+67//2 = 81
y2t = 38+34//2 = 55
Intersection coordinates:
(x1i , y1i); (x2i , y2i) - lower left and upper right point of intersection rectangle
x1i = max(x1p1 , x1t)
y1i = max(y1p1 , y1t)
x2i = min(x2p1 , x2t)
y1i = min(y2p1 , y2t)
x1i , y1i = 15, 23
x2i , y2i = 80, 53
intersection area:
intersection = max(0, (x2i - x1i))*max(0, (y2i - y1i))
union area:
union = Wp1 * hp1 + Wt* ht– intersection
IoU = intersection/union

intersection = max(0, (80- 15))*max(0, (53- 23)) = 65*30 = 1950
union = 66*30 + 67*34 - 1950 = 2308
IoU = 1950/2308 ~ 0.845

Mean IoU is a value for understanding how well the network localizes objects for the image package. It is calculated as the ratio of the sum to the number of function values IoU among all provided bbox for all images in 1 mini-package. Only the IoU values from the predicted bbox that are responsible for recognizing the object in the cell and the cell responsible for recognizing the object for that image are taken into account. This value is important because during training the algorithm is unstable and the value of the IoU function may vary greatly for an individual object. Therefore, to get a better idea of the accuracy of object recognition, IoU values are averaged over the entire mini-package.

$$Mean\_IoU = (\sum_{e}^{N}\sum_{i}^{S}\sum_{j}^{B} k_{ij}^{obj} * IoU_{pred}^{truth}) / obj\_num \quad (3)$$

N – the number of images in the mini-package;
S – the length of the grid in the cells;
B – number of bboxes;
truth – true bbox;
pred – prediction bbox;
k_{ij}^{obj} - from item 1 of section "YOLO error function";
obj_num - the number of true objects for all images in the mini-package.

MaxIoU – this value gives the numerical characteristic of localization accuracy for the best recognition cases. It can be compared to mean IoU to understand how the best IoU values for the best cases of localization differ from the average IoU values. It is calculate das the maximum value of the IoU function among all bbox predictions for mini-package images.

$$Max\_IoU = Max_N(IoU_{pred}^{truth}) \quad (4)$$

N – the number of images in the mini-package;
truth – true bbox;
pred– prediction bbox.

The YOLO error is a measure of how far network predictions differ from true values. The error function enables the network to learn. That is, it directly influences the learning process, thanks to which the optimization algorithm trains the network, changing its parameters in the direction of reducing the error. Without it, the network cannot learn to recognize objects, because without knowing what and how wrong the network will not be able to fix it.

Modified standard deviation between network prediction and true values is used to train the network. It is easy to optimize, but it is not well suited to the main goal of the network - maximize average accuracy because it equally evaluates localization and classification errors, which can be very different. Also, in each image, many cells are

not responsible for object recognition. This causes the confidence equation for the bbox of this cell to 0, often over saturating the cell gradients responsible for recognition. It can lead to instability of the network model, and cause early training discrepancies, the model will find a poor local minimum.

To prevent this, the error for bbox coordinates is increased, and for cells with no objects the confidence error is reduced, 2 parameters are added for this: $L_{coord} = 5$ and $L_{noobj} = 0.5$.

The standard deviation also equates to errors in large and small bboxes. The error should reflect that small deviations in large bboxes are less significant than in small bboxes. For this, the square root is taken from the length and height of the bbox.

"We optimize for sum-squared error in the output of our model. We use sum-squared error because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal.

Also, in every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on.

To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects. We use two parameters, λ_{coord} and λ_{noobj} to accomplish this. We setv $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$.

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly".

In YOLO, the error function consists of 4 parts, since the network output consists of several parts:

- Coordinates of the bbox center.
- Height and width bbox.
- Confidence for bbox.
- The likelihood that an object in a given cell belongs to a particular class if its center hits the cell.

Consider the individual parts of the error and an example of how to calculate it for my task: Parameters for example (Fig. 8).

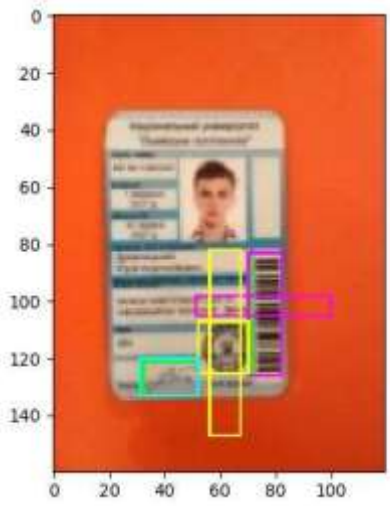

Fig. 8. Image for example of calculation of YOLO error green is indicated by true bbox, others include different classes

Image height and width = 160, 120 px respectively, S = 10, 3 classes of objects, consider b = 2 to account for a case with multiple bboxes in one cell. Also, for simplicity, let's consider only those cells that should have an object. (values are rounded for visual clarity).

True bboxes:

[0.375, 0.53125, 0.09167, 0.26875, 0.1, 1, 0, 0] - bbox1 bar code
[0.375, 0.53125, 0.09167, 0.26875, 0.935, 1, 0, 0] - bbox2 bar code
[0.0833, 0.25, 0.15, 0.1125, 0.95, 0, 1, 0] - bbox1 seal
[0.0833, 0.25, 0.15, 0.1125, 0.234, 0, 1, 0] - bbox2 seal
[0.5, 0.9375, 0.167, 0.075, 0, 0, 0, 1] - bbox1 signature
[0.5, 0.9375, 0.167, 0.075, 0.873, 0, 0, 1] - bbox2 signature

Prediction bboxes:

[0.363, 0.39, 0.409, 0.0414, 0.064, 1.01, 0.00491, 0.023] - bbox1 bar code
[0.36, 0.552, 0.097, 0.267, 0.757, 1.01, 0.00491, 0.023] - bbox2 bar code
[0.11, 0.258, 0.15, 0.11, 0.75, 0.0094, 0.994, 0] - bbox1 seal
[0.18, 0.196, 0.089, 0.4, 0.248, 0.0094, 0.994, 0] - bbox2 seal
[0.156, 0.136, 0.0657, -0.046, 0.0102, -0.0183, -0.0095, 1.033] - bbox1signature
[0.513, 0.96, 0.176, 0.0814, 0.69, -0.0183, -0.0095, 1.033] - bbox2 signature

The value of - 0.046 issued by the network as the height of bbox1 for the signature is converted to 0 because the height cannot be negative.

IoU: [0.1, 0.935, 0.95, 0.234, 0.0, 0.873] is the IoU value of true and predicted bbox for each object (used as true confidence).

1) Error for coordinates

This is the value that characterizes the difference between the coordinates of the bbox center $(\hat{x}_i, \hat{y}_i)$ predicted by the network, and true (x_i, y_i). Its purpose is to

locate the object on the image by a network, since it requires the coordinates of its center. It is calculated only for the bbox that are responsible for recognition (if their iou is maximal among other bboxes in this grid cell):"It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell)".

$$l_{coord} * \sum_{i}^{S} \sum_{j}^{B} k_{ij}^{obj} * \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \quad (5)$$

x_i, y_i - true coordinates of center bbox;

$\hat{x}_i, \hat{y}_i$ - predicted coordinates of center bbox;

k_{ij}^{obj} - coefficient equal to 1 or 0: l - when j - bbox is responsible for its recognition, ie its value for IoU is maximal among other j - bboxes in this i - cell. In addition, the cell should be responsible for recognition (if it falls into the center of the object).

0 - otherwise.

l_{coord} = 5 - the magnitude of the increase in error for the coordinates and size

loss_coords =
5*(
0 *...
+ 0*((0.375-0.363)2 + (0.53125-0.39)2)
+ 1*((0.375 - 0.36)2+(0.53125 - 0.552)2)
+0*…
+ 1*((0.0833-0.11)2 + (0.25-0.258)2)
+ 0*((0.0833 - 0.18)2+(0.25 - 0.196)2)
+ 0*…
+ 0*((0.5-0.156)2 + (0.9375-0.136)2)
+ 1*((0.5 - 0.513)2+(0.9375 - 0.96)2)
+ 0*...
) ~ 0.01054

2) Error for width and height

This is the value that characterizes the difference between the size of a bbox, its width and the length provided by the network $(\hat{w}_i, \hat{h}_i)$, and the true ones (w_i, h_i). Its purpose is to locate the object on the image by the network, since it requires its length and width. Size and coordinate errors are only calculated for the bbox that are responsible for the recognition.

$$l_{coord} * \sum_{i}^{S} \sum_{j}^{B} k_{ij}^{obj} * \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \quad (6)$$

(w_i, h_i) - true width and height of bbox;

$(\hat{w}_i, \hat{h}_i)$ - predicted width and height of bbox;

l_{coord} - from first error;

k_{ij}^{obj} - from first error.

loss_size =
5*(
+ 0*...
+ 0*((0.091671/2 - 0.4091/2)2 + (0.268751/2 - 0.04141/2)2)
+ 1*((0.091671/2-0.0971/2)2 + (0.268751/2- 0.2671/2)2)
+ 0*...
+ 1*((0.151/2 - 0.151/2)2 + (0.11251/2 - 0.111/2)2)
+ 0*((0.151/2-0.0891/2)2 + (0.11251/2- 0.41/2)2)
+ 0*...
+ 0*((0.1671/2 - 0.06571/2)2 + (0.0751/2 - 01/2)2)
+ 1*((0.1671/2- 0.1761/2)2 + (0.0751/2- 0.08141/2)2)
+ 0*...
) ~ 0.00171

The error should reflect smaller deviations in large bboxes less than in small ones, so it takes root from height and width: “Our error metric should reflect that small deviations in large boxes matter less than in small boxes”. This is well illustrated in the example below:

Height of the first and second, width of the third object reduced by 0.1 for true bbox and those responsible for recognition.

loss_size_smaller =
5*(
0*...
+ 0*((0.091671/2 - 0.4091/2)2 + (0.168751/2 - 0.04141/2)2)
+ 1*((0.091671/2-0.0971/2)2 + (0.168751/2- 0.1671/2)2)
+ 0*...
+ 1*((0.151/2 - 0.151/2)2 + (0.01251/2 - 0.011/2)2)
+ 0*((0.151/2-0.0891/2)2 + (0.01251/2- 0.41/2)2)
+ 0*...
+ 0*((0.0671/2 - 0.06571/2)2 + (0.0751/2 - 01/2)2)
+ 1*((0.0671/2- 0.0761/2)2 + (0.0751/2- 0.08141/2)2)
+ 0*...
) ~ 0.00317

3) Confidence level error:

“Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts”.

“If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IoU) between the predicted box and the ground truth”.

Therefore, the error reflects the difference between the real confidence value (iou) and the predicted network (c). The purpose of her finding is to understand how true are the results of the prediction of not having ready answers as in training.

$$\sum_{i}^{S^2}\sum_{j}^{B} k_{ij}^{obj} * (iou_i - c_i)^2 + l_{noobj} * \sum_{i}^{S^2}\sum_{j}^{B} k_{ij}^{noobj} * (iou_i - c_i)^2 \quad (7)$$

c_i - predicted confidence;

iou_i - IoU max(bbox) i – cells, because true confidence equals iou;

k_{ij}^{obj} - from first error;

k_{ij}^{noobj} - opposite to k_{ij}^{obj};

l_{noobj} = 0.5 - trust coefficient for bbox not responsible for object recognition. Their confidence for him should be 0. I think the coefficients for confidence errors are smaller than the coefficients for sizes and coordinates, since this error depend son the IOU value that changes dynamically in the learning process. So that the network does not learn when iou values are small to predict only small values of confidence, and when iou grows up sharply not to predict only high values of confidence the learning factories reduced.

loss_confidence = (
0*...
+ 0*(0.1 - 0.064)2
+ 1*(0.935-0.757)2
+0*...
+ 1*(0.95 - 0.75)2
+ 0*(0.234-0.248)2
+0*...
+ 0*(0.0 - (-0.046))2
+ 1*(0.935-0.873)2
+ 0*...
) +
0.5*(
1*...
+ 1*(0.1 - 0.064)2
+ 0*(0.935-0.757)2
+ 1*...
+ 0*(0.95 - 0.75)2
+ 1*(0.234-0.248)2
+ 1*...
+ 1*(0.0 - (-0.046))2
+ 0*(0.935-0.873)2
+ 1*...
) ~ 0.077 + 0.011 = 0.088

4) Classification error:

Object network classification error Required to improve the network's ability to classify objects.

$$\sum_{i}^{S^2} k_{ij}^{obj} * \sum_{c}^{classes} (p_i(c) - \hat{p}_i(c))^2 \quad (8)$$

$p_i(c)$ - is there an object class in this cell (1 or 0);

$\hat{p}_i(c)$ - class prediction for a given cell;

k_{ij}^{obj} - coefficient equal to 1 or 0:

1 - when the center of the object enters the i - cell.

0 - otherwise.

"Note that the loss function only penalizes classification error if an object is present in that grid cell"

loss_class= (
0*...
+ (1 - 1.01)^2 + (0 - 0.00491)^2 + (0 - 0.023)^2
+ 0*...
+ (0 - 0.0094)^2 + (1 - 0.994)^2 + (0 - 0)^2
+ 0*...
+ (0 -(-0.0183))^2 + (0 - (-0.0095))^2 + (1 - 1.033)^2
+ 0*...
) ~ 0.00229

This is the value that characterizes the overall deviation of the present network performance from the desired results. It has the effect of further optimizing the network. The next step in the network operation is to calculate the dependence of the error function on the network parameters, then the parameters change in the direction of reducing the error.

The final error is the sum of four errors: 1) coordinate errors, 2) width and height, 3) confidence level, 4) classification. For the task - localization of bar code on the student card, classification is not required, so the classification error is not included in the general error:

Loss = 0.01054 + 0.00171 + 0.088 + 0.00229 ~ 0.10254

Therefore, the final recognition error consists of the error of coordinate recognition, size recognition, prediction of how true this recognition is, and errors of object class definition. The purpose of the training is to increase the overall accuracy of object recognition by reducing the overall recognition error.

4 Experiments

To mark the data, we applied bbox boundary markers as rectangles of different colors for each class, and saved the coordinates, sizes, classes of these rectangles as true

output for the Labels algorithm by pre-converting them to YOLO (not changes). I used [16] to mark the data:

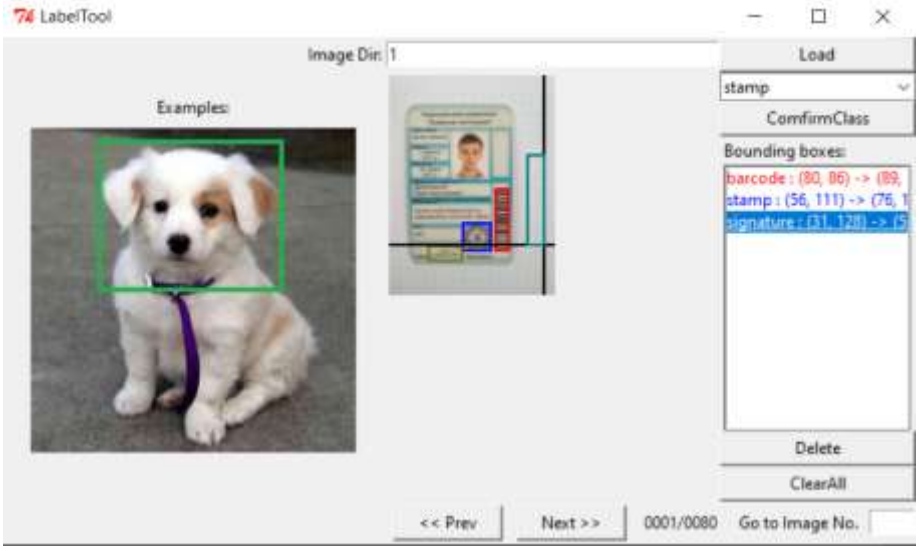

Fig. 9. Example of image markup

We trained a network with the following parameters: The total size of the training data sets 80 images. During training, I used data augmentation – certain changes to the input data in order to ensure that the algorithm works for noisy data, as well as to better study the patterns in the data. For example, in order for an algorithm to be able to recognize objects on a student card, if it sin the rotate deposition, pseudo random rotations in the training data are used. For better recognition of images made at an angle - slopes, images that are not in the center of the image - offsets, for images with different levels of brightness - fluctuations in brightness and hsv - transformation. After augmentation training set size - 720 images After each epoch, the initial 80 images are augmented up to 720 by pseudorandom translating, rotating, scaling, shearing, hsv image transformations. For this I used api [16]. Batch size - 80 images, the training era consists of 9 mini-packages (9 step / epoch).

5 Results

We managed to reach mean IoU of ~ 85% on test data over 400 training periods. At the end of each training period, the values and metrics described above were calculated and stored to represent the state of the network and the training process as a whole. For graphing I used [14]. Results for 400 training periods:

The x-axis is the index of the era, the y-axis is the value of the function

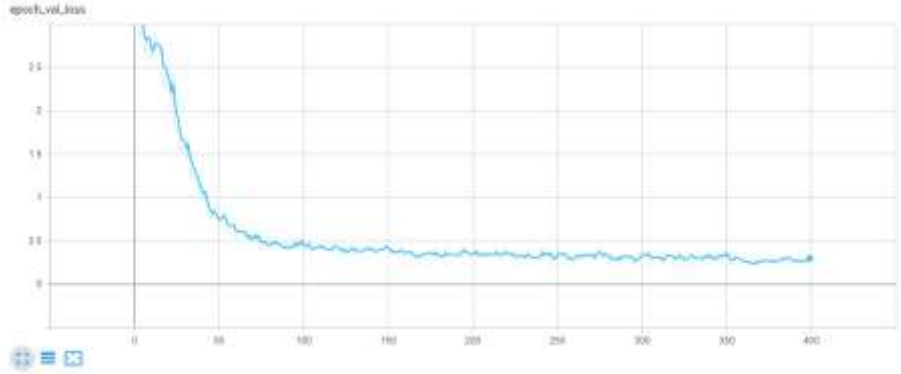

Fig. 10. General error function

The graph shows that at the initial stage of training, the error dropped rapidly, gradually its rate of decline decreased. In my opinion, this is due to the fact that certain features of the objects are easier to learn, others harder. It is also a result of the gradient damping phenomenon - in the first stages, the initial layers are rapidly training and the distant layers are slower because they have smaller gradients. Then the distant layers are slowly refinished, and the initial layers are adjusted, partly because the error is constantly fluctuating, but as a result it is reduced.

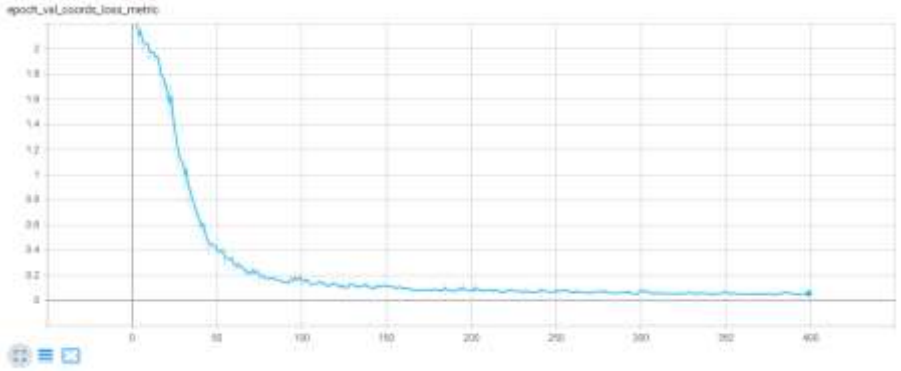

Fig. 11. Coordinate recognition error function

The graph shows that the network quickly learned to recognize the coordinates of the bbox center. I think this is because all objects look like simple geometric shapes, and they are the same color. However, when resizing go rotating the image during augmentation, the center of the object does not change.

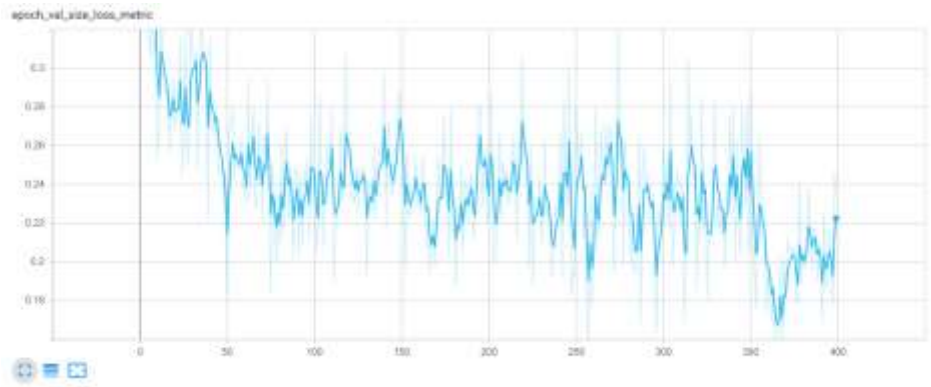

Fig. 12. Size recognition error function

The graph shows that the size recognition error decreases a there slowly and is unstable compared too there parts of the YOLO error. I think this is due to the fact that as a result of augmentation, the shape and size of the objects in the input data is constantly changing, so the network needs more time to study these patterns.

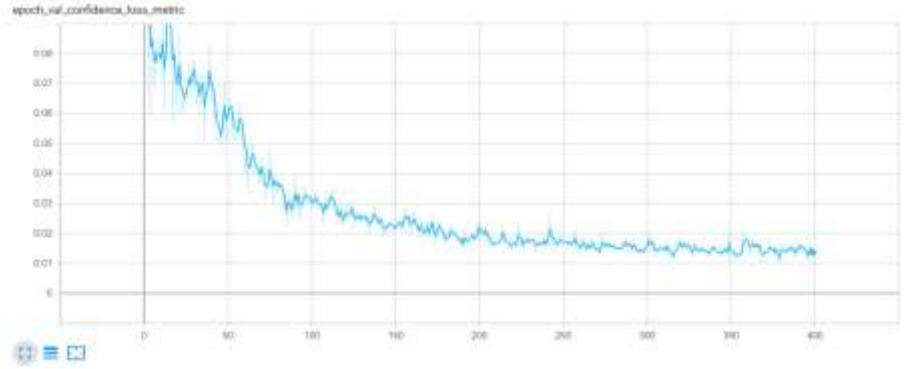

Fig. 13. Trust recognition error function

The chart shows that the confidence level error from the outset was relative to other parts of the YOLO error. I believe that this is due to several factors - it depends only on one value (confidence), and the size error (w, h), coordinates (x, y) - two, and its part is multiplied by a factor of 0.5. Also, since all recognized objects are similar to simple geometric shapes in a network, it is easier to understand if an object is in a particular cell.

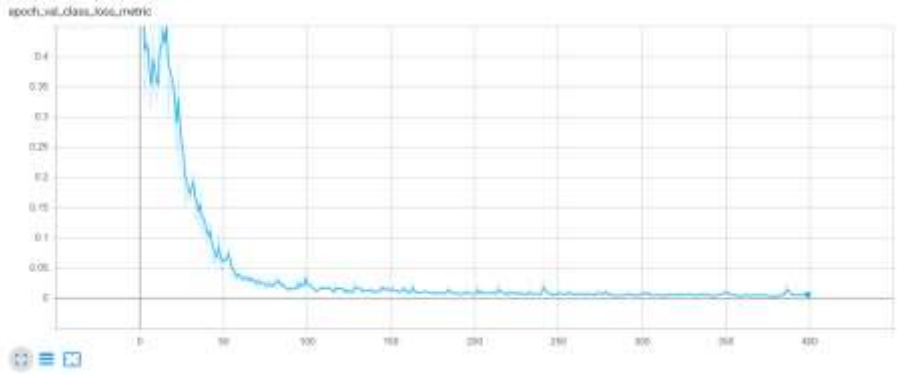

Fig. 14. Class recognition error function

The graph shows that the confidence level error decreases rapidly at the beginning of training. Further, the rate of decrease is slowed. I think this is due to the fact that the position of different classes of objects does not change relative to each other (left signature, then seal, then barcode) so it is easy to distinguish them by this feature. And slowing down the error reduction rate is because when other parts of the YOLO error are much larger, the optimization algorithm changes the network parameters more strongly to reduce them.

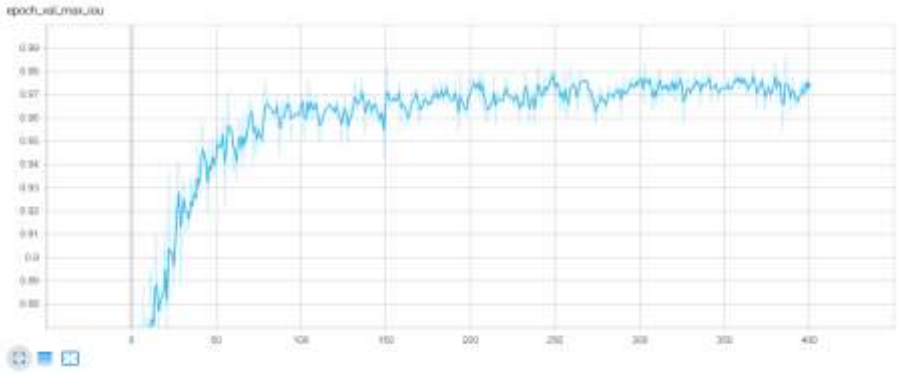

Fig. 15. Max value of IoU

The graph shows that the maximum IoU value is growing faster and more than the average IoU value. I think that because some objects area sire to localize than others, an object whose harpies very similar to a simple geometric shape (rectangle) is easier to localize than an object with a complex shape – because any complex shape can be represented as a combination of simple forms. It takes a time for the network to learn to do this.

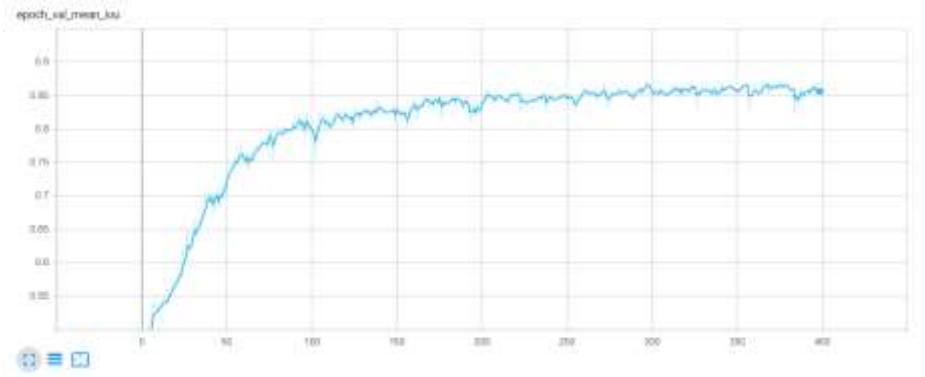

Fig. 16. Average value of IoU

The graph shows that at the beginning of training, the mean value of IoU increased rapidly, and gradually the rate of increase of iou decreased. In my opinion, this is due to the fact that the localization error initially decreased significantly, and the average IoU increased accordingly. Then the rate of decrease of the error gradually decreased, respectively, and the rate of increase of the average IoU decreased. It can be concluded that the localization error and the mean IoU are inversely proportional.

6 Conclusion

We achieved the task (Achieve at least 80% average accuracy on the test data) using the algorithm describe dab vein 4h 10min using the Intel (R) Core (TM) i5-8300H CPU @ 2.30GHz 2.30 GHz to train the network. With the YOLO approach, you can train you net work to recognize objects using a relatively small amount of computing power within a reasonable time. The speed of training depends strongly on the objects of recognition and image parameters: the complexity of the shape of the object, the possibility of the presence of several objects of the same class in the image, the number of classes of objects, or easily distinguish them, camera angle, change of illumination, distance to the object , placing the object in the image. After all, training a network for object recognition under different conditions requires a larger data set to train and test, to train the network on a larger set takes longer. Also, a more complex ANN model may be required to analyze more complex data. Therefore, such details should be determined in the first phase of building the object recognition system. In order to evaluate the network effectively, it is necessary to select the appropriate metrics (in this case IoU, yolo_loss). Selecting an error function by minimizing which will maximize recognition accuracy. Combining localization and classification steps enables the recognition of 1-step ANN calculations and simplifies the design of algorithm input. At the same time, the recognition process is optimized, since the same features of the object studied by the same network are used for localization and classification.

Research materials: labeled data set, trained model, references to useful resources - can be used for future research, and research in general - an example for future generations, which they can refine, refine. The trained model can be used to recognize barcodes, prints, signatures on an image, and then transfer them to other data processing algorithms. For example, a barcode can be transmitted to a barcode reader to obtain student information, and a stamp and signature can be transmitted to document validation algorithms.

Thanks to the development of machine learning, we have received real-time object recognition tools with fairly high precision. This facilitates process automation, as work related to the analysis of visual images by humans can be partially translated to a computer. Further research is promising, as it can increase the amount of computer-translated work and thus free people from monotonous work. To do this, the accuracy and speed of the algorithms must be at least as human as possible.

References

1. Kang, H.-Y., Lim, B.-J., Li, K.-J.: P2P Spatial query processing by Delaunay triangulation, Lecture notes in computer science, vol. 3428, pp. 136–150, Springer/Heidelberg (2005)
2. Boehm, C., Kailing, K., Kriegel, H., Kroeger, P.: Density connected clustering with local subspace preferences, IEEE Computer Society, Proc. of the 4th IEEE Intern. conf. on data mining, pp. 27–34, Los Alamitos (2004)
3. Wang, Y., Wu, X.: Heterogeneous spatial data mining based on grid, Lecture notes in computer science, vol. 4683, pp. 503–510, B.: Springer/Heidelberg (2007)
4. Kryvenchuk Y., Boyko N., Helzynskyy I., Helzhynska T., Danel R.: Synthesis control system physiological state of a soldier on the battlefield. CEUR. Vol. 2488. Lviv, Ukraine, p. 297–306. (2019)
5. Harel, D., Koren, Y.: Clustering spatial data using random walks, Proc. of the 7th ACM SIGKDD Intern. conf. on knowledge discovery and data mining, pp. 281–286, San Francisco, California (2000)
6. Gahegan, M.: On the application of inductive machine learning tools to geographical analysis, Geographical Analysis, vol. 32, pp. 113–139 (2000)
7. Boyko, N., Shakhovska, Kh., Mochurad, L., Campos, J.: Information System of Catering Selection by Using Clustering Analysis, Proceedings of the 1st International Workshop on Digital Content & Smart Multimedia (DCSMart 2019), pp. 94-106, Lviv, Ukraine (2019)
8. Boyko, N., Komarnytska, H.,Kryvenchuk ,Yu., Malynovskyy, Yu.: Clustering Algorithms for Economic and Psychological Analysis of Human Behavior, Proceedings of the International Workshop on Conflict Management in Global Information Networks (CMiGIN 2019), pp. 614-626, Lviv, Ukraine (2019)
9. Kryvenchuk Y., Vovk O., Chushak-Holoborodko A., Khavalko V., Danel R.: Research of servers and protocols as means of accumulation, processing and operational transmission of measured information. Advances in Intelligent Systems and Computing. Vol.1080. p.920-934. (2020)
10. Zhang, C., Murayama, Y.: Testing local spatial autocorrelation using, Intern. J. of Geogr. Inform. Science, vol. 14, pp. 681–692 (2000)
11. Estivill-Castro, V., Lee, I.: Amoeba: Hierarchical clustering based on spatial proximity using Delaunay diagram, 9th Intern. Symp. on spatial data handling, pp. 26–41, Beijing, China (2000)

12. Turton, I., Openshaw, S., Brunsdon, C.: Testing spacetime and more complex hyperspace geographical analysis tools, Innovations in GIS 7, pp. 87–100, London: Taylor & Francis (2000)
13. Fedushko S., Syerov Yu., Skybinskyi O., Shakhovska N., Kunch Z. (2020) Efficiency of Using Utility for Username Verification in Online Community Management. Proceedings of the International Workshop on Conflict Management in Global Information Networks (CMiGIN 2019), Lviv, Ukraine, November 29, 2019. CEUR-WS.org, Vol-2588. pp. 265-275. http://ceur-ws.org/Vol-2588/paper22.pdf
14. Tung, A.K, Hou, J., Han, J.: Spatial clustering in the presence of obstacles, The 17th Intern. conf. on data engineering (ICDE'01), pp. 359–367, Heidelberg (2001)
15. Veres, O., Shakhovska N.: Elements of the formal model big date, The 11th Intern. conf. Perspective Technologies and Methods in MEMS Design (MEMSTEH), pp. 81-83, Polyana (2015)
16. Agrawal, R., Gehrke, J., Gunopulos ,D., Raghavan, P.: Automatic sub-space clustering of high dimensional data, Data mining knowledge discovery, vol. 11(1), pp. 5–33 (2005)
17. Shakhovska, N., Boyko, N., Zasoba, Y., Benova, E.: Big data processing technologies in distributed information systems. Procedia Computer Science, 10th International conference on emerging ubiquitous systems and pervasive networks (EUSPN-2019), 9th International conference on current and future trends of information and communication technologies in healthcare (ICTH-2019), Vol. 160, pp. 561–566, Lviv, Ukraine (2019)
18. Guimei, L., Jinyan, L., Sim, K., Limsoon, W.: Distance based subspace clustering with flexible dimension partitioning, Proc. of the IEEE 23rd Intern. conf. on digital object identifier, vol. 15. Iss. 20, pp. 1250–1254 (2007)
19. Aggarwal, C., Yu, P.: Finding generalized projected clusters in high dimensional spaces, ACM SIGMOD Intern. conf. on management of data, pp. 70–81 (2000)
20. Guo, D., Peuquet, D.J., Gahegan, M.: ICEAGE: Interactive clustering and exploration of large and high-dimensional geodata, Geoinfor-matica, vol. 3, N. 7, pp. 229–253 (2003)
21. Boyko, N., Basystiuk, O.: Comparison Of Machine Learning Libraries Performance Used For Machine Translation Based On Recurrent Neural Networks, 2018 IEEE Ukraine Student, Young Professional and Women in Engineering Congress (UKRSYW), pp.78-82, Kyiv, Ukraine (2018)
22. Procopiuc, C.M., Jones, M., Agarwal, P.K., Murali, T.M.: A Monte Carlo algorithm for fast projective clustering, ACM SIGMOD Intern. conf. on management of data, pp. 418–427, Madison, Wisconsin, USA (2002)
23. Ankerst, M., Ester, M., Kriegel, H.-P.: Towards an effective cooperation of the user and the computer for classification, Proc. of the 6th ACM SIGKDD Intern. conf. on knowledge discovery and data mining, pp. 179–188, Boston, Massachusetts, USA (2000)
24. Boyko N., Pylypiv O., Peleshchak Y., Kryvenchuk Y., Campos J.: Automated document analysis for quick personal health record creation. 2nd International Workshop on Informatics and Data-Driven Medicine. IDDM 2019. Lviv. p. 208-221. (2019)
25. Kryvenchuk Y., Mykalov P., Novytskyi Y., Zakharchuk M., Malynovskyy Y., Řepka M.: Analysis of the architecture of distributed systems for the reduction of loading high-load networks. Advances in Intelligent Systems and Computing. Vol.1080. p.759-550. (2020)