DOI 10.15276/imms.v14.no3.134
UDK: 681.518

Informatics and Mathematical Methods in Simulation
Vol.14 (2024), No. 3, pp. 134-145

A METHOD FOR IMPROVING THE QUALITY OF IMAGE ANNOTATION IN SEMANTIC MONITORING GIS OF BUSINESS PROCESSES

R.M. Pasichnyk, L.V. Babala, M.V. Machuliak

West Ukrainian National University
11, Lvivska Str. Ternopil, 46009, Ukraine
Emails: Roman.pasichnyk@gmail.com, Ludaduma7@gmail.com, Mvmach9@gmail.com

This article addresses the pressing issue of automating the image labeling process for computer vision systems in agriculture. The authors investigate methods for creating image datasets and configuring parameters for image classification models using neural networks based on the TensorFlow framework. The scientific significance of the work lies in developing new approaches to automated collection of thematic image collections and formalizing the methodology for parametric training of classification models. The practical value of the research is expressed in improving the efficiency of the image labeling process for geoinformation systems in the agricultural sector. The research methodology includes analyzing existing approaches to image labeling, developing an algorithm for automated formation of thematic image collections, formalizing a method for parametric training of the classification model, and experimental verification of the proposed approaches. Main results of the work: 1. An algorithm for automated formation of thematic image collections has been developed. 2. A method for parametric training of the image classification model using the TensorFlow framework has been formalized. 3. The dependence of classification accuracy on the size of the training sample and image augmentation parameters has been experimentally established. The study showed that with optimal selection of augmentation parameters and using 48 images per label in the training sample, it is possible to reduce the classification error to an acceptable level of 8%. The work makes a significant contribution to the development of automated image processing methods for agricultural geoinformation systems. The practical significance of the results lies in improving the efficiency of monitoring and management processes in the agricultural sector.
Keywords: computer vision, image labeling, neural networks, TensorFlow, geoinformation systems, agriculture, image classification.

Introduction. Modern agriculture faces the challenge of increasing production efficiency while reducing negative environmental impact. In this context, Geographic Information Systems (GIS) serve as a powerful tool revolutionizing the agricultural sector. GIS allows processing large volumes of geospatial data, creating detailed field maps, and analyzing various factors affecting plant growth, which in turn facilitates informed decision-making in agricultural production.

Particular attention in this study is given to the integration of computer vision methods into GIS for analyzing data obtained using drones. This technology allows for automatic identification and classification of objects in images, which significantly increases the efficiency of monitoring agricultural lands.

The article will examine modern approaches to implementing computer vision systems, particularly the application of convolutional neural networks, which provide hierarchical learning of features from basic (edges, corners) to complex (specific objects of interest).

Neural networks are an effective means of image classification. However, when using this apparatus, a number of difficulties arise. These lie in the great variety of network architectures and model training methods. The situation has been simplified with the introduction of the TensorFlow framework, where recommendations have been developed for certain subject areas regarding the choice of network architectures, methods for their rapid implementation, and effective training. However, researchers still face open questions about

selecting model parameters and automating the formation of image collections for model training. This work is dedicated to the study of these issues.

Fig. 1 shows the structure of the automated monitoring system for agricultural business processes:

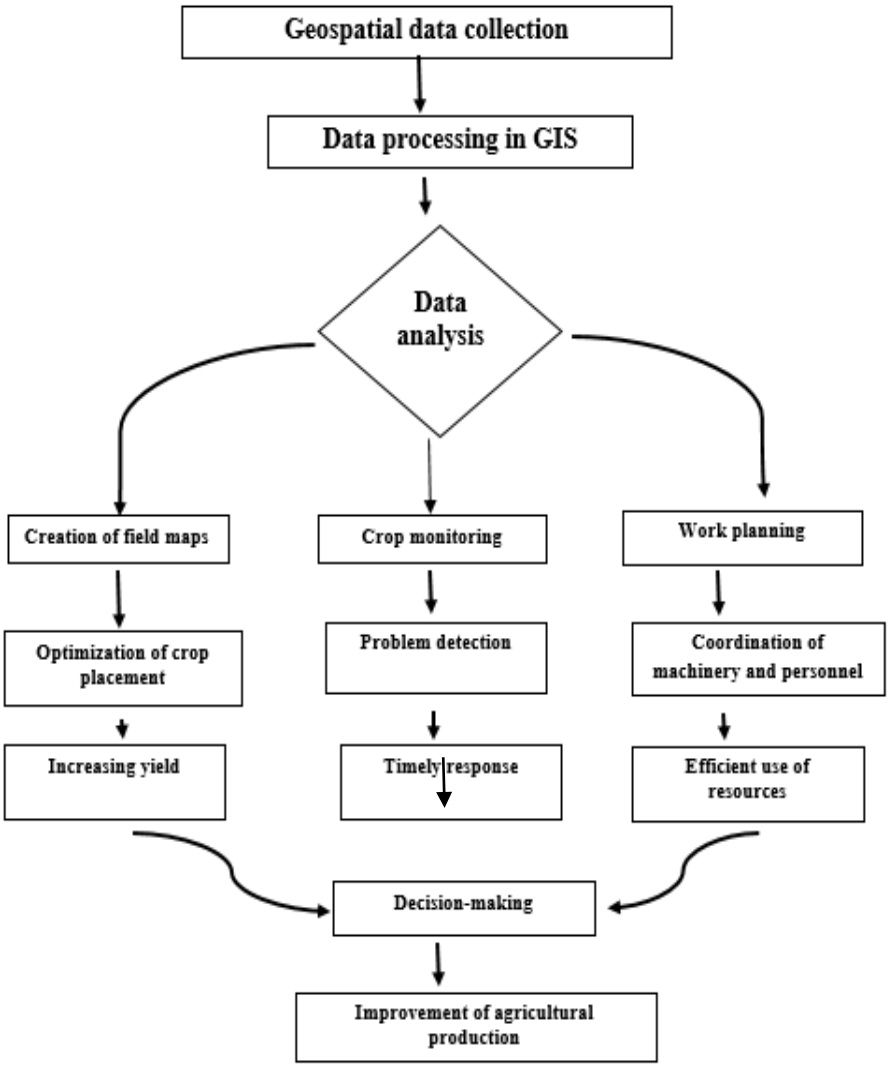

Fig. 1. Automated monitoring systems for agricultural business processes

Literature Review. Among the basic publications in the chosen research direction, we highlight reviews of approaches to image labeling problems [1] [2] [3] and approaches to building a system of semantic classes based on image segmentation [4], improving classification quality using cross-entropy similarity loss function [5], and building a semantic dictionary for image classification using deep learning methods [6].

In [1], basic approaches to forming semantic labels based on a keyword dictionary that solve the task of assessing the nature of images in the required aspect are analyzed. As these approaches are labor-intensive, various methods were used to reduce manual annotation costs. Three different types of image annotation are highlighted: free text annotation, keyword annotation, and ontology-based annotation.

Keyword annotation involves annotating images using a list of related keywords. There are two options for choosing keywords. In particular, this includes using arbitrary keywords, as well as using words from a predefined list. This information can be provided at two levels of specificity: firstly, a list of keywords related to the full image, listing what is depicted in the image. Secondly, image segmentation along with keywords associated with each segmentation area. Additionally, keywords describing the whole image can be provided. Often, segmentation

consists simply of a rectangular area drawn around the area of interest, or dividing the image into foreground and background pixels.

Ontological annotation is used when formed ontologies that cover a given subject area are available. For example, in the field of image description, ICONCLASS is a very detailed ontology for researching and documenting iconography images, designed to index or catalog the iconographic content of works of art, reproductions, literature, etc., and contains over 28,000 definitions organized in a hierarchical structure. Each definition is described by an alphanumeric code accompanied by a textual description.

For arbitrary text annotation, the user can use any combination of words or sentences. This facilitates annotation but complicates the use of annotation later for image search. Often this option is used in addition to keyword selection or ontology. Any concepts that cannot be adequately described using keyword selection are simply added in free-form description.

In [2], the search process in a system of medical articles is investigated. The developed labeled image dictionary is considered as the base vocabulary, images from which are compared with images from the analyzed document. Two image matching methods have been developed: one based on image intensity projections on coordinate axes, and the other on normalized cross-correlation. If image similarity thresholds are not reached, the analyzed image is skipped.

In [3], it is noted that to ease the load of manually labeling a large number of images, certain parts of the process can be automated where a computer vision algorithm performs preliminary annotation, and then a human user reviews and corrects the proposed labels. Accordingly, the human's role changes to supervision with the sole tasks of filtering, selection, and updating.

In the development of automated labeling, the methodology of interactive labeling is used, which is part of the "human-in-the-loop" methodology. Its goal is to reduce the limitations of fully automated labeling through purposeful interaction with the user. The method allows the user to label an initial batch of examples, trains a model, and then constantly asks the user for corrections. A developed strategy of active user feedback is used, which minimizes errors in subsequent labeling iterations and maximizes the expected information gain.

Works [4]-[6] are devoted to image labeling using semantic classes. In publication [4], the main idea is to identify areas that provide stable classification using an entropy measure. Minimizing entropy for different image segments gives its representation as a region adjacency graph. For each test image, generated contextual information is represented by a co-existence matrix.

In [5], the average similarity between the prediction and given labels is considered as a measure of semantic similarity in classification. With this type of evaluation, analysis of the loss function, which includes both cross-entropy similarity loss and deep feature loss, can improve the semantic similarity between the prediction and actual labels. It is shown that by analyzing the loss function, a model that produces more accurate predictions can be obtained. It is shown that cross-entropy similarity loss improves superclass similarity. It is proved that the average vocabulary similarity, which is a more accurate indicator, is also improved.

In [6], it is shown that although objects in the foreground of single-label images are one-level, this assumption is usually incorrect for multi-label images. Moreover, the different composition and interaction between objects in multi-label images also increases the informativeness of classifying such images. It is noted that multi-label image classification is more practical and complex than single-label image classification.

The paper presents a new end-to-end approach to multi-label image classification called Deep Semantic Dictionary Learning (DSDL), which considers the problem of multi-label image classification as a dictionary learning task. It uses an autoencoder to generate a semantic dictionary aligned with visual space, with class-level semantics. Unlike traditional approaches to multi-label image classification, DSDL not only uses correlations between label and vision spaces but also aligns relationships between label, semantic, and vision spaces.

Thus, some basic approaches to image labeling have been considered. Based on their generalization, we will form an approach to the announced automatic labeling within the framework of a geographic information system.

Semantic Label System. From the analysis of literature sources, it can be established that the basic methods of automatic labeling are based on autocorrelation image comparison or using artificial neural networks. In our view, neural networks are a more flexible tool that well supports the adaptation of the approach to detected deviations in semantic classification. Therefore, we prefer image labeling methods based on neural networks.

In particular, deep neural networks, especially convolutional neural networks (CNN), have achieved significant success in this field. Machine learning frameworks greatly simplify the process of creating and training neural networks. They provide ready-made tools, optimizations, and interfaces for interacting with data and models. In this regard, TensorFlow is one of the most popular machine learning frameworks developed by Google. It is particularly well-suited for working with deep neural networks, including convolutional neural networks. This involves the following main steps of using TensorFlow for image classification: data preparation, model formation, training, validation, and testing.

Storing trained models and quickly using them at the right moment with linking to corresponding geographical locations requires the development of a specialized geographic information system. Let's outline its basic structures that emerge from the types of information we plan to store.

From this perspective, we present the information models Im of the geographic information system being created as follows:

$$Im = \langle Bt, Ls, Gl, Md, Trs, \ Tss, Mit \rangle \quad (1)$$

where Bt - is the type of business process, Ls - is the image labeling system, Gl - is the geographic localization of the business Md - process, Trs - is the description of the method for building the image classification model, Tss - is the set of images for training, Mit - is the set of images for testing, are the types of images that the model classifies incorrectly.

$$Bt = \langle IdBt, \ NmBt \rangle \quad (2)$$

where, $IdBt$ - is the business process identifier, $NmBt$ - is the name of the business process.

$$Ls = \langle IdLi, \ NmLi, \Pr\_IdLi, IdBt \rangle \quad (3)$$

where, $IdLi$ - is the labeling element identifier, $NmLi$ - is the name of the labeling element, $\Pr\_IdLi$ - is a reference to the parent element of the hierarchy (identifier of the corresponding labeling element).

$$Gl = \langle IdLoc, \ NmLoc \rangle \quad (4)$$

where, $IdLoc$ - is the location identifier, $NmLoc$ - is the name of the location.

$$Md = \langle IdMd, \ TxtMd, \ IdBt \rangle \quad (5)$$

where, $IdMd$ - is the method identifier, $TxtMd$ - is the textual description of the method.

$$Trs = \langle IdTrs, TrPath, TrVol, Df, IdLi, IdBt \rangle \quad (6)$$

where, $IdTrs$ - is the identifier of the training image set, $TrPath$ - is the path to the training image set, $TrVol$ - is the volume of images in the training image set, Df - is the date and time of the image set formation.

$$Tss = \langle IdTss, TsPath, TsVol, Df, IdLi, IdBt \rangle \quad (7)$$

where, $IdTss$ - is the identifier of the testing image set, $TsPath$ - is the path to the training image set, $TsVol$ - is the volume of images in the testing image set.

$$Mit = \langle IdMit, IdLoc, IdLi, IdBt \rangle \quad (8)$$

where, $IdMit$ - is the identifier of the types of images that the model classifies incorrectly.

Method of Image Set Formation. After building structures for storing information, let's consider preparing a collection of images for training the neural network. We will start by building a list for labeling images from the subject area, entering it into the Ls structure. To build an image labeler for this subject area, we form a collection of images that will contain sets of images for each element of the constructed list. In the case of drones availability, the

image collection can be built from observations of objects that need to be labeled. If the system is just being formed, we create the image collection by extracting them or scraping them from the Web network.

Image scraping is the process of automatically collecting images from web pages. The Python library Beautiful Soup is often used for image extraction. It is characterized by ease of use, can work with various HTML and XML formats, and provides a wide range of methods for searching, navigating, and manipulating DOM elements.

To use the Beautiful Soup library, it is necessary to specify a Web page that contains suitable images for extraction. It can be selected by analyzing the output of a Web - search engine for a term query in the $NmLi$ image category.

Presenting the algorithm for scraping images from a multi-page structure in the form of certain stages. In particular, in the first stage, we set the tool, source, and parameters of scraping. In the second stage, we load the contents of the selected number of pages from a certain source and select the contents of image tags for a certain class from it. In the third stage, we select unique links to images from the selected tag content. And finally, in the fourth stage, we record images in a specified folder with corresponding names. Let's detail the implementation of these stages using the following steps and operators.

1. We form ***driver*** - the constructor of the *webdriver.Chrome()* class from the ***selenium*** library, which initializes a new instance of the web driver for the Chrome browser:

$$driver = webdriver.Chrome() \quad (9)$$

2. We fix ***url*** - a link to the site chosen for downloading and the ***npages*** we aim to download.

3. We fix ***class*** - the *css* class of image tags that contain links to the needed images

4. We organize a loop for the number of pages to download:

$$for\ page\ in\ range(page1, npage + 1) \quad (10)$$

5. We form ***url_page*** - a link to the next page:

$$url\_page = url + "?page = " + str(page) \quad (11)$$

6. Getting the contents of the next page:

$$driver.get(url\_page) \quad (12)$$

7. We analyze the contents of the page using an HTML parser and build a DOM - document object model:

$$content = driver.page\_source \quad (13)$$

$$soup = BeautifulSoup(content, "html.parser") \quad (14)$$

8. Loop through the elements of the image tag (IMG) with the specified class:

$$for\ image\ in\ soup.findAll("img", \{"class" : class\}) \quad (15)$$

9. Selecting a link from an element if it has not been used yet:

$$if\ image['src']\ not\ in\ results: \quad (16)$$

$$results.append(image['src']) \quad (17)$$

10. Loop through elements from downloaded images:

$$for\ b\ in\ results: \quad (18)$$

11. Getting an image by link:

$$image\_content = requests.get(b).content \quad (19)$$

$$image\_file = io.BytesIO(image\_content) \quad (20)$$

$$image = Image.open(image\_file).convert("RGB") \quad (21)$$

12. Writing the image to a specified folder with a numerical name in the order of download:

$$file\_path = Path("tractors\_plow2", str(numb).zfill(length) + ".png") \quad (22)$$

$$image.save(file\_path, "PNG", quality = 80) \quad (23)$$

$$numb = numb + 1 \quad (24)$$

Method Training Model Classification Image. We plan image scraping to have sufficient images for forming training and test sets for each marker NmLi of a specific business process NmBt. Thus, we create folders for individual business processes NmBt with subdirectories for text markers NmLi within them. For each marker, we form image directories for training Tr and testing Ts with corresponding volumes TrVol and TsVol. The number of such directories can be expanded with a link to the date and time of formation Df. Thus, paths to image directories for training Pitr(NmBt,NmLi,Df) and testing Pits(NmBt,NmLi,Df) are formed, which are recorded in the information system:

$$TrPath = Pitr(NmBt, NmLi, Df), \tag{25}$$

$$TsPath = Pits(NmBt, NmLi, Df). \tag{26}$$

After forming or replenishing sets of image directories, it is necessary to build and train the model according to a specific algorithm, which can be presented as the following sequence of stages. In the first stage, we set the values of the main algorithm parameters and form a dataset for model training. In the second stage, we create data processing pipelines that shuffle elements in the training and validation datasets. In the third stage, we create a neural network model for classification with random image transformations. In the fourth stage, we set the parameters of the method that regulates the learning process, carry out the learning process, and save the resulting model. The stages are implemented through the following steps:

1. Set the number of epochs, i.e., cycles during each of which the model uses each data sample once, and the extracted_dir directory containing photos for training, the fraction of data vs that will be allocated for the validation set

$$extracted\_dir = TrPath(NmBt, NmLi, Df) \tag{27}$$

2. Create a path object that represents the path to the directory from which images will be obtained

$$data\_dir = pathlib.Path(extracted\_dir) \tag{28}$$

3. Create a dataset of images from the directory containing photos for training and validation

$$train\_ds = tf.keras.utils.image\_dataset\_from\_directory(data\_dir, validation\_split = vs, subset = "training", seed = 123, image\_size = (img\_height, img\_width), batch\_size = batch\_size) \tag{29}$$

$$val\_ds = tf.keras.utils.image\_dataset\_from\_directory(data\_dir, validation\_split = vs, subset = "validation", seed = 123, image\_size = (img\_height, img\_width), batch\_size = batch\_size) \tag{30}$$

where the function $tf.keras.utils.image\_dataset\_from\_directory$ expects images to be organized in subdirectories, each corresponding to one class. Subdirectory names will be used as labels for images. The function recursively traverses the specified directory, loads all images and converts them into tensors that can be used for neural network training. All loaded images and their labels are combined into a *Dataset* object that can be used for iteration and feeding data into the model;

$validation\_split$ determines the fraction of data vs that will be allocated for the validation set;

$subset$ specifies which part of the data to return: training or validation;

$seed$ is used to set the initial value of the random number generator. If we set the same seed value for different runs, the data will be divided into training and validation sets in the same way;

$batch\_size$ defines the size of the data batch, i.e., a subset of data that is fed into the model simultaneously for gradient computation and weight updates. A larger batch size usually allows using larger mini-batch sizes, which can speed up training.

4. Increase the dimension of each element in the $train\_ds$ dataset. If previously each element was a pair (image, label), now it will become a triple (image, label, two identical integers).

$$counter = tf.data.Dataset.counter() \tag{31}$$

$$train\_ds = tf.data.Dataset.zip((train\_ds, (counter, counter))) \quad (32)$$

where $tf.data.Dataset.counter()$ creates simple datasets for testing or debugging models, used to create an infinite dataset where each element is a sequential integer starting from 0;

$tf.data.Dataset.zip()$ combines elements from two or more datasets into one new dataset. Elements from corresponding positions in the original datasets will be joined into tuples. A new dataset $train\_ds$ is created, which contains tuples of three elements: an image from the original train_ds set, the first integer from the counter set, and the second integer from the counter set. These additional integers can be used as indices for elements in the batch for learning algorithms.

5. We create data processing pipelines that shuffle elements in the $train\_ds$ and val_ds datasets, apply augmentation to increase data diversity for the training set, combine data into batches for efficient learning, and prefetch batches in advance to minimize model downtime

$$\begin{aligned} train\_ds = (\ & train\_ds \\ & .shuffle(1000) \\ & .map(augment) \\ & .batch(batch\_size) \\ & .prefetch(tf.data.AUTOTUNE)) \end{aligned} \quad (33)$$

$$\begin{aligned} val\_ds = (\ & val\_ds \\ & .map(resize\_and\_rescale, num\_parallel\_calls = tf.data.AUTOTUNE) \\ & .batch(batch\_size)\ ) \\ & .prefetch(tf.data.AUTOTUNE)) \end{aligned} \quad (34)$$

6. We describe a sequence of transformations for images that will be applied during model training, which will randomly change images in each training epoch

$$\begin{aligned} data\_augmentation = & keras.Sequential([\\ & layers.RandomFlip("horizontal", \\ & input\_shape = (img\_height, \text{img\_width}, 3)) \\ & layers.RandomRotation(alf1), \\ & layers.RandomZoom(alf2),]) \ ) \end{aligned} \quad (35)$$

There $keras.Sequential()$ creates a sequential model where each layer is applied to the output of the previous one;

$layers.RandomFlip("horizontal")$ randomly flips the image horizontally;

$input\_shape$ indicates the shape of input images (height, width, number of channels).

$layers.RandomRotation(alf1)$ randomly rotates the image by an angle from -$alf1$ to $alf1$ degrees.

$layers.RandomZoom(alf2)$ randomly increases or decreases the image scale by a value from –$alf2\%$ to $alf2\%$.

7. We create a sequential neural network model for image classification, consisting of several layers

$$\begin{aligned} model = Sequential([\\ & data\_augmentation, \\ & layers.Rescaling(1./255), \\ & layers.Conv2D(16, 3, padding='same', activation='relu'), \\ & layers.MaxPooling2D(), \\ & layers.Conv2D(32, 3, padding='same', activation='relu'), \\ & layers.MaxPooling2D(), \\ & layers.Conv2D(64, 3, padding='same', activation='relu'), \\ & layers.MaxPooling2D(), \\ & layers.Dropout(0.2), \\ & layers.Flatten(), \\ & layers.Dense(128, activation='relu'), \\ & layers.Dense(num\_classes) \quad]) \end{aligned} \quad (36)$$

There $keras.Sequential()$ creates a sequential model where each layer is applied to the output of the previous one;

$data\_augmentation$ - this layer (which we've already discussed) applies random transformations to images for data augmentation;

$layers.Rescaling(1./255)$ - normalizes pixel values of images to the range 0-1. This is important for most neural networks;

$layers.Conv2D(16, 3, padding = 'same', activation = 'relu')$ - applies a two-dimensional convolution with 16 filters of size 3x3;

$padding = 'same'$ - ensures that the output image size remains the same as the input;

$activation = 'relu'$ - applies the ReLU (Rectified Linear Unit) activation function to the output values;

$layers.MaxPooling2D()$ - the model contains two more convolution and pooling layers with a larger number of filters, allowing the model to extract more complex features from images;

$layers.Dropout(beta)$ - randomly turns off a fraction beta of neurons in this layer during training. This helps prevent overfitting;

$layers.Flatten()$ - transforms a three-dimensional tensor (height, width, channels) into a one-dimensional vector;

$layers.Dense(128, activation = 'relu')$ - applies a connected layer with 128 neurons and ReLU activation, allowing the model to extract abstract features;

$layers.Dense(num\_classes)$ - applies the last connected layer with the number of neurons equal to the number of classes. The output of this layer will be used for image classification.

8. Setting the optimizer, loss function, and metrics for model training

model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), (37)
metrics=['accuracy'])

There $optimizer = 'adam'$ - a method that regulates the learning process by changing the weights of the neural network to minimize the loss function;

$tf.keras.losses.SparseCategoricalCrossentropy$ - a loss function for classification tasks that measures how well the model predicts results on training data;

$metrics = ['accuracy']$ - the simplest metric that shows the percentage of correctly classified samples, which allows evaluating the quality of the model during training and validation.

9. Initiate the neural network training process

history = model.fit(train_ds, validation_data=val_ds, epochs=epochs) (38)

with training sample $train\_ds$, validation sample val_ds, and the number of epochs of training sample passes during one iteration.

10. Save the result of model training in the file mt

$$model.save(mt) \quad (39)$$

Thus, IC $(data\_dir, vs, alf1, alf2, beta)$ (formalizes the process of building and training an image recognition and classification model in the *Tensorflow* package, the main parameters of which are the set of training image $data\_dir$ directories, the fraction of images vs that will be allocated for the validation set, the limits $alf1$ of random image rotation, the limits $alf2$ of random image scaling, the fraction beta of randomly turning off network neurons during training.

Experimental Results. We experimentally assess the quality of the built image classification model and its adaptation to the proposed data. In the first stage, if necessary, we adapt the list of semantic labels to the set of images selected for training. If the quality of image recognition for a certain semantic label is sufficiently low, this label is subordinated to the one closest to it semantically. In the second stage, we optimize the accuracy of the image labeling model by

selecting the parameters of the model training algorithm and, if possible, adjusting the volume of the training set for image classification.

Let's consider the process of labeling images of field cultivation in an agricultural enterprise. Let the system of semantic labels be defined by the following one-level list:

$$NmLi = [tractors\_plow, tractors\_cultivation, tractors\_seeding, tractors\_watering\_plants, tractor\_cutting\_grass, combining] \quad (40)$$

In the first stage, we download classified images from Web resources, for example, from the site *https://www.istockphoto.com.* First, using a thematic query, we find Web sites of thematic images with the ability to select photos based on queries that correlate with semantic labels. We form such a set of images using the described method of forming an image set, in particular using the css class "yGh0CfFS4AMLWjEE9W7v". The volume of downloaded collections should be selected as significant; in this example, volumes of 1200 photos were set. Unfortunately, when viewing the collapsed images, it turns out that not all of them correspond to the queries based on which they were selected by the Web site. Therefore, automatic downloading should be supplemented with visual manual filtering. This is a labor-intensive process, so we aimed to form collections for semantic labels in volumes of 12, 24, or 48 specimens, calculating that these sets will be divided into training and validation parts. In the next stage, we train the corresponding neural networks according to the described algorithm. Randomly selected image samples used for training are shown in the following figure.

Fig. 2. Sample images used for training

The quality of training was evaluated using test samples that were not used for training. They were formed from the set of downloaded images with a volume of 5 and 10 images per semantic label.

It was established that the classification accuracy primarily depends on the size of the training sample as well as on the parameters of image collection augmentation during training. The main results of the calculations are presented in the following table.

Table 1.

Image Classification Accuracy

n	***TrVol***	***|NmLi|***	***α1***	**α2**	**ε(%)**
1	12	6	0.3	0.3	43
2	12	6	0.4	0.4	63
3	12	6	0.5	0.5	58
4	24	5	0.2	0.2	26
5	24	5	0.3	0.3	18
6	24	5	0.4	0.4	38
7	48	5	0.2	0.2	32
8	48	5	0.1	0.1	18
9	48	5	0.05	0.05	8

In the table, ***TrVol*** denotes the size of the training sample, $|NmLi|$ - the number of semantic labels in the model, $\alpha 1$, $\alpha 2$ - parameters of image collection augmentation during training, ε - relative error obtained when testing the model. The presented results indicate a low level of classification accuracy with small training sample sizes. Poor differentiation of the semantic label $tractors\_seeding$ was also recorded, which had to be combined with the label, $tractors\_cultivation$ leading to a reduction in their number from 6 to 5. With a successful selection of augmentation parameters and using 48 images per label in the training set, it was possible to reduce the classification error to an acceptable level of 8%. The dynamics of errors of the image classification model during its parameter selection are shown in the following figure. Only one level of recorded errors can be considered acceptable.

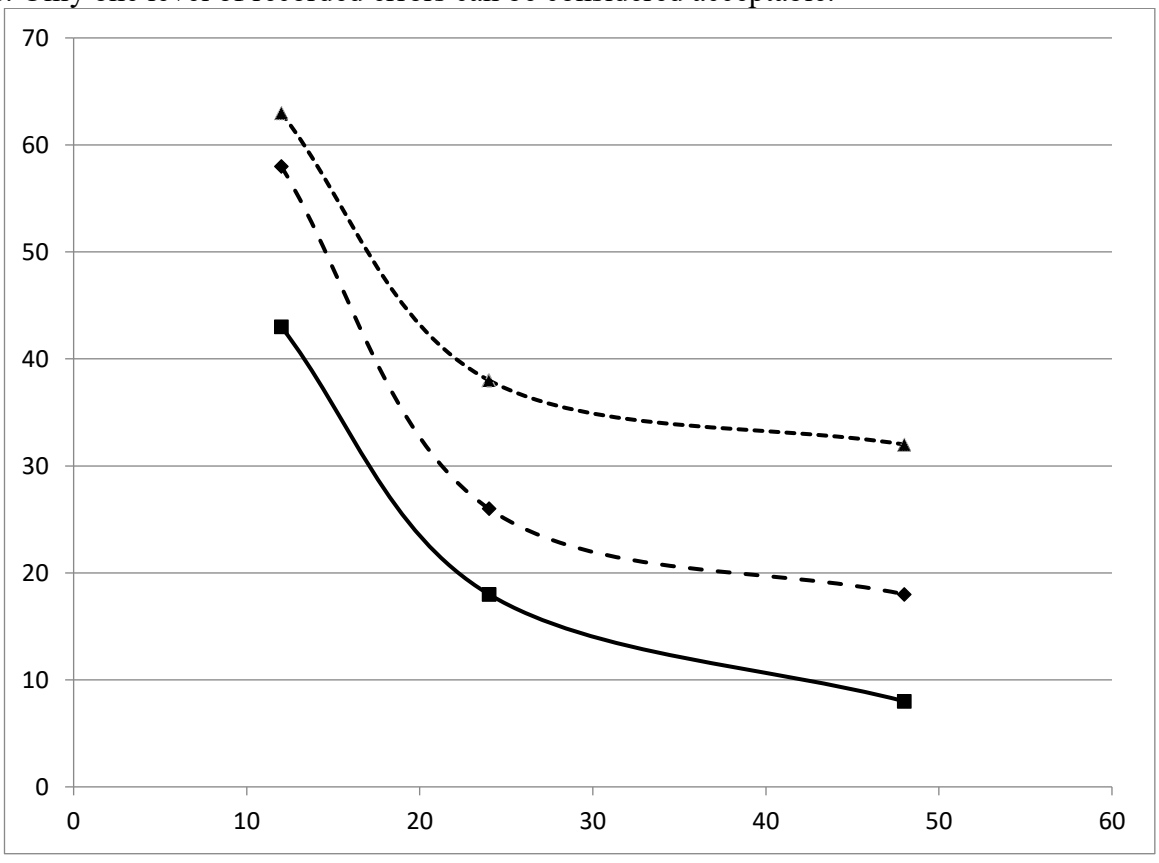

Fig.3. Dynamics of errors of the image classification model during its tuning

Conclusions. The paper investigates the issue of automating the formation of image collections and the formation of a methodology for tuning the parameters of the image classification model

using neural networks with the TensorFlow framework. An algorithm for automated formation of a thematic image collection is proposed. A method of parametric training of the thematic image classification model using the TensorFlow framework is also formalized. Experimental studies have confirmed the effectiveness of the proposed approaches.

References

1. Hanbury A. A survey of methods for image annotation. *Journal of Visual Languages and Computing*. 2008. No.19. P. 617–627.
2. Chachra S.K., Xue Z, Antani S., Demner-Fushman D., Thoma G.R. Extraction and Labeling High-resolution Images from PDF Documents. Lister Hill National Center for Biomedical Communications Bethesda, MD: U. S. National Library of Medicine,2023. 20894
3. Sager Ch., Janiesch Ch., Zschech P. A survey of image labelling for computer vision applications. *Journal of Business Analytics*. 2021. V.4. No.2, P. 91-110, DOI: 10.1080/2573234X.2021.1908861
4. Kluckner S., Mauthner T., Roth P.M., Bischof H. Semantic Image Classification Using Consistent Regions and Individual Context. URL: https://www.researchgate.net/publication/ 221259742_Semantic_Image_Classification_Using_Consistent_Regions_and_Individual _Context
5. Yu.Y. Learning Semantics of Classes in Image Classifcation. URL: https://www.zora.uzh.ch/id/eprint/259312/1/yu2023master.pdf
6. Zhou F., Huang S., Xing Y. Deep Semantic Dictionary Learning for Multi-label Image Classification. URL: https://cdn.aaai.org/ojs/16472/16472-13-19966-1-2-20210518.pdf

МЕТОД ПІДВИЩЕННЯ ЯКОСТІ РОЗМІТКИ ЗОБРАЖЕНЬ СЕМАНТИЧНОГО МОНІТОРИНГУ БІЗНЕС-ПРОЦЕСІВ ГІС

Р.М. Пасічник, Л.В. Бабала, М.В. Мачуляк

Західноукраїнський національний університет
11, Львівська, м. Тернопіль, 46009, Україна
Emails: Roman.pasichnyk@gmail.com, Ludaduma7@gmail.com, Mvmach9@gmail.com

Стаття присвячена актуальній проблемі автоматизації процесу маркування зображень для систем комп'ютерного зору в сільському господарстві. Автори досліджують методи формування наборів зображень та налаштування параметрів моделей класифікації зображень з використанням нейронних мереж на базі фреймворку TensorFlow. Наукова значущість роботи полягає у розробці нових підходів до автоматизованого збору тематичних колекцій зображень та формалізації методики параметричного навчання моделей класифікації. Практична цінність дослідження виражається у підвищенні ефективності процесу маркування зображень для геоінформаційних систем у сільськогосподарській галузі. Методологія дослідження включає аналіз існуючих підходів до маркування зображень, розробку алгоритму автоматизованого формування тематичних колекцій зображень, формалізацію методу параметричного навчання моделі класифікації та експериментальну перевірку запропонованих підходів. Основні результати роботи: 1. Розроблено алгоритм автоматизованого формування тематичних колекцій зображень. 2. Формалізовано метод параметричного навчання моделі класифікації зображень з використанням фреймворку TensorFlow. 3. Експериментально встановлено залежність точності класифікації від розміру навчальної вибірки та параметрів аугментації зображень. Дослідження показало, що при оптимальному підборі параметрів аугментації та використанні 48 зображень на мітку в навчальній вибірці можливо знизити помилку класифікації до прийнятного рівня 8%. Робота вносить значний внесок у розвиток методів автоматизованої обробки зображень для сільськогосподарських геоінформаційних систем. Практичне значення результатів полягає у підвищенні ефективності процесів моніторингу та управління в аграрному секторі.

Ключові слова: комп'ютерний зір, маркування зображень, нейронні мережі, TensorFlow, геоінформаційні системи, сільське господарство, класифікація зображень.