DOI: 10.20998/2079-0023.2024.02.10
UDC 004.89:004.932:004.4

V. I. ZIUZIUN, Candidate of Technical Sciences (PhD), Docent, Associate Professor at the Department of Management Technologies, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine; e-mail: vadym.ziuziun@knu.ua; ORCID: https://orcid.org/0000-0001-6566-8798
N. A. PETRENKO, Student, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine; e-mail: nikita.petrenko@knu.ua; ORCID: https://orcid.org/0009-0006-3921-8412

APPLICATION OF OPTICAL CHARACTER RECOGNITION AND MACHINE LEARNING TECHNOLOGIES TO CREATE AN INFORMATION SYSTEM FOR AUTOMATIC VERIFICATION OF OFFLINE TESTING

During the learning process in any field, testing and monitoring the knowledge of students or other learners is an essential part. Teachers often spend considerable time grading large volumes of standardized tests. While online testing systems have been developed to streamline this process, offline paper tests remain popular as they do not require access to computers, electricity, or a stable internet connection. Offline testing is often considered one of the most representative methods for assessment, but it leads to repetitive work for teachers during the grading process. To save time, some educators use test sheets to structure responses, simplifying grading tasks. Consequently, developing a system that automates the grading of offline tests has become increasingly relevant. The purpose of this research was to develop an information system (web platform) that simplifies the offline test grading process using optical character recognition technologies powered by machine learning algorithms. The object of this research is the processes and functionality involved in creating an information system for the automated grading and evaluation of offline tests. The scientific novelty lies in integrating machine learning algorithms with modified image processing algorithms to create a system capable of analyzing and grading a wide range of offline test tasks, including open-ended, closed-ended, sequence identification, and multiple-correct-answer questions. The practical significance of this research is the development of a web platform to automate offline test grading through optical character recognition and machine learning technologies, reducing teachers' time spent on grading, enabling analysis and improvement of educational programs, supporting various test types, and promoting scientific and technological advancement in education. The developed system can recognize handwritten text from photos, create an array of responses, and compare them to the answers provided by the teacher. This approach significantly reduces the time teachers spend on grading tests. For user convenience, a minimalist interface was created, granting access to all main system functions with intuitive controls. A detailed description of the developed algorithms and machine learning models is provided. This project offers broad potential for further development, including integration with other educational platforms, enhancements in recognition technology, and system scalability.

Keywords: information system, web platform, IT project, machine learning, neural networks, algorithm, IAM dataset, optical character recognition, testing, educational process.

Introduction. Before starting the development of the information system, it is helpful to conduct a brief analysis of software solutions aimed at simplifying the testing process. It is important to note that while there is a wide range of online testing systems, there are relatively few programs specifically specializing in handwritten answer recognition. Among the most popular and successful ones are ZipGrade, GradeCam, and Essay Grader.

The ZipGrade mobile app is designed to automate the test grading process and offers users a range of functionalities [1]. At first glance, it is a convenient and efficient tool, but it has limitations in its free version. While the system is user-friendly, it has several drawbacks: limited flexibility in creating tests; no option to assign specific point values to questions; the need to print answer sheets for the entire group; restrictions on test types (inability to create closed-ended tests, multiple-answer tests, matching tests, or sequence-based tests) and limitations on the number of test versions available.

GradeCam is a web application that offers a range of functionalities beyond those of the previous product. Like ZipGrade, its primary function is to automate the test grading process. However, GradeCam distinguishes itself with a more appealing analytics interface and the ability to recognize text [2]. This system is designed to automate the entire educational process – from creating tests and storing them in a database to assigning grades, monitoring student performance, and generating reports for administration, parents, and students.

Among the advantages of GradeCam are the following: it is easier to create tests on a computer than on a mobile device; the response format is adapted for various subjects; it allows scanning of student work using any camera: on a mobile device, laptop, or computer, as well as an external camera; there is an option to save images of questions for subsequent manual grading by the instructor; it recognizes handwritten text in open-ended answers; all grades and student work are stored; it generates reports on group performance and individual student success; the system automatically assigns final grades based on test results.

The drawbacks of this system include: limitations in that it only works with forms created within the system itself; the need to print a large number of answer sheets; the requirement to input open-ended questions in printed letters, placed separately in each cell; lack of support for languages other than English; high product prices, which depend on the number of students or the volume of assessed work.

Although the system offers numerous useful features, instructors are still required to print specialized answer sheets, resulting in additional time and financial costs. Moreover, this complicates the administration of tests without prior preparation, such as if there is free time during a class. For individual instructors, using this system can become costly if their university or educational institution does not collaborate with the platform.

Essay Grader is an application designed to simplify the process of grading written works, particularly comprehensive handwritten texts rather than just closed-ended tests. This system allows educators to upload images of essays, check for plagiarism, identify grammatical errors, and provides an interface for digital editing of sentences, along with the ability to add comments and send results to students [3]. Although the system includes handwriting recognition, it is not intended for automated test grading and is exclusively focused on assisting in the grading of essays. This application does not have features for reporting, storing student grades, or analyzing statistics. Therefore, while this system shares a similar theme, it is not an equivalent to the project being developed.

Analysis of recent research and publications. After analyzing well-known solutions in the field of test grading, it can be noted that the issue of effective and rapid grading of test items remains relevant.

However, many existing software products aimed at addressing this issue have rigid logic that limits users in customizing tests to their needs. For example, they may not allow the creation of open-ended tests, matching questions, or sequencing questions, and they may restrict the ability to change the point value for specific items. Even when a system permits flexible test creation, it may still impose limitations on the ways in which answers can be entered.

The process of handwriting recognition consists of the following subtasks [4]: extracting the page from the photograph and establishing text boundaries; processing the image by enhancing contrast and line thickness; compensating for lighting by illuminating dark areas; aligning the text tilt; aligning the line tilt; segmenting the page into lines; segmenting lines into words; recognizing words; and correcting recognized words and sentences using natural language processing techniques.

There are algorithms [5] that allow for the immediate identification and recognition of words on a page, bypassing the corresponding steps outlined above. In such cases, the technique of «transfer learning» can be applied, meaning that a model is initially trained on one dataset and then fine-tuned on real data. However, since the data collection process is time-consuming and labor-intensive, it is essential to choose more robust algorithms. Thus, statistical approaches will be used to address the problem of text segmentation. During the analysis of existing text segmentation solutions, a module was found that implements the methodologies from the articles referenced [6, 7]. Despite being published in 2007 and 1999, respectively, these articles remain competitive, yielding good results and operating relatively quickly. The module also includes implementations of algorithms for document scanning, finding the contour of an image using a four-point transformation; binarization techniques such as Niblack, Sauvola, and Wolf; lighting compensation according to the work materials; segmentation based on the algorithms referenced [8, 9]; and text tilt alignment according to the referenced methodology.

Additionally, all of the aforementioned algorithms are implemented in C++ and utilize OpenCV, allowing them to operate much more quickly.

Since there are no ready-made modules for handwriting recognition, an analysis of machine learning algorithms was conducted. Over the past several decades, researchers have developed various models for this task. Historically, this problem has been viewed as a sequence alignment task: a sequence of features derived from the input image is compared to a sequence of letters, Hidden Markov Models were used for this purpose. However, the primary drawback of such models is the limitation in utilizing the full context of the data due to the nature of Markov models, where each observation depends only on the current state and is independent of previous states [9]. This limitation has been overcome through the use of networks such as RNNs and LSTMs, which can «remember» long sequences. A true breakthrough occurred with the invention of Connectionist Temporal Classification (CTC) [10] and its combination with recurrent networks.

As noted in the research [11, 12], models composed of layers such as convolutional layers (CNNs), recurrent layers (RNNs), specifically the LSTM layer, and the Connectionist Temporal Classification (CTC) layer are the most effective. Two metrics were used to determine effecttiveness: Character Error Rate (CER) and Word Error Rate (WER). Therefore, for the implementation of this IT project, a model was chosen that demonstrates the highest efficiency in text recognition.

Based on the conducted research [13, 14], it can be concluded that the IAM (IAM OnLine Handwriting Database) dataset is the best choice for training the model.

The IAM is a collection of handwritten English text on a white background. It features the handwriting of 657 writers who contributed to the creation of the dataset. The database contains a total of 1,539 scanned pages of text, 5,685 isolated and annotated sentences, 13,353 isolated and annotated lines of text, as well as 115,320 individual words [15].

In addition to the aspects of direct IT product development, it is also important to analyze the approaches to managing IT projects and making managerial decisions. These issues are discussed in the works of [16, 17].

Presentation of the main material. The information system is being developed to optimize the quality control of education, and it will include the following key elements:

- Participants (teachers (instructors), students, learners).
- Methods of knowledge assessment (closed tests, open tests, and matching questions).
- Information system (web pages, database, test grading module).
- Reporting documents (logs, records).

As mentioned earlier, the proposed information system is considered in the context of quality control of knowledge, specifically to optimize this process during the grading of offline tests. Therefore, let us first examine the mathematical processes involved in grading tests manually and automatically.

Let's define the variables:

- Set of instructors (1):

$$T = \{t_1, t_2, ..., t_n\}, \quad (1)$$

where $t_1, t_2, ..., t_n$ are the instructors.

- Set of courses (2):

$$C = \{c_1, c_2, ..., c_n\}, \quad (2)$$

where $c_1, c_2, ..., c_n$ are the courses.

- Set of questions (3):

$$Q = \{q_1, q_2, ..., q_n\}, \quad (3)$$

where $q_1, q_2, ..., q_n$ are the questions.

- Set of students (4):

$$S = \{s_1, s_2, ..., s_n\}, \quad (4)$$

where $s_1, s_2, ..., s_n$ are the students.

- Set of groups (5):

$$G = \{g_1, g_2, ..., g_n\}, \quad (5)$$

where $g_1, g_2, ..., g_n$ are the groups.

Let's describe the constraints:

- Constraints for the category of instructors (6):

$$\text{teaches: } C \to T, \text{ then } \forall c \in C, \exists! t \in T: \text{teaches}(c) = t \quad (6)$$

- Constraints for the category of questions (7):

$$\text{correct.answer: } Q \to A, \text{ then}$$

$$\forall q \in Q, \exists! a \in A : \text{correct.answer}(q) = a;$$

$$|\text{correct.answers}(q)| = 1, \forall q \in Q. \quad (7)$$

Let's describe mathematically the test grading functions using the information system and manually.

Common:

- Function (8) associates a combination of instructor and course with a set of tests:

$$f_{\text{test}} : T \times C \to \text{Tests}. \quad (8)$$

- Function (9) maps a test to a set of questions:

$$f_{\text{questions}} : \text{Tests} \to 2^Q, \quad (9)$$

where 2^Q is the power set of Q, representing all possible combinations of questions.

- Function (10) maps a student and a test to a real number representing the student's score on that test:

$$f_{\text{marks}} : S \times \text{Tests} \to R. \quad (10)$$

Manual grading of assignments by the instructor.

Let $T_m(q,s)$ be the time the instructor spends grading question q for a single student s. The total time for manual grading of all questions for all students is calculated using formulas (11):

$$T_{\text{m(total)}} = \sum(T_m(q,s), \forall q \in Q, \forall s \in S). \quad (11)$$

Grading using the information system. Let $T_a(q)$ represent the time the automated system takes to grade question q. This time should be significantly less than $T_m(q,s)$, as the system should maintain a consistent (or negligibly variable) grading time regardless of question complexity or answer length. The total grading time by the automated system for all questions for all students is calculated using formulas (12):

$$T_{\text{a(total)}} = \sum(T_a(q), \forall q \in Q). \quad (12)$$

Thus, we have provided formulas for calculating the time required for grading tests both manually and with the application. Now, let us present the formulas for determining the efficiency gain. The efficiency gain G can be determined using formulas (13):

$$G = \left(\frac{E_m - E_a}{E_m}\right) \cdot 100\%, \quad (13)$$

where E_m – efficiency of manual test grading;

E_a – efficiency of grading using the application.

The efficiency of manual test grading can be determined using formulas (14):

$$Em = \left(\frac{T_m}{N}\right), \quad (14)$$

where T_m – time for manual test grading;

N – number of tests graded.

The efficiency of test grading using the application can be determined using formulas (15):

$$Em = \left(\frac{T_a}{N}\right), \quad (15)$$

where T_a – time for grading tests using the application;

N – number of tests graded.

Formulas (16) can be used to calculate the time saved by using the application:

$$E = \sum(T_m(q,s) - |S|) \cdot \sum(T_a(q), \forall q \in Q, \forall s \in S). \quad (16)$$

Thus, we have obtained all the necessary formulas that describe the processes of grading tests manually and automatically. We have also derived the formulas to calculate the efficiency gain from using the application, as well as the formula for calculating the time saved.

Now, let us mathematically define the task of the application, specifically to automatically grade and assign scores for the test. Formulas (17) provides a mathematical description of how to calculate the scores based on the responses:

$$\text{score}\left(A_{\text{s(test)}} Q_{\text{test}}\right) = \sum(\text{points}(q) \cdot \delta(a_q, A_{\text{s(test)}}[q])), \quad (17)$$

where $A_{\text{s(test)}}$ – the set of answers provided by the student for a specific test;

$\text{points}(q)$ – represents the number of points for question q;

Q_{test} – the set of questions included in the specific test;

δ – the Kronecker delta function [18].

Database of the system. Since the system being developed will work with a large amount of unstructured data, MongoDB was chosen as the database. MongoDB is

a document-oriented NoSQL database [19]. Here's why MongoDB is the best fit:

- Document-oriented storage. Since MongoDB is a NoSQL database that stores data in a document-oriented format similar to JSON (known as BSON), it is well-suited for storing and manipulating large volumes of text data. This creates a natural correspondence between the data storage format and the data itself.
- Schema-less nature. MongoDB does not enforce schemas, meaning you can store documents without needing to define their structure in advance. This flexibility is particularly beneficial for text data, which may have varying structures, allowing different documents to be stored in the same collection without adhering to a strict schema.
- Designed for scalability. MongoDB is built for scalability, supporting sharding (distributing data across multiple servers) and replication (creating copies of data on different servers). This ensures that the database can handle increasing volumes of text data and user queries without significant performance degradation.
- Aggregation framework. MongoDB provides a powerful aggregation framework that allows for the processing and analysis of large amounts of text data in the database. This can be utilized for tasks such as text analysis, data transformation, and generating summaries or reports based on text data.
- Development efficiency. Given the popularity of JSON in modern web applications, using MongoDB can simplify the development process. Developers can work with a familiar format, reducing the complexity of data conversion between the application and the database.

Structural diagram of the system. The software will feature a client-server architecture, as depicted in fig. 1.

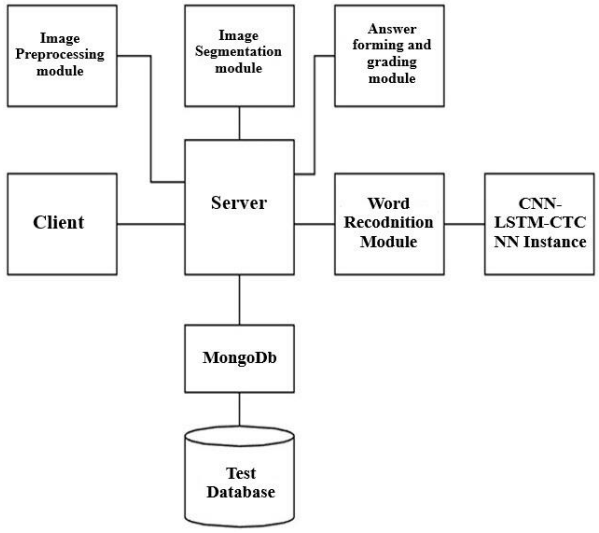

Fig. 1. Structural diagram of the system

The client provides the user interface for interacting with the server, sends requests to the server, receives responses from it, and loads the results in a web browser. The server receives requests from the client, processes them, generates a webpage, and returns it to the client.

In addition, it performs the following functions:

- Interacts with the database containing information about user-created tests.
- Interacts with the image preprocessing module.
- Interacts with the text segmentation module.
- Interacts with the text recognition module.
- Interacts with the response generation and grading module.

Algorithm for processing incoming images. Since the images in the dataset selected for training the neural network contain clear, high-contrast handwritten text with large spaces between lines and words, it is necessary to transform each image to resemble the images from the training dataset as closely as possible. To test this part of the project, data from real English language tests were collected. Unfortunately, only previously graded work was available, which contains corrections in red ink. At this stage, it was necessary to remove the red color from the images using OpenCV tools. The image processing algorithm is outlined below (fig. 2).

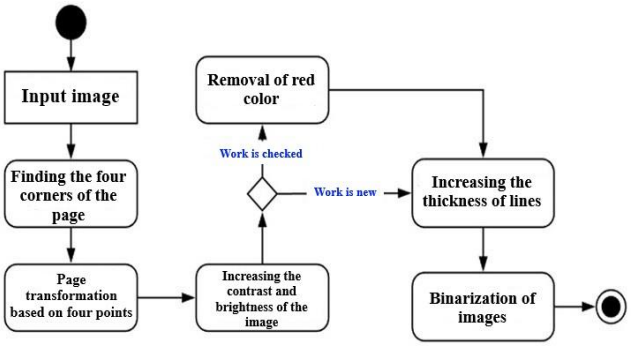

Fig. 2. Image preprocessing algorithm

The image processing algorithm shown in Figure 2 includes the following steps:

- Page boundary detection and corner coordinates fixing. The page boundary is identified by finding the convex hull using OpenCV's convexHull function, which is then approximated to a rectangle with the approxPolyDP function.
- Four-point perspective transformation. The page is transformed based on the four detected corner points (as mentioned above) to obtain an image as if taken at a 90° angle to the page.
- Contrast and brightness enhancement. This step clarifies the distinction between text and paper while reducing potential background defects, such as paper creases, grid markings, and text bleed-through from the opposite side.
- Red color removal. This is applied only to graded papers with red marks or those with a red margin line separating the page borders.
- Line thickening. After previous steps, some lines may appear thin or broken. They are thickened to restore word integrity and match the training data as closely as possible.
- Image binarization. Wolf's binarization method is used to convert the image into a 2D array of zeros and ones, which serves as the input vector for the segmentation algorithm.

Four-point transformation algorithm. If a user takes a photo of a handwritten document at an angle other than 90° to the page, the word recognition algorithm will not function correctly, as the computer perceives the same word at different angles as entirely different words. This issue can be addressed during the training data preparation stage by introducing all possible distortions to the images, allowing the machine learning algorithm to adapt to such variations and accurately classify even highly skewed words.

However, a much faster and more efficient approach is to use perspective transformation. This transformation was implemented using functions from the OpenCV library.

To apply the transformation, it is necessary to specify the coordinates of the four corners of the page (fig. 3), which are identified during the initial processing stage, as well as the coordinates of the new image's corners, which are defined as follows.

Let A be the top-left corner of the page, B the top-right, C the bottom-left, and D the bottom-right, as shown in fig. 3. Their coordinates will be as follows (18):

$$\left(x_a, y_a\right), \left(x_b, y_b\right), \left(x_c, y_c\right), \left(x_d, y_d\right). \quad (18)$$

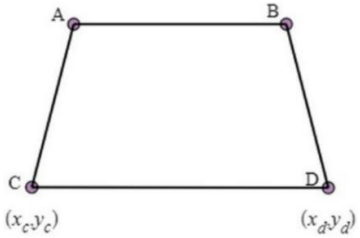

Fig. 3. Schematic representation of a page photographed at an angle

We find the side lengths using formulas (19-22):

$$AB = \sqrt{\left(x_b - x_a\right)^2 + \left(y_b - y_a\right)^2}. \quad (19)$$

$$CD = \sqrt{\left(x_d - x_c\right)^2 + \left(y_d - y_c\right)^2}. \quad (20)$$

$$CA = \sqrt{\left(x_a - x_c\right)^2 + \left(y_a - y_a\right)^2}. \quad (21)$$

$$BD = \sqrt{\left(x_d - x_b\right)^2 + \left(y_d - y_b\right)^2}. \quad (22)$$

We calculate the width and height of the new image using formulas 23 and 24:

$$W = \max(AB, CD). \quad (23)$$

$$H = \max(CA, BD). \quad (24)$$

The coordinates of the corners of the new image are then defined as follows: the top-left corner has coordinates $(0, 0)$, the top-right corner has coordinates $(W, 0)$, the bottom-left corner has coordinates $(0, H)$, and the bottom-right corner has coordinates (W, H). The getPerspectiveTransform function [20] is then applied to obtain the transformation matrix, followed by the warpPerspective function to apply this matrix and generate the new image. The algorithm's schematic is shown in fig. 4.

Word recognition algorithm for images. The machine learning model used for image classification is a neural network consisting of five convolutional layers and two recurrent layers, followed by a CTC (Connectionist Temporal Classification) layer. Formally, this model can be represented as a function that maps an image – specifically, a matrix of size $W \times H$ – to a sequence of characters $\left(c_0, c_1, ..., c_L\right)$ with a length ranging from 0 to L.

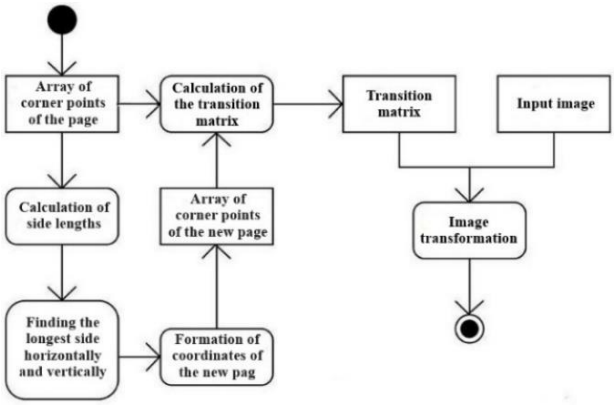

Fig. 4. Four-point transformation algorithm

Thus, text recognition occurs character by character, enabling the classification of words that were not present in the training data, as well as names and misspelled words. The architecture of the neural network is schematically illustrated in fig. 5.

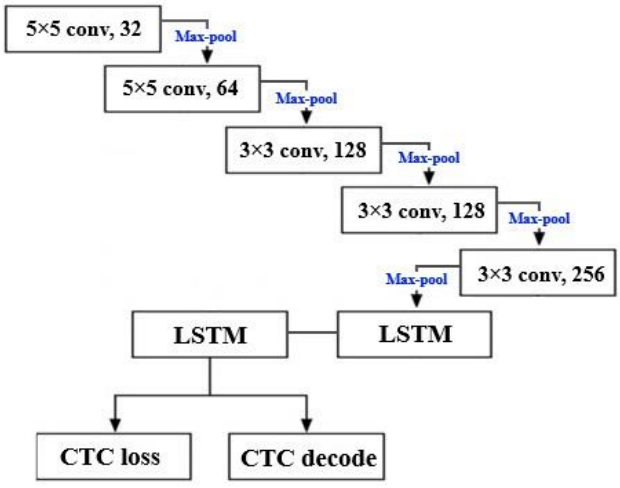

Fig. 5. Neural network architecture

The algorithm takes a binary image of size 128×32 as input. Since not all word images will have this exact size, the image is resized until either the height reaches 32 pixels or the width reaches 128 pixels. After resizing, the image is padded with white pixels to make it 128×32. The image then enters the convolutional layers, which consist of five layers trained to identify relevant features in the image. Each layer includes three operations: first, a filter of size 5×5 is applied in the first two layers and 3×3 in the last three layers. Then, the nonlinear activation function ReLU, defined by formulas (25), is applied:

$$f(x) = \max(0, x), \quad (25)$$

where x – input value of the neuron.

Next is the pooling layer, which reduces the dimensionality of the input image. In this model, max-pooling is used, which retains only the maximum value within each 4×4 region, thus capturing a generalized version of the input image's features. This is highly beneficial, as minor variations in features identified by the convolutional layer won't affect the resulting feature matrix. After passing through these five layers, we obtain a feature matrix of size 32×256, which is then fed into the recurrent layers. This matrix contains 32 sequences, each with 256 features per time step.

LSTM layers were chosen as the recurrent layers because they can propagate information over long distances and learn long sequences of characters. The output of the recurrent layers is a matrix of size 32×80, where 32 represents the number of time steps, and 80 represents the total number of unique characters in the IAM dataset (79) plus 1 additional symbol required for Connectionist Temporal Classification (CTC). During training, CTC receives the 32×80 matrix along with the correct transcription of the text in the image and calculates the CTC loss. During prediction, CTC only receives the feature matrix and decodes it into text. RMSProp was chosen as the optimizer. The trained model achieved a Character Error Rate (CER) of 10.63% on the validation dataset.

The answer array generation algorithm. To facilitate comparison with the reference answer, an array is created containing answers for each question in a row format. After processing the image, the image is segmented into rows. Each row is then split into words. At this stage, we know the order of the rows in the text and the order of words within each row.

After obtaining an array of images for each word in the current line, we begin recognition. If the word in the image is a number and appears first in the line, it is assumed to indicate the question number for which this is the beginning of an answer. If it appears second or later in the line, it is added to the answer as regular text.

If a new answer start is identified, all subsequent words following this number are part of the answer until another number is found at the beginning of a line.

The algorithm produces an array where each element represents the student's answer to a specific question, which will then be used for test grading in the next step. The algorithm for generating the answer array is shown in fig. 6.

Answer verification algorithm. After image processing, text segmentation, word recognition, and answer array generation, the test grading begins. The system supports the following question types: open-ended, single-answer multiple-choice, sequence ordering, and multiple-answer (or set) questions. The «set» question type can be further divided into two cases: where points are awarded for each correct answer, or where points are only awarded if all answers are correct.

The simplest case – single-answer multiple-choice – is graded by comparing letters written in lowercase.

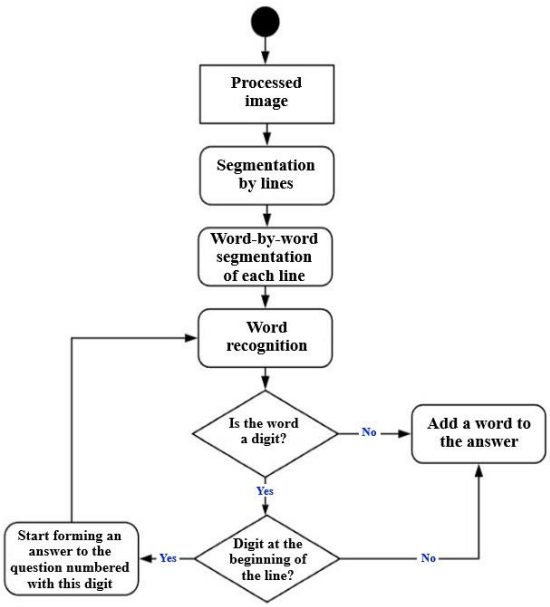

Fig. 6. Answer array generation algorithm

The simplest case – single-answer multiple-choice – is graded by comparing letters written in lowercase.

If a multiple-answer question (set type) contains several correct answers, there are two possible grading scenarios. If the instructor specified that all correct answers must be identified, the following steps are performed:

- Convert letters to lowercase.
- Sort answers alphabetically.
- Concatenate into a single string.
- Compare with the correct answer.

If the instructor intends to award points for each correct answer identified, the grading process is as follows:

- Convert letters to lowercase.
- Calculate the intersection between the sets of correct answers and the student's answers.
- Count the number of correct answers.

If the question requires sequence ordering, it is graded similarly to a set-type question where all correct answers must be identified. After checking multiple-choice questions, the student is awarded the points specified by the instructor for these questions. For grading open-ended text responses, the «cosine similarity» metric and Tf-idf vectorization from the sklearn package are used. The verification algorithm proceeds as follows:

- Create an instance of the TfidfVectorizer class.
- Use the fit_transform function to obtain two vectors of length n, representing the correct answer and the student's response.
- Construct a matrix $A_{2\times n}$, where the first row is the vector for the correct answer, and the second row is the vector for the actual response.
- Calculate a new matrix as the product of matrix A and the transpose of matrix A using formulas (26):

$$C = A \cdot A^T. \quad (26)$$

Matrix C has the following form (27):

$$\begin{pmatrix} 1 & s \\ s & 1 \end{pmatrix}, \quad (27)$$

where s – measure of similarity between the correct answer and the actual response. If the answers match completely, $s = 1$. If there is no overlap between them, $s = 0$. Open-ended responses are not graded; however, the instructor is informed of the similarity level.

Peculiarities of server-side implementation. The server side of the application is implemented using the Flask microframework, which has minimal out-of-the-box functionality and allows the developer to extend it as needed. In this case, the MVC (Model-View-Controller) model is implemented. This architectural design pattern for user interfaces aims to separate the internal data structure from its external representation and increase code modularity.

The model's module is the central component of MVC [21]. It is responsible for managing the data of the web platform using the MongoEngine ORM library, which allows manipulation of data from the MongoDB database. This module contains the definitions of all classes corresponding to the collections and documents in the MongoDB database.

The controller is a specific abstraction composed of three parts:

- Initialization. This involves creating an instance of the Flask application, configuring the necessary settings, and connecting to the database.
- Routing. This defines the URL paths through which the server loads specific pages using decorators in Python. Each route is associated with a controller, specifically with a particular action of the controller. During the execution of this action, models are used to retrieve the necessary data from the database, which is then sent to the views module for rendering.
- Execution. This is the process of launching the web application

The routes.py file contains the routing methods associated with the Model and View.

The views module is an abstraction responsible for the user interface. It receives data from the model via the controller and determines how to present it. For this purpose, the Flask framework provides the Jinja2 template engine for generating HTML pages, which also allows for template inheritance. The base template is the top navigation bar defined in base.html, which is inherited by the left navigation bar in dashboard.html, and all other templates inherit from it as well.

Design and Page Content. After registering and logging into the system, the user sees the main navigation panel, described in dashboard.html, which inherits from the authorization panel page, base.html. All other pages in the system inherit from dashboard.html, creating a four-level template inheritance structure. This approach maintains a consistent page style and reduces the amount of code.

The navigation panel contains essential controls, helping to minimize the number of intermediate steps required to work within the system. The menu items include links to the test page, group lists, statistics page, and new test creation page, along with graphical elements like Feather icons for each tab to aid in user navigation.

The group list page includes a form for uploading a new list from a CSV file, as well as a list of all groups that have already been created. The group import form consists of a text field for entering the group number and a file upload field. If the current user hasn't created any groups yet, they will see a message in place of the group list indicating this.

If the user attempts to upload a file in any format other than CSV, an error message will appear indicating a file read issue. Once a group is added, it automatically appears in the list as a link. By clicking on this link, the user is directed to a page showing the current grades for that group across all tests.

The new test creation page contains a form with fields for the course name, test description, and a section for creating new questions (fig. 7). This section includes the following fields:

- Question Type. A dropdown menu with options such as single-choice, sequence, set, and text, which correspond to valid values in the database.
- Question Text. An optional text field.
- Correct Answer. A required text field.
- Question Score. A required numeric field, with a default value of 1.
- Add Question Button.

Fig. 7. New test creation form

The question list appears in a table format, with columns corresponding to the question's properties and rows for each question. Upon clicking the «Add Question» button, the new question immediately displays in the list below the form, using JavaScript, and the Save Test button shifts down by the height of one table row. When the Save Test button is clicked, or if the user attempts to leave the page, a confirmation dialog appears. The user must select the desired option to proceed with the system.

The tests page displays a list of all courses created by the current user. If the user did not specify a course name when creating a test, those tests are grouped under the category «Other Tests».

Each course item in the list is an active link to the test view page for the selected course, which also includes a list of created tests.

When hovering over a list item, it changes color, and clicking it opens the test view page (fig. 8). This page displays the course name, test description, a table of questions, and a «Start Checking» button, which leads to the grading page for the selected test.

The Test Grading Page (fig. 9) includes the following fields:

- Group. A dropdown menu containing all groups imported by the current user.
- Student. A dropdown list of student names in the selected group. The values in this field update dynamically based on the selection in the first dropdown menu.
- File. An upload button for adding an image of the student's work from the file system.
- Recommended Score. Automatically populated by the system after the uploaded file is graded.
- Score. The actual grade that the instructor can assign based on the recommended score.
- Save Score Button.

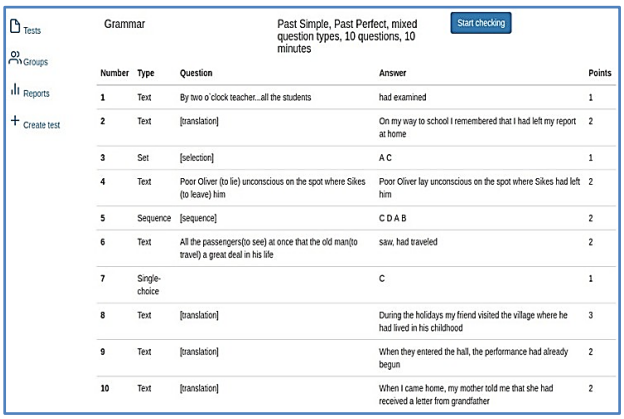

Fig. 8. Test view page

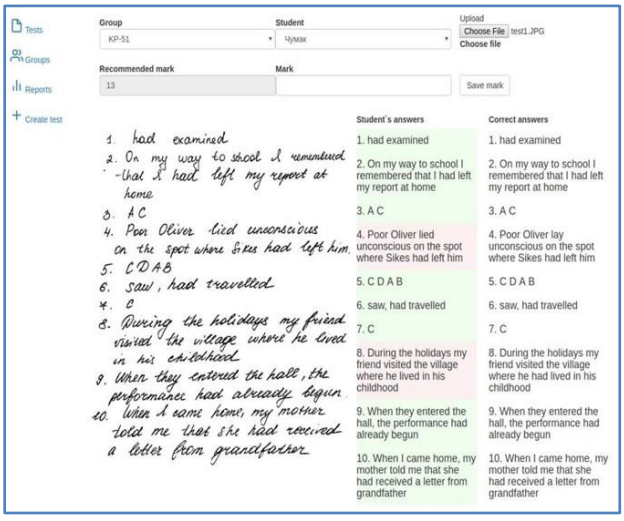

Fig. 9. Test grading page

After uploading the image, the processed black-and-white version of the work appears below this form, along with a column displaying the student's answers obtained from the image recognition process and a column with the correct answers. Each correct student answer is highlighted in green, while incorrect answers are shown in red. If the user tries to enter anything other than a simple whole number in the score field, an error message will be displayed.

The Reports page contains three dropdown menus:

- the type of chart (histogram or boxplot);
- group (all groups for the current instructor, as well as an «All» option for viewing statistics across all groups simultaneously);
- test (all tests for the current instructor, along with an «All» option for viewing statistics for all tests).

Usage recommendations for the information system. When using the web platform, teachers or instructors are advised to follow several guidelines. First, it is best to take photos in well-lit areas and use a high-quality camera, as inadequate lighting or camera quality may lead to unpredictable word recognition results. Ideally, the page with the work should be a neutral color, free from additional markings, drawings, etc. Regular lined or grid paper is acceptable.

Second, when designing the test structure, it is recommended to avoid nested tasks. The test can include numerous questions, but grouping them into separate tasks, sections, or subsections should be avoided.

Third, students should be reminded to write words with sufficient spacing between them. For closed-ended tests with multiple answers, responses should be written in a single line, without numbering, and with adequate spacing between each answer.

Additionally, students should write answers sequentially in a single column, avoid flipping the answer sheet perpendicularly, and refrain from drawing sequence arrows. Corrections, cross-outs, drawings, or any elements differing from words and numbers may significantly impair the image recognition quality.

Recommendations for further improvement. During the development of this software, a machine learning model was trained, achieving satisfactory results on the IAM dataset. However, real-world images differ significantly from those in the training data, leading to occasional issues with page segmentation and incorrect word classification by the algorithm. For future enhancement, it is necessary to collect and annotate a custom training dataset. This step was not performed during development due to the substantial time required [22]. One way to collect training data is through crowdsourcing, which would be a viable method for future use.

It should be noted that it is not necessary to discard the IAM dataset and the existing model. Instead, the transfer learning technique could be applied, whereby a model is initially trained on one dataset, and then its ability to learn low-level data abstractions in the early stages is leveraged to train on a different dataset. This approach is commonly used in computer vision and has a strong potential for success in handwritten text recognition.

Additionally, recognition accuracy could be improved by adding more layers to the neural network and increasing the input vector size to enable the recognition of whole words, sentences, and paragraphs. This approach would require more training data.

During the development process, the machine learning model was designed with limited depth because training such models is time-intensive. To achieve better results, it is recommended to both deepen the neural network and perform training on multiple GPUs instead of a CPU. It is worth considering that cloud services like AWS and Google Cloud offer access to their GPUs, though these are paid services, and training may incur significant costs.

To improve the evaluation of open-ended questions, an additional algorithm using NLP technologies could be

developed. This would better align two answers and determine whether the text contains an answer to the question.

Conclusions. As a result of this research, an information system (web platform) was developed to automate the grading of offline tests, significantly easing the assessment process for educators through optical character recognition (OCR) technology based on machine learning algorithms.

At the outset, the problem domain was defined, with an analysis of existing research and publications on potential solutions, along with a review of existing software solutions. The research problem was then formulated mathematically, and a mathematical rationale for the development's usefulness was provided.

The study includes a brief overview of the database structure, which facilitates the storage and management of large volumes of data, as well as examples of the software in operation that demonstrate the system's capabilities. The developed system can recognize handwritten text from photos, create an array of responses, and compare them with the answers provided by the teacher. This approach significantly reduces the time teachers spend on test grading.

To enhance usability, a minimalist user interface was designed, providing access to all key system functions with intuitive controls. A detailed description of the developed algorithms and machine learning models is included.

This IT project has considerable potential for future development, including integration with other educational platforms, improvement of recognition technologies, and system scalability.

Thus, the web platform not only reduces teachers' time spent on grading tests but also supports the analysis and enhancement of educational programs, accommodates various types of tests, and fosters scientific and technological progress in the education sector. The system can be adapted for use in different educational institutions, making it a universal tool for automating knowledge assessment for learners.

References

1. *ZipGrade: Touchless grading when in class, online & remote for students that are not*. URL: https://zipgrade.com (access date: 04.10.2024).
2. *GradeCam: Assessment made easy*. URL: https://gradecam.com (access date: 03.10.2024).
3. *Essay Grader for Quality Assessment*. URL: https://edubirdie.com/essay-grader (access date: 04.10.2024).
4. Klieshch D., Fedorchenko V. *Analysisof approaches to solving the problem of picture recognitionusing artificial intelligence. Management, Navigation, and Communication Systems*. URL: https://journals.nupp.edu.ua/sunz/article/view/2835/2243 (access date: 06.10.2024).
5. Axler G., Wolf L. *Toward a Dataset-Agnostic Word Segmentation Method*. URL: https://ieeexplore.ieee.org/document/8451570 (access date: 06.10.2024).
6. Arivazhagan M., Srinivasan H., Srihari S. N. *A statistical approach to line segmentation in handwritten documents*. URL: https://cedar.buffalo.edu/~srihari/papers/SPIE-2007-lineSeg.pdf (access date: 06.10.2024).
7. Manmatha R., Srimal N. *Scale space technique for word segmentation in handwritten documents*. URL: https://works.bepress.com/r_manmatha/11/ (access date: 08.10.2024).
8. Azmi R., Mohseni A., Maleki S., Layeghi K. *Handwriting Recognition: A Comprehensive Review*. URL: https://civilica.com/doc/924197 (access date: 08.10.2024).
9. Awad M. & Khanna R. *Hidden Markov Model*. URL: https://doi.org/10.1007/978-1-4302-5990-9_5 (access date: 09.10.2024).
10. Graves A., Fernández S., Gomez F., Schmidhuber J. *Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks*. URL: https://www.cs.toronto.edu/~graves/icml_2006.pdf (access date: 10.10.2024).
11. Chowdhury A., Vig, L. *An Efficient End-to-End Neural Model for Handwritten Text Recognition*. URL: https://arxiv.org/abs/1807.07965 (access date: 05.10.2024).
12. Puigcerver J. *Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?* URL: https://doi.org/10.1109/ICDAR.2017.20 (access date: 10.10.2024).
13. Zhang Hui, Yao Quanming, Kwok James, Bai Xiang. *Searching a High-Performance Feature Extractor for Text Recognition Network*. URL: https://doi.org/10.48550/arXiv.2209.13139 (access date: 12.10.2024).
14. *IAM Handwritimg Database*. URL: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database (access date: 10.10.2024).
15. Ziuziun V. *Analysis of the impact of information technologies for making management decisions, including project ones*. URL: https://www.researchgate.net/publication/371492759_Analysis_of_Aspects_of_Increasing_the_Efficiency_of_IT_Project_Management (access date: 10.10.2024).
16. Ziuziun V. *Substantiation of the importance of the role of using information technologies in business process reengineering*. URL: https://doi.org/10.46299/ISG.2023.1.32 (access date: 10.10.2024).
17. Kronecker Delta. URL: https://mathworld.wolfram.com/KroneckerDelta.htm (access date: 10.10.2024).
18. Ziuziun V., Petrenko N. *Formation of Conceptual and Logical Databases for the Project of Creating a Web Platform for Automated Verification and Evaluation of Written Tests*. URL: https://www.researchgate.net/publication/Formation_of_Conceptual_and_Logical_Database (access date: 13.10.2024).
19. *OpenCv Perspective Transformation*. URL: https://medium.com/analytics-vidhya/opencv-perspective-transformation-9edffefb2143 (access date: 15.10.2024).
20. *MVC Design Pattern*. URL: https://www.geeksforgeeks.org/mvc-design-pattern (access date: 15.10.2024).
21. *Machine Learning Mastery: A Gentle Introduction to Transfer Learning for Deep Learning*. URL: https://machinelearningmastery.com/transferlearning-for-deep-learning (access date: 15.10.2024).

References (transliterated)

1. *ZipGrade: Touchless grading when in class, online & remote for students that are not*. Available at: https://zipgrade.com (accessed 04.10.2024).
2. *GradeCam: Assessment made easy*. Available at: https://gradecam.com (accessed 03.10.2024).
3. *Essay Grader for Quality Assessment*. Available at: https://edubirdie.com/essay-grader (accessed 04.10.2024).
4. Klieshch D., Fedorchenko V. *Analysisof approaches to solving the problem of picture recognitionusing artificial intelligence. Management, Navigation, and Communication Systems*. Available at: https://journals.nupp.edu.ua/sunz/article/view/2835/2243 (accessed 06.10.2024).
5. Axler G., Wolf L. *Toward a Dataset-Agnostic Word Segmentation Method*. Available at: https://ieeexplore.ieee.org/document/8451570 (accessed 06.10.2024).
6. Arivazhagan M., Srinivasan H., Srihari S. N. *A statistical approach to line segmentation in handwritten documents*. Available at: https://cedar.buffalo.edu/~srihari/papers/SPIE-2007-lineSeg.pdf (accessed 06.10.2024).
7. Manmatha R., Srimal N. *Scale space technique for word segmentation in handwritten documents*. Available at: https://works.bepress.com/r_manmatha/11/ (accessed 08.10.2024).
8. Azmi R., Mohseni A., Maleki S., Layeghi K. *Handwriting Recognition: A Comprehensive Review*. Available at: https://civilica.com/doc/924197 (accessed 08.10.2024).

9. Awad M. & Khanna R. *Hidden Markov Model*. Available at: https://doi.org/10.1007/978-1-4302-5990-9_5 (accessed 09.10.2024).
10. Graves A., Fernández S., Gomez F., Schmidhuber J. *Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks*. Available at: https://www.cs.toronto.edu/~graves/icml_2006.pdf (accessed 10.10.2024).
11. Chowdhury A., Vig, L. *An Efficient End-to-End Neural Model for Handwritten Text Recognition*. Available at: https://arxiv.org/abs/1807.07965 (accessed 05.10.2024).
12. Puigcerver J. *Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?* Available at: https://doi.org/10.1109/ICDAR.2017.20 (accessed 10.10.2024).
13. Zhang Hui, Yao Quanming, Kwok James, Bai Xiang. *Searching a High-Performance Feature Extractor for Text Recognition Network.* Available at: https://doi.org/10.48550/arXiv.2209.13139 (accessed 12.10.2024).
14. *IAM Handwritimg Database*. Available at: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database (accessed 10.10.2024).
15. Ziuziun V. *Analysis of the impact of information technologies for making management decisions, including project ones*. Available at: https://www.researchgate.net/publication/371492759_Analysis_of_Aspects_of_Increasing_the_Efficiency_of_IT_Project_Management (accessed 10.10.2024).
16. Ziuziun V. *Substantiation of the importance of the role of using information technologies in business process reengineering*. Available at: https://doi.org/10.46299/ISG.2023.1.32 (accessed 10.10.2024).
17. *Kronecker Delta*. Available at: https://mathworld.wolfram.com/KroneckerDelta.htm (accessed 10.10.2024).
18. Ziuziun V., Petrenko N. *Formation of Conceptual and Logical Databases for the Project of Creating a Web Platform for Automated Verification and Evaluation of Written Tests*. Available at: https://www.researchgate.net/publication/Formation_of_Conceptual and_Logical_Database (accessed 13.10.2024).
19. *OpenCv Perspective Transformation*. Available at: https://medium.com/analytics-vidhya/opencv-perspective-transformation-9edffefb2143 (accessed 15.10.2024).
20. *MVC Design Pattern*. Available at: https://www.geeksforgeeks.org/mvc-design-pattern (accessed 15.10.2024).
21. *Machine Learning Mastery: A Gentle* Introduction *to Transfer Learning for Deep Learning*. Available at: https://machinelearningmastery.com/transferlearning-for-deep-learning (accessed 15.10.2024).

Received 29.10.2024

УДК 004.89:004.932:004.4

В. І. ЗЮЗЮН кандидат технічних наук (PhD), доцент, Київський національний університет імені Тараса Шевченка, доцент кафедри технологій управління, м. Київ, Україна; e-mail: vadym.ziuziun@knu.ua; ORCID: https://orcid.org/0000-0001-6566-8798

Н. А. ПЕТРЕНКО, Київський національний університет імені Тараса Шевченка, студент, м. Київ, Україна; e-mail: nikita.petrenko@knu.ua; ORCID: https://orcid.org/0009-0006-3921-8412

ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ОПТИЧНОГО РОЗПІЗНАВАННЯ СИМВОЛІВ ТА МАШИННОГО НАВЧАННЯ ДЛЯ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АВТОМАТИЧНОЇ ПЕРЕВІРКИ ОФЛАЙН-ТЕСТУВАННЯ

Під час навчання в будь-якій сфері, тестування та моніторинг знань учнів, студентів, або здобувачів освіти є невід'ємною частиною процесу. Викладачі часто витрачають багато часу на перевірку великої кількості типових тестів. Хоча системи онлайн-тестування були розроблені для полегшення цього процесу, паперові офлайн-тести залишаються популярними, оскільки не вимагають доступу до комп'ютерів, електроенергії та надійного інтернет-з'єднання. Офлайн-тестування є найбільш репрезентативним видом перевірки, але призводить до монотонної роботи для викладачів під час перевірки. Деякі викладачі використовують бланки для тестів, щоб структурувати відповіді здобувачів знань та зекономити час. У зв'язку з цим стає актуальним створення системи, яка автоматизує перевірку офлайн-тестів. Метою дослідження стала розробка інформаційної системи (вебплатформи), яка спростить процес перевірки офлайн-тестів за допомогою технологій оптичного розпізнавання символів на основі алгоритмів машинного навчання. Об'єктом дослідження стали процеси та функціональні можливості створення інформаційної системи для автоматизованої перевірки та оцінювання офлайн-тестів. Наукова новизна полягає у інтеграції та використанні алгоритмів машинного навчання разом з модифікаціями алгоритмів оброблення зображень, для створення інформаційної системи, яка дозволить аналізувати та оцінювати широкий спектр тестових офлайн завдань (відкриті, закриті, визначення послідовності, закриті з декількома правильними відповідями). Практичне значення дослідження полягає в розробці вебплатформи для автоматизації перевірки офлайн-тестів за допомогою технологій оптичного розпізнавання символів та машинного навчання, що зменшує витрати часу викладачів, дозволяє аналізувати та вдосконалювати навчальні програми, підтримує різноманітні типи тестів і сприяє науковому та технологічному прогресу у сфері освіти. Розроблена система здатна розпізнавати рукописний текст з фотографій, формувати масив відповідей та порівнювати їх з відповідями, внесеними викладачем. Це значно зменшує час, який викладачі витрачають на перевірку тестів. Для зручності використання був створений мінімалістичний інтерфейс користувача, який надає доступ до всіх основних функцій системи та забезпечує інтуїтивно зрозуміле управління. Було надано детальний опис розроблених алгоритмів та моделей машинного навчання. Даний ІТ проєкт має широкі перспективи для подальшого розвитку, зокрема інтеграція з іншими освітніми платформами, покращення технологій розпізнавання та масштабування системи.

Ключові слова: інформаційна система, вебплатформа, ІТ проєкт, машинне навчання, нейронні мережі, алгоритм, IAM датасет, оптичне розпізнавання символів, тестування, освітній процес.

Повні імена авторів / Author's full names

Автор 1 / Author 1: Зюзюн Вадим Ігорович / Ziuziun Vadym Ihorovych
Автор 2 / Author 2: Петренко Нікіта Андрійович / Petrenko Nikita Andriiovych