Lecture Notes in Electrical Engineering 1324

Andrey A. Radionov
Vadim R. Gasiyarov *Editors*

Advances in Automation VI

Proceedings of the International Russian Automation Conference, RusAutoCon2024, September 8–14, 2024, Sochi, Russia

Springer

Identification of Metal Sheets in the Flow, Based on the Marking Imprint, Using Neural Networks

A. V. Lednov(✉), D. D. Prokhorov, and A. V. Shvaleva

Novotroitsk Branch of the National Research Technological University MISIS, Frunze Str. 8, Novotroitsk 462359, Russia
aled nov@mail.ru

Abstract. The purpose of this work is the application of neural network technologies in the production flow of metallurgical enterprises. The object of the study is a sheet-rolling shop. The subject of the research is the use of neural network technologies to track and identify sheets. In real workshop conditions, it is not possible to identify a sheet using text recognition technologies due to the impact of a huge number of disturbances, as can be seen from the figures in the article. The article presents the first important idea - to use the marking imprint as a master image for further sheet recognition. The second postulate of the article is training a neural network and identifying an object from its presence in the technological network. The third idea is to create a digital twin to simulate sheet movement. And, as a result, a neural network is configured to solve the above problems. The paper discusses in detail the use of the YOLOv8 model for object detection and convolutional neural networks for comparing pairs of images. The research results are expected to be used to improve the efficiency of monitoring and control of production processes at metallurgical enterprises.

Keywords: Plate identification · Neural network · Machine learning · Convolutional network · Object modeling · Production process · Metallurgical enterprise

1 Introduction

In today's manufacturing environment, where accuracy and efficiency are critical factors, plate tracking and identification in metallurgical plants has become a critical task. Moreover, it is necessary to identify the object both in the flow and at various static points. It is proposed to use video analytics - the technology using methods of computer vision for the automated obtaining different data on the basis of the analysis of the sequence of the images arriving from video cameras in real time or from historical records [1].

A production experiment for training a video analytics model for plate tracking is technically and organizationally complex, expensive, and has a large number of disturbing influences. The work proposes the use of neural network technologies and machine learning methods on the created digital twin.

This paper discusses the use of libraries based on neural network technologies, such as Ultralytics, Tensorflow, Keras, PyTorch, to create models that help track and identify

A. A. Radionov and V. R. Gasiyarov (Eds.): RusAutoCon 2024, LNEE 1324, pp. 85–95, 2025.
https://doi.org/10.1007/978-3-031-82494-4_9

plates in the production flow at a metallurgical plant. The main goal of the research is to develop software that can effectively identify and track plates, which will improve the accuracy and efficiency of monitoring production processes.

The production process model for machine vision assumes training based on photos of real plates with markings, technological equipment (roller tables and transfer beds), which made it possible to bring testing conditions as close as possible to real production conditions, and, accordingly, increase the reproducibility and reliability of the results obtained [2–4].

The work proposes identification of a plate by a fragment with markings, since computer vision technologies do not allow reading the marking text due to the large number of disturbing influences. The search is carried out from an array of data about objects located at the technological site.

2 Dataset Formation and Neural Network Training

In this work, a dataset has been generated for training the neural network and testing the operation of the program. An image of a plate with markings is taken as a digital twin (Fig. 1), as close as possible to real conditions and with which various experiments can be carried out to improve, analyze and understand the operation of a particular model.

Fig. 1. Digital twin of a sheet with markings.

The work presents the result of the functioning of a simulation model, in which a photo of a sheet is randomly selected from a pre-generated array and used as a background for the drawing, then a marking image is generated from a test database of plate parameters in accordance with the fonts and this marking is applied to the images with plates. Also, due to special distortion of images, blurring them, and blurring inscriptions (simulating plate grinding in real conditions), it was possible to simulate more real conditions.

Also, using images from the digital twin, an animation of the movement of sheets along the roller table was made, the required dataset was generated, and the operation of the program was checked.

The images of the digital twin have different inscriptions, which differ in numbers and letters from each other, and also have different types of substrates. The required

dataset was formed by frame-by-frame cropping of images from video animation in different positions of the plate on it. For more accurate operation of the model, blurred and distorted images were also included in the dataset, for example, some inscriptions were blurred, which simulates plate grinding, images were blurred, which is possible, for example, when working with video cameras, or both were done simultaneously on one image [5, 6].

After that, a working environment was created, including all the necessary libraries for training the neural network. The popular Python distribution Anaconda [7], which is designed for scientific computing and development in the field of data, machine learning and artificial intelligence, is perfect for this task. Anaconda is not just a set of individual libraries, but a full-fledged integrated development environment. That is, all the packages and tools included in it are configured to work together. Its main difference from other package managers, such as pip, is that the installation of libraries takes into account the already installed versions and their features. When pip installs a package, it automatically installs all Python dependencies without checking whether they are compatible with those already on the computer. For example, a user who already has a production build installed, such as Tensorflow, uses pip to install a package that requires a different version of the dependent Keras library than the one currently used in Tensorflow. Because of this, he may encounter the fact that the assembly either stops working altogether or gives distorted results when performing calculations.

At the same time, the Conda package manager first checks the current environment, including all currently installed packages, as well as their version restrictions. And based on this analysis, the program either selects a set of dependencies compatible with them, or notifies the user that this is impossible.

The environment is an isolated environment (virtual machine) that simulates the operation of a full-fledged physical PC. The virtual environment manager built into Anaconda is responsible for its creation and operation, which allocates part of the PC's physical resources (RAM, disk space, processor power, etc.) for this purpose. With this tool, you can create multiple workspaces at the same time. This feature simplifies the work, since for each specific task you can create your own environment with specific libraries.

Using Anaconda, a working environment was installed with all the necessary libraries for training and testing the model.

To train the neural network to detect plate markings, major libraries were used, such as Ultralytics with its YOLOv8 model, PyTorch and LabelImg. Using LabelImg [8], labels were placed and classes were specified on the images for training the model. (Fig. 2). Labels and classes provide the model with information about where objects are in the image (via bounding boxes) and what classes those objects belong to. This allows the model to learn to recognize objects of different classes in images. Labels and classes also help evaluate how well a model has learned to recognize different objects by comparing the model's predictions with the actual labels and their classes. For each object in the image, a label is created, usually in the form of a bounding box, that indicates the spatial location of the object. The mark contains coordinates that determine the location and size of the frame. Each label is assigned a class that indicates the type of object within

the bounding box (for example, “dog”, “cat”, “car”). Depending on the task, there may be several different classes.

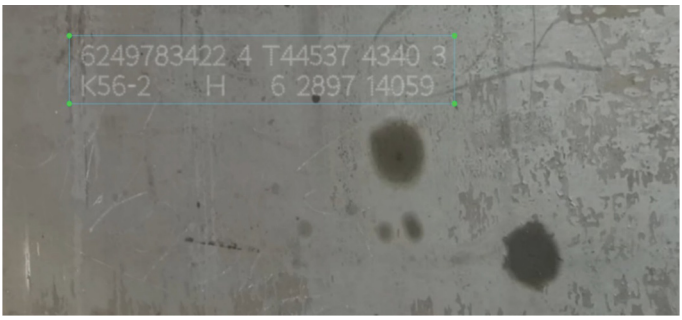

Fig. 2. Placement of label and class on the image.

As can be seen from Fig. 2, the name List was chosen as the class, which is also used in the Python script to access data sets. After placing all the labels and specifying the classes, the frame coordinates and object classes are saved in the corresponding text files, through which the model is provided with information about where the objects are located in the image and what classes these objects belong to. Also, the operation of the program during any actions or changes is displayed in the Anaconda command line.

The PyTorch library [9] has CUDA technology, which allows you to run calculations both on the processor and on the video card. The peculiarity of deep learning is a serious resource-intensive task; models consume significant computer resources during training and operation. Therefore, to run them, you can use graphics processors of video cards, as they are more powerful and suitable for complex calculations. This technology has been successfully applied for faster training.

Choosing the Ultralytics library to work with the YOLOv8 model is due to several key advantages:

1. Ultralytics YOLOv8 is one of the fastest and most accurate models for real-time object detection. This makes it ideal for projects that require high performance with limited computing resources, such as real-time monitoring on production lines.
2. YOLOv8 allows you to easily adapt the model to specific tasks by retraining on a new data set, which is ideal for a specific production task - detecting markings on a sheet.

The next stage is determining the volume of the dataset and the number of epochs for training, since this is very important for the correct and more accurate operation of the model. Using a small or large number of images in combination with a different number of training epochs has asignificant impact on the process and results of model training, here is how this can affect depending on the number of epochs and the volume of images.

1. Few images, many eras. Fewer images require less time per epoch, which speeds up the learning process. The model can carefully study the provided examples, identifying

subtle features and patterns. But the main risk when training on a small dataset with a large number of epochs is overfitting. The model "remembers" training examples rather than generalizing patterns, which reduces its effectiveness on new data. Also, the model may not learn to recognize objects in a variety of conditions due to the limited training set.

2. Many images, few eras. A larger data set allows the model to see more variation across objects and conditions, leading to better generalization. The risk of overfitting is reduced because the model has to be trained on a larger, more diverse set of examples. If the number of epochs is too small, the model may not have time to adequately adapt to the training data, which will reduce its accuracy. In addition, training on a large dataset requires more time and computational resources, even if the number of epochs is smaller.

The choice of the optimal combination of the number of images and the number of epochs depends on the specific conditions and goals of the project. It is ideal to strive for a balance that provides sufficient training without overtraining, given time and resource constraints. Various methods can be used to achieve this balance. For example, using regularization techniques that can help prevent overfitting when working with a small data set. Using data augmentation can effectively increase the size and diversity of the training set without requiring additional images. Monitoring model performance on the validation dataset and stopping training when improvements are no longer observed helps avoid overfitting. Using cross-validation to evaluate model performance on different subsets of data can help determine the best tradeoff between dataset size and number of epochs.

It is worth noting that the Ultralytics library has a built-in early stopping function when the model's performance on the validation dataset stops improving within a given number of epochs, which avoids overfitting.

The dataset was also divided into validation and training sets, which has its own goals and advantages in the process of training models. The main purpose of splitting into training and validation sets is to evaluate how well the model is able to generalize information from data it has not seen before. This provides important information about the model's performance in the real world, where it will encounter previously unseen data. Dividing the dataset helps detect overfitting, which is a situation where the model adapts too well to the training data, losing its ability to generalize. The validation set makes it possible to check whether the model has started to "remember" the training data instead of "understanding" it. In our case, the dataset was divided into approximately 80% training and 20% validation data sets, which were placed in appropriate folders. These folders are accessed during the learning process through a pre-prepared Python script.

After determining the volume and number of epochs, the YOLOv8 model [10] was retrained for our production task. At the end of model training, the Ultralytics library [11] creates a folder with various graphs and indicators for model training (Figs. 3 and 4), for example, loss plots that allow you to track how the model's loss changes throughout the training process, for training and validation sets. This is critical to determining how well the model is learning and helps in identifying overfitting or underfitting. Accuracy plots, which display the change in model accuracy on the training and validation datasets.

This allows you to evaluate how well the model is able to generalize to new data. Error matrices, which provide a detailed view of the model's performance for each class, showing which classes are most likely to be confused with each other. Ultralytics also allows you to visualize examples of detections in images, which is a clear way to check the quality of the trained model. All this allows us to better evaluate the accuracy and performance of the model. At the end of training, the model itself is also created with the extension pt.

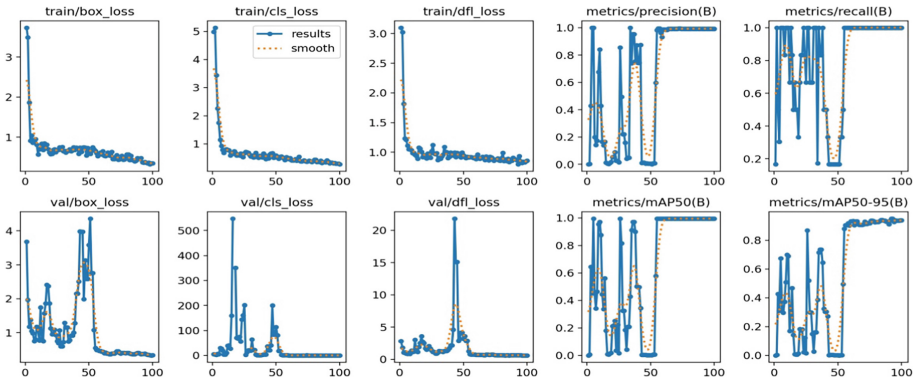

Fig. 3. Performance graphs.

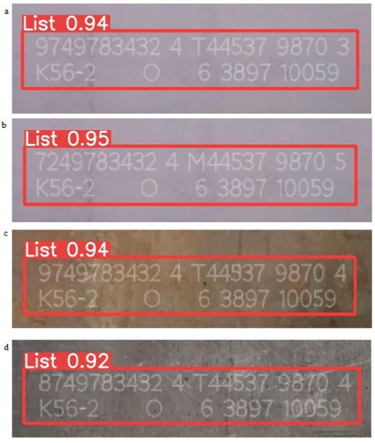

Fig. 4. Examples of marking detection.

To ensure correct comparison of images in the sheet identification task, it is necessary to take into account that the images must be on different substrates. This is due to the fact that convolutional neural networks primarily recognize external features of an image rather than specific text. Even if the text in the images differs by only one letter or number, the likelihood of a correct comparison is significantly reduced. However, if the images are located on different substrates, the convolutional neural network is more likely to be

able to correctly identify them. For example, in Fig. 5 a and b, the sheets have the same background, but the convolutional neural network recognizes these images by several different letters and numbers. In the case of Fig. 5 c and d, where the images differ by only one digit, the convolutional neural network correctly recognizes them due to differences in the substrates.

Next, it was necessary to identify the plate in real time. Since recognition of the text itself is impossible within the framework of production, due to the fact that during production the markings are partially erased, it was decided to make an imprint of the marking (similar to a person's biometrics) and use it to identify it. Accordingly, it was decided to train a convolutional neural network to identify features of similarity and difference. With the help of which two images with markings will be compared.

To train the convolutional neural network, the TensorFlow [12] and Keras [13] libraries were used, as well as labeled images that were automatically taken after detecting images with plates. The output was a model with h5 extension capable of identifying signs of similarity and difference in labeled images.

3 Software Development and Algorithm of Its Work

All trained models had to be integrated into Python code and its operating algorithm thought through (Fig. 5) in order to form a single working system, which would later be combined into a software product [14, 15].

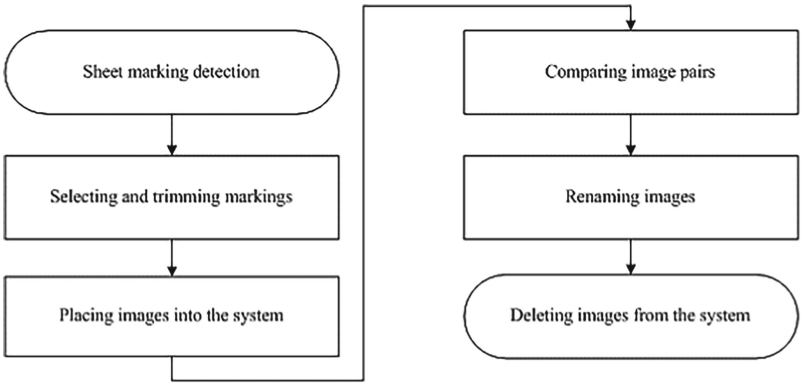

Fig. 5. Algorithm flowchart.

The key element of the system when modeling the production process is a conditional "master camera", with which the reference image of passing plates will be read and all subsequent comparisons will be made with its image. And also in the model, other cameras are conditionally installed in other areas, from which images of the plates will be taken, and they will be compared with the image of the "master camera" [16]. The identification of plates will be the same name of the images in the software, and to track them, it will be important to simply remove similar pairs of images from previous sections from the system (Fig. 6).

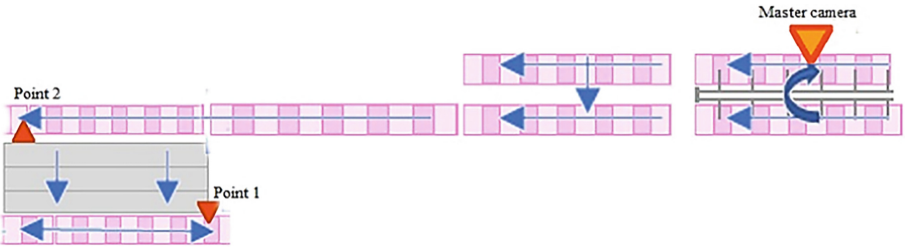

Fig. 6. Camera installation.

Initially, markings on a plates are detected using a trained model on a video stream, while the conditional center of the screen of this video stream is taken to prevent detection of the same plate, then these images with the selected markings are cropped and placed in the system. Next, using another trained convolutional neural network model, features of similarity and difference are extracted from pairs of images. In this case, the method of comparing images through the distances between feature vectors obtained at the output of the model layers is used. After training the model on input images and extracting image features from the database, these features are compared with the image features detected in the video file. Cosine similarity is used for comparison. This comparison method evaluates the similarity between images based on their feature representations. It calculates the cosine of the angle between feature vectors, where values close to 1 indicate high similarity between images, and values close to 0 indicate little or no similarity. This method allows identifying the most similar images from the database for each detected frame from a video file, based on their feature representations. Then the image pairs with the highest similarity values are renamed. Finally, images from the system that are responsible for a specific area are removed.

Next, the software was developed in visual studio in C# (Figs. 7, 8, and 9). The main work is done through running Python scripts. This program visualizes the video stream for each area where the camera is located, visualizes the resulting images after they have been selected and cropped in the appropriate panels, and also displays the work of Python scripts in text fields.

.

4 Conclusions

The results of the study showed that the use of neural networks to improve the quality of production processes at metallurgical enterprises is a very promising direction. Visualization and analysis of the program's operation confirmed its ability to accurately identify plates in real production conditions, which helps to increase the overall efficiency and productivity of the Plate Rolling Mill, and also confirms the correct operation of neural network technologies.

Thus, the developed software and the methodology for its application can be successfully implemented at other metallurgical enterprises to optimize the monitoring and control process, which will ultimately lead to improved product quality and reduced production costs.

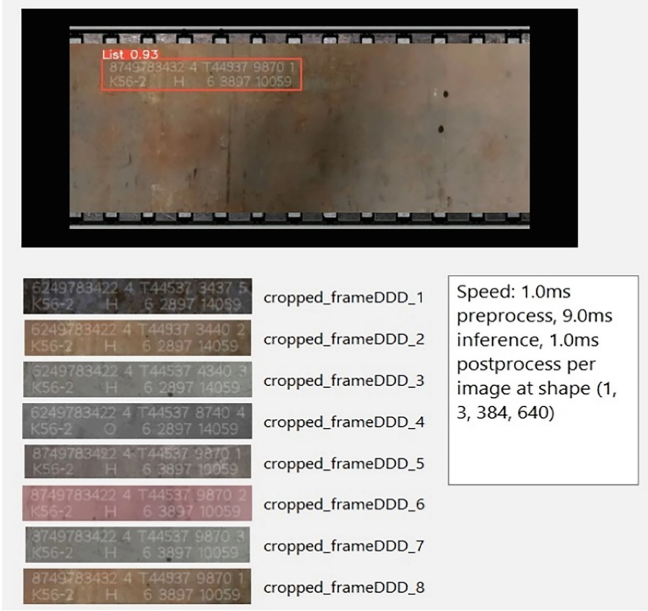

Fig. 7. Program interface ***(master camera)***.

Fig. 8. Program interface ***(point 1)***.

The study showed the high efficiency of using neural network technologies and machine learning methods to track and identify plates in the production flow of a metallurgical enterprise. The main factors that ensured the success of the study were the following:

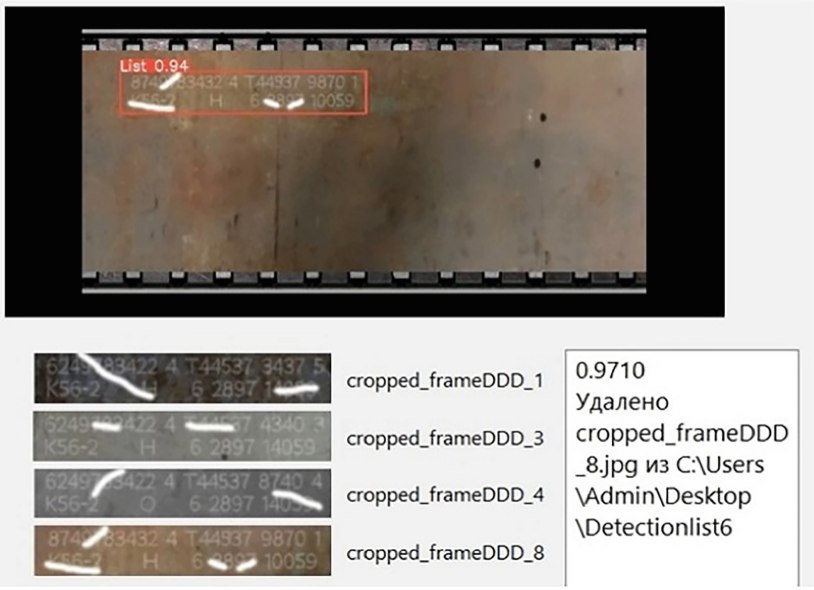

Fig. 9. Program interface (point 2).

1. A digital twin of the Plate Rolling Mill was developed and successfully tested, which made it possible to simulate real production conditions and conduct experiments in a safe and controlled environment.
2. The created dataset, which included images of plates with markings, as well as their distorted and blurred versions, allowed us to train a neural network and create appropriate models that help effectively track and identify plates. This approach allowed us to get as close as possible to real production conditions and ensured high accuracy of the model even in conditions of significant visual noise.
3. The use of modern libraries for machine learning and neural networks, such as Ultralytics, TensorFlow, Keras and PyTorch, made it possible to develop and implement complex models that showed excellent results in conditions close to real production.
4. The developed software that integrates trained models into a single system has proven its effectiveness in real time. It allows you to track and identify plates with high accuracy and efficiency, which helps improve the monitoring and control of production processes.

As a result of the research, the goal of creating software for tracking plates using neural network technologies was achieved. The results obtained can be successfully used to optimize production processes at metallurgical enterprises, which will ultimately lead to improved product quality.

References

1. Video analytics: Terms, scopes of application, technologies. In: Official website of TAdviser State Business It (2024). https://tadviser.com/index.php/. Accessed 30 Jun 2024

2. Parsunkin, B.N., Lednov, A.V., Sukhonosova, T.G., Lednova, J.: Testing signals formation for identification of heat power objects and control systems comparison. Int. J. Adv. Manuf. Technol. **93**(9–12), 3429–3436 (2017). https://doi.org/10.1007/s00170-017-0788-x
3. Itskovich, E.L.: Methods for rational automation of production. Infra, p 255 (2009)
4. Parsunkin, B.N., Obukhov, G.F., Lednov, A.V., et al.: Formation of testing signals to identify heat and power management objects. Energy: Proc. Univ. **6**, 65–70 (1988)
5. Shilov, R.E., Logunova, O.S., Lednov, A.V.: Methods and algorithms for segmentation of the image in control flotation process. RusAutoCon. International Russian Automation Conference, Sochi, 09–16 September, pp 1–4 (2018). https://doi.org/10.1109/RUSAUTOCON.2018.8501678
6. Lednov, A.V., Logunova, O.S., Kukhta, Y.B., Khudyakov, P.Y.: Centralized control system of the flotation department: Ross-eshby principle. ISAE Conference Series: International Conference on Automatics and Energy, p 012079 (2021). https://doi.org/10.1088/1742-6596/2096/1/012079
7. Operating system for artificial intelligence (2024) Anaconda. https://www.anaconda.com/. Accessed 30 Jun 2024
8. Image markup tool (2024) LabelImg. https://github.com/HumanSignal/labelImg. Accessed 30 Jun 2024
9. Machine learning framework (2024) PyTorch. https://pytorch.org/. Accessed 30 Jun 2024
10. Complete YOLO v8 Custom Object Detection Tutorial Windows & Linux (2024). https://www.youtube.com/watch?v=gRAyOPjQ9_s&list=PLR4jMwpFn4C2fuZgBZRtOAYH7NoIE3mdO&index=7. Accessed 30 Jun 2024
11. An AI platform for building, training and deploying machine learning models (2024) Ultralytics. https://www.ultralytics.com/ru. Accessed 30 Jun 2024
12. Comprehensive platform for machine learning (2024) TensorFlow. https://www.tensorflow.org/?hl=ru. Accessed 30 Jun 2024
13. Machine learning library (2024) Keras. https://keras.io/. Accessed 30 Jun 2024
14. Lednov, A.V., Parsunkin, B.N., Obukhov, G.F., Ryabkov, V.M., Degtyarev, V.V.: Operative determination of the work effectiveness of the microprocessor process control systems. Steel **9**, 101–104 (1987)
15. Shvaleva, A.V.: Professional orientation of the system of mathematical training of students of technical direction. In: Bubnov, J.A. (ed.) Vestnik of Voronezh State University. Series: Problems of Higher Education, vol. 3, pp. 107–112 (2014)
16. Anfilatov, V.S., Emelyanov, A.A., Kukushkin, A.A.: System analysis in management: a tutorial, p. 368. Finance and Statistics, Moscow (2009)