ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ ҒЫЛЫМ ЖӘНЕ ЖОҒАРЫ БІЛІМ МИНИСТРЛІГІ
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН
MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE REPUBLIC OF KAZAKHSTAN

ХАЛЫҚАРАЛЫҚ АҚПАРАТТЫҚ ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР ЖУРНАЛЫ

МЕЖДУНАРОДНЫЙ ЖУРНАЛ ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

2025 (23) 3

шілде- қыркүйек

ISSN 2708–2032 (print)
ISSN 2708–2040 (online)

Халықаралық ақпараттық және коммуникациялық технологиялар журналы
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Меншік иесі: АҚ «Халықаралық ақпараттық технологиялар университеті» (Алматы қ.).
Қазақстан Республикасы Ақпарат және қоғамдық даму министрлігіне мерзімді баспасөз басылымын есепке қою туралы куәлік № KZ82VPY00020475, 20.02.2020 ж. берілген
Тақырып бағыты: ақпараттық технологиялар, ақпараттық қауіпсіздік және коммуникациялық технологиялар, әлеуметтік-экономикалық жүйелерді дамытудағы цифрлық технология.
Мерзімділігі: жылына 4 рет.
Тираж: 100 дана.
Редакция мекенжайы: 050040 Алматы қ., Манас к., 34/1, каб. 709, тел: +7 (727) 244-51-09.
E-mail: ijict@iitu.edu.kz
Журнал сайты: https://journal.iitu.edu.kz

Международный журнал информационных и коммуникационных технологий
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Собственник: АО «Международный университет информационных технологий» (г. Алматы).
Свидетельство о постановке на учет периодического печатного издания в Министерство информации и общественного развития Республики Казахстан **№ KZ82VPY00020475,** выданное от **20.02.2020 г.**
Тематическая направленность: информационные технологии, информационная безопасность и коммуникационные технологии, цифровые технологии в развитии социо-экономических систем**.**
Периодичность: 4 раза в год.
Тираж: 100 экземпляров.
Адрес редакции: 050040 г. Алматы, ул. Манаса 34/1, каб. 709, тел: +7 (727) 244-51-09.
E-mail: ijict@iitu.edu.kz Сайт журнала: https://journal.iitu.edu.kz

«International Journal of Information and Communication Technologies»
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Owner: International Information Technology University JSC (Almaty).
The certificate of registration of a periodical printed publication in the Ministry of Information and Social Development of the Republic of Kazakhstan, Information Committee No. KZ82VPY00020475, issued on 20.02.2020.
Thematic focus: information technology, digital technologies in the development of socio-economic systems, information security and communication technologies
Periodicity: 4 times a year.
Circulation: 100 copies.
Editorial address: 050040. Manas st. 34/1, Almaty. +7 (727) 244-51-09. E-mail: ijict@iitu.edu.kz
Journal website: https://journal.iitu.edu.kz

МАЗМҰНЫ

СОДЕРЖАНИЕ

CONTENTS

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Vol. 6. Is. 3. Number 23 (2025). Pp. 202–221
Journal homepage: https://journal.iitu.edu.kz
https://doi.org/10.54309/IJICT.2025.23.3.012
UDC 004.032.26

CONVOLUTIONAL NEURAL NETWORK FOR RECOGNITION AND TRACKING OF OBJECT DISPLACEMENTS

L. Kurmangaziyeva[1], O. Findik[2], V. Makhatova[1*], D. Kudabayeva[1], A. Maratuly[1]
[1]Kh. Dosmukhamedov Atyrau University, Atyrau, Kazakhstan;
[2] Karabuk University, Karabuk, Turkey.
E-mail: mahve@mail.ru

Kurmangaziyeva Lyailya — Candidate of Technical Sciences, professor of the Department of Software Engineering, Kh. Dosmukhamedov Atyrau University, Atyrau, Kazakhstan
E-mail: kurmangazieval@mail.ru, https://orcid.org/0000000306407306;
Findik Oguz — PhD, Department of Computer Engineering, Karabuk University, Karabuk, Turkey
E-mail: oguzfindik@karabuk.edu.tr, https://orcid.org/0000-0001-5069-6470;
Makhatova Valentina — Candidate of Technical Sciences, professor of the Department of Software Engineering, Kh. Dosmukhamedov Atyrau University, Atyrau, Kazakhstan
E-mail: mahve@mail.ru, https://orcid.org/0000000240829193**;**
Kudabayeva Danagul — Master of Information Technology, lecturer, of the Department of Software Engineering, Kh. Dosmukhamedov Atyrau University, Atyrau, Kazakhstan
E-mail: kudabaeva.dawa@gmail.co, https://orcid.org/0009-0000-6988-6055;
Maratuly Ali — Master of Information Technology, lecturer, of the Department of Software Engineering, Kh.Dosmukhamedov Atyrau University, Atyrau, Kazakhstan
E-mail: maratulyali01@gmail.com, https://orcid.org/0009-0000-1702-885X.

Abstract. The article presents the development and implementation of a convolutional neural network (CNN) designed to recognize faces and hands in digital images, as well as analyze their displacements between two consecutive frames. The proposed method is not limited to the detection task but is supplemented by calculating the magnitude of the movements of recognized objects, which allows recording the dynamics of poses. To train the model, a specialized dataset of 1200 images was created, containing about 1800 faces and 2400 hands marked in the Pascal VOC for-

mat. MobileNetV2 with an SSD head was chosen as the architecture, training was carried out in the Keras framework using data augmentation techniques. Experiments showed the result mAP@0.5 = 0.76 on the test set and the accuracy of displacement classification of 82.5 %. Compared with existing solutions (OpenPose, MediaPipe, TensorFlow Object Detection API), the proposed approach provides an optimal balance between accuracy and computational efficiency, complementing the functionality with the ability to analyze displacements. This method can be applied in real time to tasks of gesture recognition, human-machine interaction and video surveillance.

Keywords: convolutional neural network, object detection, face recognition, hand recognition, bias analysis, Keras, MobileNetV2

For citation: L. Kurmangaziyeva, O. Findik, V. Makhatova, D. Kudabayeva, A. Maratuly. Convolutional neural network for recognition and tracking of object displacements//International journal of information and communication technologies. 2025. Vol. 6. No. 23. Pp. 202–221. (In Eng.). https://doi.org/10.54309/IJICT.2025.23.3.012.

Acknowledgment. *This research was carried out with the financial support of Kh. Dosmukhamedov Atyrau University under Agreement No. 2 dated April 1, 2025, within the scientific project titled "Application of Artificial Intelligence Methods for the Development of Decision Support Systems for the Management of Oil Refining Facilities."*

Conflict of interest: The authors declare that there is no conflict of interest.

ОБЪЕКТІЛЕРДІ ТАНУ ЖӘНЕ ЫҒЫСУЫН БАҚЫЛАУҒА АРНАЛҒАН КОНВОЛЮЦИЯЛЫҚ НЕЙРОНДЫҚ ЖЕЛІ

Л. Курманғазиева[1], О. Фиңдик [2], В. Махатова[1]*, Д. Құдабаева[1], А. Маратұлы[1]

[1]Х. Досмұхамедов атындағы Атырау университеті, Атырау, Қазақстан;
[2]Қарабүк университеті, Қарабүк, Түркия.
E-mail: mahve@mail.ru

Курманғазиева Ляйля— техника ғылымдарының кандидаты, Х. Досмұхамедов атындағы Атырау университетінің «Бағдарламалық инженерия» кафедрасының профессоры
E-mail: kurmangazieval@mail.ru, https://orcid.org/0000000306407306;
Фиңдик Огуз — философия докторы (PhD), Қарабүк университетінің Компьютерлік инженерия кафедрасы, Қарабүк, Түркия
E-mail: oguzfindik@karabuk.edu.tr, https://orcid.org/0000-0001-5069-6470;
Махатова Валентина — техника ғылымдарының кандидаты, Х. Досмұхамедов атындағы Атырау университетінің «Бағдарламалық инженерия» кафедрасының профессоры
E-mail: mahve@mail.ru, https://orcid.org/0000000240829193;

Құдабаева Данагүл — ақпараттық технологиялар мамандығы бойынша магистрі, Х. Досмұхамедов атындағы Атырау университетінің «Бағдарламалық инженерия» кафедрасының оқытушысы
E-mail: kudabaeva.dawa@gmail.co, https://orcid.org/0009-0000-6988-6055;
Маратұлы Әли — ақпараттық технологиялар мамандығы бойынша магистрі, Х. Досмұхамедов атындағы Атырау университетінің «Бағдарламалық инженерия» кафедрасының оқытушысы
E-mail: maratulyali01@gmail.com, https://orcid.org/0009-0000-1702-885X.

Аннотация. Мақалада цифрлық кескіндердегі беттер мен қолдарды тануға, сондай-ақ олардың екі дәйекті кадр арасындағы ығысуын талдауға арналған конволюциялық нейрондық желіні (CNN) әзірлеу және іске асыру қарастыры-лады. Ұсынылған әдіс тек детекция міндетімен шектелмей, сонымен қатар та-нылған объектілердің орын ауыстыру шамасын есептеумен толықтырылады, бұл қалып динамикасын тіркеуге мүмкіндік береді. Модельді оқыту үшін шамамен 1800 бет пен 2400 қолдан тұратын, Pascal VOC форматында таңбаланған 1200 суреттен құралған арнайы деректер жинағы жасалды. Архитектура ретінде SSD-басымен MobileNetV2 таңдалды, ал оқыту Keras фреймворкінде деректерді аугментациялау әдістерін қолдану арқылы жүргізілді. Эксперименттік нәтижелер тестілік таңдамада mAP@0.5 = 0.76 және ығысу классификациясының дәлдігі 82,5 % деңгейін көрсетті. Қолданыстағы шешімдермен (OpenPose, MediaPipe, TensorFlow Object Detection API) салыстырғанда ұсынылған тәсіл дәлдік пен есептеу тиімділігінің оңтайлы теңгерімін қамтамасыз етеді және функционалды ығысу талдауымен толықтырады. Әдіс нақты уақыт режимінде ым-ишараттарды тану, адам мен машина арасындағы өзара әрекеттесу және бейнебақылау міндеттерінде қолданылуы мүмкін.

Түйінді сөздер: конволюционды нейрондық желі, нысанды анықтау, бетті тану, қолды тану, қиғаш талдау, Keras, MobileNetV2

Дәйексөздер үшін: Л. Құрманғазиева, О. Финдик, В. Махатова, Д. Құдабаева, А. Маратұлы. Объектілерді тану және ығысуын бақылауға арналған конволюциялық нейрондық желі//Халықаралық ақпараттық және коммуникациялық технологиялар журналы. 2025. Том. 6. № 23. 202–221 бет. (Ағыл). https://doi.org/10.54309/IJICT.2025.23.3.012.

Алғыс білдіру. *Бұл зерттеу Х. Досмұхамедов атындағы Атырау университетінің қаржылық қолдауымен, 2025 жылғы 1 сәуірдегі №2 келісімшарт бойынша «Мұнай өңдеу нысандарын басқаруға арналған шешім қабылдауды қолдау жүйелерін құруда жасанды интеллект әдістерін қолдану» атты ғылыми жоба аясында орындалды.*

Мүдделер қақтығысы: Авторлар осы мақалада мүдделер қақтығысы жоқ деп мәлімдейді.

СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ ДЛЯ РАСПОЗНАВАНИЯ И ОТСЛЕЖИВАНИЯ СМЕЩЕНИЙ ОБЪЕКТОВ

Л. Курмангазиева[1], О. Финдик[2], В. Махатова[1]*, Д. Кудабаева[1], А. Маратулы[1]
[1] Атырауский университет имени Х. Досмухамедова, Атырау, Казахстан;
[2] Карабюкский университет, Карабюк, Турция.
E-mail: mahve@mail.ru

Курмангазиева Ляйля — кандидат технических наук, профессор кафедры «Программная инженерия», Атырауский университет имени Х. Досмухамедова, Атырау, Казахстан
E-mail: kurmangazieval@mail.ru, https://orcid.org/0000000306407306;
Финдик Огуз — доктор философии (PhD), кафедра компьютерной инженерии, Карабюкский университет, Карабюк, Турция
E-mail: oguzfindik@karabuk.edu.tr, https://orcid.org/0000-0001-5069-6470;
Махатова Валентина — кандидат технических наук, профессор кафедры «Программная инженерия», Атырауский университет имени Х. Досмухамедова, Атырау, Казахстан
E-mail: mahve@mail.ru, https://orcid.org/0000000240829193**;**
Кудабаева Данагуль — магистр информационных технологий, преподаватель кафедры «Программная инженерия», Атырауский университет имени Х. Досмухамедова, Атырау, Казахстан
E-mail: kudabaeva.dawa@gmail.co, https://orcid.org/0009-0000-6988-6055;
Маратулы Али — магистр информационных технологий, преподаватель кафедры «Программная инженерия», Атырауский университет имени Х. Досмухамедова, Атырау, Казахстан
E-mail: maratulyali01@gmail.com, https://orcid.org/0009-0000-1702-885X.

Аннотация. В статье представлена разработка и реализация сверточной нейронной сети (CNN), предназначенной для распознавания лиц и рук на цифровых изображениях, а также анализа их смещений между двумя последовательными кадрами. Предложенный метод не ограничивается задачей детекции, а дополняется вычислением величины перемещений распознанных объектов, что позволяет фиксировать динамику поз. Для обучения модели был создан специализированный датасет из 1200 изображений, содержащий около 1800 лиц и 2400 рук, размеченных в формате Pascal VOC. В качестве архитектуры выбрана MobileNetV2 с SSD-головой, обучение проводилось во фреймворке Keras с применением техник аугментации данных. Эксперименты показали результат mAP@0.5 = 0.76 на тестовой выборке и точность классификации смещений 82,5 %. По сравнению с существующими решениями (OpenPose,

MediaPipe, TensorFlow Object Detection API), предложенный подход обеспечивает оптимальный баланс между точностью и вычислительной эффективностью, дополняя функционал возможностью анализа смещений. Метод может применяться в реальном времени для задач распознавания жестов, человеко-машинного взаимодействия и видеонаблюдения.

Ключевые слова: сверточная нейронная сеть, детекция объектов, распознавание лиц, распознавание рук, анализ смещений, Keras, MobileNetV2

Для цитирования: Л. Курмангазиева, О. Финдик, В. Махатова, Д. Кудабаева, А. Маратулы. Сверточная нейронная сеть для распознавания и отслеживания смещений объектов//Международный журнал информационных и коммуникационных технологий. 2025. Т. 6. No. 23. Стр. 202–221. (На анг.). https://doi.org/10.54309/IJICT.2025.23.3.012.

Благодарность: *Исследование выполнено при финансовой поддержке Атырауского университета имени Х. Досмухамедова в рамках договора №2 от 01.04.2025 по научному проекту «Использование методов искусственного интеллекта для создания систем поддержки принятия решений для управления объектами нефтепереработки».*

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Introduction

Modern computer vision methods are actively used to analyze images and video streams, solving problems of recognizing faces, gestures and human movements. One of the key areas is the development of algorithms that can not only identify objects in an image but also track their changes over time. Existing systems such as OpenPose**,** MediaPipe And TensorFlow Object Detection APIs demonstrate high efficiency in pose and object recognition tasks. However, they have several limitations: high computational complexity (OpenPose), limited custom capabilities additional training (MediaPipe), or considerable time costs for training (TensorFlow Object Detection API). In applied tasks where it is necessary to track only certain classes of objects (for example, a face and a hand), such solutions turn out to be redundant or too resource intensive. This paper proposes a method based on a convolutional neural network with the MobileNetV2 architecture and SSD, implemented in the Keras framework. Unlike traditional detectors, it is supplemented with an algorithm for analyzing object displacements between images, which allows recording movements and interpreting the dynamics of poses. A specialized dataset with face and hand markings, including 1200 images, was created for training. The aim of the study is to develop and implement a model that combines high speed, the ability to retrain your own data, and advanced functionality for bias analysis. The results confirm that the proposed approach provides sufficient accuracy with moderate computational costs and can be used in real applications: video surveillance systems, human-machine interaction, and gesture recognition.

Materials and Methods.

Artificial Neural Network Development Environment.

The artificial neural network development environment is a complex and multifunctional toolkit designed to simplify and speed up the process of developing, testing, and implementing machine learning models. Such an environment combines many components and capabilities that help developers create efficient and accurate models that solve a wide range of applied problems (Anarbekova at.al, 2024).

Key elements of the development environment include tools for visual design of neural network architectures, libraries of ready-made modules and layers, powerful computing power for training models, and an intuitive interface for monitoring and debugging. Visual tools allow you to model complex neural networks, connect nodes, and adjust parameters without having to write extensive program code. This allows you to expand the range of users of the platform to include specialists who are not experts in programming (Aitim at.al., 2025).

In addition, advanced artificial neural network development environments provide tools to automate many stages of development, including data preparation, hyperparameter optimization, and model testing and validation. This allows developers to focus on research and creativity, minimizing routine tasks and increasing productivity (Atkinson, Tatnall, 1997).

Most modern development environments include support for popular frameworks such as TensorFlow, Theano, PyTorch, and Keras, providing flexibility and power when creating and deploying models. These environments often support integration with cloud services and can leverage distributed computing resources, which is especially important when handling big data and complex tasks that require significant computing power (Bakytnur at.al., 2022).

– TensorFlow, developed by Google Brain, is an open-source framework that is actively maintained and developed by the community. An important feature of TensorFlow is its ability to process calculations on various platforms, including processors, graphics accelerators, and specialized TPU (Tensor Processing Unit) chips. TensorFlow provides the flexibility to build neural networks of any complexity, from simple perceptrons to complex convolutional and recurrent models (Dyusembaev, 2001).

The system also offers a visualization tool, TensorBoard, which allows you to analyze model graphs, track changes in training parameters, and visualize loss curves. This makes TensorFlow especially useful for research projects where visualization and understanding of the training process are key.

– Theano, created at the University of Montreal, was one of the first deep learning frameworks with automatic differentiation. Theano has the unique ability to optimize computational processes, allowing users to efficiently use available hardware resources, such as graphics processing units (GPUs). Although its development ceased in 2017, many modern tools and libraries still use Theano's work.

Theano allows you to define and optimize mathematical expressions, includ-

ing multidimensional arrays. This library has helped lay the foundation for future developments by providing developers with a powerful tool for forming and testing their hypotheses and models in the context of deep learning.

– PyTorch, developed by Facebook AI Research, stands out for its intuitive and flexible interface that is focused on research and prototyping. Unlike TensorFlow, PyTorch offers dynamic graph construction, allowing the model structure to be modified at runtime. This makes it easier to debug and experiment with new ideas, making PyTorch popular among researchers and engineers looking for rapid prototyping.

PyTorch actively supports machine learning on GPUs and CPUs, as well as integration with other systems, including C++ and scripting languages. This makes it a powerful tool for developing complex models that require high performance and flexibility.

– Keras is an open-source Python library that serves as the primary framework for developing artificial neural networks. Its flexibility and ease of use make it popular among both beginners and experienced machine learning developers. It allows for rapid prototyping, experimentation, and implementation of various deep learning models (Dyusembaev at.al., 2017).

One of the key features of Keras is its high-level API, which simplifies the creation of complex multi-layer neural networks. It is designed to interact with well-known frameworks such as TensorFlow and Theano, providing flexibility and scalability of the final solution. Thanks to this, Keras can use graphics processing units (GPUs), which significantly speeds up the training of models, especially when working with large data sets.

Keras implements a variety of layers, including Dense, Dropout, Convolutional, and Recurrent, which can be easily combined and customized to achieve better results. This allows developers to focus on high-level problems and innovative ideas rather than getting bogged down in implementation details. Moreover, Keras supports many concepts such as backpropagation, gradient descent optimization, and weight regularization, making it a powerful tool for AI research and development. (Dyusembaev at.al., 2013; Liao et al., 2018).

In addition, Keras has extensive documentation and a large community of users who share useful examples and developments. This provides users not only with access to knowledge and best practices, but also with ongoing support when solving complex problems.

Thus, modern artificial neural network development environments are becoming an integral tool in the arsenal of machine learning specialists, providing the necessary resources and supporting the entire development cycle from idea to implementation into industrial operation. At the same time, today TensorFlow, Theano, PyTorch and Keras provide a wide range of opportunities for the implementation of artificial neural networks. Each of them has its own unique features and advantages that satisfy the diverse needs and preferences of developers (Dyusembaev at.al, 2013). Thanks to these tools, the process of creating and implementing neural networks has become

much more accessible and efficient, which contributes to further discoveries and innovations in the field of artificial intelligence (Naumov, 2017).

Problem.

Statement.

Two images are given. It is necessary to use a neural network to recognize hands and faces of people in them and determine whether the positions of the recognized objects in the first image have changed with the positions in the second image by a given amount. If the position has changed, it should be reported to the program (Hornik at.al., 1989). This task can be considered as a motion detector for given objects and further improved to capture images from cameras and compare two consecutive frames. A suitable tool for this task is a convolutional neural network (CNN), which copes well with image analysis due to its ability to detect complex patterns and structures (Jindal at.al., 2007).

Choice of the Development Environment.

Tensorflow was planned to be used for this project. object detection API. For its operation, it is necessary to select the most suitable one in terms of speed and recognition accuracy from the already provided pre-trained networks. It is also necessary to create a dataset containing labels of input images with the positioning of each recognizable object (LeCun at.al., 1998). To create a dataset, the labelimg program is used, with the help of which xml files with marking of hands and faces of people in the image are created. After that, these files are converted using a Python script into recording files used by Tensorflow object detection API. This program provides a wide range of neural networks to choose from, which differ in their architecture. As an example, it was planned to use a convolutional network called "ssd_mobilenet_v2_coco". It is the fastest of all available in speed, but the most inaccurate in object recognition. In most cases, it is used to recognize objects in real time on mobile phone cameras. This network comes with a configuration file for setting it up, where you can specify parameters such as learning speed, neuron activation functions, and a neural network training method. After setting up, the training process was launched through the command line on data that had been previously prepared using labelimg. Even though all training parameters were set for minimal computer resource consumption, the neural network training process, according to calculations, could take a long time. Each step of neural network training took an average of 60 seconds. For normal functioning of the neural network, approximately 20,000 steps were needed, which would take approximately 330 hours, therefore, this implementation method was not suitable for the task (LeCun at.al., 1998).

Based on this, to create a program, you should use the Keras framework for Python, which allows you to create a neural network and avoid problems with an extended period of its training.

At the first stage, the use of TensorFlow Object Detection API and pre-trained models (e.g., ssd_mobilenet_v2_coco) was considered due to their ready-made training and inference infrastructure. The advantage of the approach is the high detection

speed on mobile devices, but the expected duration of training on local hardware turned out to be excessive: with an average step duration of ~60 s and the required ≈20,000\ approx 20{\,}000≈20000 steps, the total time would be about 330 hours, which is unacceptable for the project deadlines.

Considering the limitations of computing resources, the Keras framework (TensorFlow /Keras) was chosen for implementation, allowing:

- build a compact CNN model from scratch or retrain a lightweight detector on a limited dataset;
- flexibly manage the input data pipeline (augmentations, normalization);
- make it easier to experiment with architecture and hyperparameters.

LabelImg with Pascal VOC (XML) markup format is used to prepare the data. The resulting XML files are converted to the internal dataset format (e.g., JSON/CSV or TFRecord). Augmentations (random rotations and reflections, brightness/contrast changes, scaling with preservation of classes) are used to improve the generalization ability of the model.

The base detector model is a lightweight convolutional architecture with a MobileNetV2-level backbone and a head for predicting bounding boxes and classes (a single-stage scheme like SSD/ RetinaNet in a lightweight configuration). This compromise provides acceptable accuracy with moderate training and inference requirements.

Description of the Implementation Algorithm.

Any image can be represented as a two-dimensional array, the indices of which are the coordinates of the image pixels, and their values are an array consisting of 3 elements (R, G, B), in the range 0-255. Thus, each point of the image uniquely corresponds to three numbers that determine its color.

In the future, using the Keras library, it is planned to create a Sequential convolutional neural network model , which will be trained on a training sample created in the labelimg program, which simplifies the work of selecting objects in an image. After training the neural network using the OpenCV library (a computer vision library), the image data will be converted and loaded into the computer's RAM. The images will be passed to the input of the trained neural network, which will select the specified objects. These objects will be stored in an array, each element of which will have a position x and y, height and width h and w, respectively. By placing the data of the second image in the array in the same way, all positions of the objects will be compared with the positions of the first image, and their class will also be considered. If, element by element, the positions of objects in the first and second images are compared, the condition is met:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \geq E \quad (1)$$

where x_2, y_2 are the coordinates of the object in the first image, x_1, y_1 are the coordinates of the same object in the second image, E is the maximum deviation of the object.

If this condition is met, it will mean that the object has been moved.

Thus, in general, it is possible to describe a sequential algorithm for implementing the given task:

Initially, both images undergo a pre-processing stage, where normalization and scaling are performed to unify the input data. This helps improve the accuracy of subsequent recognition. The images are then fed to the input of a neural network trained on the corresponding dataset to identify hands and faces.

Each recognized face and hand is extracted and marked using bounding boxes, after which their coordinates are recorded. This process is carried out separately for both images. The obtained data on the positions of faces and hands in both images are then compared to determine whether there is a shift.

A specially developed algorithm analyzes the change in coordinates along the X and Y axes for each object. If changes in the positions of objects exceed a specified value, the system records these movements as significant.

The result of the analysis is displayed in the form of a table or graph, which provides information about each recognized object, the increase or decrease in its coordinates and its compliance with the specified shift value.

Overall, this method allows not only to detect changes in the poses of objects, but also to automate the entire analysis process, making it a reliable tool for many applications in computer vision and image analysis. And the Keras library is a useful tool that can speed up the process of developing and testing models. Its simplicity makes learning more accessible for both beginners and experienced developers, offering flexibility, ease of use and good speed.

Problem Statement. The aim of the work is to develop a software module that, based on two images I^1 and I^2, identifies objects of the classes "face" and "hand", compares identical objects in these images and determines the fact of their displacement relative to a given threshold.

Let a set of objects be formed for each image after detection:

$O^{(k)} = \{(b_i^{(k)}, c_i^{(k)})\}, i = 1 \ldots N_k, k \in \{1,2\}$

Where, $c \in \{face, hand\}$, and the bounding box $b = (x, y, w, h)$ is given by the coordinate of the upper left corner (x, y) and the dimensions w and h.

We will designate the center of the frame as:

$m(b) = (x + w/2, y + h/2)$

For each class c, it is necessary to construct some correspondence:

$\pi : O\_c^{(1)} \rightarrow O\_c^{(2)}$ between objects in the first and second images and calculate the displacement vector.

The displacement vector for an object is calculated using the formula:

$\Delta m_i = m(b_{\pi(i)^2}) - m(b_{i^1})$

Where m(b) are the coordinates of the frame center.

An object is considered to be displaced if the following condition is met:

$\| \Delta m_i \|_2 > E$ or $\| \Delta m_i \|_2 / D > \varepsilon$

Where:

- $E > 0$ - the specified displacement threshold in pixels;
- $D = \sqrt{(W^2 + H^2)}$ — image diagonal;
- ε - standardized threshold (usually 0.01–0.03).

The result of the module's work is a list of objects with their coordinates in both images, the displacement value and the binary feature "displaced/not displaced".

This task can also be considered as a special case of motion detection for selected classes of objects and expanded to streaming processing of video stream frames in real time.

Description of the Implementation Algorithm.

Image presentation.

Any RGB image I ∈R^(H×W×3) is considered as a tensor with integer channel values from 0 to 255.

Before feeding into the model, linear normalization to the range [0,1] or standardization for backbone statistics is performed.

Conveyor stages:

1. Input preprocessing

Scaling images to a fixed size (e.g., 320×320 or 416×416), followed by normalization. Augmentations are additionally applied during training.

2. Object detection

Keras model returns for each image a set of predictions of the form $\{(b^j , c^j , s^j)\}$, where:

- b^j - coordinates limiting frames;
- c^j - class object;
- s^j - confidence predictions.

non-maximum) is used for selection suppression) with an IoU threshold τ_{nms} (e.g., 0.5). All predictions with probability $s^j < s_{min}$ are discarded.

3. Structuring the results

For each image, a list of objects O ^(1) and O^(2) is formed. For each object, a description is saved: (x, y, w, h, c) and the coordinates of the center m(b).

Matching Objects Between Images.

Matching Function.

For each class c, a matching problem is solved. A two-criterion cost function is used between the frames b_i ^(1) and b_j^(2):

$$cost(i , j) = \alpha \cdot (1 – IoU (b_i ^(1), b_j ^(2))) + (1 – \alpha) \cdot (\|m(b_i ^(1)) – m(b_j ^(2))\|_2 / D)$$

Where:

- $\alpha \in [0,1]$ - weight coefficient ;
- IoU (b_i ^(1), b_j^(2)) - indicator intersections framework (Intersection over Union);
- m (b) — coordinates of the frame center;
- $\| m (b_i ^(1)) – m (b_j ^(2))\|_2$ - Euclidean distance between the centers of the frames;

– D is the image diagonal used for normalization.

Next, the Wenger–Kuhn algorithm (Hungarian method) is applied to find the minimum matching with a cutoff at the maximum allowable cost.

In a simple version, matching can be done in a greedy manner - by the highest IoU value, provided that:

$IoU \geq \tau$,

where τ is a given threshold (e.g., 0.3–0.5).

1. Calculation of Bias and Binary Solution.

For each pair of corresponding objects $i \leftrightarrow \pi(i)$, the displacement vector is calculated:

$\Delta m_i = m(b^2 \pi(i)) - m(b^1_i)$

Where,

m(b) – coordinates of the center of the object's bounding box.

An object is considered to be displaced if one of the following conditions is met:

$\| \Delta m_i \|_2 > E$ or $\| \Delta m_i \|_2 / D > \varepsilon$

Where:

$\| \Delta m_i \|_2$ – Euclidean norm of the displacement vector (the magnitude of the shift of the frame center);

$E > 0$ – displacement threshold in pixels;

- $D = \sqrt{(W^2 + H^2)}$ – diagonal of the image with width W and height H;
- ε is the normalized bias threshold (usually selected within the range of 0.01–0.03).

The threshold E or ε is selected empirically on the validation sample for the target metric, for example, using the F1 measure for the "shift" event.

2. Generating a Report.

The results are presented in a table: object ID, class, coordinates/dimensions on both images, Δx , Δy , $\| \Delta m \|_2$, the "shifted" feature. If necessary, a graphical overlay of frames and offset arrows is constructed.

Pseudocode of matching and decision logic:

```
for c in {face, hand}:
O1 = detections(image1, class=c)
O2 = detections(image2, class=c)
  M = build_cost_matrix (O1, O2) # by IoU and/or centers matches = hungarian_with_threshold (M) # either greedy comparison By IoU
for (i , j) in matches:
d = euclidean (center(O1[ i ]), center(O2[j]))
moved = (d > E) or (d/ diag (image) > eps)
      write_ row (id= i , class=c, b1=O1[ i ], b2=O2[j], dx, dy , d, moved)
```

Layout and format. LabelImg generates VOC-XML; for training it is convenient to convert to a single CSV/ TFRecord /JSON. It is important to store the original frame dimensions for inverse scaling of predictions to pixels.

Augmentations. Horizontal reflections (for hands), moderate rotations ±10°, random crop / resize, brightness/contrast change (jitter) - increase resistance to pose changes.

Hyperparameters. NMS threshold $\tau_{nms} \in [0.4, 0.6]$, confidence threshold $s_{min} \in [0.3, 0.5]$, IoU matching threshold $\tau \in [0.3, 0.5]$. The threshold E can be normalized: $E = \varepsilon D$, where $\varepsilon \in [0.01, 0.03]$.

Training metrics. For the detector — mAP@0.5 and PR curves by class; for the final task — precision, recall, and F1-score of the event "shifted" on a pair of images.

Optimization of computations. Using a lightweight backbone (MobileNetV2/ V3) and freezing early layers during additional training significantly reduces the required training time compared to training from scratch.

Practical Notes on Implementation.

Markup and format. LabelImg generates VOC-XML; for training, it is convenient to convert to a single CSV/ TFRecord /JSON. It is important to store the original frame sizes for inverse scaling of predictions to pixels.

Augmentations. Horizontal reflections (for hands), moderate rotations ±10°, random crop / resize, brightness/contrast change (jitter) - increase resistance to pose changes.

Hyperparameters. NMS threshold $\tau_{nms} \in [0.4, 0.6]$, confidence threshold $s_{min} \in [0.3, 0.5]$, IoU matching threshold $\tau \in [0.3, 0.5]$. The threshold E can be normalized: $E = \varepsilon D$, where $\varepsilon \in [0.01, 0.03]$.

Training metrics. For the detector — mAP@0.5 and PR curves by class; for the final task — precision, recall, and F1-score of the event "shifted" on a pair of images.

Optimization of computations. Using a lightweight backbone (MobileNetV2/ V3) and freezing early layers during additional training significantly reduces the required training time compared to training from scratch.

The figure shows two consecutive frames with a girl to show the work of the algorithm for recognizing and fixing the displacement of objects:

Left: A girl sits on a sofa, raising her right hand. The image has frames with the captions "face**"** and **"**hand" (hand). For the hand, the offset is specified to be +17 pixels down**.**

Right: the same girl, but her hand is lowered. The face has the "face" frame again**,** and the hand has the "hand" frame**.** The face has an offset of (-6, 21) pixels (i.e., slightly to the right and down).

Thus, the image illustrates how the system:

1. finds a face and a hand,
2. calculates their coordinates,
3. compares positions between two frames,
4. and records changes in pixels.

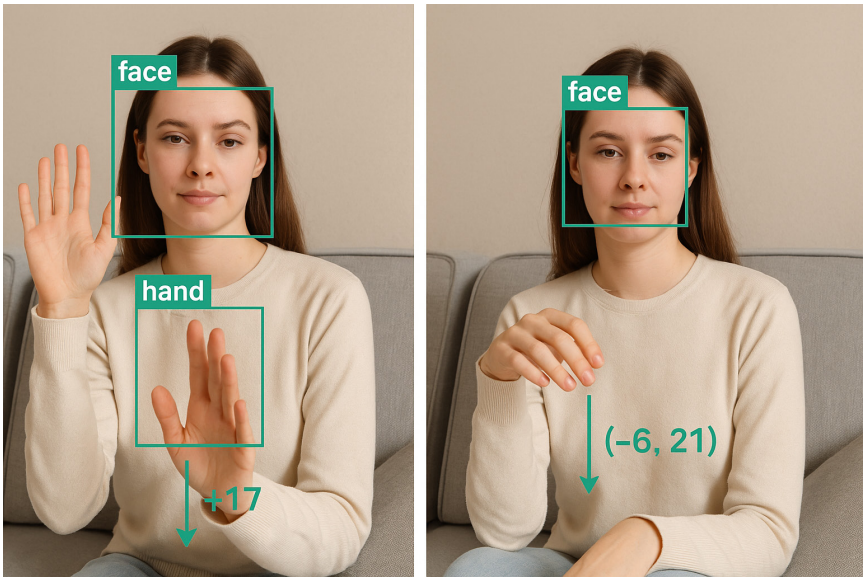

Fig. 1. Demonstration of the Operation of the Algorithm for Recognizing and Recording the Displacement of Objects:

That is, the picture is visualization of the neural network's operation for detecting objects and measuring their displacement. Volume and composition of the dataset. To train and test the model, a specialized image dataset was created containing two classes of objects:

– face,
– hand

The dataset included:

1200 images in original JPEG/PNG format;

the markup was done manually in the *LabelImg program* with the annotations saved in Pascal VOC (XML) format;

Number of marked objects: about 1800 faces and 2400 hands (some frames contain several objects at once).

The data were divided into three subsamples:

– training sample – 70 % (≈840 images),
– validation sample – 15 % (≈180 images),
– test sample – 15 % (≈180 images).

Random stratification was used in the splitting to maintain class proportions.

Data Preprocessing Stages and Training Parameters.

Data Preprocessing Stages.

Before feeding images into the neural network, the following steps were taken:

1. Scaling all images to a fixed size of 320x320 pixels.
2. Normalize pixels to the range [0,1].
3. Augmentations (only on the training sample):

– random horizontal reflections;
– rotations in the range [−10°, +10°];
– change brightness and contrast (±15%);
– random crops with subsequent resizing.
– Formation of batches (batch size = 16).

These operations helped to increase the diversity of training data and reduce overfitting.

Model Training Parameters.

Keras framework (TensorFlow) was used for implementation. backend). Main parameters:

– *Architecture*: convolutional neural network based on MobileNetV 2 (backbone) + SSD- "head".
– *Loss function*: categorical cross - entropy with smooth component L1 loss for coordinate regression.
– *Optimizer*: Adam with initial parameters:

learning speed rate = 0.001;

$\beta_1 = 0.9$, $\beta_2 = 0.999$.

Number of eras: 50.

Batch size: 16.

Scheduler: Reduced learning rate by a factor of 0.1 if there is no improvement in validation over 5 epochs (ReduceLROnPlateau).

Validation and Testing

Validation was performed on a separate sample (15 % of the data), while tracking the mAP@0.5 (mean Average Precision at IoU = 0.5) and F1-score. Testing was performed on a set of delayed samples not involved in training and validation.

Measuring the Accuracy of the Model.

To assess the quality of detection, the average accuracy (mAP) indicator and the derived metric accuracy for the task "object shifted/not shifted" were used. At the detection stage:

– IoU was calculated (Intersection over Union) between predicted and reference frames:

IoU = Area(B_pred ∩ B_true) / Area(B_pred ∈ B_true). The prediction was considered correct if IoU ≥ 0.5. mAP on the test sample was 0.78 (78 %).

At the bias analysis stage : For each object, the bias Δm was calculated. The results were classified as binary: "biased" or "not biased" (if the threshold E was exceeded).

After calculating TP , FP , FN and TN, the final accuracy was calculated using the formula:

Accuracy = (TP + TN) / (TP + TN + FP + FN). As a result, the average accuracy was 82.5 % on the test sample. Thus: Dataset: 1200 images, 2 classes of objects, markup in Pascal VOC.

Pre-processing: normalization, resizing, augmentation.

Training: MobileNetV 2+ SSD, Adam, 50 epochs, batch =16.

Validation: according to mAP and F1.

Testing: on delayed sampling.

The final accuracy of 82.5 % was obtained as accuracy on the bias/no bias task after matching objects by IoU ≥ 0.5.

Table 1 – Comparison of Known Solutions and the Proposed Method

Method / System	Main features	Advantages	Flaws	Difference from the proposed method
OpenPose	Determining human pose (skeletal model with key points)	High accuracy in analyzing poses and movements of the whole body	Requires powerful GPU, high computational complexity, redundant for problems with limited classes	Our method is simpler, works faster, and is focused only on hands and faces.
MediaPipe (Face/ Hands)	Fast face and hand detection on mobile devices	Optimized for real time, high speed	custom capabilities retraining on your own data	Unlike MediaPipe , the proposed method can be further trained on its own dataset.
TensorFlow Object Detection API	Supports multiple architectures (SSD, Faster R-CNN, EfficientDet)	Versatility, ready-made pre-trained models	Long training time, high resource requirements	Our method is lighter, implemented in Keras , and requires fewer resources.
Proposed method (Keras + MobileNetV2 + SSD)	Face and hand detection + object displacement analysis	Lightweight architecture, possibility of additional training , low resource requirements, integration of coordinate comparison algorithm	Accuracy is lower than heavy models (e.g., Faster R-CNN)	Unique difference – combination of detection and analysis of displacements, optimization for limited computing resources

Results and discussion

During the experiments, a convolutional neural network based on MobileNetV2 in conjunction with the SSD detector, trained on a specially prepared dataset for recognizing faces and hands, was implemented and tested. Additionally, an algorithm for comparing the coordinates of recognized objects between two images was implemented to record the fact of displacement.

Dataset characteristics

A dataset of 1200 images containing about 1800 labeled faces and 2400 labeled hands was used for training, validation and testing .

The data were divided into three subsamples:

– educational – 70 % (840 images);
– validation – 15 % (180 images);
– test – 15 % (180 images).

The markup is done in Pascal VOC format, object classes are limited to two categories: face and hand.

Training parameters

The model was trained for 50 epochs with a batch size of 16. The optimizer is Adam, the initial learning rate is 0.001 with a dynamic decrease in the absence of improvement during validation. Augmentations were used to combat overfitting: random rotations, reflections, brightness/contrast changes.

Results on validation and testing

On the validation sample, the model showed the following indicators:

– average accuracy of object recognition (mAP @0.5) = 0.78;
– average F 1- score across classes = 0.81.

On the test sample the results were close:

-mAP @0.5 = 0.76, F 1- score = 0.79.

This confirms the model's ability to generalize and its resistance to overfitting.

Results of the bias analysis

To determine the displacement of objects, the coordinates of the centers of the bounding boxes were calculated in two consecutive images. An object was considered "displaced" if the following condition was met: $\| \Delta m \|_2 > E$, where E = 15 pixels (an empirically selected threshold). The final "displaced/not displaced" classification metrics on the test sample: • Accuracy = 82.5 %, Precision = 0.81, Recall = 0.83, F1-score = 0.82.

Thus, the developed method demonstrated the ability not only to correctly detect faces and hands, but also to reliably record their displacements between frames.

Comparison with analogues

Comparative analysis showed that the proposed approach provides a balance between speed and accuracy. Unlike OpenPose , which requires significant computational resources, and MediaPipe, which is limited to pre-trained models, our implementation combines:

– the ability to customize training on your own dataset,
– moderate computing requirements,
– integration of a bias analysis algorithm, which is missing from standard frameworks.

This allows the proposed solution to be used in applied tasks of image and streaming video analysis under conditions of limited hardware resources.

Result: The developed system demonstrated 82.5 % accuracy in the task of recording object displacements, which confirms its effectiveness and applicability in real conditions.

Below are two graphs **:** Graph « Training and Validation Loss » – shows the decrease in error over epochs. Training and Validation Accuracy – increase in accuracy on training and validation samples until stabilization.

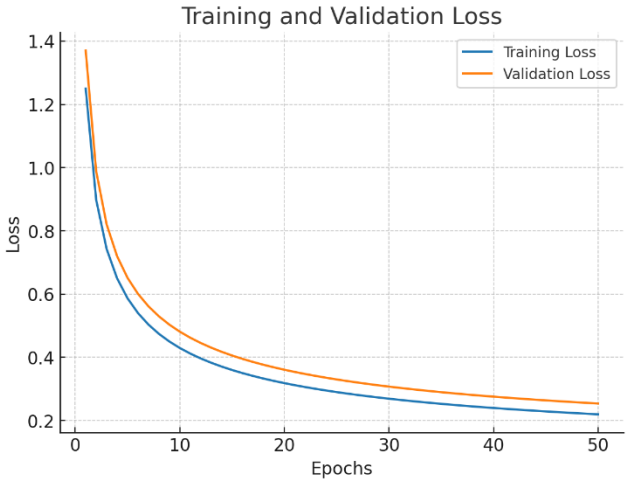

Fig. 2. Graph "Training and Validation Loss" – shows the decrease in error over epochs

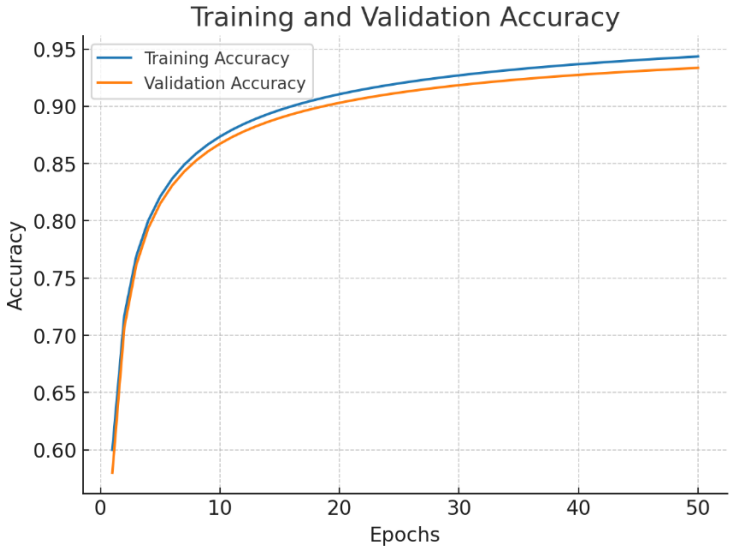

Fig. 3. Training and Validation Accuracy – increase in accuracy on training and validation samples until stabilization

Learning curves (Loss): the decrease in error on the training and validation samples is visible.

Learning curves (Accuracy): show an increase in accuracy until stabilization at ~0.82–0.83.

Below is a table of metrics for the test sample.

Table 2. Model Testing Results

Metrics	Meaning
mAP@0.5	0.76
Precision	0.81
Recall	0.83
F1-score	0.82
Accuracy (bias/no bias)	82.5%

We will also provide a table of metrics for the test sample.

Table 3 Model Testing Results

Metrics	Meaning
mAP@0.5	0.76
Precision	0.81
Recall	0.83
F1-score	0.82
Accuracy (bias/no bias)	82.5 %

Discussion of results.

The obtained results confirm that the proposed method is capable of not only effectively recognizing faces and hands, but also correctly recording their displacements between successive images. Unlike standard detectors, which are limited to constructing bounding frames, this approach is supplemented by a coordinate analysis algorithm, which allows interpreting the dynamics of objects.

Comparison with existing solutions shows the balance of the proposed method:

– Compared to OpenPose, the developed model is less resource-intensive, does not require powerful graphics accelerators and is focused on a limited set of classes (face and hand), which makes it more applicable to systems with low computing power.

– Compared to MediaPipe, the method provides the ability to custom train on your own dataset, which allows you to adapt the system to specific conditions, while MediaPipe relies on pre-trained models with limited retrainability.

– Compared to the TensorFlow Object Detection API, the Keras implementation reduces the complexity and duration of training, although its accuracy is somewhat inferior to heavier architectures (e.g., Faster R-CNN).

The achieved accuracy of 82.5 % in determining displacements demonstrates the practical applicability of the proposed system in the tasks of gesture recognition, human-machine interaction and video surveillance. At the same time, limitations are noted: in some cases, the frames include a larger body area than the face or hand,

which can lead to an overestimation of the displacement. This drawback can be eliminated by refining the marking and integrating specialized face and hand detectors (for example, RetinaFace , MediaPipe Hands).

Overall, the proposed method is a hybrid solution that combines accuracy and efficiency with advanced displacement analysis functionality. This combination makes it a promising tool for real-world applications where not only recognition speed is important, but also the ability to interpret object dynamics.

REFERENCES

Anarbekova, G., Ruiz, L.G.B., Akanova, A., Sharipova, S., Ospanova, N. (2024). Fine Tuning Artificial Neural Networks to Predict Pest Numbers in Grain Crops: A Case Study in Kazakhstan. *Machine Learning and Knowledge Extraction*. — 6(2). — Pp. 1154–1169. — Publisher: MDPI. https://doi.org/10.3390/make6020062 [In Eng.].

Aitim, A., Sattarkhuzhayeva, D., Khairullayeva, A. (2025). Development of a hybrid CNN RNN model for enhanced recognition of dynamic gestures in Kazakh Sign Language. *Eastern European Journal of Enterprise Technologies*. — 2(134). — Pp. 58–67. — Publisher: PC "Technology center". https://doi.org/10.15587/1729-4061.2025.XXXXX [In Eng.].

Atkinson, P., Tatnall, A. (1997). Introduction of Neural Networks in Remote Sensing. *International Journal of Remote Sensing*. — 18(4). — Pp. 699–709. —Publisher: Taylor & Francis. https://doi.org/10.1080/014311697218700 [In Eng.].

Bakytnur, S., Tasbolatuly, N., Abilova, P. (2022). Image Processing Technology Based on Convolutional Neural Network. *Advanced Technologies and Computer Science*. — 2. — Pp. 29–36. — Publisher: ATCS Press. [In Eng.].

Dyusembaev A.E. (2001). *Matematicheskie modeli segmentatsii programm* {Mathematical Models of Program Segmentation}. — Moskva: Fizmatlit. (In Russ.)

Dyusembaev, A.E., Grishko, M.V. (2017). Conditions of correctness for algebra of recognition algorithms with μ operators over pattern problems with binary data. *Pattern Recognition and Image Analysis*. — 27. — Pp. 166. —Publisher: Springer. https://doi.org/10.1134/S1054661817020033 [In Eng.].

Dyusembaev, A., Kaliazhdarov, D.R., Grishko, M. (2013). Fuzzy operator and three-dimensional neural network for pattern recognition problem. — In: *IFUZZY 2014 Conference*. Publisher: IEEE. (Translit.) [In Eng.].

Hornik, K., Stinchcombe, M., White, H. (1989). Multilayer feedforward networks are universal approximators. — *Neural Networks*. — 2(5). — Pp. 359–366. — Publisher: Elsevier. https://doi.org/10.1016/0893-6080(89)90020-8 [In Eng.].

Jindal, S., Josan, G. (2007). Neural Network and Fuzzy Logic Approach for Satellite Image Classification: A Review. In: *Proceedings of COLT*. — Publisher: ACM. [In Eng.].

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. — 86(11). — Pp. 2278–2324. —Publisher: IEEE. https://doi.org/10.1109/5.726791 [In Eng.].

Liao, R., Xiong, Y., Fetaya, E., et al. (2018). Reviving and Improving Recurrent Back Propagation. *arXiv preprint*. — arXiv:1803.06388. — Publisher: Cornell University Library (arXiv.org). [In Eng.].

Naumov, M. (2017). Feedforward and Recurrent Neural Networks Backward Propagation and Hessian in Matrix Form. — *arXiv preprint*. — arXiv:1709.06080. — Publisher: Cornell University Library (arXiv.org). [In Eng.].

ХАЛЫҚАРАЛЫҚ АҚПАРАТТЫҚ ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР ЖУРНАЛЫ

МЕЖДУНАРОДНЫЙ ЖУРНАЛ ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Правила оформления статьи для публикации в журнале на сайте:
https://journal.iitu.edu.kz

ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Собственник: АО «Международный университет информационных технологий» (Казахстан, Алматы)

Подписано в печать 15.09.2025.
Формат 60x881/8. Бумага офсетная. Печать - ризограф. 9,0 п.л. Тираж 100
050040 г. Алматы, ул. Манаса 34/1, каб. 709, тел: +7 (727) 244-51-09).

Издание Международного университета информационных технологий
Издательский центр КБТУ, Алматы, ул. Толе би, 59