TECHNOLOGIES AND ENGINEERING

Journal homepage: https://technologies-engineering.com.ua/en
Vol. 26, No. 1, 2025

Received: 08.10.2024
Revised: 15.01.2025
Accepted: 26.02.2025

UDC 004.021

Doi: 10.30857/2786-5371.2025.1.2

Automatic recognition of UML diagrams in images: Approaches, trends, and challenges

Volodymyr Polischuk[*]
Postgraduate Student
Kyiv National University of Technologies and Design
01011, 2 Mala Shyianovska Str., Kyiv, Ukraine
https://orcid.org/0009-0000-2161-4560

Abstract. The purpose of the study was to analyse and generalise modern methods for recognising UML diagrams in images. The main focus was on automated extraction of text and graphic elements to further reproduce models in text formats. The research methodology covered the analysis of scientific publications, which included 23 papers available in open sources. The study focused on exploring existing approaches to recognising UML diagrams in images. Analysis of scientific publications has shown what modern methods of UML diagram recognition allow achieving more than 90% accuracy in recognising UML diagrams in images. The advantages, limitations, and effectiveness of classical algorithms for computer vision, machine learning, and deep neural networks were investigated. It was found that the best results in classification were provided by deep neural networks, while classical algorithms remain effective for interpreting and extracting elements of UML diagrams. It was found that the main areas in the field of UML diagram recognition are classification of UML diagram types, and interpretation and conversion of UML images to text formats. The main problems were identified: poor image quality, limited training data, and format variability. Possible areas of further research are presented, such as creating large annotated sets of UML diagrams to improve accuracy, and summarising modern approaches to support recognition of more chart types. The findings will contribute to improving the automation processes for working with UML diagrams, and provide an understanding of the current state of the information technology and software development industry, opening up new prospects for development

Keywords: image recognition; computer vision; machine learning; deep learning; automation

Introduction

UML (Unified Modelling Language) diagrams provide a standardised, structured representation of system architecture, which significantly improves communication between development participants, in particular, programmers, system architects, and analysts. However, with the growing number and complexity of software projects, the need for automating the processing of UML diagrams became more acute, especially in terms of recognising information directly from graphic images, which further allows integrating the obtained data into development processes, search information systems, etc.

In the modern world of information technology and software development, UML diagrams play an important role in the process of system design, visualisation, and documentation of software systems. According to a survey in this area, UML diagrams are the most common standard diagrams for design modelling and reach 67% (Chen *et al.*, 2022).

Research shows that most UML diagrams are stored and distributed as images. In particular, based on the results of a large-scale study by R. Hebig *et al.* (2016), conducted on the basis of models stored in various formats on GitHub, 93,596 UML models from 24,717 different repositories were automatically processed and partially manually analysed. Of this number, 57,822 models (61.8%) were presented in image format, while the rest were presented in .xml or .uml formats.

More recent studies by F. Chen *et al.* (2022) showed that 73.72% of UML diagrams collected from projects on GitHub were stored as bitmaps (PNG, JPG, BMP, GIF, etc.), while the remaining 26.28% were stored in text format. Therefore, since UML models are often stored as images embedded in documents, the original versions of the model's text format

[*]Corresponding author

are easily lost. This makes it difficult to use and evolve such models, as it becomes impossible to quickly edit, integrate, and update them in new environments. The researchers proposed ReSECDI, a method designed to automatically recognise semantic elements (such as classes, relationships) in UML class diagram images. Their approach uses classical image processing technologies, including clustering rectangles and combining lines. Problems such as different chart resolutions and styles are solved, depending on the tools in which these charts were created. Despite achieving an accuracy rate of about 90%, the study focuses mainly on UML class diagrams and does not cover other types of UML diagrams or broader methodological issues.

S. Shcherban *et al*. (2021a) developed a neural network-based approach for classifying four types of UML diagrams, including class, activity, sequence, and use case diagrams. Their study used convolutional neural networks (CNNs) and achieved a high accuracy of more than 90% in classifying various types of diagrams. However, it was limited to the task of classification and did not consider the extraction of structural or semantic elements. A. Conrardy & J. Cabot (2024) investigated the use of large language models to transform UML class diagrams into formal machine-readable representations such as PlantUML. Although their approach was innovative, it focused only on converting handwritten diagrams and highlighted the limitations of the LLM (Large Language Model), such as the dependence on quality hints.

M. Axt (2023) introduced SketchToPlantUML, a tool for converting sketchy UML class diagrams to formal PlantUML models using OpenCV. The tool focuses on preprocessing and segmenting static images, but has difficulties with more complex relationships such as associations and dependencies. V. Moreno *et al*. (2020) developed a machine learning tool for classifying static UML diagrams from web images. Their approach achieves 95% accuracy by using rule induction, but does not consider textual content, which limits its capabilities for semantic analysis.

A. Koenig *et al*. (2023) developed NEURAL-UML, a training framework for identifying and classifying semantic elements in UML class diagrams. The study presented a new annotated data set for training and evaluated the model on complex diagrams, achieving accuracy rates of more than 90%. Statistics highlighted the importance of research in this area and pointed out the need for efficient methods that can automatically process, classify, and extract content from UML diagrams stored in various graphic formats. M. Baraban *et al*. (2021) investigated the features of using intelligent technologies for the problem of image recognition. This is especially relevant in the context of rapid growth in data volumes and the number of UML diagrams that require fast and accurate analysis.

These studies focus on specific aspects of UML diagram recognition, such as classification of chart types, identification of semantic elements, or transformation of thumbnails into formal models. However, as of the beginning of 2025, there was no comprehensive review of existing approaches or comprehensive analysis of key challenges of UML diagram recognition, so the purpose of the study was to systematise available sources and implement a generalised analysis of UML diagram recognition methods to provide a holistic vision of problems, challenges, and trends.

The objectives of this study included:

- To analyse classical image processing methods used to recognise UML diagrams.
- To overview current approaches using machine learning and deep learning for UML diagram recognition.
- To identify current trends and major challenges faced by researchers and practitioners in the industry.

Materials and Methods

The main method of research was the analysis of 23 scientific publications that were publicly available. It included searching for relevant sources in scientific databases such as IEEE Xplore, Springer, Scopus, Web of Science, and open archives such as arXiv. For the selection of materials, criteria such as relevance of the topic (recognition of UML diagrams, in particular, classification, extraction of semantic elements, image processing), and the availability of experimental data or descriptions of the methods were used. Table 1 contains a list of papers that were considered.

***Table 1**. Research papers that have been considered and their subject areas*

Research paper	Subject area
E. Lank *et al.* (2000)	Extraction of elements and recognition of UML semantics in images
B. Karasneh & M.R.V Chaudron (2013)	
T. De-Wyse *et al*. (2018)	
F. Chen *et al*. (2022)	
M. Axt (2023)	
A. Koenig *et al*. (2023)	
A. Conrardy & J. Cabot (2024)	
T. Hammond & R. Davis (2006)	

Table 1. Continued

Research paper	Subject area
V. Moreno *et al.* (2020)	Classification of UML diagrams
B. Gosala *et al*. (2021)	
S. Shcherban *et al*. (2021a)	
S. Shcherban *et al*. (2021b)	
L. Wang *et al*. (2022)	
J. Hjaltason & I. Samúelsson (2014)	
T. Ho-Quang *et al*. (2014)	
M.H. Osman *et al*. (2018)	
S. Rashid (2019)	
G. Bergström *et al*. (2022)	Other
J. Ott *et al*. (2019)	
S.W. Munialo *et al*. (2020)	
R. Hebig *et al*. (2016)	
M. Baraban *et al*. (2021)	
A. Jha (2019)	

Source: *compiled by the author*

The study stages included several consecutive steps. First, relevant papers were searched and collected in the above-mentioned databases. Further, the collected papers were grouped by area, which allowed structuring the analysis. The next step was to extract key information from each paper, such as the tasks set by the authors (classification of UML diagrams, recognition of semantics, etc.), and the approaches to solving these problems themselves, including data on the accuracy of methods, their speed, and the amount of data used. The collected information was organised into tables and charts for further evaluation. The final stage was the generalisation of the obtained data to systematise existing approaches and formulate conclusions at each stage.

The classification method was used to systematise the collected data. The publications were grouped into categories: problems solved by researchers, solution methods (in particular, native algorithms, classical machine learning methods, deep learning), and types of UML diagrams that were studied. Challenges for recognising UML diagrams were detected during the analysis of papers. Each paper was analysed to identify key approaches, algorithms used, and their effectiveness. In addition, the statistical analysis method was applied. Quantitative data on the accuracy of classification of UML diagrams, the amount of data used for training models, and the percentage of correctly recognised semantic elements were collected. The collected information was structured in the form of tables and diagrams for further analysis.

Key indicators for evaluating methods were recognition accuracy, quantity, and quality of data for training and testing, algorithm execution time (if specified by the authors), and universality, i.e., the ability to apply methods to different types of UML diagrams, rather than just a specific type. The methods were evaluated in three separate groups: classical machine learning algorithms, deep learning methods, and proprietary algorithms. In addition, aspects such as noise resistance, variable chart styles, and adaptability to different working conditions were considered. The chosen methodology ensured reproducibility of the study, since all stages of data collection and analysis are described, and all sources used have links for verification.

Results and Discussion

As part of the study, modern approaches to UML diagram recognition were classified, including classical computer vision algorithms, machine learning methods, and neural networks. Special attention was paid to analysing the effectiveness of methods, their limitations, and prospects. The main trends in the use of deep neural networks and large language models for automating UML diagram processing were presented.

Classification of UML diagrams by subject areas

In the field of UML diagram recognition in images, researchers have focused on solving a number of problems related to automatic processing, classification, and interpretation of graphic information. Given the growing need for tools that can effectively work with UML models stored as images, various research initiatives are focused on creating solutions to automate these processes. The main areas of research were classification of chart types, interpretation and recognition of UML structural elements, and conversion of images to formats suitable for use in design software.

Among the scientific publications available in open sources, 23 papers were identified that dealt with the task of recognising UML diagrams in images. Figure 1 shows the main research areas in the field of UML diagram recognition and processing – classification and conversion of UML images to XML (Extensible Markup Language). The rest of the papers were difficult to group and assign to a separate area, because they are focused on solving and researching specific problems, such as analysing the complexity of the system architecture or finding the model's compliance with standardised rules for plotting diagrams, etc.

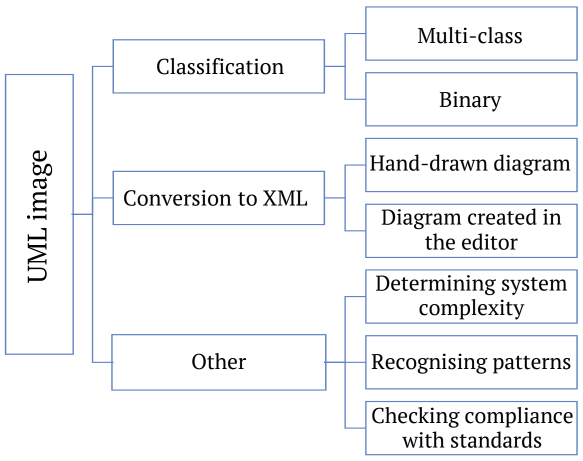

Figure 1*. Areas of research in UML image recognition*
Source: *developed by the author*

Classification of UML diagrams consists of determining whether an image belongs to UML diagrams, and dividing diagrams into types such as classes, sequences, states, etc. This process automates image analysis using characteristics inherent in UML diagrams, including shapes, text elements, semantic attributes, and other properties. In general, the classification of UML images can be divided into two types (Jha *et al.*, 2019), in particular:

- Binary classification – a classification that gives a "yes" or "no" answer to a given question. It is used when it is necessary to determine whether an object belongs to a certain category. For example, in the case of UML diagrams, binary classification can determine whether a given image is a UML diagram.
- Multiclass classification – a classification that allows dividing objects into several different categories. In the context of UML diagrams, multiclass classification can help to automatically recognise which of the many UML types a model image belongs to (for example, a class diagram, sequence diagram, state diagram, etc.).

Various approaches to automatic classification of UML diagrams are presented in the scientific literature. T. Ho-Quang *et al.* (2014) investigated the possibility of automatic classification of UML class diagrams using image analysis and machine learning methods. The paper offers a set of 23 features describing the structure of diagrams, and experiments with various classification algorithms are performed. The researchers focus on identifying the most informative features for recognising UML classes, and on the need to create databases of UML diagrams, which is the basis for academic research. S. Rashid (2019) proposed extending methods for automatic classification of UML diagrams by adding recognition of UML sequences. The main focus is on the development of sequence analysis algorithms based on machine learning methods. This study is a continuation of previous research in the field of automatic recognition of UML diagrams and demonstrates the potential for extending classification to other types of UML models.

V. Moreno *et al.* (2020) investigated the possibility of automatic classification of UML diagrams among web images using machine learning techniques. The main contribution of the study is the creation of a large sample of UML images, which helped to train the model to effectively separate UML diagrams from irrelevant graphic diagrams. The proposed method is aimed at improving the accuracy of search engines and automating the analysis of software repositories.

B. Gosala *et al.* (2021) presented an approach to automatic classification of UML class diagrams using deep learning using convolutional neural networks (CNN). The researchers considered the advantages of CNNs over classical machine learning methods and evaluated the impact of various hyperparameters on the quality of recognition. The proposed method provides high automation of the UML image analysis process, but its effectiveness may depend on the size and quality of the training sample. The researchers also pointed out the limited availability of UML diagram repositories and the importance of creating open and high-quality repositories. J. Ott *et al.* (2019) reviewed the use of low-shot learning for classifying UML diagrams, which allows efficient training of models on small data samples. The researchers showed that even with a limited number of training examples, it is possible to achieve high accuracy in classifying UML classes and sequences. The study demonstrated the possibilities of optimising educational processes in UML image recognition.

S. Shcherban *et al.* (2021b) presented an approach to multiclass classification of UML diagrams, which includes recognition of class diagrams, sequences, activities, use cases, and other types of UML. Deep learning was used, namely, popular neural architectures with transfer learning. The study made a significant contribution to expanding the capabilities of automatic analysis of UML diagrams, which contributes to the creation of scalable systems for processing software models. Since images are one of the most common formats for storing and sharing UML diagrams, determining whether they belong to UML notation and classifying them by type is an important condition for creating specialised repositories. Manually sorting a large number of images is a time-consuming process that requires significant time resources. Therefore, automatic identification of UML diagram types based on images is becoming particularly relevant, which leads to an increase in research attention to image classification methods.

Convert UML images to text formats

A significant group of studies concerns automating the recognition of individual elements and the semantics of diagrams and converting UML images to text formats for further use in software tools. Saving UML diagrams in image format leads to loss of structural information, which makes them difficult to process and integrate into software tools. Since such images are static, they cannot be edited, which creates difficulties in maintaining and developing software projects. Therefore, there is a need for automated methods for recognising UML images and converting them to text formats to ensure that models are saved and editable. The following papers show that research in element recognition and UML diagram semantics focuses on two main areas.

Recognition of hand-created UML sketches – research aimed at transforming informal sketches into formal UML models, which allows their further use in development. This is important to maintain the flexibility of the design approach, reduce the manual work of transferring sketches to CASE (Computer-Aided Software Engineering) tools, and speed up the modelling process.

Image recognition of UML diagrams created in CASE tools – considers methods for automatically extracting structural information from UML images used in documentation. This allows converting UML models to editable formats, reducing information loss, and facilitating their further processing in software systems.

E. Lank *et al*. (2000) proposed an interactive UML sketch recognition system that works with electronic tablets, interactive whiteboards, and the mouse. It uses a multi-level approach to segmentation and recognition of UML graphic characters. First, general recognition of graphic primitives is performed, then UML characters, and at the final stage – their semantic location on the diagram. The researchers also integrated an error correction system that allows users to manually edit segmented elements to improve recognition accuracy. A different approach was used by T. Hammond & R. Davis (2006), who created the Tahuti system. It uses geometric analysis techniques to automatically convert UML class sketches to formal models. The main feature of this approach is its ability to adapt to different drawing styles, reducing the impact of the developer's handwriting on the quality of recognition. The system also considers the relative spatial distances between elements, which improves the accuracy of determining relationships between classes.

M. Axt (2023) proposed a method for automatic recognition of UML thumbnails, based on the use of the OpenCV library for segmenting elements, analysing relationships between them, and converting diagrams to PlantUML format. A distinctive feature of this approach is the emphasis on geometric contour processing and the use of noise filtering algorithms to improve recognition accuracy. The method works well for well-structured sketches, but has limitations when recognising handwritten annotations, non-standard notation, and complex diagrams with a large number of interrelated elements, which requires additional manual adjustment of the results. T. De-Wyse *et al*. (2018) investigated the possibility of automatic recognition of UML sketches to extract useful information from them and then use them in the software modelling process. The main contribution of this paper is to investigate the relationship between initial sketches and final UML models. By analysing sketches, the system identifies the key elements that are most important for developers, and offers their automatic conversion to digital format.

A. Conrardy & J. Cabot (2024) explored the possibility of using large language models (LLMs) to automatically generate UML diagrams from images, with a particular focus on handwritten sketches. The paper conducted experiments with various LLMs, including GPT-4V, Gemini Pro/Ultra, and CogVLM, to evaluate their ability to convert UML-class images to the appropriate PlantUML text format. The study showed that GPT-4V performed better, while other models had difficulties with syntactic correctness of the source code and correct interpretation of class relationships. Automatic recognition of LLM-based UML diagrams shows the potential to speed up the modelling process, but the results remain unstable, which requires mandatory user involvement to correct errors.

B. Karasneh & M.R.V. Chaudron (2013) introduced Img2UML, a system for automatically extracting UML models from images and saving them in XML format, which allows editing the resulting models in CASE tools. The basic idea is that UML models are often stored as images in documentation, which makes them impossible to edit and use in further development. Img2UML uses image processing techniques to detect UML classes, relationships between them, and recognise them. It uses a segmentation algorithm to detect rectangles representing classes, and OCR (Optical Character Recognition) technologies to recognise text labels. The main limitations of this approach are related to difficulties in recognising diagonal lines, dependency notation, and errors in OCR text recognition, especially in cases of poor image quality or non-standard placement of UML elements.

F. Chen *et al*. (2022) presented ReSECDI, a method for automatic recognition of UML class diagrams from images that uses rectangle clustering to identify classes and polyline pooling algorithms to correctly detect relationships. This method is effective for high-quality images, but its accuracy is reduced on low-resolution and non-standard UML diagrams. A. Koenig *et al*. (2023) presented a NEURAL-UML system based on deep learning for automatic recognition of structural elements of UML class diagrams from images. The main focus is on categorising and localising classes, and link types (association, inheritance, aggregation, etc.), which significantly improves the automated processing of UML models. The system uses an extended set of annotated UML diagrams to train and validate the neural network, demonstrating high recognition accuracy. However, the researchers note that the method has certain limitations when working with diagrams containing unusual or atypical graphic designations, and low-quality images, which may affect the accuracy of the analysis.

Specialised research areas

In addition to classifying and interpreting UML diagrams, there are also studies that focus on highly specialised tasks in the field of UML image recognition. These areas include identifying design patterns, which helps to automatically recognise standard architectural patterns, evaluating the layout quality of chart elements to improve their readability, and analysing the complexity of the system, which allows evaluating the scale and structure of software based on UML models.

Identifying design patterns is an important task for recognising standard architectural patterns in UML models. L. Wang *et al*. (2022) proposed a pattern detection method that uses deep learning and advanced graphical information from UML diagrams. The researchers proposed a colour UML model in which information about classes, relationships, and other aspects is encoded using different colours and geometric shapes. This allows turning the problem of recognising design patterns into an image classification problem that can be solved using convolutional neural networks.

Evaluating the layout quality of UML diagrams is another area of research aimed at improving the readability of models. G. Bergström *et al*. (2022) developed an automatic method for evaluating the layout of UML classes using machine learning methods. To build the model, a sample of 600+ UML class diagrams was used, for which experts manually assessed the quality of the layout. The automated system analysed parameters such as the number of line intersections, the length of links between classes, the uniformity of element placement, and other metrics that affect the usability of the diagram.

Another area is the complexity analysis of UML models to assess the scalability and structure of software. S.W. Munialo *et al*. (2020) proposed an automated approach to measuring the size of service-oriented architectures (SOA) based on UML diagrams. Their method uses deep learning to automatically extract the characteristics of UML interfaces and sequential diagrams, including analysing relationships between components, classifying text labels, and determining the level of complexity of the software architecture. The implemented tool uses OCR algorithms for text recognition, classification algorithms for link analysis, and neural networks to assess system complexity.

Basic approaches to recognising UML diagrams in images include proprietary processing algorithms for converting images to XML, classical machine learning techniques, and deep learning techniques.

Custom processing and conversion algorithms

The UML image recognition method using its own processing and transformation algorithms is mainly used to interpret and recognise the semantics of UML class diagrams for their subsequent conversion to structured XML format. This can be traced, for example, in the papers by B. Karasneh & M.R.V. Chaudron (2013), F. Chen *et al*. (2022), etc. However, it is also used at the preprocessing stage to extract the characteristics used by classical machine learning algorithms for classification problems, which is confirmed by T. Ho-Quang *et al*. (2014) or J. Hjaltason & I. Samúelsson (2014).

Proprietary image processing algorithms are usually based on the use of classical computer vision techniques, such as image segmentation, contour detection, and geometric shape recognition. They allow directly detecting UML elements, such as classes, attributes and relationships between them, key shapes and contours (Fig. 2).

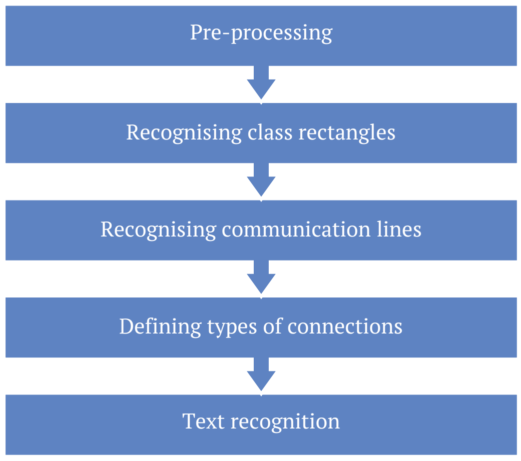
Pre-processing
Recognising class rectangles
Recognising communication lines
Defining types of connections
Text recognition

***Figure 2**. General algorithm of the UML class diagram recognition software*
***Source:** developed by the author*

The implementation details and algorithms used at each individual stage differ in different research papers. However, diagram 2 describes typical steps of the UML class diagram recognition algorithm, which can be considered as a generalised approach. At the first stage, the image is pre-processed to eliminate noise that may interfere with the accuracy of element recognition. The second stage focuses on identifying rectangles that represent classes. Each class can have one or more rectangles to indicate the name, attributes, and methods. To classify rectangles as components of a single class, clustering methods are used, where rectangles are distributed in a hierarchy: the class name is at the top, attributes are in the middle, and methods are at the bottom.

Once classes are detected, all rectangles are removed from the image to avoid interference when recognising link lines. The method of combining polygonal lines recognises different types of link lines: straight and diagonal. This helps to pinpoint the source and target class for each link. Character recognition is used to define link types, such as dependency, implementation, and aggregation. This includes identifying geometric shapes that represent the type of connection (for example, triangles for generalisation or rhombuses for aggregation). At the final stage, text recognition is performed in all class rectangles. This process allows getting class names, attribute names, and method names. After text recognition, the information is combined with class rectangles to form a complete class with its characteristics.

During the analysis of papers that specialised in building their own recognition algorithms, data from the experimental results of these studies were extracted, and Table 2 was compiled. According to each paper, the table contains information about the accuracy of class recognition, linking, and text processing. It is worth noting that the table contains only data from three studies, since other studies did not provide results confirming the effectiveness of their algorithms.

***Table 2.** UML chart element recognition accuracy indicators*

	B. Karasneh & M.R.V. Chaudron (2013)	F. Chen *et al.* (2022)	M. Axt (2023)
Classes	100%	98.62%	92%
Connections	97%	93%	84%
Text and symbols	85%	not specified	not specified

***Source:** developed by the author*

Notably, the study by M. Axt (2023) could have had lower accuracy indicators, since the problem was solved by recognising sketches of diagrams that are created in an informal form (for example, by hand or on a blackboard). This approach had significant problems due to the uncertainty and non-standard styles of elements in such diagrams. Instead, B. Karasneh & M.R.V. Chaudron (2013) and F. Chen *et al*. (2022) recognised diagrams created in specialised editors, which greatly facilitated the process and improved recognition accuracy due to a well-defined element structure.

In addition, B. Karasneh & M.R.V. Chaudron (2013) provided accuracy metrics tested on only 10 sample charts created in a specific editor, which may limit the generalisation of the results. However, it is worth noting that among the methods considered, only the study by B. Karasneh & M.R.V. Chaudron (2013) contained data on the accuracy of text character recognition, which is an important aspect in preserving the semantics of UML diagrams. In contrast, the accuracy specified by F. Chen *et al*. (2022) was tested on 598 sample diagrams found in various open source projects. This provides more grounds for generalising the results obtained and demonstrates the stability of the methodology using real-world examples of UML diagrams.

Classical machine learning methods

Classical machine learning methods are widely used to classify UML diagrams. A common feature of classical machine learning algorithms is that they are based on the use of predefined features to classify UML diagrams. Each of these methods relies on extracted graphical characteristics such as shapes, contours, element sizes, distances between them, and their spatial location, helping to clearly identify UML classes, relationships, and other components. This approach ensures good interpretability of the model, since the classification process can be tracked and explained based on the selected features.

In the research papers mentioned in Table 1 and specialising in classification using classical machine learning methods, a common pattern that the authors use when solving the problem of classifying UML diagrams can be distinguished. Figure 3 shows a general approach to UML classification experiments for class diagrams. The first step of the process is to get an input image of the UML class diagram. At the next stage, image processing is performed to identify contours and shapes, recognise horizontal and vertical lines, and identify rectangles and link lines, which is the basis for UML connections. To avoid delays in processing complex images, images are pre-checked before basic processing. Next, special attributes or metrics are defined (for example, size, area, etc.). These attributes are calculated based on the objects identified in the previous step and presented as numeric values. Selected features are fed to the input of a machine learning algorithm, such as SVM (Support Vector Machine), which was trained based on these features to classify UML class diagrams.

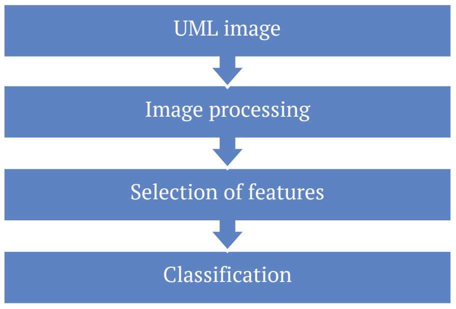

***Figure 3.** Experimental classification process*
***Source:** developed by the author*

The classical machine learning algorithms most commonly used for UML diagram classification problems are SVM, LR (Logistic Regression), RF (Random Forests), DT (Decision Tables), and J84 (a subspecies of Decision Tree). Some studies use RI (Rule Induction) and K-NN (K-Nearest Neighbours). Table 3 shows the machine learning methods whose accuracy data was taken from the relevant studies.

***Table 3.** UML diagram recognition accuracy in various studies*

	J48	LR	DT	RF	SVM	RI	K-NN
T. Ho-Quang *et al.* (2014)	90%	91%	89%	93.1%	89%		
J. Hjaltason & I. Samúelsson (2014)	88.65%	91.1%	89.4%	91.9%	91.5%		
S. Rashid (2019)					90.8%		
V. Moreno *et al.* (2020)						94.4%	
M.H. Osman *et al.* (2018)	88.6%	88.9%	88.3%	90.7%	87.28%		89.1%
L. Wang *et al.* (2022)					91.9%		

***Source:** developed by the author*

Comparative analysis of the classification accuracy of UML diagrams between different studies was not performed due to differences in experimental conditions, such as the set of classification features, image quality, chart type, and volume of training data. However, the Random Forests algorithm stands out in three papers with accuracy results of 93.1%, 91.9%, and 90.74%, which indicates its high efficiency in classifying UML diagrams. The highest results of classification accuracy among all the considered papers were achieved by V. Moreno *et al*. (2020), who showed that their Rule Induction method, with the correct set of rules, achieves an accuracy of 94.4%. This demonstrates the advantage of the Rule Induction method in the context of classifying UML diagrams, although its use may be specific to the conditions of this experiment. In general, all the considered algorithms provide a fairly high accuracy, reaching about 90%, so under the appropriate conditions they can be successfully used for classification problems of UML diagrams.

Deep learning methods and neural networks

With the development of new neural network architectures, such as CNN (Convolutional Neural Network), RNN (Recurrent Neural Network), Transformers, etc., there has been a noticeable interest among researchers in applying these methods to UML diagram recognition problems. Their advantage over classical machine learning algorithms is the ability to automatically extract relevant features from UML diagrams and adapt to different formats and styles.

J. Ott *et al*. (2019) investigated the possibility of using low-shot learning to classify UML diagrams using CNN and the VGG (Visual Geometry Group) architecture as a basic comparison. However, the VGGNet method showed low accuracy in this paper (about 50%), which did not exceed random guessing due to the excessive number of parameters and the complexity of training. Instead, using a smaller CNN with four convolutional layers allowed achieving 70% accuracy already with 100 training examples and almost 90% with several hundred. This highlights the effectiveness of low-shot learning in cases of limited data for training. B. Gosala *et al*. (2021) applied CNN to automatically classify UML-class diagrams, investigating the effectiveness of CNN using regularisation techniques to improve model accuracy.

In classification problems, the use of deep learning helped to rapidly move from binary classification to multiclass classification, which expanded the ability to recognise various types of UML diagrams and elements in them. S. Shcherban *et al*. (2021a) experimented with MobileNet, DenseNet (Densely Connected Convolutional Network), NasNet (Neural Architecture Search Network), ResNet (Resident Network), Inception, and its own architecture for the multiclass classification problem. In the paper S. Shcherban *et al*. (2021a), the researchers indicated the accuracy of algorithms when solving the problem of multiclass classification of UML diagrams. Among the tested models, DenseNet showed the highest accuracy – 94.44%, followed by MobileNet with 94.22%, while ResNet reached 93.50%. The NasNetMobile architecture showed lower accuracy compared to other models – 90.21%. This suggests that DenseNet and MobileNet are more efficient for this task, probably due to their ability to use parameters efficiently and store important information during deep learning.

In general, the performance indicators of these neural network methods are quite high for UML diagram recognition tasks. On average, they recognise UML in images in more than 90% of cases. However, the researchers demonstrated that native neural network architectures designed specifically for a specific task provide even better results for recognising UML diagrams in images and can reach up to 95%. However, native architectures had significantly fewer parameters (2.4 million) and demonstrated the fastest classification time (0.0135 s/image) (Shcherban *et al*., 2021a). Moreover, the researchers point out that the disadvantage of building own neural network architecture is its complexity. Developing a specialised architecture to perform a specific task, such as UML diagram recognition, requires considerable effort in designing and configuring the model, and a deep understanding of the task features and the data being processed.

By interpreting UML semantics, researchers can recognise individual diagram elements such as classes and types of relationships, which helps to automate software architecture analysis. However, there are currently no papers that solve the problem of completely converting UML images to XML format using deep learning, which indicates a potential area for future research. A. Koenig *et al*. (2023) developed a system for recognising structural elements in UML class diagrams using YOLOv8 (You Only Look Once) for semantic analysis of UML diagram images. YOLO provides high performance for the task of recognising individual UML elements of a class diagram, which averages 94.11%, and the accuracy of recognising class elements reaches 98.69%.

The ability of deep learning methods to generalise and automatically identify features for problem solving allows researchers to go beyond typical problems. Therefore, in addition to the problems of classifying and converting UML images to XML, they also explore such areas as, for example, determining the complexity of the system architecture (Munialo *et al*., 2020). It is difficult to perform a comparative analysis of all architectures together, because they are used in separate studies and often solve different problems. In addition, in each individual paper, the authors have their own set of training data, and separate experimental conditions in which the tests were conducted.

Advantages and limitations of UML recognition methods

After considering each approach separately, it can be concluded that each has its own advantages and limitations (Table 4). Deep neural networks have shown the best results in recognising different diagrams of different types and formats. However, they have significant computational costs and also require a sufficient amount of marked-up data to train the model. Classical machine learning methods may be the best choice in situations where there are limited resources for training neural networks, but higher accuracy and interpretability of the model is required, since the layout works according to a clearly defined set of characteristics. Proprietary algorithms remain relevant for fast and easy processing of standard UML diagrams without significant complexity in their structure.

***Table 4**. Advantages and limitations of UML recognition approaches*

Approach	Advantages	Disadvantages
Custom processing algorithms	High processing speed, easy implementation	Low adaptability to complex UML diagrams
Classical machine learning	Good balance between performance and adaptability	Requires manual selection of attributes
Deep learning	High accuracy and ability to summarise new data	Requirements for computing resources, the need for a large amount of data

***Source:** developed by the author*

The analysis showed that deep neural networks are slightly superior to classical machine learning methods in the accuracy of recognising UML diagrams in the classification problem, achieving 94-95% accuracy in the case of DenseNet, MobileNet, ResNet, while the best classical algorithms (Random Forest, Rule Induction, SVM) show results in the range of 91-94%. However, classical methods have the advantage of processing speed and explanatory solutions, while deep neural networks provide better generalisation capabilities and recognise more complex UML diagrams with variable representation styles. Thus, the choice of UML classification approach depends on the trade-off between accuracy, processing speed, and available computing resources, but the general trend indicates the advantage of deep learning in automatic UML recognition tasks.

It was found that for the tasks of recognising individual elements and semantics of UML diagrams in images, among the various approaches to automatic recognition of UML diagrams, only NEURAL-UML uses deep neural networks, while other studies are based on conventional image processing methods and computer vision algorithms (Koenig *et al.*, 2023). The resulting accuracy for recognising various elements using NEURAL-UML varies between 92.62-96.09%, while Table 2 shows the recognition rates of native algorithms, the accuracy of which varies in the range of 93-98.62% in the study by F. Chen *et al.* (2022). Although ReSECDI and similar classical methods may show slightly higher accuracy rates for individual elements, NEURAL-UML has a number of important advantages. Firstly, the neural network approach is more resistant to the diversity of UML diagram styles, in particular, to changes in fonts, element positions, and circuit structural features. Secondly, NEURAL-UML works better with UML diagrams that contain low-quality artefacts, such as distortion, blurring, or poor contrast, which is a common problem in UML images stored in bitmap format. Thirdly, the method demonstrates a higher generalisation ability, which allows it to work effectively with UML diagrams created in various software environments without the need for significant refinement. Thus, while conventional algorithms can show high accuracy under certain controlled conditions, it is deep neural networks that provide greater reliability, scalability, and adaptability for automatic UML diagram recognition.

Challenges and trends for further research

Current research on recognising UML diagrams from images faces a number of technical challenges that become obstacles to achieving full automation and accuracy. Despite this, the development of new approaches and methods allows gradually expanding the possibilities of analysing and automating work with UML. Among the analysed papers (Table 1), several problems of UML diagram recognition can be distinguished, which are most often solved by the researchers. Figure 4 shows those that were mentioned more than once.

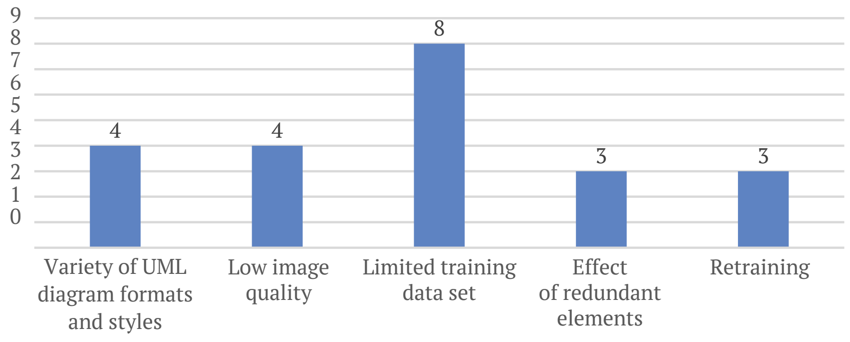

***Figure 4**. The most frequently mentioned problems with UML diagram recognition*

***Source:** compiled by the author*

In different UML software projects, diagrams may differ not only in content, but also in styles, fonts, colours, and element placement. This makes automatic recognition difficult, as the systems used for classification are often trained in standard styles. For example, diagrams can have different types of arrows to indicate dependencies or associations, or they can be created in different tools, which adds variability that is not provided for in training models. Effective recognition requires additional algorithms that can adapt to different styles or standardise the diagrams themselves.

The quality of UML diagram images significantly affects the performance of recognition algorithms. Low resolution, blurry borders, background noise, or other artefacts can cause important details to be lost. In the case of blurry or low-quality images, the model may not correctly recognise text names, roles, or relationships between elements. This causes the need for image preprocessing, such as improving contrast, eliminating noise, or even image reconstruction.

One of the most serious limitations is limited access to large data sets containing various UML diagrams. The small amount of training images limits the ability of models to generalise, which increases the likelihood of errors when working with new or unusual UML diagrams. There is also often a problem of uneven distribution between UML chart types (for example, more class diagrams, less sequential ones), which can affect the accuracy of classification.

Real UML diagrams often contain additional elements that may not be important for recognising a particular type or pattern, such as comments, markers, or other graphic labels. Such elements can "clog" the image, complicating model analysis and reducing classification accuracy. Algorithms often require first removing or ignoring such elements, which, however, requires additional resources.

Overfitting occurs when the model performs well on the training set, but shows low accuracy on new data. This is especially problematic with a limited data set, because the model can "learn" the specifics of specific diagrams, but cannot generalise knowledge to new images. Regularisation techniques are often used to solve the problem, and data augmentation strategies, but this requires additional computational resources.

Modern research on UML diagram recognition shows a clear evolution of methods and approaches covering the period from classical image processing algorithms to deep learning and the use of large language models. Early approaches were based on geometric techniques and optical character recognition (OCR) algorithms that made allowed extracting UML elements from scanned or digital images (Lank *et al.*, 2000). In the following years, the researchers integrated machine learning techniques to classify UML diagrams based on features extracted from graphical objects. Since 2014, there has been an active development of methods for automatic classification of UML diagrams using machine learning technologies, in particular SVM and Random Forest (Hjaltason & Samúelsson, 2014). In 2018, the first papers appeared using deep neural networks to recognise UML diagrams, including convolutional neural networks (CNN) to extract semantic structures (Osman *et al.*, 2018). Recent trends in UML diagram research are related to the integration of large language models (LLM), which allow automatic conversion of UML thumbnails into formal models (Conrardy & Cabot, 2024).

In addition, there is an active introduction of low-shot learning methods, which allows training models on limited data sets, as an attempt to reduce the need for large marked-up datasets (Otts *et al.*, 2019). The low-shot learning method has become an effective approach for teaching UML diagram recognition models based on limited samples of training data. The researchers proved that even with 100 training examples, 70% accuracy can be achieved, and with several hundred – almost 90%. The study showed that a compact CNN with four convolutional layers showed significantly better performance in small samples than the large VGGNet model, which highlights the importance of adapting the architecture to the specifics of the task.

An important area is also the combination of conventional machine learning and deep learning methods, which helps to improve the accuracy of recognising UML elements in complex diagrams (Koenig *et al.*, 2023). The use of CNN allowed researchers to automatically extract key graphical features of UML elements, such as class forms and relationships between them, while SVM was used for final classification and verification of the obtained characteristics, in particular, recognition of text elements (class names, attributes, and methods).

Most classification studies focused on only one type of UML diagram, but noted the need to improve methods for recognising other types. In such studies, classical machine learning algorithms were most often used, in particular, SVM (Rashid, 2019), Random Forest (Osman *et al.*, 2018), logistic regression (Ho-Quang *et al.*, 2014). However, the use of deep neural networks, such as different CNN architectures, allowed achieving a generalised approach that covers the classification of various UML diagrams within a single model by automatically extracting characteristics, which reduced the need for manual feature design. Thus, studies of UML diagram classification are gradually shifting the focus from highly specialised approaches based on classical algorithms to generalised solutions based on deep learning. The development of UML diagram recognition goes through several stages-from basic geometric algorithms to modern hybrid methods that combine deep neural networks, large language models, and adaptive approaches to limited samples. Further research is likely to focus on improving the accuracy, adaptability, and efficiency of UML diagram recognition in real-world software environments.

Conclusions

The study confirmed the relevance and effectiveness of various methods of automatic recognition of UML diagrams, especially for tasks of classifying, extracting elements, and converting UML images to text format. An analysis of

23 publications showed that 53% of studies are devoted to classifying UML diagrams, 33% to extracting elements and converting them to text format, and 14% to other areas.

In UML diagram classification problems, neural networks show the highest accuracy – up to 95%, while classical machine learning methods, such as SVM, reach 94%. Studies show that the Random Forest method is one of the most effective for classifying UML diagrams, reaching 93.1%, 91.9%, and 90.74% in various experiments.

In problems of extracting individual elements of UML diagrams, proprietary computer vision algorithms show accuracy in the range of 93-98.62%, while deep neural networks show accuracy in the range of 92.62-96.09%. Despite the high accuracy of conventional algorithms under controlled conditions, it is deep neural networks that provide greater reliability, scalability, and adaptability when automatically recognising UML diagrams. The main problems in this area remain the poor quality of input images, the limited number of annotated data sets, and the significant variability of UML chart formats. There is also a problem integrating recognised UML diagrams into CASE tools, as many of them do not support editing UML images without first converting them to text format.

Among the promising areas of further research are low-shot learning methods, the use of large language models) for UML diagram recognition, and combined approaches that combine classical machine learning algorithms with deep neural networks. Most often, researchers point out the need to create large-scale open repositories of UML diagrams for training models, improve algorithms for converting sketches into formal UML diagrams, and develop new methods for automatic semantic interpretation of UML models. Additionally, based on the results obtained, appropriate areas for further research are the development of self-learning methods to improve the accuracy of recognising UML diagrams without the need for a large amount of annotated data, ways to reduce the computational complexity of algorithms for real use in resource-dependent environments, and the development of methods for automatic detection of anomalies in UML diagrams to identify potential errors in models.

Acknowledgements

None.

Conflict of Interest

None.

References

[1] Axt, M. (2023). *Transformation of sketchy UML class diagrams into formal PlantUML models*. Retrieved from https://www.diva-portal.org/smash/record.jsf?pid=diva2:1786365&dswid=-4498.

[2] Baraban, M., Baraban, S., & Garmash, V. (2021). Development of an advanced web application with a convolutional neural network for image recognition. *Information Technologies and Computer Engineering*, 18(1), 7-14. doi: 10.31649/1999-9941-2021-50-1-7-14.

[3] Bergström, G., Hujainah, F., Ho-Quang, T., Jolak, R., Rukmono, S.A., Nurwidyantoro, A., & Chaudron, M.R.V. (2022). Evaluating the layout quality of UML class diagrams using machine learning. *The Journal of Systems & Software*, 192, article number 111413. doi: 10.1016/j.jss.2022.111413.

[4] Chen, F., Zhang, L., Lian, X., & Niu, N. (2022). Automatically recognizing the semantic elements from UML class diagram images. *Journal of Systems and Software*, 193, article number 111431. doi 10.1016/j.jss.2022.111431.

[5] Conrardy, A., & Cabot, J. (2024). *From image to UML: First results of image-based UML diagram generation using LLMs*. doi: 10.48550/arXiv.2404.11376.

[6] De-Wyse, T., Renaux, E., & Mennesson, J. (2018). Using sketch recognition for capturing developer's mental models. In *Proceedings of the ACM/IEEE 21st international conference on model driven engineering languages and systems* (pp. 23-28). Copenhagen: IEEE.

[7] Gosala, B., Chowdhuri, S.R., Singh, J., Gupta, M., & Mishra, A. (2021). Automatic classification of UML class diagrams using deep learning technique: Convolutional neural network. *Applied Sciences*, 11(9), article number 4267. doi: 10.3390/app11094267.

[8] Hebig, R., Ho-Quang, T., Robles, G., Fernandez, M.A., & Chaudron, M.R.V. (2016). The quest for open source projects that use UML: Mining GitHub. In *Proceedings of the ACM/IEEE 19th international conference on model driven engineering languages and systems* (pp. 173-183). New York: Association for Computing Machinery. doi: 10.1145/2976767.2976778.

[9] Hjaltason, J., & Samúelsson, I. (2014). *Automatic classification of UML class diagrams through image feature extraction and machine learning*. Sweden: University of Gothenburg and Chalmers University of Technology.

[10] Ho-Quang, T., Chaudron, M.R.V., Karasneh, B., & Osman, M. (2014). Automatic classification of UML class diagrams from images. In *Proceedings of the 21st Asia-Pacific software engineering conference* (pp. 422-429). Jeju: IEEE. doi: 10.1109/APSEC.2014.65.

[11] Jha, A., Dave, M., & Madan, S. (2019). Comparison of binary class and multi-class classifier using different data mining classification techniques. In *Proceedings of international conference on advancements in computing & management (ICACM) 2019*. (pp. 894-903). Rochester: SSRN. doi: 10.2139/ssrn.3464211.

[12] Karasneh, B., & Chaudron, M.R.V. (2013). Extracting UML models from images. In *Proceedings of the 2013 5th international conference on computer science and information technology* (pp. 134-137). Amman: IEEE. doi: 10.1109/CSIT.2013.6588776.

[13] Koenig, A., Allaert, B., & Renaux, E. (2023). NEURAL-UML: Intelligent recognition system of structural elements in UML class diagram. In *Proceedings of the 5th workshop on artificial intelligence and model-driven engineering*. (pp. 605-613). Västerås: IEEE. doi: 10.1109/MODELS-C59198.2023.00099.

[14] Lank, E., Thorley, J.S., & Chen, S.J. (2000). An interactive system for recognizing hand drawn UML diagrams. In *Proceedings of the IBM center for advanced studies conference (CASCON)* (pp. 1-15). DBLP: Mississauga: doi: 10.1145/782034.782041.

[15] Moreno, V., Génova, G., Alejandres, M., & Fraga, A. (2020). Automatic classification of web images as UML static diagrams using machine learning techniques. *Applied Sciences*, 10(7), article number 2406. doi: 0.3390/app10072406.

[16] Munialo, S.W., Muketha, G.M., & Omieno, K.K. (2020). Automated feature extraction from UML images to measure SOA size. *International Journal of Recent Technology and Engineering,* 9(2), 1132-1136. doi: 10.35940/ijrte.B4131.079220.

[17] Osman, M.H., Ho-Quang, T., & Chaudron, M.R.V. (2018). An automated approach for classifying reverse-engineered and forward-engineered UML class diagrams. In *Proceedings of the 44th EUROMICRO conference on software engineering and advanced applications* (pp. 123-130). Prague: IEEE. doi: 10.1109/SEAA.2018.00070.

[18] Ott, J., Atchison, A., & Linstead, E. (2019). Exploring the applicability of low-shot learning in mining software repositories. *Journal of Big Data*, 6, article number 35. doi: 10.1186/s40537-019-0198-z.

[19] Rashid, S. (2019). *Automatic classification of UML sequence diagrams from images*. Sweden: University of Gothenburg and Chalmers University of Technology.

[20] Shcherban, S., Liang, P., Li, Z., & Yang, C. (2021a). Multiclass classification of four types of UML diagrams from images using deep learning. In *Proceedings of the 33rd international conference on software engineering and knowledge engineering*. Pittsburgh: SEKE. doi: 10.18293/SEKE2021-185.

[21] Shcherban, S., Liang, P., Li, Z., & Yang, C. (2021b). Multiclass classification of UML diagrams from images using deep learning. *International Journal of Software Engineering*, 31(11), 1683-1698. doi: 10.1142/S0218194021400179.

[22] Wang, L., Song, T., Song, H.-N., & Zhang, S. (2022). Research on design pattern detection method based on UML model with extended image information and deep learning. *Applied Sciences*, 12(17), article number 8718. doi: 10.3390/app12178718.

[23] Hammond, T., & Davis, R. (2006). Tahuti: A geometrical sketch recognition system for UML class diagrams. *In Proceedings of the 2006 working conference on Advanced visual interfaces* (pp. 372-375). New York: ACM. doi: 10.1145/1185657.1185786.

Автоматизоване розпізнавання UML діаграм на зображеннях: підходи, тенденції та виклики

Володимир Поліщук

Аспірант
Київський національний університет технологій та дизайну
01011, вул. Мала Шияновська, 2, м. Київ, Україна
https://orcid.org/0009-0000-2161-4560

Анотація. Мета дослідження полягала в аналізі та узагальненні сучасних методів розпізнавання UML-діаграм на зображеннях. Основна увага була приділена автоматизованому вилученню текстових і графічних елементів з метою подальшого відтворення моделей у текстових форматах. Методика дослідження охоплювала аналіз наукових публікацій, що включав 23 роботи, доступних у відкритих джерелах. Дослідження зосереджувалося на вивченні існуючих підходів до розпізнавання UML-діаграм на зображеннях. Аналіз наукових публікацій показав, які сучасні методи розпізнавання UML-діаграм дозволяють досягти точності понад 90 % у розпізнаванні UML-діаграм у зображеннях. Досліджено переваги, обмеження та ефективність класичних алгоритмів комп'ютерного зору, машинного навчання та глибоких нейронних мереж. Встановлено, що найкращі результати у класифікації забезпечують глибокі нейронні мережі, тоді як класичні алгоритми залишаються ефективними для інтерпретації та вилучення елементів UML-діаграм. З'ясовано, що основними напрямками у сфери розпізнавання UML-діаграм є класифікація типів UML-діаграм, а також інтерпретація та перетворення UML-зображень у текстові формати. Виявлено основні виклики: низьку якість зображень, обмеженість навчальних даних і варіативність форматів. Наведено можливі напрямки робіт для подальших досліджень, такі як створення великих анотованих наборів UML-діаграм для підвищення точності, узагальнення сучасних підходів для підтримки розпізнавання більшої кількості типів діаграм. Результати роботи сприятимуть вдосконаленню процесів автоматизації роботи з UML-діаграмами, а також забезпечать розуміння сучасного стану галузі інформаційних технологій та розробки програмного забезпечення, відкриваючи нові перспективи для розвитку

Ключові слова: розпізнавання зображень; комп'ютерний зір; машинне навчання; глибоке навчання; автоматизація