ACKNOWLEDGEMENT

I would like to thank my tutor Ing. Petr Čech, Ph.D. for always answering all my questions diligently, for proposing the types of damage that would be used to create the dataset, and for guiding me through the process and technicalities of creating a bachelor thesis. I would also like to thank doc. Ing. Filip Lankaš, Ph.D. for his feedback on the first draft of the theoretical part of this thesis.

SUMMARY

Much chemical information is stored in the form of chemical structure images within chemical publications. Optical Chemical Structure Recognition is a field focused on taking these images and converting them into a machine-readable format that can then be stored and queried in chemical databases. Many tools have been developed to solve this problem throughout the years. This work includes the installation of freely available ones and the evaluation of their performance on a dataset developed specifically for thesis. The dataset consists of a base of structurally variable images, which were damaged in numerous ways and degrees to form subsets. Results are gathered as the performance of each of the installed tools on each of the unique subsets and are then evaluated both individually and against each other.

SOUHRN

Velké množství chemických informací je uloženo ve formě obrázků chemických struktur v chemických publikacích. Optické rozpoznávání chemických struktur je obor zaměřený na převod těchto obrázků do strojově čitelného formátu, který je poté možné ukládat a vyhledávat v chemických databázích. Během let bylo vyvinuto mnoho nástrojů, které se pokouší tento problém řešit. V rámci této práce byli instalovány volně dostupné nástroje a hodnoceny podle jejich výkonu na data setů vyvinutém speciálně pro účely této práce. Data set se skládá z báze obrázků s variabilními strukturami, která byla poškozena různými způsoby a stupni, aby vytvořila podmnožiny. Výsledky jsou shromážděny jako výkonnost každého nainstalovaného nástroje na každé jedinečné podmnožině a jsou následně hodnoceny jak individuálně, tak i ve srovnání s ostatními nástroji.

TABLE OF CONTENTS

1 INTRODUCTION

A vast amount of information is hidden within the chemical publications created before annotation and curation of publications became a widespread practice. Due to the number of publications today and the exponential growth taking place, literature databases are the primary way of storing and accessing chemical information. Chemical information is often represented in the form of chemical structures, which are present in the literature in the form of graphical images – a form that is easily interpretable by humans. Databases however require a more machine-readable format to present relevant results when being queried – these formats are called chemical structure identifiers and are usually not meant to be directly interpreted by humans. This need for conversion from graphical representation to identifiers is the crux of the OCSR problem and many approaches have been proposed and applied in recent years.

One of the projects taking place in the Department of Informatics and Chemistry at UCT Prague is the development of a database of chemical structures that were produced by Czech scientists. One of the key steps in the workflow of this database is the proper annotation of chemical structures included within the publications through OCSR. This bachelor's thesis aims to arrive at conclusions that will help determine the appropriate OCSR tool for this task. This included research of the currently available tools in the field, installation of the tools, creation of a testing dataset of chemical images and their corresponding identifiers, and performance testing. Since a substantial portion of the targeted publications is stored only in the form of scans of physical publications – images of chemical structures often contain graphical damage which reduces the success rate of OCSR. For this reason, the performance test is done on multiple versions of the dataset with varying and progressive artificial graphical damage that attempts to simulate several types of damage present on the scanned images. The goal of the testing is then not only to compare the different OCSR tools but to also establish which kinds of damage have an effect on their performance and to what extent.

2 STATE OF ART

2.1 Chemical structure graphical representations

The goal of including an image of a chemical structure in a publication is to provide information that is readable and comprehensible to a human. Recommendations for how this is done exist, but since a chemical structure of a single molecule can be drawn in an infinite number of ways (see Figure 1) – this representation can never be ubiquitous. [1]

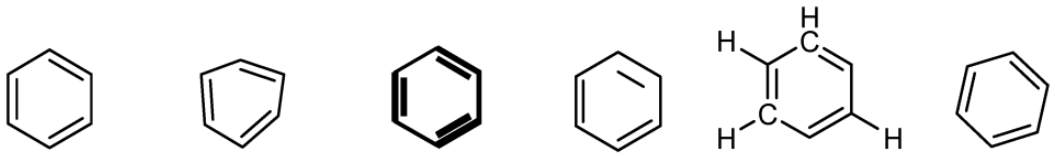


Figure 1: *Six valid representations of benzene* [1]

Most chemical publications are available in the Portable Document Format (PDF) which is either exported from the user's editor of choice or generated from scans of physical prints. Within, chemical structures are represented in the form of either raster or vector images – raster being the more common of the two. [2]

2.2 Chemical structure identifiers

If the purpose of vector and raster representations of chemical structures is to be identifiable by humans – the purpose of chemical structure identifiers is to be identifiable by computer, particularly to be stored and queried in chemical databases. [3]

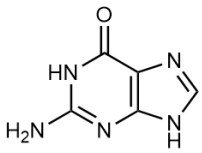
These identifiers should be standardized, easy to store, and convertible between different formats, including raster and vector images mentioned above. The methods of achieving this can be generally divided into line notations and connection tables.

2.2.1 Line notations

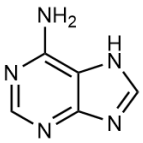
Line notations are a single-string representation of a chemical structure. In the field of cheminformatics, the most common line notation systems are The International Chemical Identifier (InChI) and Simplified Molecular-Input Line-Entry System (SMILES) (and its many derivatives). [4]

2.2.1.1 SMILES

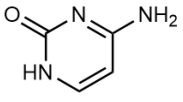
SMILES encodes exclusively a two-dimensional chemical structure by building a connection graph of its molecular structure. It is designed to be both easy to learn and comprehend by humans and to be easily generated by computer algorithms. For example, ethanol is represented as simply as CCO, a mere 3 bytes of data. [5] Its simplicity has led to SMILES becoming the most common method of encoding structure to date; however, its main disadvantage is that one structure can be represented in multiple ways. Ethanol for example can be represented in two separate ways (OCC, CCO), which means that without outside guidance two unrelated algorithms will generate matching SMILES strings 50% of the time, and the amount only gets worse for larger molecules. Derivatives of SMILES have been developed to address this issue and provide a 1:1 structure-representation relationship, such as SMARTS, DeepSMILES, and SELFIES. [2]



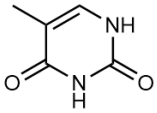
guanine
Nc1nc2[nH]cnc2c(=O)[nH]1
[#7]-[#6]-1=[#7]-[#6]-2=[#6](-[#7]=[#6]-[#7]-2)-[#6](=O)-[#7]-1



adenine
Nc1ncnc2nc[nH]c12
[#7]-[#6]-1=[#6]-2-[#7]-[#6]=[#7]-[#6]-2=[#7]-[#6]=[#7]-1



cytosine
Nc1cc[nH]c(=O)n1
[#7]-[#6]-1=[#7]-[#6](=O)-[#7]-[#6]=[#6]-1

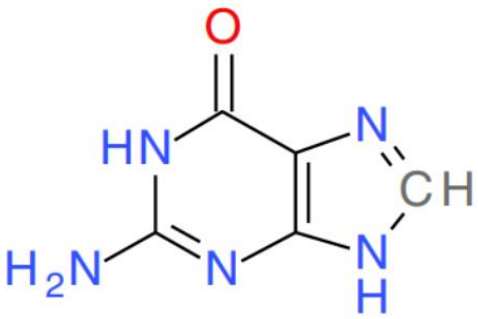


thymin
Cc1c[nH]c(=O)[nH]c1=O
[#6]-[#6]-1=[#6]-[#7]-[#6](=O)-[#7]-[#6]-1=O

Figure 2: *Structures of DNA nucleic bases and their corresponding SMILES and SMARTS*

2.2.1.2 InChI

InChI is a hierarchical layered approach. A chemical structure is encoded as a single line string composed of layers, where each layer is a sequence initiated by a forward slash. This is best shown in an example (see Figure 3).



The InChI for this structure is:
InChI=1/C5H5N5O/c6-5-9-3-2(4(11)10-5)7-1-8-3/h1H,(H4,6,7,8,9,10,11)/f/h8,10H,6H2

Figure 3: *Chemical structure and corresponding InChI encoding of Guanine*.[6]

"InChI" represents the identifier standard used, afterwards the layers in order are formula, connectivity, isotopes, stereochemistry, and tautomers.[6] It is apparent that InChI is meant to be primarily machine-readable and lacks easy human readability when compared to SMILES. Its main advantage is that a particular molecule has exactly one way to be encoded, allowing for easier database searching.[2]

2.2.2 Connection tables

The connection table approach is based on the idea of defining the structure as atoms with 3-dimensional coordinates (x, y, z) and their mutual connectivity. Since the approach is naturally close to a graphical representation – the generation of graphical structures from connection tables is a trivial task. Many implementations of this approach exist, but by far the most widespread are MDL molfiles.

2.2.2.1 MDL molfiles

A molfile can be divided into three distinct sections. First, a header block contains the title, timestamp, and comments. Second, a CTab, which represents the actual structure, and third an end line reading “M END.” Molfiles are once again best understood through an example (see Figure 4)

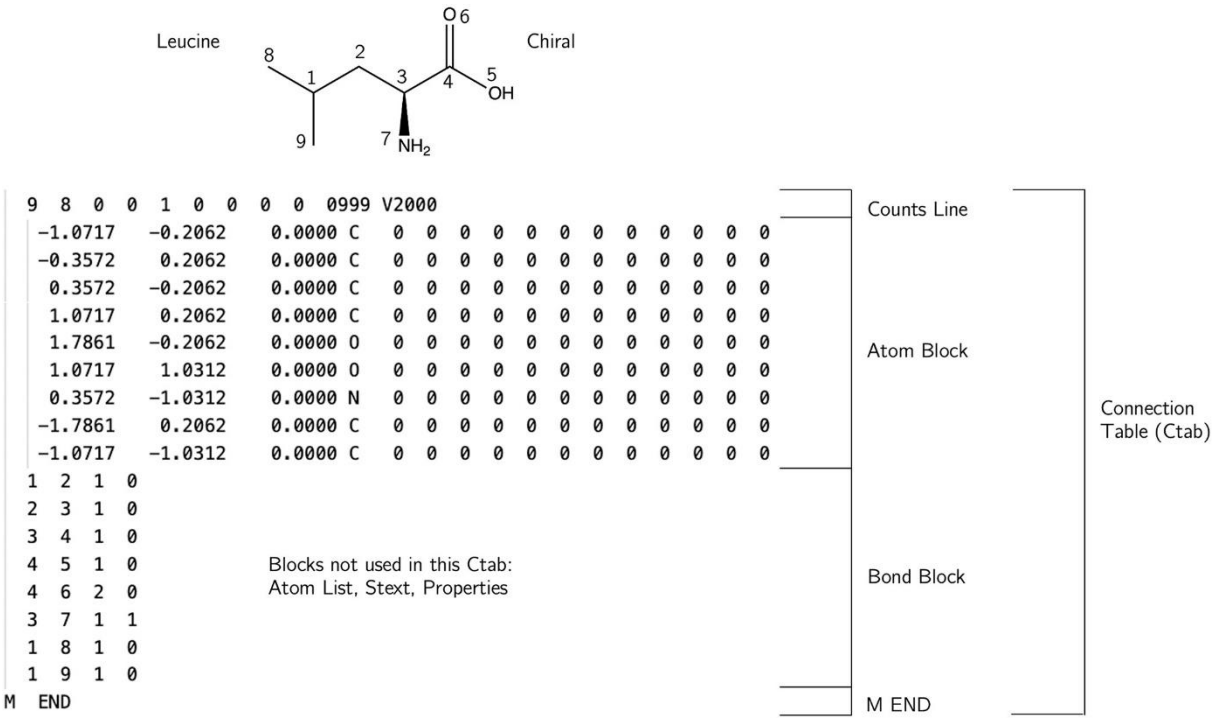


Figure 4: *Example of a connection table and end line within an MDL molfile V2000 for the molecule leucine generated by ChemDraw* [7]

Molfiles also have their drawbacks, the main one being the simplicity of the depiction of bonds as covalent sharing of electrons, supporting only single, double, and triple bonds as a result. Another common issue with molfiles is the handling of implicitly stated hydrogen atoms combined with non-trivial valency leading to a possible misinterpretation of implied hydrogen atoms. [7]

2.3 Goals of OCSR

Optical Structure Chemical Recognition (OCSR) is the middle step between human-readable formats and machine-readable chemical structure identifiers. This generally entails three steps: identifying the chemical structures within the document as opposed to other graphical elements, compiling them into chemical graphs, and interpreting them. The first two steps can be seen as segmentation while the third step is the recognition and generation of the specified output in the form of one of the standard chemical structure identifiers. OCSR tools either dedicate themselves to both segmentation and recognition or focus only on the recognition – requiring the input to be processed into a single image of a chemical structure.[2, 8] This thesis focuses only on the recognition step.

2.4 OCSR Tools

While many techniques and models can be used to fulfill the goals of OCSR, the approach can be categorized into two main sub-sections. Most of the initially developed tools can be described as *Rule-based*, while more recently developed systems apply *Machine-learning* principles to the problem of image caption of chemical structures. [2]

2.4.1 Rule-based OCSR

One of the first systems to fulfill the needs of OCSR was Kekulé, published in 1992. Kekulé's workflow consists of scanning the input image, vectorizing it (raster to vector conversion), searching for bond lines, using Optical Character Recognition (OCR) to recognize atoms, using gathered information to complete a graph, and producing the output available in multiple machine-readable formats such as SMILES.[9] This sequence of steps forms a pattern that is present in most rule-based systems. The distinguishing features of the various tools tend to be deviations from this pattern or extra steps that build upon it.

As can be seen in Figure 5 taken from a 2020 review of OCSR tools, the majority of early OCSR systems were commercial. This thesis focuses on freely available tools and as such the following chapters will focus on the workflows of OSRA, Imago, and MolVec – three representatives of open-source rule-based OCSR tools.

Tool name	Programming language used	Operating System compatibility	Open-source	Commercial or free availability (2020)	Ongoing development
Kekulé	C++	Windows	No	Yes	No
OROCS	C	IBM OS/2	No	No	No
CLiDE Pro	C++	Windows	No	Yes	Yes
OSRA	C++	Independent	Yes[a]	Yes	Yes
ChemReader	C++	Windows	No	No	No
MolRec	Unknown	Unknown	No	No	Unknown
Imago	C++	Independent	Yes	Yes	No
ChemOCR	Java	Independent	No	Yes	Yes
ChemInfty	Unknown	Windows	No	No	No
eChem	Unknown	Unknown	No	No	No
MLOCSR	Unknown	Only Web interface	No	Only web interface	Unknown
OCSR	Unknown	Unknown	No	No	Unknown
ChemRobot	Unknown	Unknown	No	No	Unknown
MolVec	Java	Independent	Yes	Yes	Yes
MSE-DUDL	Python	Independent	No	No	No
Chemgrapher	Python	Independent	No	No	Yes

[a] Precompiled tool is only available commercially

Figure 5: *Comparison of tools and methods from a 2020 OCSR Review* [2]

2.4.1.1 OSRA

The first open-source OCSR tool developed was Optical Structure Recognition Application (OSRA). OSRA's workflow is like the above-mentioned general Rule-based workflow and the fact that it is open source allowed for participation in the development of not only OSRA but the OCSR problem in general. OSRA does not require specific image specifications such as resolution, color depth, or font type. It makes use of the ImageMagick library, which enables the processing of a wide variety of formats, including TIFF, JPEG, GIF, PNG, Postscript, and PDF. OSRA is implemented as a command-line utility and a web interface is available for demonstrative purposes and remains under development, the most recent version as of 2023 being 2.1.0. [10]

An obstacle in the free use of OSRA, when compared to other OCSR tools, is its complicated compilation process due to a high amount of version specific dependencies (GraphicsMagick, POTRACE, GOCR, OCRA, TCLAP, and OpenBabel) that need to be compiled from source code as well, a fact that is mentioned in multiple different attempts at benchmarking or implementing OSRA.[2, 11] The 2020 OCSR review that benchmarked OSRA used a PyOSRA environment and a conda recipe for OSRA 2.1.0 is available through bioconda. For this thesis, however, none of the mentioned methods of installation have been successful.

2.4.1.2 Imago

The publication of OSRA allowed for easier development of further OCSR systems. One such system is Imago – an open-source toolkit for 2D chemical structure image recognition. Imago was built with the intent of developing a cross-platform library that could be used in various applications, including mobile devices. In contrast to OSRA, it does not have any dependencies. This is highlighted by the fact that the installation of Imago requires just the download of a single executable. The workflow of Imago is once again like other rule-based systems. Imago can be used both as a command-line utility or a graphical user interface (GUI) program. The most recent version of Imago is 2.0.0. Plans for further development of the program are not mentioned in the related publication.

One of the focuses highlighted by Imago that are important for this thesis is the impact of noise and graphical damage. The authors consider the impact of low resolution, small number of symbols, and atom labels containing multiple symbols leading to a significant reduction in correct recognitions. These and similar ideas have become an overreaching focus in the ongoing development of OCSR tools. [12]

2.4.1.3 MolVec

In 2019 a Java library known as MolVec was developed to answer the need for a freely available OCSR tool that is lightweight, fully self-contained, and does not require higher-level programming knowledge to be implemented successfully. Unfortunately, as of 2023, MolVec does not have a paper published yet. All the relevant information can be found on the author's GitHub page.[13] MolVec is entirely a recognition tool, meaning that it does not contain a segmentation module and must only be provided images containing single chemical structures. Other than being used as a Java library, Molvec also contains a command-line runnable Main class. Further development of Molvec does not appear to be ongoing, with the latest version 0.9.8 being released on October 14, 2020. [13]

2.4.2 Machine-learning based OCSR

Rule-based systems make their decisions based on a set of pre-defined rules. These systems achieve solid results when dealing with datasets that contain structures whose properties can be described by a lower amount of such rules. When dealing with higher-complexity chemical structures, unique sub-structure and rule exceptions become harder to accurately predict using rule-based systems. Chemical structures found in chemical publications simply contain too much variation to be possible to account for only through pre-defined rules. The principle of machine-learning based OCSR is then to approach the problem as an image captioning problem by utilizing neural networks – a deep-learning and data-driven approach. [2, 14]

A common approach in image captioning is the use of an encoder-decoder network, where the encoder – a Convolutional Neural Network (CNN) is used to extract features from the image, and a decoder – usually a Recurrent Neural Network (RNN) is used to decode image features and generate text. Alternate approaches exist using Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) instead of the RNN. Extracted image features are also weighed using an attention model, which allows the decoder to focus on the most noteworthy features of the image. A self-attention mechanism used in a model known as Transformer has achieved performance that justifies further exploration. This framework became the basis for emerging machine-learning based systems. [14, 15]

2.4.2.1 Decimer

Deep-learning for chemical image recognition (DECIMER) was one of the first developed open-source deep-learning models. Based on existing show-and-tell neural networks that were developed for a more general image captioning problem – allowing for the problem to be approached without relying on assumptions of earlier models, potentially avoiding their shortcomings. According to the authors, the first iteration of DECIMER did not achieve performance levels that could rival established rule-based methods. [3, 16]

DECIMER 1.0, an updated version of DECIMER developed a year later implemented a successful model of Transformers, leading to a significant increase in performance. The DECIMER project continues to be open source and in further development, with a plan to

integrate further deep-learning technologies and increase the training datasets to 50-100 million structures, which the authors claim will lead to a further increase in performance. [3, 17]

At the time of this thesis, the most recent published version of DECIMER is 2.2.0 released on February 20th, 2023. The project also created a webpage called decimer.ai that can be used for the extraction and recognition of structures from a PDF file.[18]

2.4.2.2 MolMiner

MolMiner, published in September 2022, works with neural networks developed for the tasks of semantic segmentation and object detection, mapped to the task of recognizing atom and bond elements from documents. These elements are then connected into a molecular graph using a distance-based construction algorithm.

The authors present the results of their testing on both self-collected real-world data sets and a benchmark evaluation, comparing MolMiner to MolVec, Imago, and OSRA, using accuracy on InChI and MCS output formats. MolMiner achieves the highest accuracy and lowest run-time on all tests.

Unfortunately, MolMiner is only available as a GUI application. Users can submit a PDF file and MolMiner attempts to segregate chemical structures, asks the user for manual segregation – which it then attempts to recognize. The user can then manually modify the segregation of unrecognized molecules. Unfortunately, as of April 2023, there is no way to automate the recognition process through the MolMiner application, which makes it not suitable for the comparative testing that is the subject of this thesis. [19]

2.4.2.3 SwinOCSR

Another open-source machine-learning based system was SwinOCSR published in July 2022. SwinOCSR is based on implementing the Swin Transformer as the backbone of the model followed by a Transformer encoder and decoder. The backbone is used to extract image features into a high-dimensional patch sequence. This patch sequence is flattened and fed into the Transformer encoder, generating a representation sequence, which the decoder uses to decode a corresponding DeepSMILES. The advantages of this approach are that instead of pooling, which is used in CNN and leads to information loss, Swin Transformer merges

neighboring patches of the sequence, lowering the size of feature maps to avoid information loss.

SwinOCSR authors used the Cheminformatics toolkit (CDK) to generate their dataset in the form of molecular diagrams which allowed them to train their model on a big dataset. Several procedures were used during the generation of the dataset to make it more representative of how chemical structures are present in publications. They also used four different evaluation metrics to test the validity of their model – Accuracy, Tanimoto distance, BLUE, and ROUGE. The first two metrics are common for evaluating OCSR, while the second two are common to evaluate image captioning methods. The model achieved satisfactory results when evaluated on testing data that is similar to training data, however, it only achieved 25% accuracy when tested on real literature data, which the authors deemed unsatisfactory. They propose that improvements to the model can be made by expanding the dataset with high-complexity low-resolution images.[15] For this reason and due to very slow execution time, SwinOCSR was omitted from the testing part of this thesis.

2.4.2.4 MolScribe

The most recent development in OCSR, published in March 2023 was MolScribe. MolScribe model uses three approaches that complement each other, to handle high data variation common in chemistry. Data augmentation was used to generate a high variety of chemical images to cover as many drawing styles as possible during training – leading to an increase in robustness. This appears to be the first OCSR implementation that attempts to take principles of rule-based systems and extend them into the realm of machine-learning based systems to get the best out of both worlds.

MolScribe was benchmarked against other open-source tools, both from the rule-based and machine-learning based categories. The benchmarking covers synthetic, realistic, and perturbed benchmarks. MolScribe achieved the highest recognition rates on most of the benchmarks, occasionally being outperformed by Decimer by a margin of about 1 percent.

All code, data, and model checkpoints are publicly available on the authors' GitHub repository. A public interface is also available on huggingface.co. [20]

3 EXPERIMENTAL PART

3.1 Dataset

To evaluate the selected OCSR tools a dataset of chemical structures was created. The idea was to take a representative set of images of chemical structures and subject it to various kinds and degrees of graphical damage to ascertain which of them would negatively affect the recognition of a given OCSR tool. The following chapters describe the individual steps that were taken in the creation of the dataset.

3.1.1 Structure selection

The first step was to select the structures that would be included in the dataset. The molecules were selected using ChatGPT[21] as trivial or systemic names with a combination of prompting and trial and error. A considerable number of suggestions were generated and then a combination containing varying substituents, lengths, and bond amounts was manually selected from the suggestions. The representation of the dataset was decided to be:

- Twenty simple linear molecules
- Twenty simple branched molecules
- Thirty cyclic molecules
    - Five simple structures featuring a benzene core
    - Five heterocyclic structures
    - Five structures with fused rings
    - Five structures with bridged rings
    - Five polycyclic structures
    - Five macrocyclic structures
- Forty biochemically relevant molecules
    - Ten amino acids
    - Five saccharides
    - Five lipids
    - Five hormones
    - Five vitamins

    - Five molecules with unique or unusual structures
    - Five more random biochemically significant molecules
- Ten molecules that combine linear, branched, and cyclic molecules
- Nine pharmaceutically significant molecules

The result of this step was a .txt file containing 129 systematic or trivial names of chemical structures that would form the base of the dataset.

3.1.2 Structure generation

The names selected by ChatGPT were used to generate corresponding chemical structures using the "Convert name to structure" command in ChemDraw Professional version 22.2.0. ChemDraw was then used to export the images of the structures as 300dpi TIFF files, without the specification of dimensions, size, or resolution of the images. ChemDraw automatically selected the dimensions and size of the TIFF files to allow for slight padding around the outer edges of the molecule. ChemDraw was also used to export the corresponding molfiles, which would then be used as a reference for testing.

The result of this step was the base dataset of 129 TIFF files and their corresponding molfiles.

3.1.3 Compression damage subset

Compression is probably the most common form of graphical damage that stored images undergo – the most common form of compression is JPEG compression. This was achieved using the Image module from the Pillow fork. [22] Original TIFF images were saved using the Image.save() function with the quality parameter set to the values of 80, 60, 40, 20, and 1 resulting in the formation of five *tiffs_compressed_(compression percentage)* subsets. See Figure 6 for an example of gradual compression damage.

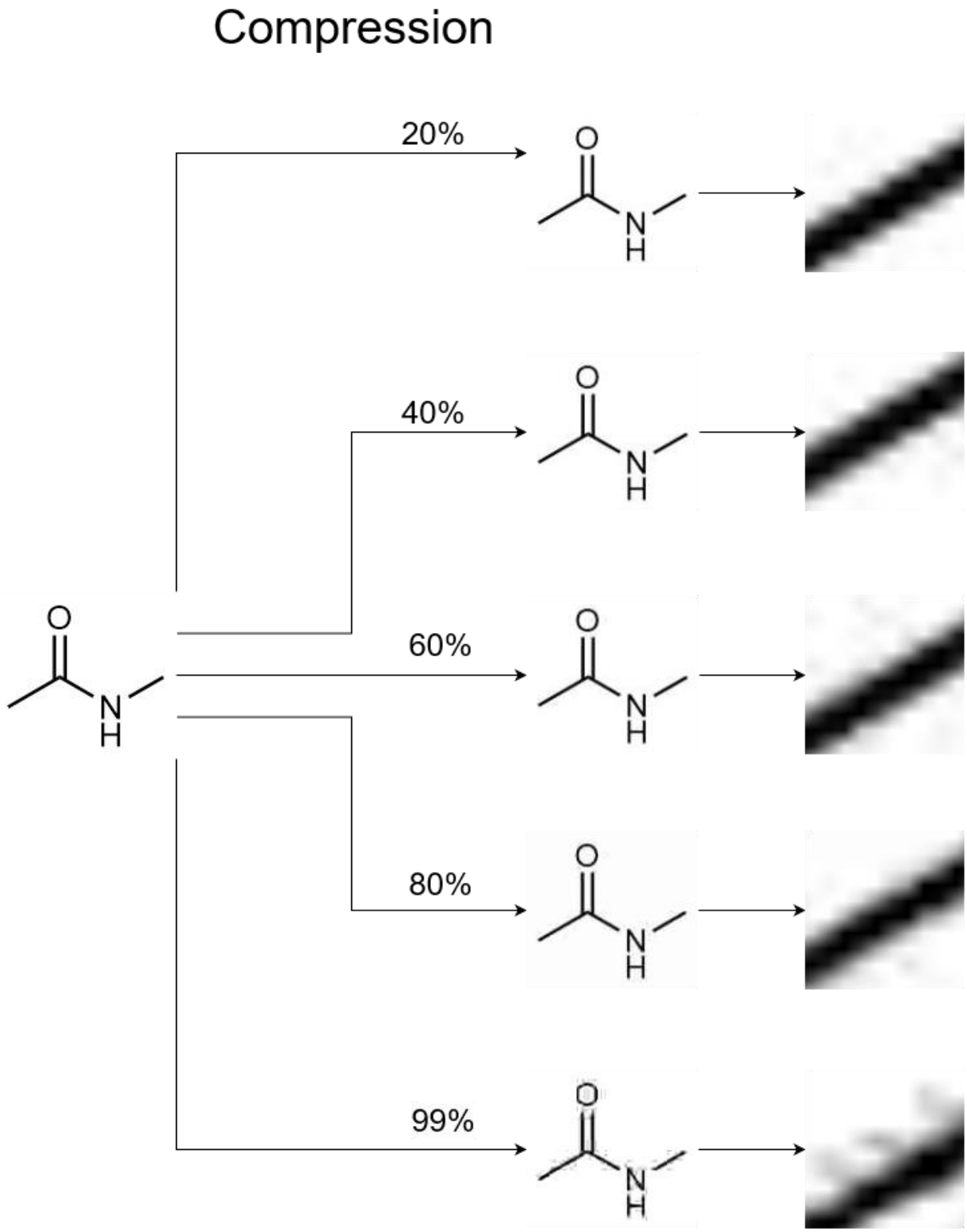


Figure 6: *Visualization of gradual compression damage*

3.1.4 Convert damage subset

Another ordinary form of damage is random noise. This was achieved using the convert command from ImageMagick, with the type of noise specified as grey Gaussian noise of gradually increasing attenuation of values 0.5-2.5, generating the *tiffs_convert_(attenuation_value)* subsets. See Figure 7 for an example.

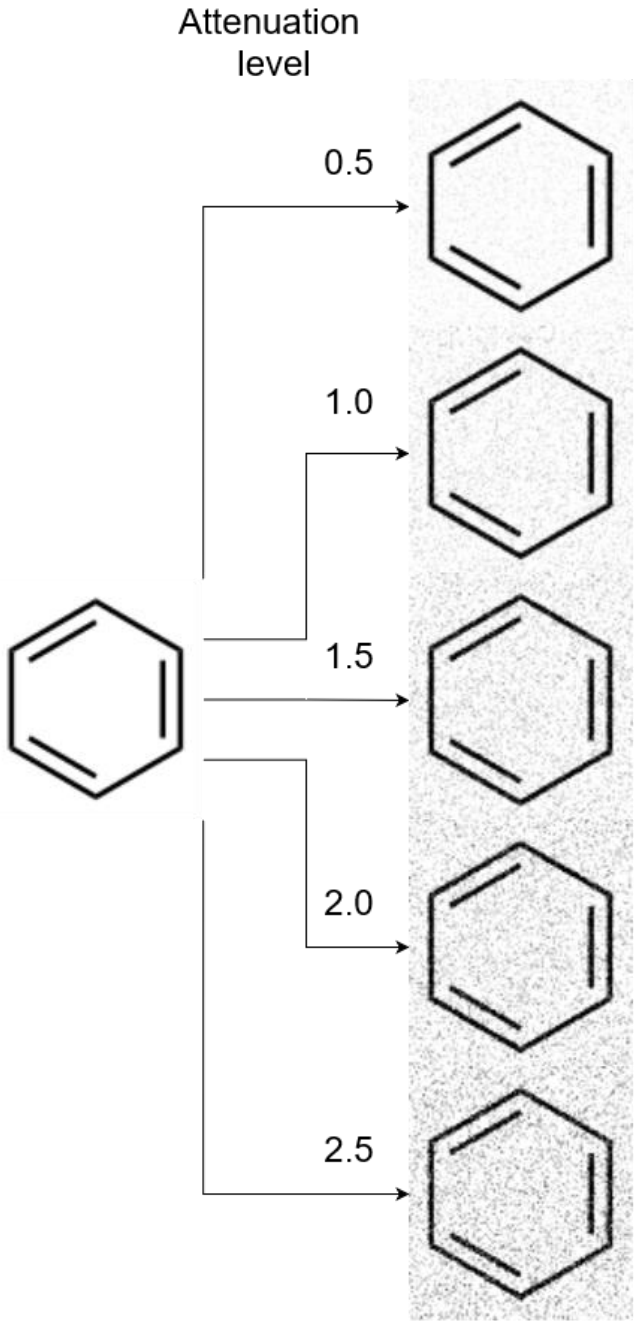


Figure 7: *Visualization of gradual damage via Gaussian noise*

3.1.5 Blend damage subset

Scanned files often feature a background that is not perfectly white. To simulate this, a subset was created by gradually overlaying the images with a black mask of an increasing overlay percentage (20, 40, 60, and 80), using the blend function from ImageMagick. The result of this overlay were the subsets *tiffs_blend_(overlay_percentage)*. See Figure 8 for an example.

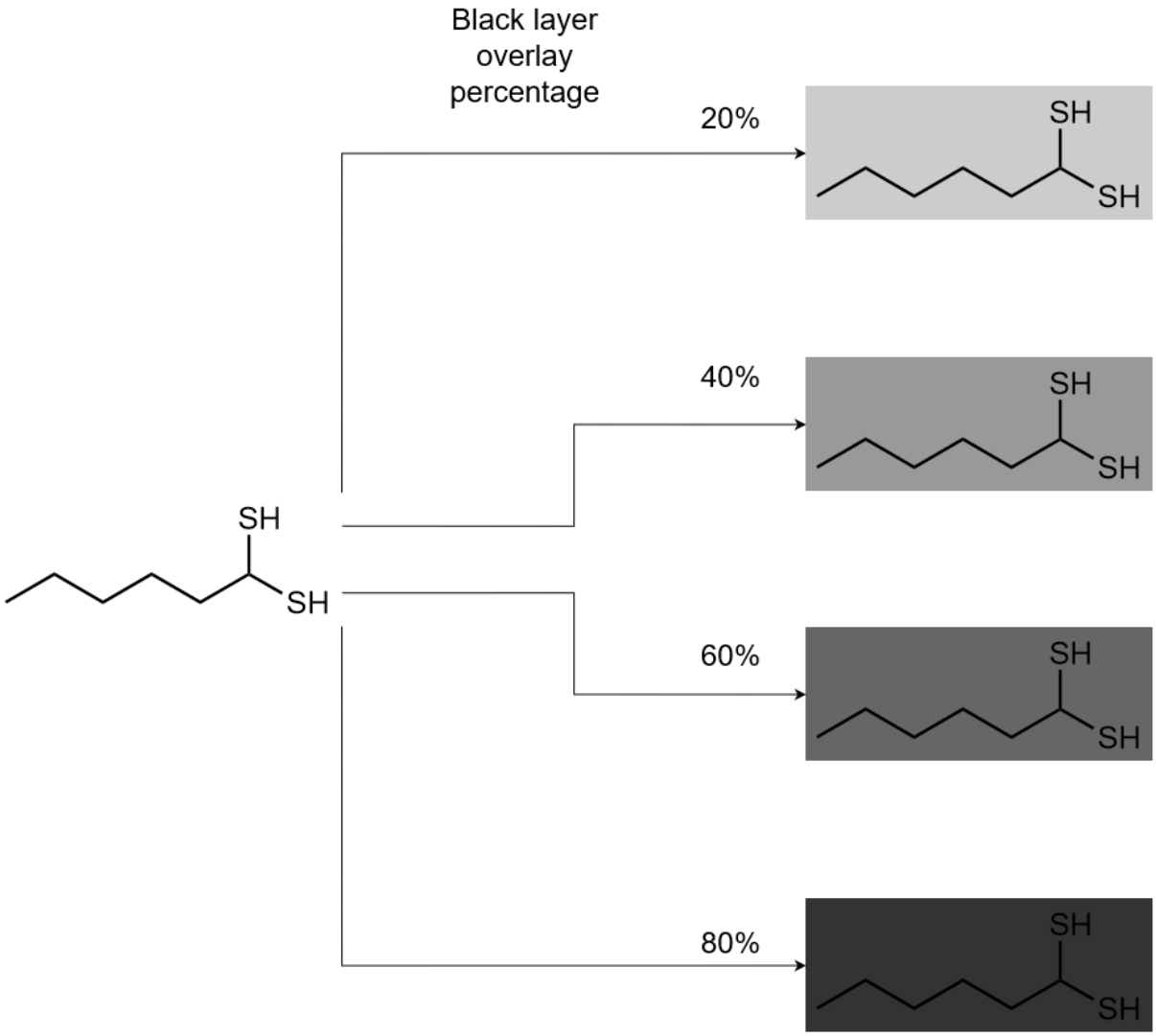


Figure 8: *Visualization of gradual damage via the blending of a black layer*

3.1.6 Distortion damage subset

Imperfect scanning may lead to the creation of creases and distortions of the original image. A simulation of this damage was implemented through ten repeated uses of the img.distort() function from ImageMagick with the method specified as ‘shepards’ with randomly selected points. 30-pixel white padding was added to all pictures to prevent the distortion from moving the bodies of the structures outside of the frame.

Shepard’s distortion moves a specified source point to a destination coordinate. In this case, a source point was selected as a random x, y coordinate in the range of 0.3 to 0.7 times the maximum x or y, and the destination point at a random distance away from the source point – the scale of this distance increasing through a *scale* parameter that indicated the scale of the distortion (from 0.1 to 0.5, but it should be noted that this was an arbitrary choice based on the quality of the resulting distortion). The result of this manipulation are the subsets *tiffs_distort_(scale_parameter)*. See Figure 10 for an example. An unexpected result of the distortion was the creation of “hand-drawn-like” images of chemical structures through a quick and automated process, see Figure 9 for examples of resulting structures.

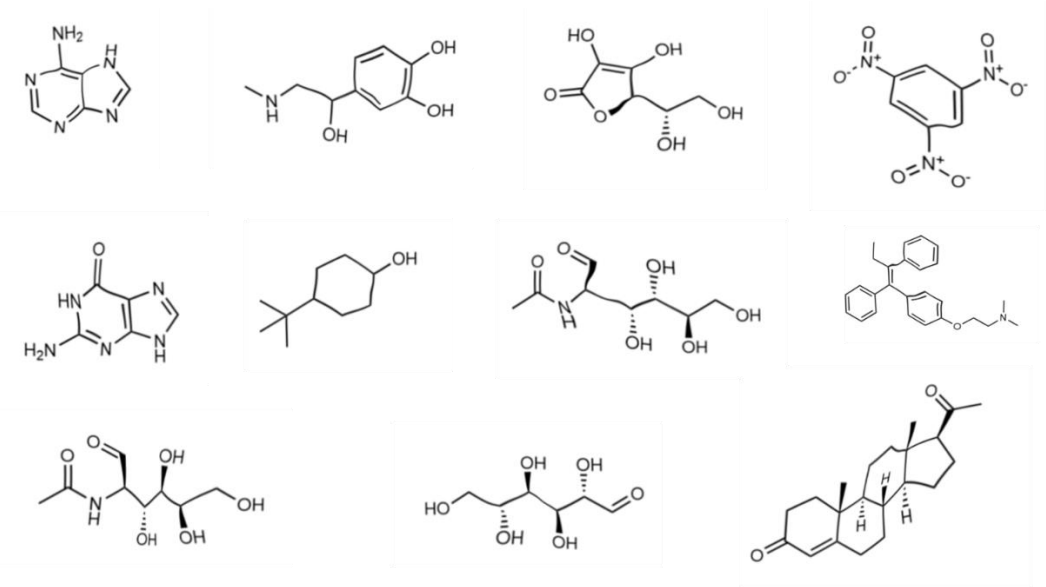


Figure 9: *Examples of distorted structures that resemble hand-drawn chemical images*

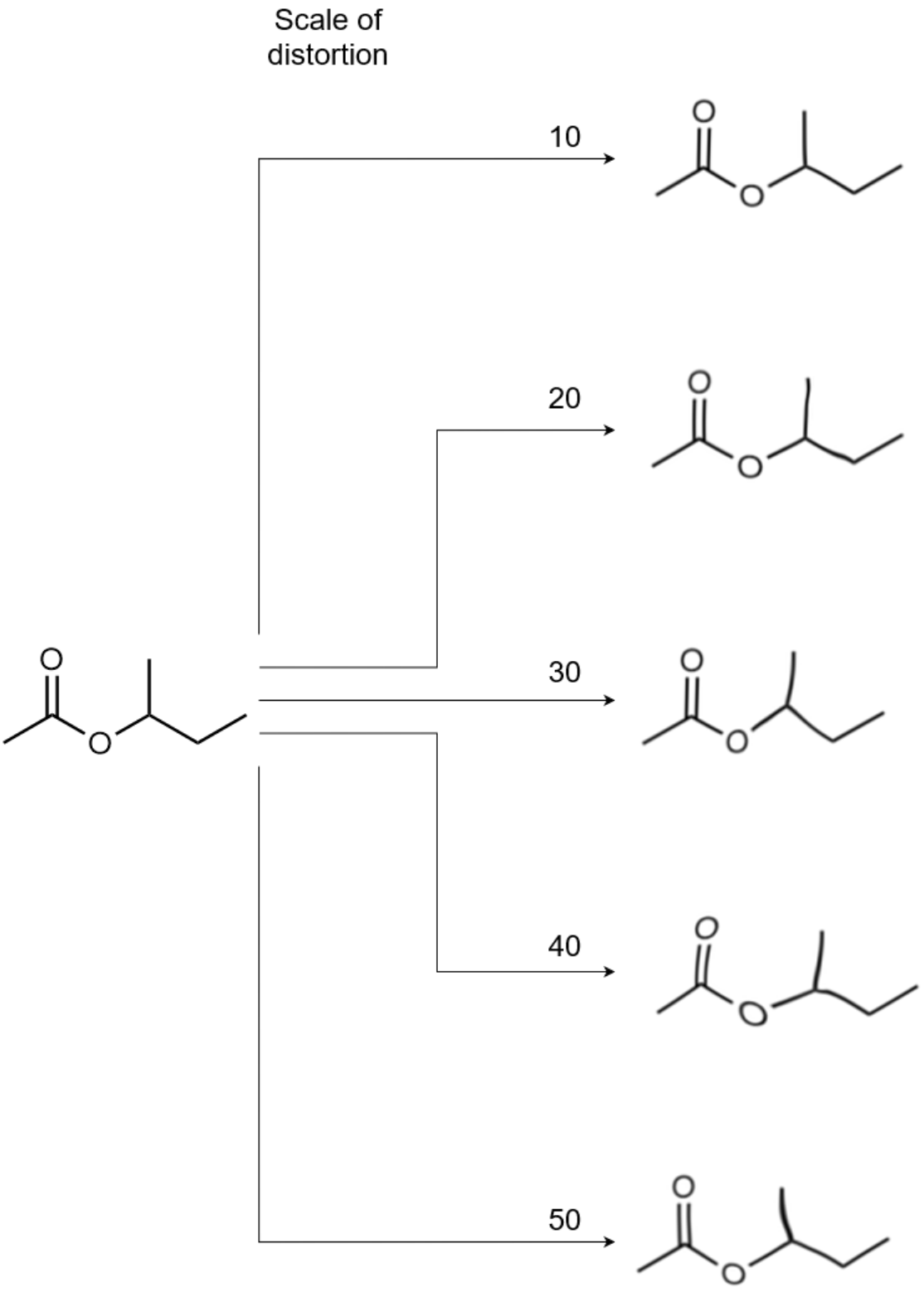


Figure 10: *Visualization of gradual distortion via Shepards distortion*

3.2 Testing

The main goal of the testing phase was to generate recognition success rate data for each of the tested OCSR tools on each of the subsets. This was achieved by comparing molfiles corresponding to the tool's attempted recognition to the reference molfiles exported by ChemDraw. The general workflow of testing can be summarized as follows: TIFF files with chemical structure images are given to the tool as input, the tool attempt to recognize the structure in the image, the tool generates a chemical structure identifier (molfile if possible), this is repeated for all the subsets of the dataset. In case the recognition process fails, is aborted, loops, or produces an empty molfile – a placeholder molfile that is always evaluated as an unsuccessful recognition is generated because of that attempt. Once all images are processed, each subset of generated molfiles is compared to the reference molfiles. The workflow is visualized in Figure 11.

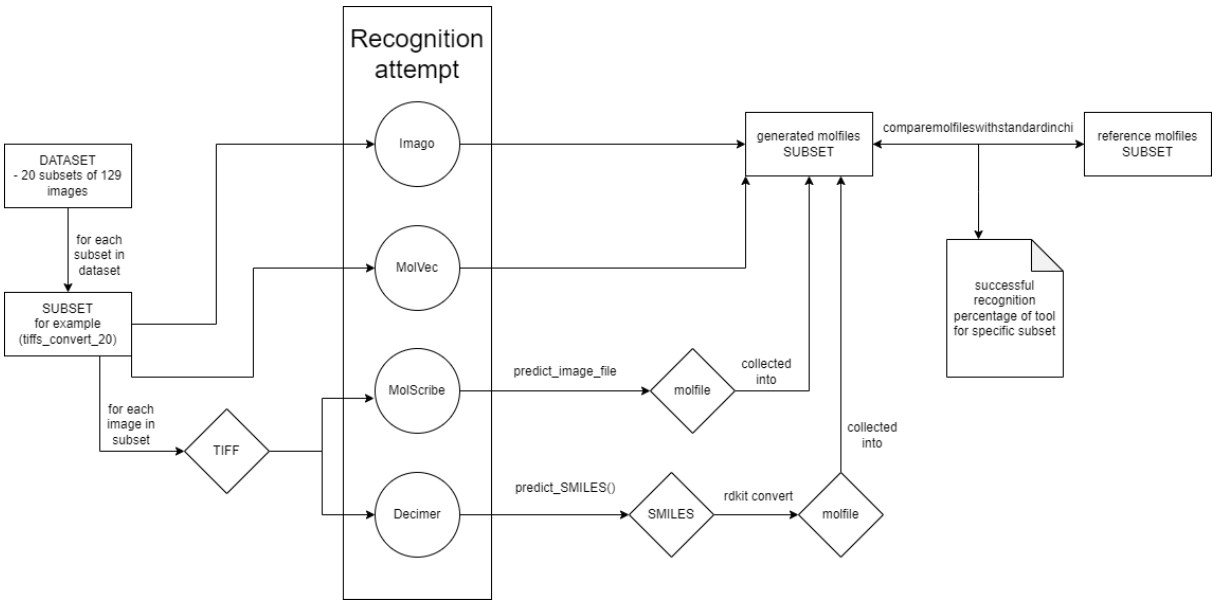


Figure 11: *Visualization of the OCSR tool testing*

3.2.1 Testing environment

Each of the tools was installed and tested within a separate miniconda environment, to ensure that their dependencies would not interfere with each other. Since execution time was only considered on a relative scale, rather than measured (see chapter Results and discussion), the

processor and graphics card were not relevant to the testing process. The following versions and forms of OCSR tools were chosen for testing:

Imago 2.0.0 command-line utility for 64-bit Windows (imago-console).

Molvec 0.98 runnable Main class obtained from a Molvec jar built using Maven.

DECIMER V2.0.0 installed as a Python package from PyPi.

MolScribe (version not specified) installed through pip, using a checkpoint from HuggingFace Hub provided on the MolScribe GitHub repository.

3.2.2 Results formatting and evaluation

Once all the recognized molfiles were created a modified version of the script from the 2020 OCSR benchmarking study was used. The script takes both the generated and reference molfile and converts them to their corresponding InChI string using RDKit, afterwards, it evaluates whether the two InChI strings match and logs the result of the evaluation, including both the generated and reference InChI string into a text file that is shared for the entire subset. After comparing all molfiles of a given subset are compared the result of the evaluation is calculated as the number of matching InChI divided by the number of molfiles in the subset.

3.2.3 Tool-specific modifications

Some tool-specific behaviors had to be addressed manually for the recognition and evaluation process to not be interrupted.

Imago got stuck in a loop while attempting to recognize three of the structures in the tiffs_convert_20 subset (but not on the more damaged tiffs_convert_25 subset). To prevent this, structures that had caused the tool to loop were temporarily removed from the dataset and replaced with a copy of a random different structure from the subset, which would ensure that the recognition attempt would be evaluated as unsuccessful.

Decimer's only option for output of recognition was SMILES strings generated by the predict_SMILES() function. A testing script was created to add SMILES to molfile conversion so that the evaluation process could be the same as for other tools. Decimer tended

to occasionally generate invalid SMILES strings that lead to a parsing error when conversion to molfiles was attempted. If this occurred, the script produced a placeholder molfile that would lead to the recognition being evaluated as unsuccessful.

4 RESULTS AND DISCUSSION

4.1 Individual results

This section is focused primarily on the effects of distinct types of graphical damage on OCSR tools by comparing the recognition success rate on damaged subsets to the rate on unmodified subsets.

4.1.1 Imago

Table 1: *Recognition rates per subset of Imago*

	base	blend_20	blend_40	blend_60	blend_80	
blend	73.6%	34.9%	33.3%	32.6%	29.5%	
	base	compress_20	compress_40	compress_60	compress_80	compress_99
compress	73.6%	37.2%	37.2%	36.4%	34.9%	20.2%
	base	convert_5	convert_10	convert_15	convert_20	convert_25
convert	73.6%	34.9%	38.0%	15.5%	6.2%	9.3%
	base	distort_10	distort_20	distort_30	distort_40	distort_50
distort	73.6%	62.8%	64.3%	59.7%	55.8%	54.3%

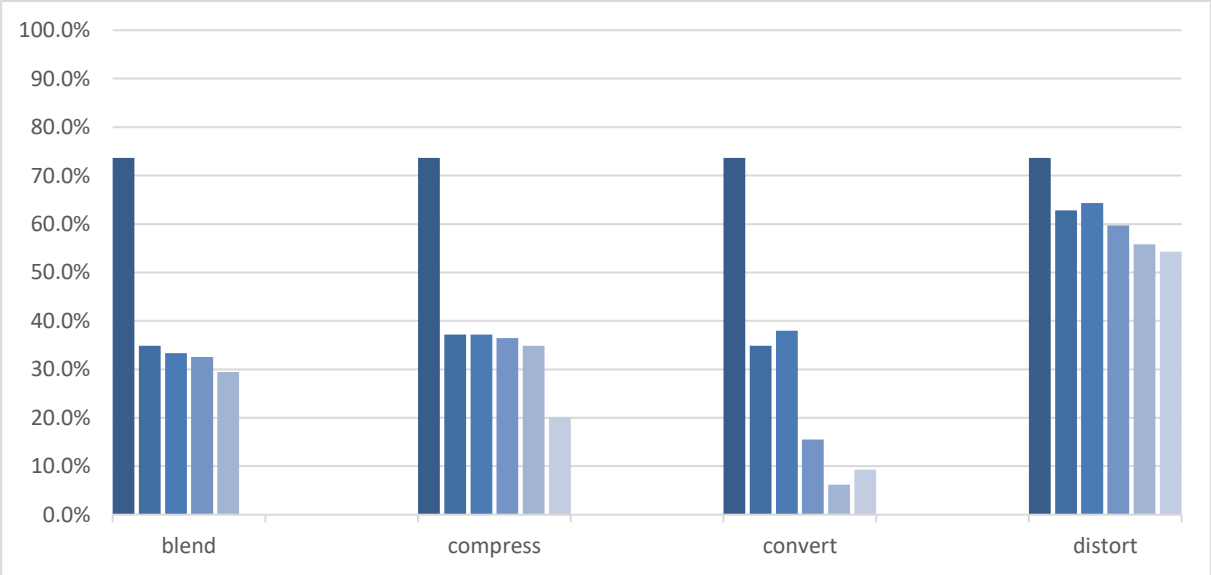


Figure 12: *Results of testing on subsets affected by a type of damage (the first column represents an undamaged subset) of Imago*

The introduction of even a small degree of a black overlay, compression damage, or noise led to significant drop-offs in performance. High degrees of compression or noise led to very low rates of recognition. Distortion damage led to a gradual reduction of recognition success.

4.1.2 MolVec

Table 2: *Recognition rates per subset of MolVec*

	base	blend_20	blend_40	blend_60	blend_80	
blend	89.1%	89.1%	88.4%	89.1%	89.1%	
	base	compress_20	compress_40	compress_60	compress_80	compress_99
compress	89.1%	86.8%	86.8%	89.1%	84.5%	43.4%
	base	convert_5	convert_10	convert_15	convert_20	convert_25
convert	89.1%	88.4%	81.4%	48.1%	31.0%	43.4%
	base	distort_10	distort_20	distort_30	distort_40	distort_50
distort	89.1%	74.4%	75.2%	70.5%	62.0%	62.0%

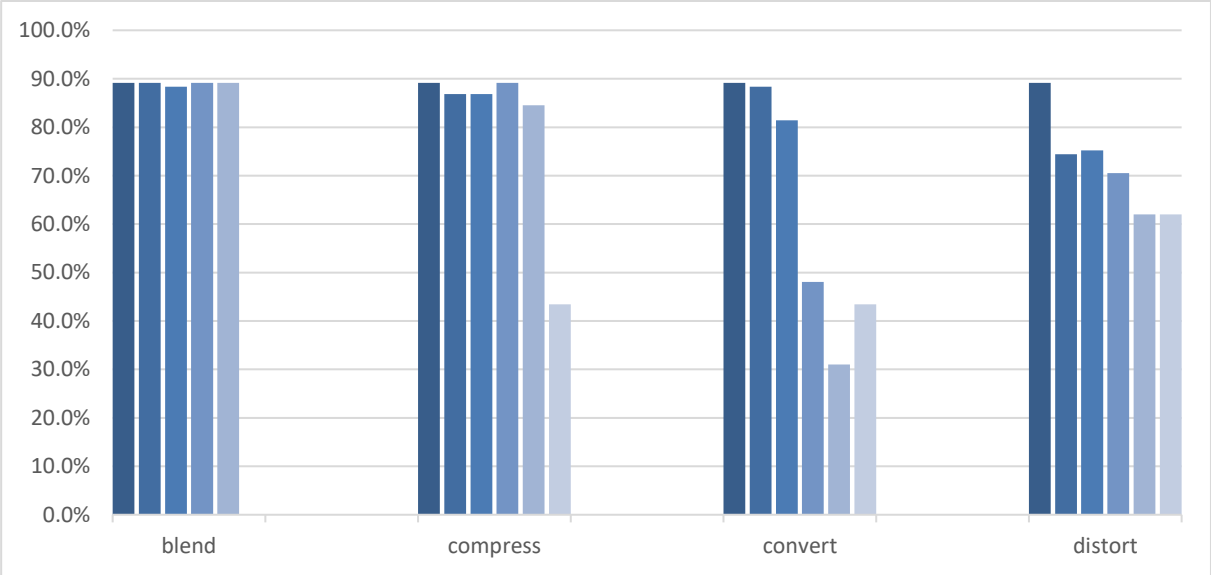


Figure 13: *Results of testing on subsets affected by a type of damage (the first column represents an undamaged subset) of MolVec*

The presence of a black overlay did not appear to affect the recognition success of MolVec at all. MolVec also shows an elevated level of resilience towards compression damage and noise, with only higher degrees of this damage leading to a significant reduction in recognition rate. The presence of any distortion leads to an almost 20% reduction in success rate, decreasing as the distortion increases.

4.1.3 Decimer

Table 3: *Recognition rates per subset of Decimer*

	base	blend_20	blend_40	blend_60	blend_80	
blend	82.2%	23.3%	26.4%	0.8%	0.0%	
	base	compress_20	compress_40	compress_60	compress_80	compress_99
compress	82.2%	83.7%	81.4%	75.2%	71.3%	44.2%
	base	convert_5	convert_10	convert_15	convert_20	convert_25
convert	82.2%	15.5%	4.7%	1.6%	1.6%	0.8%
	base	distort_10	distort_20	distort_30	distort_40	distort_50
distort	82.2%	79.8%	79.1%	79.1%	77.5%	73.6%

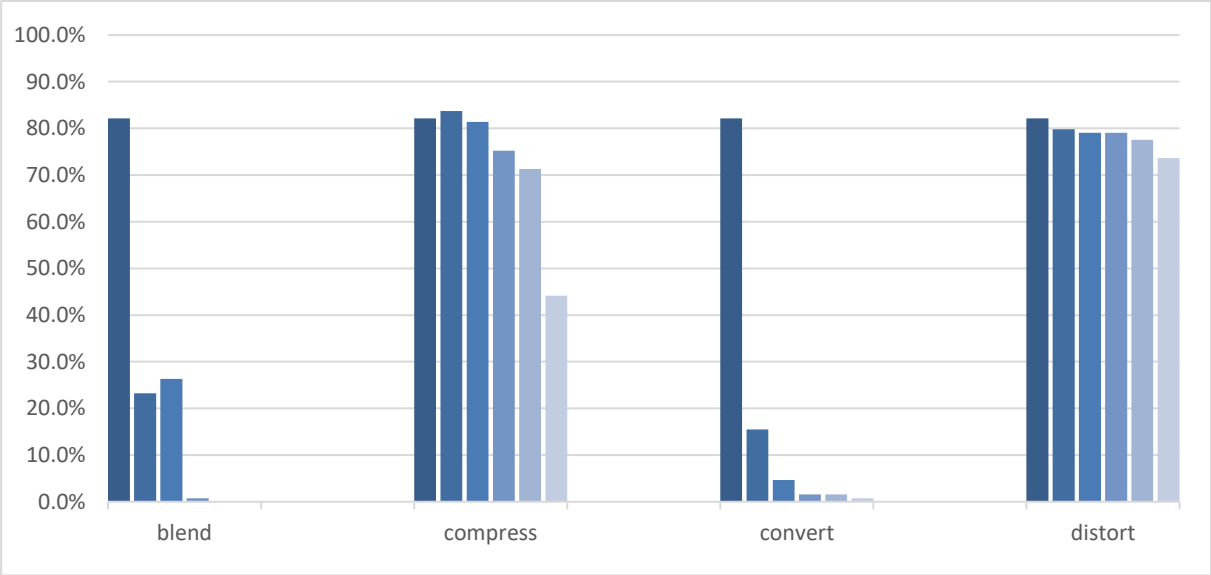


Figure 14: *Results of testing on subsets affected by a type of damage (the first column represents an undamaged subset) of Decimer*

The recognition ability of Decimer showed extreme sensitivity to the presence of a black overlay and noise with both forms of damage leading to sub-30% success and high rates of these damages leading to a recognition rate below 2%. The reason for this is Decimer interpreting the black overlay and dots of noise as carbon atoms with the most usual form of output on these subsets being long SMILES strings consisting mostly of Cs. Compression damage only began to significantly affect the recognition rate of Decimer at the highest compression percentage subset. Decimer also showed a remarkably high capability of dealing with distorted structures.

4.1.4 MolScribe

Table 4: *Recognition rates per subset of MolScribe*

	base	blend_20	blend_40	blend_60	blend_80	
blend	94.6%	48.1%	63.6%	46.5%	10.1%	
	base	compress_20	compress_40	compress_60	compress_80	compress_99
compress	94.6%	94.6%	93.0%	94.6%	94.6%	55.8%
	base	convert_5	convert_10	convert_15	convert_20	convert_25
convert	94.6%	89.9%	49.6%	20.9%	7.0%	3.1%
	base	distort_10	distort_20	distort_30	distort_40	distort_50
distort	94.6%	93.0%	93.0%	87.6%	77.5%	72.9%

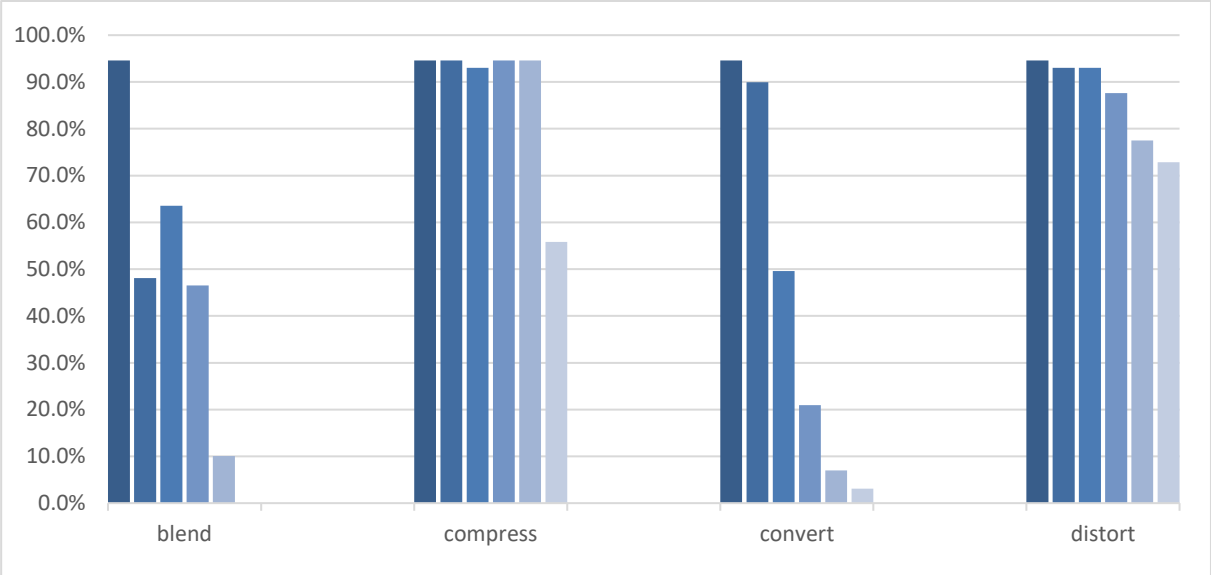


Figure 15: *Results of testing on subsets affected by a type of damage (the first column represents an undamaged subset) of MolScribe*

The effect of a black overlay on MolScribe was inconsistent but can be described as negative with a drop-off of about 50% and the highest percentage overlay leading to a breakdown of recognition. MolScribe appeared especially capable of handling compressed images, with only a 99% conversion leading to a significant reduction in recognition. MolScribe could handle low levels of noise decently, but higher levels lead to recognition rates below 20%. Finally, the effect of distortion was minimal at lower levels but grew in significance at higher levels.

4.2 Comparison between individual results

Table 5 shows recognition rates and Table 6 shows the percentage of performance decrease due to the specified type and degree of damage. Of the tested OCSR tools, MolScribe and MolVec showed the highest recognition success rate on the undamaged subset, while Imago performed worse than others by a margin of about 10%. MolVec was the only tool that was not affected by the black overlay, while Decimer proved to be especially vulnerable to this type of damage. Decimer also showed higher sensitivity to noise than other tools, while MolVec appeared most resistant to noise (but not immune). All tools except for Imago showed a capability to handle compression rates up to 80% without losing much accuracy (only Decimer showed a noticeable decrease when dealing with compression rates of 60% and 80%). All tools suffered around a 50% reduction in accuracy on the 99% compression rate subset. Machine-learning based tools proved to be better equipped at dealing with distortion (Decimer especially only suffered a 10% reduction at the highest distortion level), while rule-based tools recognition rate reduction rate was in the realm of 15-30% (with MolVec being the most vulnerable).

Table 5: *Comparison of recognition rates per subset of all tested tools*

	base	blend_20	blend_40	blend_60	blend_80
decimer	82.2%	23.3%	26.4%	0.8%	0.0%
molscribe	94.6%	48.1%	63.6%	46.5%	10.1%
imago	73.6%	34.9%	33.3%	32.6%	29.5%
molvec	89.1%	89.1%	88.4%	89.1%	89.1%
	convert_5	convert_10	convert_15	convert_20	convert_25
decimer	15.5%	4.7%	1.6%	1.6%	0.8%
molscribe	89.9%	49.6%	20.9%	7.0%	3.1%
imago	34.9%	38.0%	15.5%	6.2%	9.3%
molvec	88.4%	81.4%	48.1%	31.0%	43.4%
	compress_20	compress_40	compress_60	compress_80	compress_99
decimer	83.7%	81.4%	75.2%	71.3%	44.2%
molscribe	94.6%	93.0%	94.6%	94.6%	55.8%
imago	37.2%	37.2%	36.4%	34.9%	20.2%
molvec	86.8%	86.8%	89.1%	84.5%	43.4%
	distort_10	distort_20	distort_30	distort_40	distort_50
decimer	79.8%	79.1%	79.1%	77.5%	73.6%
molscribe	93.0%	93.0%	87.6%	77.5%	72.9%
imago	62.8%	64.3%	59.7%	55.8%	54.3%
molvec	74.4%	75.2%	70.5%	62.0%	62.0%

Table 6: *Comparison of percentage decrease of recognition rate of damaged subsets, 0% means that the subset was recognized just as well as the undamaged, and 50% means that the damaged subset recognition success rate was half of the undamaged recognition success rate.*

	base	blend_20	blend_40	blend_60	blend_80
decimer	0.0%	71.7%	67.9%	99.1%	100.0%
molscribe	0.0%	49.2%	32.8%	50.8%	89.3%
imago	0.0%	52.6%	54.7%	55.8%	60.0%
molvec	0.0%	0.0%	0.9%	0.0%	0.0%
	convert_5	convert_10	convert_15	convert_20	convert_25
decimer	81.1%	94.3%	98.1%	98.1%	99.1%
molscribe	4.9%	47.5%	77.9%	92.6%	96.7%
imago	52.6%	48.4%	78.9%	91.6%	87.4%
molvec	0.9%	8.7%	46.1%	65.2%	51.3%
	compress_20	compress_40	compress_60	compress_80	compress_99
decimer	-1.9%	0.9%	8.5%	13.2%	46.2%
molscribe	0.0%	1.6%	0.0%	0.0%	41.0%
imago	49.5%	49.5%	50.5%	52.6%	72.6%
molvec	2.6%	2.6%	0.0%	5.2%	51.3%
	distort_10	distort_20	distort_30	distort_40	distort_50
decimer	2.8%	3.8%	3.8%	5.7%	10.4%
molscribe	1.6%	1.6%	7.4%	18.0%	23.0%
imago	14.7%	12.6%	18.9%	24.2%	26.3%
molvec	16.5%	15.7%	20.9%	30.4%	30.4%

4.3 Execution speed

While execution speed was not explicitly measured, the amount of time it took for each tool to process the entire dataset was significantly different enough to be mentioned on a relative scale. It should be noted that execution time depends on a considerable number of factors that will inevitably differ based on the testing environment.

MolVec and Imago processed the entire dataset in a matter of minutes, with MolVec being noticeably faster than Imago. In contrast, both Decimer and MolScribe processed the dataset in a matter of hours, with Molscribe being about six times slower than Decimer.

5 CONCLUSIONS

Four different OCSR tools, two rule-based and two machine-learning based were installed and tested on several subsets featuring various levels and types of damage, generating a recognition success rate per each tool per each subset. Two types of information were gathered – the performance of tools while working with both undamaged and damaged subsets, and the effect distinct types of damage have on recognition success rates.

Of the tested tools, MolVec and MolScribe stood out from the rest for varied reasons. MolScribe for its high performance and robustness in the face of compression and distortion damage. MolVec for its combination of reliable performance and high speed while showing a unique resilience towards a blending background and noise. Decimer was the least affected by distortion damage. Imago was by far the easiest to install and use, but it was outperformed by the other tools on almost all subsets.

Except for extreme compression, high amounts of noise damage, and to an extent any distortion – all other tested damage has been handled without a decrease in performance by at least one of the tools. This suggests that a combination of the aspects of the tools that allow them to handle a particular part of damage could lead to better results. Particularly, aspects of MolVec could be introduced into MolScribe to improve its handling of blending background and noise.

Further testing could be done by adding a binarization step to the testing process since a correctly executed binarization step should entirely remove the effect of a blending black background. This appears to be a reasonable concern – considering that scanned pages often do not have a perfectly white background or black text. It is possible that many contemporary OCSR tools are losing successful recognition due to imperfectly implemented or omitted binarization steps.

An interesting proposal could be the utilization of two rule-based OCSR tools with one machine-learning based (MolVec + OSRA + MolScribe for example). If MolVec and OSRA outputs match, they are accepted as the output, if they do not, MolScribe is used to recognize the structure. This could lead to a compromise between execution speed and performance, and the benefits of rule-based tools combined with those of machine-learning based.

REFERENCES

(1) Brecher, J. Graphical representation of stereochemical configuration (IUPAC Recommendations 2008). *Pure Appl Chem* **2008**.

(2) Rajan, K.; Brinkhaus, H. O.; Zielesny, A.; Steinbeck, C. A review of optical chemical structure recognition tools. *J. Cheminf.* **2020**, *12* (1), 60. DOI: 10.1186/s13321-020-00465-0.

(3) Musazade, F.; Jamalova, N.; Hasanov, J. Review of techniques and models used in optical chemical structure recognition in images and scanned documents. *J Cheminform* **2022**, *14* (1), 61. DOI: 10.1186/s13321-022-00642-3.

(4) O'Boyle, N. M. Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI. *Journal of Cheminformatics* **2012**, *4* (1), 22. DOI: 10.1186/1758-2946-4-22.

(5) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* **1988**, *28* (1), 31-36.

(6) Heller, S. R.; McNaught, A.; Pletnev, I.; Stein, S.; Tchekhovskoi, D. InChI, the IUPAC International Chemical Identifier. *J. Cheminf.* **2015**, *7*, 1-63. DOI: 10.1186/s13321-015-0068-4.

(7) Wigh, D. S.; Goodman, J. M.; Lapkin, A. A. A review of molecular representation in the age of machine learning. *WIREs Computational Molecular Science* **2022**, *12* (5), e1603. DOI: https://doi.org/10.1002/wcms.1603.

(8) Valko, A. T.; Johnson, A. P. CLiDE Pro: The Latest Generation of CLiDE, a Tool for Optical Chemical Structure Recognition. *J. Chem. Inf. Model.* **2009**, *49* (4), 780-787. DOI: 10.1021/ci800449t.

(9) McDaniel, J. R.; Balmuth, J. R. Kekule: OCR-optical chemical (structure) recognition. *J. Chem. Inf. Comput. Sci.* **1992**, *32* (4), 373.

(10) Filippov, I. V.; Nicklaus, M. C. Optical structure recognition software to recover chemical information: OSRA, an open source solution. *J. Chem. Inf. Model.* **2009**, *49* (3), 740-743. DOI: 10.1021/ci800067r.

(11) Novotný, J. *Molminer Github repository*. 2020. https://github.com/gorgitko/molminer (accessed 2023)

(12) Chutkov, R.; Rybalkin, M.; Smolov, V.; Andrea, K. Imago: Open-source toolkit for chemical structure image recognition. 2012; American Chemical Society: p CINF.

(13) NCATS. *Molvec GitHub repository*. 2020. https://github.com/ncats/molvec (accessed 2023)
(14) Staker, J.; Marshall, K.; Abel, R.; McQuaw, C. M. Molecular Structure Extraction from Documents Using Deep Learning. *J. Chem. Inf. Model.* **2019**, *59* (3), 1017-1029. DOI: 10.1021/acs.jcim.8b00669.

(15) Xu, Z.; Li, J.; Yang, Z.; Li, S.; Li, H. SwinOCSR: end-to-end optical chemical structure recognition using a Swin Transformer. *J Cheminform* **2022**, *14* (1), 41. DOI: 10.1186/s13321-022-00624-5.

(16) Rajan, K.; Zielesny, A.; Steinbeck, C. DECIMER: towards deep learning for chemical image recognition. *J. Cheminf.* **2020**, *12* (1), 65. DOI: 10.1186/s13321-020-00469-w.

(17) Rajan, K.; Zielesny, A.; Steinbeck, C. DECIMER 1.0: deep learning for chemical image recognition using transformers. *J Cheminform* **2021**, *13* (1), 61. DOI: 10.1186/s13321-021-00538-8.

(18) Rajan, K.; Zielesny, A.; Steinbeck, C. DECIMER-V2. 2022.

(19) Xu, Y.; Xiao, J.; Chou, C.-H.; Zhang, J.; Zhu, J.; Hu, Q.; Li, H.; Han, N.; Liu, B.; Zhang, S.; et al. MolMiner: You Only Look Once for Chemical Structure Recognition. *J. Chem. Inf. Model.* **2022**, *62* (22), 5321-5328. DOI: 10.1021/acs.jcim.2c00733.

(20) Qian, Y.; Guo, J.; Tu, Z.; Li, Z.; Coley, C. W.; Barzilay, R. MolScribe: Robust Molecular Structure Recognition with Image-to-Graph Generation. *Journal of Chemical Information and Modeling* **2023**, *63* (7), 1925-1934. DOI: 10.1021/acs.jcim.2c01480.

(21) OpenAI. *ChatGPT(March 23 version) [Large language model]*. 2023. https://chat.openai.com (accessed 2023)

(22) A., C. *Pillow (PIL Fork) Documentation. readthedocs*. 2015. https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf (accessed 2023)

LIST OF ABBREVIATIONS

OCSR	Optical Chemical Structure Recognition
UCT	University of Chemistry and Technology
PDF	Portable Document Format
InChI	International Chemical Identifier
SMILES	Simplified Molecular-Input Line-Entry System
CTab	Connection Table
OCR	Optical Character Recognition
TIFF	Tag Image File Format
JPEG	Joint Photographic Experts Group
GIF	Graphics Interchange Format
PNG	Portable Network Graphics
GUI	Graphical User Interface
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit