universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Component Extraction from Scientific Publications using Convolutional Neural Networks“

verfasst von / submitted by

György Katona

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2019 / Vienna, 2019

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 921
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Informatik
Betreut von / Supervisor:	Univ. Prof. Torsten Möller, PhD

Contents

Abstract

Extracting components from scientific publications is a widely researched information mining problem. We are presenting a component extraction system that is not limited to figures and tables, thereby it is more flexible than already existing solutions. In order to support training process, we introduce our annotation tool for article type documents that is more than 4 times faster than existing tools according to our experiments. After labeling 350 publications from three different scientific fields, we investigate accuracy of our system on three different sets of classes and ran an error analysis to provide help for future improvement. As our system is designed in a way that the image segmentation layer can be replaced, our experiments also cover the comparison of using two different neural network models.

Abstract

Das Extrahieren von Komponenten aus wissenschaftlichen Publikationen ist ein weit erforschtes Informations-Mining-Problem. In dieser Arbeit präsentieren wir ein Komponentenextraktionssystem, das nicht auf Abbildungen und Tabellen beschränkt ist und daher flexibler ist als bereits vorhandene Lösungen. Zur Verbesserung des Trainingsprozesses entwickeln wir ein Annotationstool für Dokumente, das unseren Experimenten zufolge viermal schneller als vorhandene Annotationstools ist. Nachdem wir 350 Veröffentlichungen aus drei verschiedenen wissenschaftlichen Bereichen für das Trainingsverfahren bereitgestellt haben, untersuchen wir die Präzision unseres Systems in drei verschiedenen Klassen und führen eine Fehleranalyse durch, um zukünftige Verbesserungen zu ermöglichen. Da die Image-Segmentation-Schicht in unserem System ersetzt werden kann, vergleichen wir in unserem Experiment zwei unterschiedliche neuronale Netzwerkmodelle miteinander.

1 Motivation

The extensive number of publications available serve as a great library of knowledge in any scientific field. Apart from the textual content, some key components like tables and figures hold major parts of the information contained. These elements serve as concise and effective way of communicating results and showcasing the most important features of the research carried out, therefore extracting them is in a great interest of many researchers. Looking at these key components gives us an overview on the scientific contribution and further analysis of them serves the better semantic understanding.

Most scientific publications are published in PDF format, which is built for optimized presentation, rather than to contain well defined structural information for further processing. These files contain texts, vector and raster graphics with their format and position. Finding the boundaries between elements in separated components is a difficult segmentation problem. As figures are frequently comprised of multiple vector elements, existing tools often extract them separately or only a subset of them. The identification of the component can be also problematic, as a line can be part of a figure, table, mathematical equation or even merely a border on the document's margin.

There are several solutions for the extraction of figures from PDF documents, but the existing approaches face a number of limitations. Most of them are built on algorithmic, rule-based methods identifying regions of the documents, which are then classified into one of the predefined types. As various fields and conferences use different structural formats, in order to maximize performance, most of these solutions are optimized only for a number of conferences or one scientific domain.

The problem of component extraction can be interpreted as an image segmentation problem, therefore Neural Networks also show great potential in providing solution. The volume and quality of the training data has a great contribution in the overall accuracy of the neural network. In order to train a highly accurate model for image segmentation, we need to process a set of documents and annotate the regions of interest that we want the system to recognize. The traditional labeling process is manual; people select the prominent fields of the documents using tools similar to general image editing software. The problem of this approach is that it is not only expensive due to the slow annotation process, but human subjectivity also affects the performance of the final model. In order to maximize the quality of the training data, we should precisely define where is the border between a "figure" and "not figure".

This is not a task for humans, expecting them to use the same number of bordering pixels throughout the annotation of hundreds or thousands of components is unrealistic. There are ways of automatically derive training data, if a Latex or XML file is also provided alongside the PDF, but these methods also face a number of limitations. Although having an automatic way of scraping documents has the advantage of being able to process a great number of publications, available solutions suffer from quality concerns at finding the exact borders and the types of the components are also limited. Users might

be interested in finding components more complicated or abstract than figures or tables, like figures containing a specific element or mathematical equations. In order to train a model which is able to find such components, we are have to use manual labeling during the annotation process.

Our solution for document annotation manages to incorporate the benefits of both manual and automatic labeling. We developed and released an annotation tool specifically designed for article type documents, making region selection semi-automated. By using clustering and post-processing techniques at the time of the annotation, we make labeling faster, more simple and we can even eliminate the uncertainty of the human made selections. Our experiments show evidence of that using this tool makes the labeling process multiple times faster than traditional manual methods. By combining manual labeling with neural networks we are also capable of extracting arbitrarily defined component types.

Our component extraction system uses a Convolutional Neural Network for image segmentation, which has been successfully deployed in many image recognition domains, then in a final post-processing step we use clustering techniques to derive the eventual rectangular component. In Section 4 we investigated the performance of the annotation tool in the terms of required time and difference in the image segmentation accuracy and investigated the whole component extraction system's performance using seven different component types. We also provide an analysis of typical errors, which helps future improvements.

2 Related Work

In this section we are introducing existing component extraction tools (Section 2.1), describe popular tools for image annotation (Section 2.2), then lastly present two image segmentation models (Section 2.3).

2.1 Component extraction

In recent years the problem of extracting non-textual components from scientific publications has been in the interest of many researchers. Traditional methods are rule-based algorithmic approaches, mining structural information from PDF files. Existing solutions focus on the problem of extracting figure and table components including their captions. In order to achieve higher performance, many rule-based methods exploit the typical structure of a conference or a scientific field, which make them less sufficient on general domains. The solution of Praczyk et al. PDFFigures [7] and PDFFigures 2.0 [6] are both focused on conferences in the field of computer science, PDFigCapX [16] is tested on computer science and biomedical publications and there are solutions optimized for high-energy physics as well [18].

Although Machine Learning is highly efficient for many object recognition task, using them on digitally born scholarly articles is relatively new. Machine Learning has been used on the level of elements composing the PDF files, thereby avoiding the need of converting documents to images [5]. In order to train a highly accurate model, it is essential to construct training data with great care. One of the main contributions of DeepFigures [21] was developing a method to automatically generate training data from publications, where either Latex or XML files have been also published alongside the PDF format. This way they were able to train their model on thousands of papers, however their scraped data also suffer from quality concern caused by the undefined borders. Another limitation of this approach is the fact that these Latex and XML files contain only a number of predefined types of components, like figures and tables, therefore identification of more complex or abstract component types is impossible. Other solutions use Neural Networks not for image segmentation, but as a further analysis step. Neural Networks are also used for the classification of previously extracted figures, like recognizing if they contain a face or not [8].

The standard way of evaluating performance is using the measure intersection over union (IOU) with a value of 0.8, meaning that the identification of a component is correct, if the proportion of the intersection and the union of the predicted and ground truth area is more or equal than 0.8. Different structures of different conferences and the fact that publications use different test data makes it difficult to compare the precision and recall rates. Table 1 shows the precision, recall and F1 scores of different solutions of figure extraction. CS-150 is a data set containing 150 publications in the field of computer science, while CS-Large aims to be more diverse and challenging, containing 346 computer science publications. All solutions except FigCapX uses IOU value of 0.8,

FigCapX uses 0.6, which might result in a higher result.

	Precision	**Recall**	**F1**	**Dataset**
Praczyk et al.	0.624	0.500	0.555	CS-150
PDFFigures	0.957	0.915	0.936	CS-150
PDFFigures 2.0	0.936	0.897	0.916	CS-Large
FigCapX	0.935*	0.880*	0.907*	CS-150
DeepFigures	-	-	0.879	CS-Large

Table 1: Precision, recall and F1 scores of different figure extraction solutions. The different datasets make comparison difficult, CS-Large is a larger and more diverse dataset than CS-150, which makes the problem more challenging. All system uses Intersection Over Union value of at least 0.8 to classify an extraction as correct, except of FigCapX, which uses 0.6.

2.2 Image annotation

Image annotation is the process of labeling regions of images to describe what it contains so we can train a model that recognizes these annotated objects. In most cases annotation is done manually by humans, who use either bounding boxes or a geometric shapes to indicate presence of the desired objects. As the volume and quality of the training data is essential in order to train a highly accurate model, image annotation should be done with great care, therefore it is usually a time consuming and costly process. There are multiple commercial tools supporting the annotation process, aiming to make it as user friendly and easy to use, as possible.

Computer Vision Annotation Tool (CVAT) [13], Labelbox [15] and LabelMe [20] all provide multiple options for indicating objects of interests (Table 2). Our tool is different to all solutions mentioned above in that it is focused on fast and precise selection of article type sections, rectangular blocks bordered by white regions. It is a free, open source java application, widely accessible for anyone facing similar challenges.

	Free	**Browser based**	**Collaborative**	**Shapes**
CVAT	Yes	Yes	Yes	polygons, rectangles, lines, points
LabelMe	Yes	No	No	polygons, rectangles, circles, lines, points
LabelBox	No	Yes	Yes	polygons, rectangles, lines, points; brush- and superpixel selected areas

Table 2: Features of popular image annotation tools

2.3 Image segmentation

Image segmentation is the field of Supervised Machine Learning, which divides visual input into segments, which represent objects or parts of objects, then classifies them into meaningful real-world categories. Self driving cars interpreting pictures, by segmenting them into categories like *road*, *car*, *pedestrian* and biomedical diagnostic software classifying *healthy* and *not healthy* cells are both popular topics using image segmentation. In this section we describe two Convolutional Neural Network models we chose to experiment with, as they both achieved great results in image segmentation problems.

2.3.1 U-Net

U-Net [19] architecture was developed for biomedical image segmentation, it won the Dental X-Ray Image Segmentation Challenge and it was among the 4 winners of the Cell Tracking Challenge at the International Symposium on Biomedical Imaging (ISBI) in 2015. It is designed to work with high efficiency even when the size of the training data is limited, as in biomedical environments it is especially difficult to build large training data sets due to its sensitivity by nature. The model's efficiency is also important for our use of component extraction, using a smaller set of training data means that the costs of image annotation can be reduced.

2.3.2 SegNet

SegNet [4] is developed by members of the Computer Vision and Robotics Group at the University of Cambridge, UK. It labels each pixel by using an encoder-decoder architecture: a sequence of non-linear processing layers (encoders) and a corresponding set of decoders followed by a pixelwise classifier. The predictions of the model tend to be smooth, even without a following post-processing step. The developers provide examples of both indoor and outdoor scene understanding.

3 Model Overview

In order to cover the whole preparation and use of our component extraction system, we describe each step separately in the following sections. Figure 1 shows an overview of the entire system. In Section 3.1 we describe the conversion of the PDF documents to PNG images, then we present our annotation tool designed specifically for article type documents in Section 3.2. The masks generated by the annotation tool are used to train the Neural Network component (described in Section 3.3). The last step in our system is the conversion of the pixel prediction given by the Neural Network: in a post-processing step we convert them into rectangular components. This last step is described in Section 3.4.

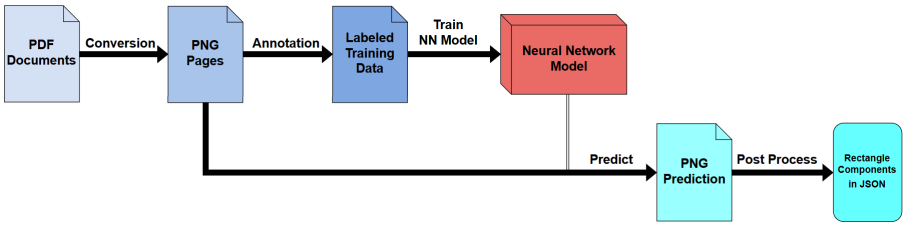


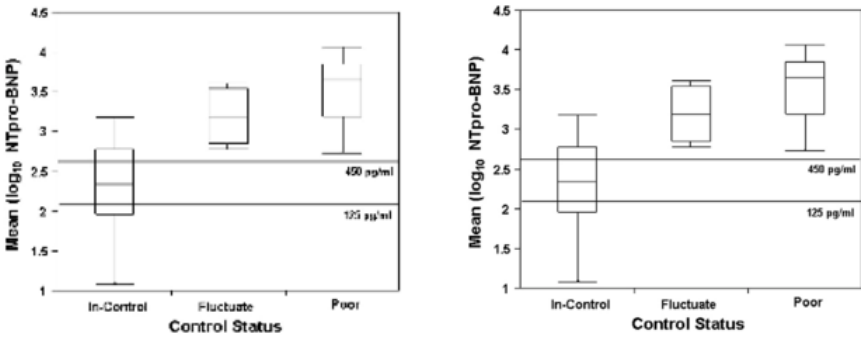
Figure 1: Component extraction model

3.1 PDF to PNG Conversion

In order to train a Neural Network which makes predictions on the component types of documents, the first step is to define the input dimensions. There is a trade-off between resolution and computational time, therefore it is important to find a method which results in a high image quality relative to the fixed resolution. As scientific publications almost exclusively use the page size of A4, we have chosen 724x1024 resolutions, where the document is still readable with a high conversion quality. By choosing a resolution too low, we could lose important information, but after a certain point working with higher resolution only results in a longer training process without incorporating any additional information. For making the conversion we used the popular open-source Linux tool, ImageMagick [12].

```
convert -verbose          \
        -density 300      \
        ${pdfile}"        \
        -quality 100      \
        -sharpen 0x1.0    \
        -resize 724x1024  \
        -gravity center   \
        -background white \
        -extent 724x1024  \
        -alpha remove     \
        "${pdfile%.*}".png
```

Choosing the right conversion parameters makes a great difference in the quality of the outcome. Figure 2 shows the result of making the conversion without any custom parameters on the left, and using the optimized parameters on the right.



Side-by-side box plots of the average $\log_{10}$ of NT-proBNP by control status group ("In Control", n = 20; "Fluctating", n = 4; "Poor Control", n = 6). Vertical lines indicate the range; the boxes, the 25th to the 75th percentiles; and the lines in the box, the medians. The reference lines at 125 and 450 pg/ml indicate the upper limit of normal and the diagnostic value for HF, respectively.

Side-by-side box plots of the average $\log_{10}$ of NT-proBNP by control status group ("In Control", n = 20; "Fluctating", n = 4; "Poor Control", n = 6). Vertical lines indicate the range; the boxes, the 25th to the 75th percentiles; and the lines in the box, the medians. The reference lines at 125 and 450 pg/ml indicate the upper limit of normal and the diagnostic value for HF, respectively.

Figure 2: Difference of choosing the right conversion parameters - result of using no custom parameters on the left, optimized parameters on the right

3.2 Document Annotation

Manual image annotation tools aim to provide a fast and efficient way of defining regions using rectangular boxes, polygons, polylines and points. Tools like Computer Vision Annotation Tool (CVAT) [13], Labelbox [15] and LabelMe [20] are widely used, user friendly solutions of producing training data for neural networks. One of the most useful tools provided by these programs is a smart selection tool, similar to Photoshop's magic wand, which helps to select a similarly colored area by a single click. Unfortunately however, these tools are not working properly in the context of digitally born textual documents, where the regions of interests are not separated by the contrast in color: the main

content of these documents are texts, which are a mixture of black pixels of the texts on white background pixels. Our goal therefore is to design an annotation tool specifically for images of textual documents, which makes it possible to select regions by singular clicks. Such a tool would make the annotation process much faster and thereby also less expensive. Making the annotation semi-automated might also increase the performance of the model built upon the produced labelled data, as defining borders would no longer be a subject of human subjectivity.

3.2.1 Smart Rectangular Selection

Scientific publications use different structures but they all of them can be divided into rectangular blocks. Both one and double column documents use a combination of text, figure, table, algorithm, formula and reference blocks, separated by whitespace padding. During our first manual labeling process, we noticed that selecting the exact borders of the components took a great amount of time and focus. It is necessary to decide the "right amount of borders" before labeling, especially if it is done by multiple people. By using human resource it is unrealistic to agree on a 0 pixel border width, as it would need absolute pixel correctness. The same argument can be made with any specific number of pixels. The best solution for the problem is to define the border as "as small as possible, but it must contain all pixels of the component in interest". The problem now is that border width still varies, which might affect the quality of the final neural network predictions, and achieving a lower error still requires a great level of concentration which extends labeling time and increases costs.

The most simple solution to this problem is to require the labeling person only to select the component roughly, where the width of the whitespace border does not matter, then remove all rows and columns on the edges which contain white pixels only (Figure 3). After finding the non-white borders, it is also possible to extend the padding with a specific number of pixels, the goal is to use the same border width for all the selected components.

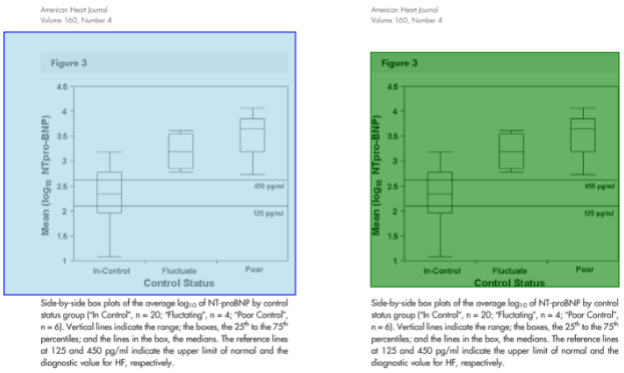


Figure 3: Smart rectangular selection - our tool can eliminate all white rows and columns on the borders in order to support precise labeling

3.2.2 Smart Clique Selection

In order to make the labeling process even faster, we developed a one-click solution of selecting rectangular components. The ideal solution of defining regions is that the user only clicks in the middle of the component, then the smart selection tool automatically finds the edges. As the components are bounded by whitespace regions, the problem can be interpreted as a clustering problem in the two dimensional pixel space, where data points are the non-white pixels. Using a well defined tolerance value, which defines the minimum space between two separated clusters (components) results in an optimal solution. However, as the optimal value depends not only on the structure of the publication, but also on the resolution of the image, we made a user interface which makes it easy to modify this value.

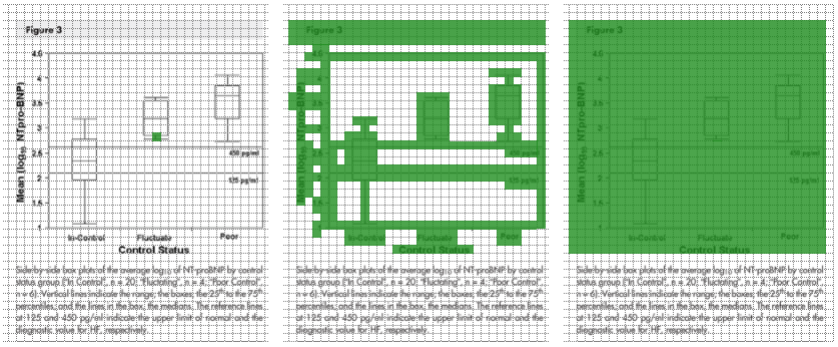


Figure 4: Smart CLIQUE selection finding connected units containing non-white pixels, thereby making one click selection possible

Our solution is using a similar approach to CLIQUE [2], it divides the docu-

ment into a grid, consisting of square shaped units of a size based on the tolerance value. Clicking on one of the units selects the largest connected component, separated by units containing white pixels only (Figure 4). We recommend using a smaller tolerance value and if the automatically selected region is only a part of the full component, the user can hold the control button while clicking on the unselected region of the same component in order to join the two clusters as one selected rectangular region (Figure 5).

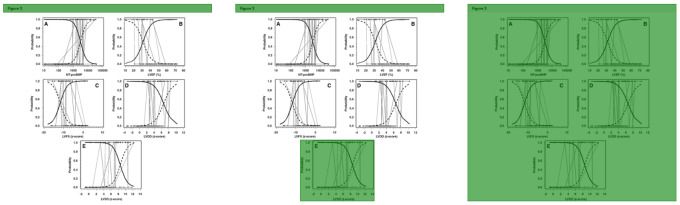


Figure 5: Joining sub-components by clicking and holding control key

3.2.3 Implementation

Our image annotation tool, DocLabels [14] is a free, open source java application focused on the annotation of article type documents. Using the software consists of three steps, importing images, annotation itself and exporting the output files. The user can select multiple images in a popup window, after clicking on the "Open Images" button, then "Next" and "Previous" buttons or the arrow keys help navigating between them. Annotation is done page-by-page with the tools described in section 3.2.1 and 3.2.2. Smart CLIQUE is done by simply clicking on a non-white area, while the smart rectangular selection is by dragging the mouse over the area to select. Omitting whitespace regions on borders at the rectangle selection and the helping axis lines are both optional features, they can be turned off using the user interface. To make the annotation process faster, class id-s can be selected not only by clicking on a spinner, but also using the mouse wheel or the up/down arrow keys. Different class types are displayed by different color for easier error detection (Figure 6). If an error still occurs, undo can be done by a simple right click.

Figure 6: Document annotation using DocLabels. Using different colors for different classes help error prevention and detection.

DocLabels exports the components using two formats. One is an ImageMagick script that produces mask images with the same resolution as the labelled images, using the blue RGB channel as class id. The other output is a JSON file, containing the original resolution and rectangular component positions along with their class ids. Using this JSON format makes it easier to evaluate predictions, while mask images are used as an input for training the Neural Network model.

3.3 Neural Network

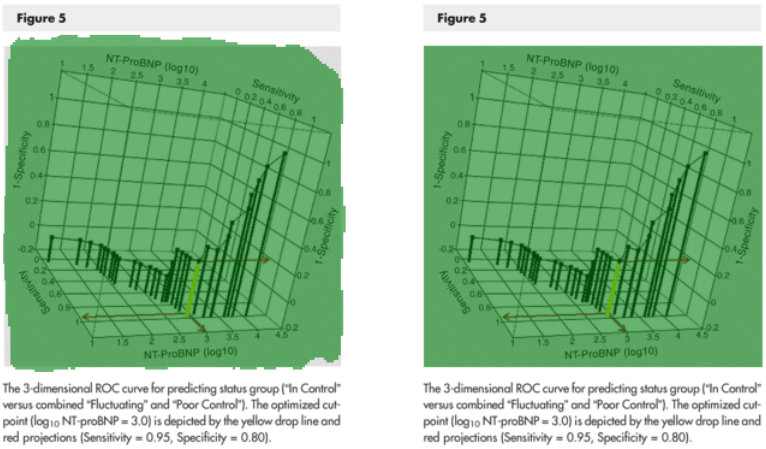
Our component extraction system is designed in a way, that the Neural Network model is an independent layer, used for image segmentation, therefore it is easy to replace or to investigate performance of different models. In our experiments in Section 4, we inspected the results of two models: U-Net [19] and SegNet [4]. We chose these two models due to their great results in other contexts and to investigate if there is significant difference in their performance.

Training the models requires the original images of the pages along with their masks, which contain class ids for each pixel. To avoid overfitting, we train as long, as both training and validation accuracy increases. On the output, the Neural Network model produces a pixelwise prediction, which is then used in the following post-processing step.

3.4 Post-Processing

We use a final post-processing step in order to convert pixel predictions to rectangular components. As dense regions in the two dimensional pixel-space indicate high probability of components, we run DBSCAN clustering [9] to find

clusters of pixels with the same predicted class id, then find the smallest rectangle that encapsulates them. Components can be interpreted as a number of colored pixels, bordered by white regions, therefore we eliminate all rows and columns containing only white pixels in the original image - just like we did in the annotation tool. This step makes sure that in the case of a slight uncertainty in the prediction of the Neural Network, the resulting rectangular component is still perfectly accurate (Figure 7).



The 3-dimensional ROC curve for predicting status group ("In Control" versus combined "Fluctuating" and "Poor Control"). The optimized cut-point ($\log_{10}$ NT-proBNP = 3.0) is depicted by the yellow drop line and red projections (Sensitivity = 0.95, Specificity = 0.80).

The 3-dimensional ROC curve for predicting status group ("In Control" versus combined "Fluctuating" and "Poor Control"). The optimized cut-point ($\log_{10}$ NT-proBNP = 3.0) is depicted by the yellow drop line and red projections (Sensitivity = 0.95, Specificity = 0.80).

Figure 7: Performing post-processing step of clustering segmented pixels, finding smallest encapsulating rectangle and removing all-white rows and columns on the edges

4 Experiments

In this section we are presenting the results of all our experiments. In Section 4.1 we introduce all datasets that have been used during the analysis. To investigate the efficiency of our annotation tool, we have run a pilot user test and compared its performance to one of the popular labeling tools in Section 4.2. All component segmentation and extraction results are presented in Section 4.3, including experiments with multiple component type settings and applying multiple accuracy measures. To provide help for further improvement, we have also run an error analysis which is presented in Section 4.4.

4.1 Data Sets

Table 3 shows a summary of the three datasets that we labelled and used in our experiments. VIS-CHI-100 contains 100 scientific publications - 50 from IEEE's VIS [11] and 50 from ACM's CHI [1] conference from the period between 2009 and 2018. It has 903 pages labelled with 8 classes: *Text*, *Figure*, *Table*, *Reference*, *Algorithm*, *Formula*, *Picture* (an image that is not indicated as Figure) and *Background*. There are 9309 *Text*, 666 *Figure*, 90 *Table*, 251 *Reference*, 20 *Algorithm*, 154 *Formula* and 11 *Picture* components.

We have also built a dataset of scientific papers from the biomedical field. PUBMED-100 consists of 100 publications from 2009 to 2018, 10 publication randomly selected for each year. In this dataset we manually annotated 223 *figures* on 772 pages.

CS-150 is a dataset introduced by Clark et al. [7] which consists of publications in the field of computer science from between 2010 and 2014. It contains 50-50 publications from AAAI [3], ICML [10] and NIPS [17] conferences, manually labelled with the classes of *figure* and *table*.

Dataset	**Publications Included (Pages)**	**Release Period**	**Field**
VIS-CHI-100	100 (903)	2009-2018	Visualization, Human-Computer Interaction
PUBMED-100	100 (772)	2009-2018	Medical Biology
CS-150	150 (1176)	2010-2014	Computer Science

Table 3: Summary of the used datasets

4.2 Annotation Efficiency

One of our main contributions is the manual labeling technique we developed and the annotation tool implemented on top of it. In order to investigate the difference of using traditional techniques and our approach specialized for article type documents like scientific publications, we have prepared and run a pilot

user test with 4 users. We selected Labelbox [15] as a reference tool, as it is a popular choice for image annotation chosen by companies like Airbus or FLIR Systems. Each of our test users got an introduction to the problem, then learned the controls and got comfortable with using the tool by labeling 2 publications. After the tutorial phase, we measured the time of annotating all *figures* in 5 publications (50 pages) using one tool, then we repeated the whole process using the other tool. The users were asked to make reasonably accurate annotations with all non-white pixels of the figure included in the rectangular area.

We selected two subsets of the dataset VIS-CHI-100 publications for the two iterations of the tests. To avoid bias caused by the differences in the data, we used them equally with the two tools and we also alternated which tool was introduced to the user first. The users were all new to the problem of image annotation but they use computers in their work on a daily basis. The two male and two female participants are in between 25 and 60 years old.

The measured annotation times using our tool was multiple times faster for each user than the same process using Labelbox. Figure 8 shows the measured times that was required to label each dataset using each tool. The average annotation speed was 4.44 times faster using our tool. As our labeling technique is designed to find the perfect borders of rectangular components formulated by non-white pixels, the labelled regions produced by our tool are perfectly accurate. In contrast, perfect consistency is not achievable using traditional tools, regardless how much more time we might allocate.

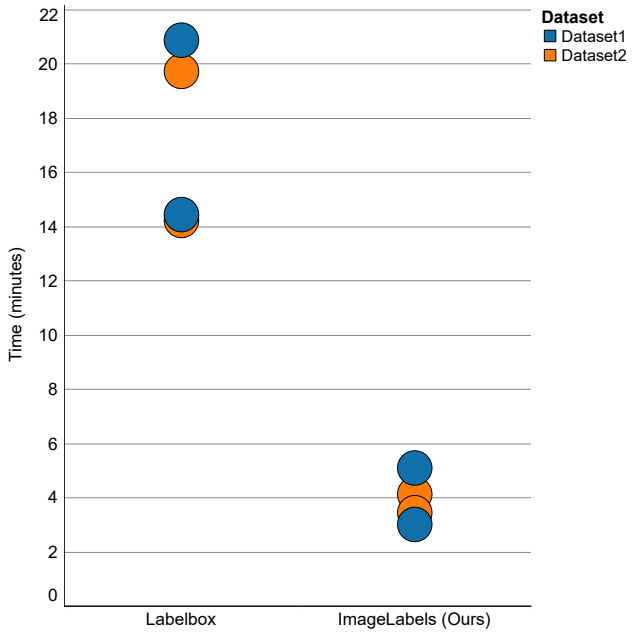


Figure 8: Times used for annotating *figure* components of five publications using Labelbox and ImageLabels - an annotation tool implemented by us

Running a user test with only 4 users already shows the benefits of using our annotation tool for article type documents, but to get a better overview of the exact difference and benefits, we should run a user test with an increased number of users and multiple labeling tools.

4.3 Component Segmentation and Extraction

In this section we are presenting four experiments. In Section 4.3.1 we investigate pixel accuracy of segmenting 8, 4 and then 2 classes. Each of these experiments use the dataset VIS-CHI-100 and we run all setups both with U-Net and Segnet neural network models in order to get an insight into the effects of using different models. In Section 4.3.2 we use the metric Intersection Over Union (IOU) to evaluate the accuracy of extracting figures from three different datasets: VIS-CHI-100, PUBMED-100 and CS-150.

4.3.1 Pixelwise Evaluation

Table 4 and Figure 9 present precision, recall and F1-score results of segmenting publications from VIS-CHI-100 into 8 different classes, without applying the post-processing step. As the segmentation means predicting the class of each individual pixel, the numbers in the table are based on pixelwise evaluation.

	Unet			Segnet		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Reference	0.971	0.978	0.975	0.970	0.977	0.974
Text	0.956	0.984	0.970	0.964	0.978	0.971
Background	0.968	0.955	0.961	0.962	0.959	0.960
Figure	0.889	0.914	0.902	0.885	0.925	0.905
Picture	0.957	0.703	0.810	0.698	0.868	0.773
Formula	0.765	0.523	0.622	0.666	0.501	0.572
Table	0.610	0.439	0.510	0.774	0.512	0.616
Algorithm	0.407	0.020	0.037	0.449	0.049	0.089

Table 4: Results of identifying 8 classes on the pixel level - U-Net vs Segnet models

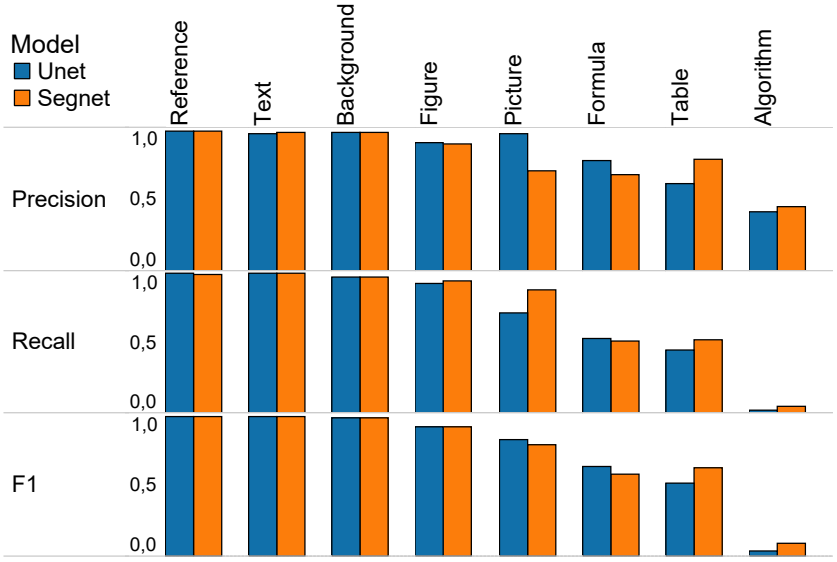


Figure 9: Differences in accuracy of U-net and Segnet neural network models segmenting 8 classes

Comparing segmentation results of the system using U-Net and Segnet shows us that there are only small differences in the accuracy when using these two models. On the other hand the accuracy of segmenting different classes are highly divergent. The F1 score of *Text*, *Reference* and *Background* segmentation is above 96%, and the F1 score of *Figure* is also around 90%. The segmentation of *Algorithm* was the most difficult task for both models, it is the only class with F1 score below 50%. Segnet was more efficient in segmenting *Tables*, while U-net performed better for the class *Formula*. The segmentation of *Pictures* is higher in precision using U-Net, but Segnet has a better recall rate.

In the next experiment, we investigated the performance of both U-Net and Segnet models with a smaller number of classes. Table 5 and Figure 10 show segmentation results with merged classes. As *Algorithm* and *Formula* was difficult to distinguish from *Text* and could be interpreted as a specific subclass of it, we merged them into one class. The class *Picture* in the VIS-CHI-100 dataset was only used for the same logo in 11 publications, therefore we decided to investigate performance when we do not distinguish it from the *Background*. Tables and Figures are arguably the most important non-textual component types, therefore extracting them with high accuracy is greatly desirable. We decided to merge these classes to see if it results in a better precision and recall. If the segmentation of the joint classes can be done with high accuracy, then the classification of the extracted component might be an easier task than using all classes at once.

	Unet			**Segnet**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Reference	0.965	0.980	0.972	0.987	0.966	0.976
Text U Algorithm U Formula	0.961	0.971	0.966	0.965	0.966	0.966
Background U Picture	0.966	0.957	0.961	0.963	0.960	0.961
Figure U Table	0.917	0.904	0.910	0.904	0.920	0.912

Table 5: Results of identifying 4 classes on the pixel level - U-Net vs Segnet models

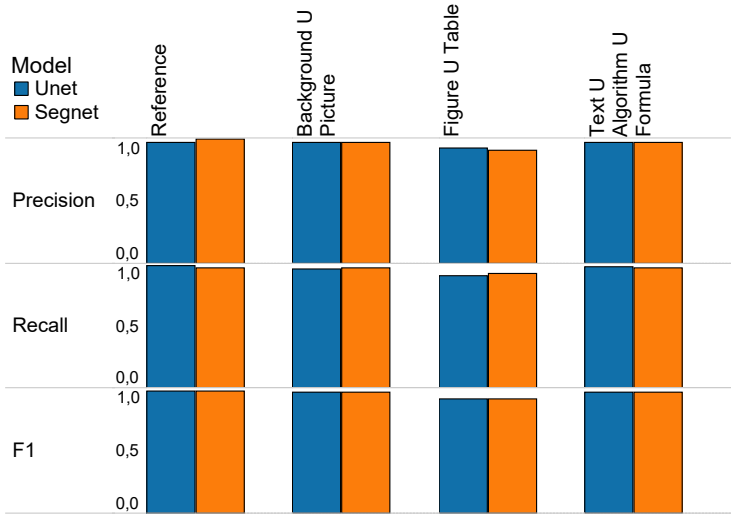


Figure 10: Differences in accuracy of U-net and Segnet neural network models segmenting 4 classes

Segmenting four classes also shows only small differences in the performance of U-Net and Segnet models. The joint *Figure U Table* class has now the lowest F1 score, but still around 91%. The F1 score of all other classes are higher than 96%.

	Unet			**Segnet**		
Post-Proc.	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Yes	0.881	0.991	0.933	0.874	0.979	0.923
No	0.899	0.923	0.911	0.899	0.915	0.907

Table 6: Figure segmentation results with- and without post-processing, evaluated on the pixel level

Table 6 and Figure 11 show the benefit of using our post-processing, evaluated on the pixel level. It manages to increase the recall rate by more than 6% for the cost of only around 2% in the precision score.

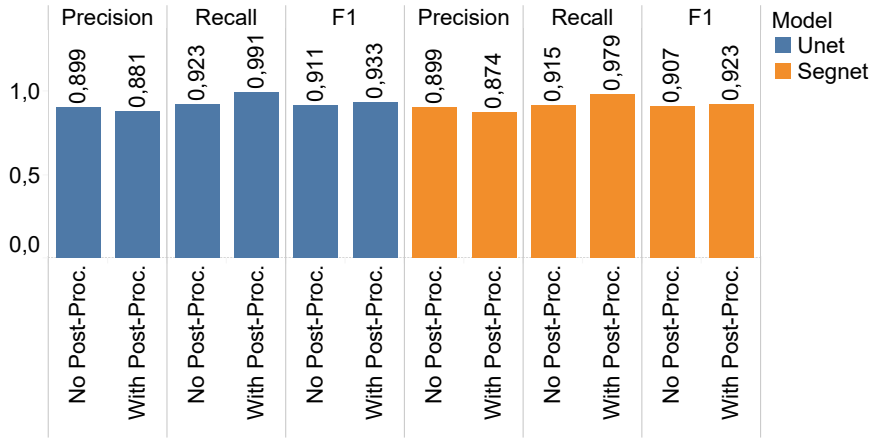


Figure 11: Differences in accuracy before and after the post-processing step. The main reason of post-processing is to convert pixel predictions to rectangular components, yet it achieves a higher recall rate and F1 score.

4.3.2 Intersection Over Union

Another way of evaluating performance is by using the concept Intersection Over Union (IOU). Most of the prior works addressing the problem of figure extraction use IOU value of 0.8, meaning that the extraction is considered correct if IOU of the ground truth and the predicted value is higher or equal to 0.8. In order to support comparison to other methods, we also consider an extraction correct only if IOU is at least 0.8.

$$IOU = \frac{ground\_truth \cap prediction}{ground\_truth \cup prediction}$$

The last post-processing step produces the rectangular components which are ready for the extraction. Table 7 and Figure 12 show precision, recall and F1 scores of figure extraction trained and tested for three datasets of different contexts. In these experiments we used epsilon=10 and min_sample=250 pixels as parameters for the DBSCAN clustering algorithm of the post-processing step. Choosing these parameters are greatly dependent of the page resolution, we decided to choose a relatively small epsilon which could separate two independent components right next to each other, and a min_sample rate that requires an at least around 80% dense region of pixels classified as *Figure*.

	U-Net			SegNet		
Dataset	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
VIS-CHI-100	0.710	0.765	0.736	0.662	0.748	0.702
CS-150	0.718	0.689	0.703	0.700	0.662	0.681
PUBMED-100	0.208	0.412	0.276	0.766	0.706	0.735

Table 7: Figure extraction results - an extraction is considered correct, if Intersection Over Union is greater than or equal to 0.8

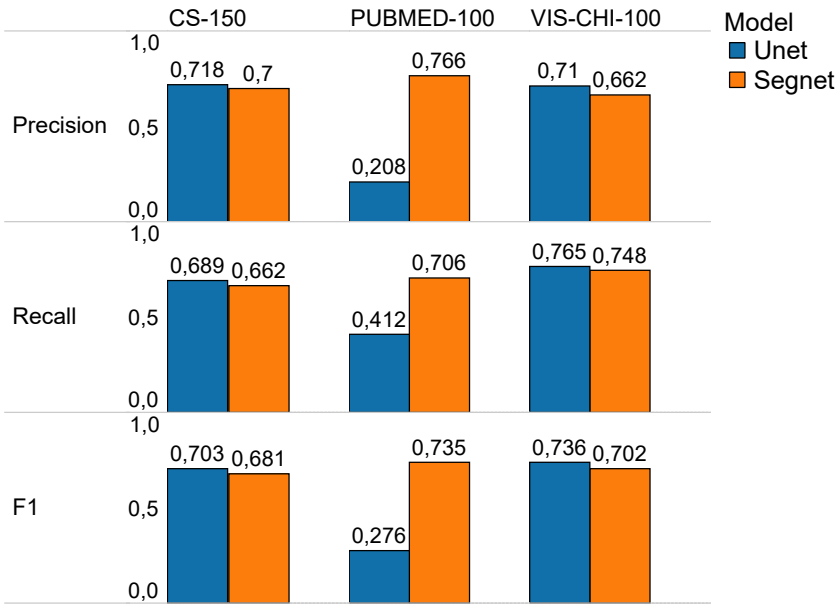


Figure 12: Differences in accuracy of figure extraction from different datasets. Although U-net is slightly better on CS-150 and VIS-CHI-100, Segnet has a much better performance on the PUBMED-100 dataset.

In this last experiment, U-Net and Segnet produced largely different results, especially when they were applied at the dataset PUBMED-100. Figure extraction from this dataset was way more efficient using the Segnet neural network model, but on VIS-CHI-100 U-Net was able to produce better results both in recall and precision.

4.4 Error Analysis

In this section we are investigating the results of our experiments, which classes were difficult for the system to distinguish and what could be the main reasons behind the errors. The goal of this analysis is to provide help for further improvement.

In Figure 13 each sub-figure shows the error rates of classes falsely predicted for a specific class using either U-Net or Segnet. In the cases, where the class *Background* has the highest error rates, the most difficult problem of the segmentation was to find the ground truth border of the component. These classes are *Text*, *Figure* and *Reference* for both neural network model, and *Formula* at the Segnet experiment. On total, more than 27% of *Table* pixels were falsely predicted as *Figures* for both models, which indicates, that the class *Table* is difficult to distinguish from *Figures*. *Algorithm* turned out to be the most difficult class to recognize, more than 50% of *Algorithms* were predicted as *Figures*. The class *Picture* is also problematic, as it was often confused with *Figures* and *Tables*, however the class *Picture* is only used for a background logo of a conference, while *Figures* and *Tables* are important source of information. For this reason *Pictures* were merged with *Background* for the next experiment, involving 4 segmented classes.

The 4 class image segmentation error rates are shown in Figure 14. Average error rates are much lower in these experiments, which means that these four classes were defined in a way that the system was able to find the important features that distinguish them. On the other hand, segmenting only four classes naturally provides us less information, for example using this model might be only a first step if we are also interested in further classification of the class *Figure U Table*. The class Reference was hardly ever confused with any other classes using both models, and in general the *Background* class was the only class with a higher error rate, than 2%. Having this error rate somewhat higher is also expected, as finding the perfect rectangular boundaries for each component type is difficult for these article type documents, where there are no differences in neither color or contrast that would indicate the edges of components.

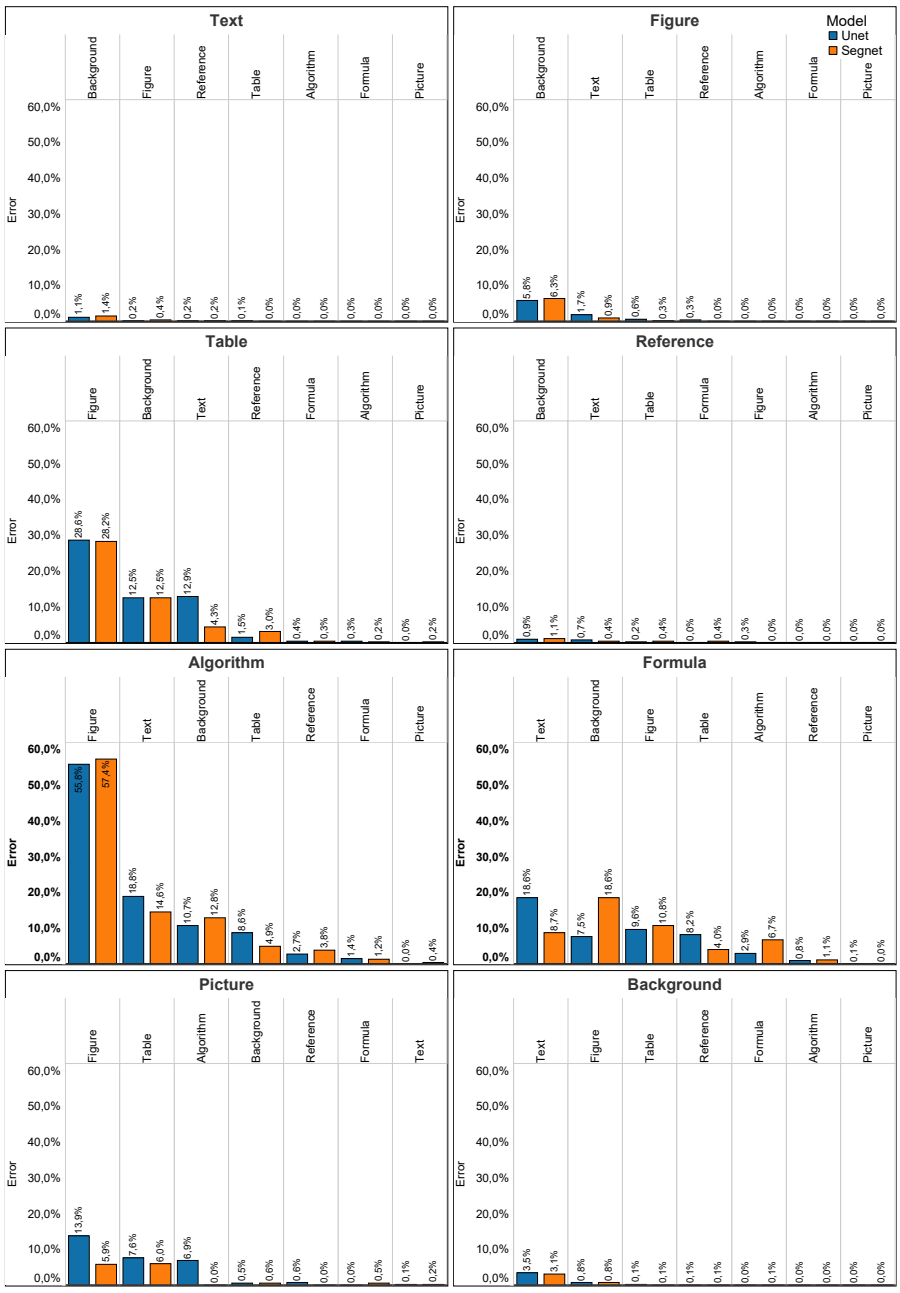


Figure 13: Classification errors using 8 classes

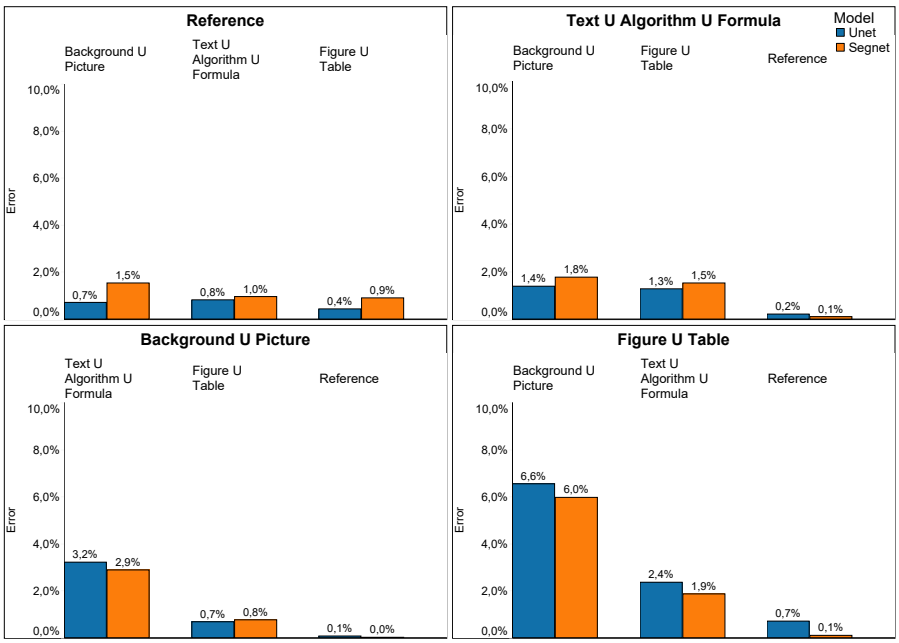


Figure 14: Classification errors using 4 classes

The last experiment was extracting rectangular *Figure* components from three different datasets and accepting components only if their Intersection Over Union score is at least 0.8. Here we are analyzing the main causes of errors, which was carried out by manually investigating and counting errors.

The first issue we are addressing is the problem of "bordering figures". There are cases, where it is difficult to decide if two or more components right next to each other are constructing one large or multiple individual components, without analyzing the textual clues and labels. Another problem is the inconsistency over the use of Figures, tables or any other component types. Writers of different scientific fields and conferences have different styles and sometimes they use atypical components. During our manual labeling process we have seen Tables marked as Figures and components like table of figures which are difficult to classify right. We refer to errors rising from this issue as "atypical components". There are also cases, where the image segmentation identifies the correct region, but the accuracy is not perfect. We refer to extracted components with Intersection Over Union less than 0.8 but larger than 0.6 as "inaccuracy" errors. Some solutions like FigCapX [16] use 0.6 as the primary decision boundary, therefore we decided to investigate the number of errors in this range.

In this analysis we are focusing on those components that the system was not able to extract. Table 8 and Figure 15 show the proportion of the defined error types of these components. As we saw in Table 7, U-Net worked slightly better than SegNet for VIS-CHI-100 and CS-150, but had major drawback at figure ex-

traction from PUBMED-100. According to our analysis, the most problematic issue of extracting all figures from a paper is due to the difficulty of distinguishing large components including sub-figures from separated components being right next to each other. One way of resolving this problem would be to combine image segmentation techniques with text analysis. As most already existing figure extraction solutions build up from finding keywords like "Figure", they are less effected by this problem, the number of components to extract is already given. Finding an efficient way of combining these methods could result in a robust and accurate solution. Our analysis also shows that different datasets might need a slightly different approach, in CS-150 a significant amount of errors were caused by that all text components, like tables and algorithms were also labelled as Figures by the authors. These textual figures are difficult to detect for component extraction systems building on image segmentation.

Dataset	Model	Bordering Figures	Atypical Comp.	Inaccuracy	Other
VIS-CHI-100	U-Net	81.5%	0%	14.8%	3.7%
VIS-CHI-100	SegNet	69.0%	0%	24.1%	6.9%
PUBMED-100	U-Net	51.6%	0%	9.7%	38.7%
PUBMED-100	SegNet	80.0%	0%	20.0%	0%
CS-150	U-Net	47.8%	17.4%	8.7%	26.1%
CS-150	SegNet	53.8%	26.9%	3.9%	15.4%

Table 8: Error types behind not being able to extract figure components

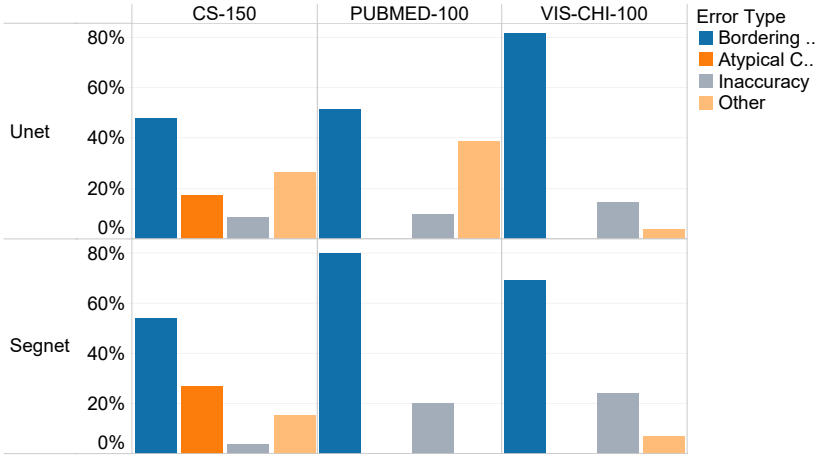


Figure 15: The proportion of different error types (Bordering Figures, Atypical Component, Inaccuracy and Other) in three different datasets, using using U-net and Segnet neural network models

5 Conclusion

We presented a component extraction system that is built upon image segmentation, thereby it is able to extract components of multiple types. In order to support the training process, we introduced a new annotation tool for article type documents, and ran a pilot user test to demonstrate the benefits of using it over already existing solutions. We used the same tool to annotate three datasets from different fields, which consists of 350 publications and more than 2500 pages in total. We investigated the performance of our segmentation and extraction system by running experiments with different sets of classes over these three datasets using two different neural network models for image segmentation. Finally we ran an analysis on the segmentation errors, to examine the weaknesses of our system and to provide help for future improvements.

Future work concerns improving extraction accuracy based on our error analysis: our experiment implies that a great improvement can be made by handling the problem of components being right next to each other. For ordinary components such as *Figure* and *Table*, the analysis of the PDF's textual content could be promising, as it could help to determine the number of components on a page. We are also considering further investigation and development of our annotation tool such as publishing a web-based version in order to provide an accessible solution for higher audiences.

References

[1] ACM SIGCHI. Acm conference on human factors in computing systems. `https://chi2018.acm.org/`. Accessed: 2019-07-23.

[2] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1998), SIGMOD '98, ACM, pp. 94–105.

[3] Association for the Advancement of Artificial Intelligence. Aaai. `https://www.aaai.org/`. Accessed: 2019-07-23.

[4] Badrinarayanan, V., Kendall, A., and Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[5] Choudhury, S. R., Mitra, P., and Giles, C. L. Automatic extraction of figures from scholarly documents. In *Proceedings of the 2015 ACM Symposium on Document Engineering, DocEng 2015, Lausanne, Switzerland, September 8-11, 2015* (2015), pp. 47–50.

[6] Clark, C., and Divvala, S. Pdffigures 2.0: Mining figures from research papers. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries* (New York, NY, USA, 2016), JCDL '16, ACM, pp. 143–152.

[7] Clark, C. A., and Divvala, S. K. Looking beyond text: Extracting figures, tables and captions from computer science papers. In *Scholarly Big Data: AI Perspectives, Challenges, and Ideas, Papers from the 2015 AAAI Workshop, Austin, Texas, USA, January, 2015.* (2015).

[8] Dawson, M., Zisserman, A., and Nellker, C. Mining faces from biomedical literature using deep learning. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB 2017, Boston, MA, USA, August 20-23, 2017* (2017), pp. 562–567.

[9] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, pp. 226–231.

[10] ICML Organizing Committee. International conference on machine learning. `https://icml.cc/`. Accessed: 2019-07-23.

[11] IEEE Computer Society Visualization and Graphics Technical Committee. Ieee visualization conference. `http://ieeevis.org/`. Accessed: 2019-07-23.

[12] ImageMagick Studio LLC. Imagemagick. https://imagemagick.org/. Accessed: 2019-07-16.

[13] Intel. Computer vision annotation tool. https://github.com/opencv/cvat. Accessed: 2019-07-16.

[14] Katona, G. DocLabels: Image Annotation for Article Type Documents. https://github.com/georgekatona/DocLabels, 2019.

[15] Labelbox, Inc. Labelbox. https://labelbox.com/. Accessed: 2019-07-16.

[16] Li, P., Jiang, X., and Shatkay, H. Extracting figures and captions from scientific publications. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018* (2018), pp. 1595–1598.

[17] NIPS Organizing Committee. Neural information processing systems. https://nips.cc/. Accessed: 2019-07-23.

[18] Praczyk, P., and Nogueras-Iso, J. A semantic approach for the annotation of figures: Application to high-energy physics. In *Metadata and Semantics Research - 7th Research Conference, MTSR 2013, Thessaloniki, Greece, November 19-22, 2013. Proceedings* (2013), pp. 302–314.

[19] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III* (2015), pp. 234–241.

[20] Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision 77*, 1-3 (2008), 157–173.

[21] Siegel, N., Lourie, N., Power, R., and Ammar, W. Extracting scientific figures with distantly supervised neural networks. In *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries* (New York, NY, USA, 2018), JCDL '18, ACM, pp. 223–232.