Project Number: **770299**

NewsEye:

A Digital Investigator for Historical Newspapers

Research and Innovation Action
Call H2020-SC-CULT-COOP-2016-2017

D2.4: Layout Analysis (final)

Due date of deliverable: M24 (30 April 2020)

Actual submission date: 08 April 2020

Start date of project: 1 May 2018 **Duration:** 36 months

Partner organization name in charge of deliverable: UROS

Project co-funded by the European Commission within Horizon 2020		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	-
RE	Restricted to a group specified by the Consortium (including the Commission Services)	-
CO	Confidential, only for members of the Consortium (including the Commission Services)	-

Revision History

Document administrative information		
Project acronym:	NewsEye	
Project number:	770299	
Deliverable number:	D2.4	
Deliverable full title:	Layout Analysis (final)	
Deliverable short title:	Layout Analysis (final)	
Document identifier:	NewsEye-T21-D24-LayoutAnalysis-Submitted-v3.0.pdf	
Lead partner short name:	UROS	
Report version:	V3.0	
Report preparation date:	08.04.2020	
Dissemination level:	PU	
Nature:	Report	
Lead author:	Max Weidemann (UROS)	
Co-authors:	Bastian Laasch (UROS)	
Internal reviewers:	Sébastien Cretin (BNF), Günter Hackl (UIBK-DEA)	
Status:		Draft
		Final
	x	Submitted

The NewsEye Consortium partner responsible for this deliverable has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

Change Log

Date	Version	Editor	Summary of changes made
16/03/2020	0.1	Max Weidemann (UROS)	First full draft
17/03/2020	1.0	Max Weidemann, Bastian Laasch (UROS)	Minor changes prior to reviews
24/03/2020	2.0	Max Weidemann (UROS)	Taking reviewer comments into account: minor modifications regarding spelling, expression and layout
08/04/2020	3.0	Max Weidemann (UROS), Antoine Doucet (ULR)	Minor modifications ahead of submission

Executive summary

This report describes the first step of the information processing pipeline of work packages 2–5, the layout analysis. Specifically, the baseline detection problem is described and tackled as a first important step for automated text recognition and article separation. In addition we tackled the text block detection problem in year 2 and show the achieved results. In order to make these processes as automatic as possible, mathematical models from the field of machine learning are applied and combined with state-of-the-art image processing based techniques. These models involve special deep artificial neural networks designed for image processing tasks. The algorithms are tested and evaluated for a given set of newspaper pages.

This document presents the outcomes of layout analysis research as a whole, including information from previous Deliverable D2.1. As a result, we added Sections 3 to 6 and made some minor changes in the remaining text like updating the results for the baseline detection experiments in Section 2.4.

Comparing our models from year one and year two, we improved the F-Value from 96.17% to 97.65%, which makes a relative improvement of 1.54%. We also compared ourselves to the OCR engine ABBYY, which resulted in a relative improvement of 3.96%, from 93.93% to 97.65%. When interpreting the F-Value in terms of an error measure, i.e. using (1 - F-Value), the relative improvements respectively reach 38.64% and 61.29%.

Contents

1 Introduction

The first newspapers were printed in the early 17th century. Since then, their number is growing steadily, holding information to cultural, political and social events in almost every language. As a consequence, tens of millions of newspaper pages from European libraries have been digitized in the last decades. To extract the key information from these files automatic processing pipelines are necessary, i.e. not relying on handcrafted rules, to handle a broad class of documents. This is one of the main goals of the Horizon 2020 NewsEye project, including layout analysis (LA), automated text recognition (ATR)[1], article separation (AS), named entity recognition (NER), topic modeling (TM) and more. Work package (WP 2) comprises the first three topics, where the LA part is described in this deliverable.

This document is organized as follows: In Section 2 we describe the baseline detection problem which forms the basis for the AS in Task 2.3 and the text line extraction which is used by the ATR models in Task 2.2. In Section 2.4 we present the results achieved on four newspapers from the Austrian National Library (ONB)[2] comprising 232 pages. Section 3 introduces the text block detection problem and proposes two methods to solve it. Instead of training the machine learning models on whole images we experimented with approaches working on patches of images in Section 4. This makes it possible to use other algorithms for larger input images like the capsule networks introduced in Section 5. In Section 6 we compare our algorithms to outputs generated from OCR engines like ABBYY. We conclude the deliverable with a brief overview of the hardware and tools being used in Section 7 and give a conclusion in the last section. This deliverable is the second and last output of Task 2.1, whose initial description from the Description of action is copy-pasted below.

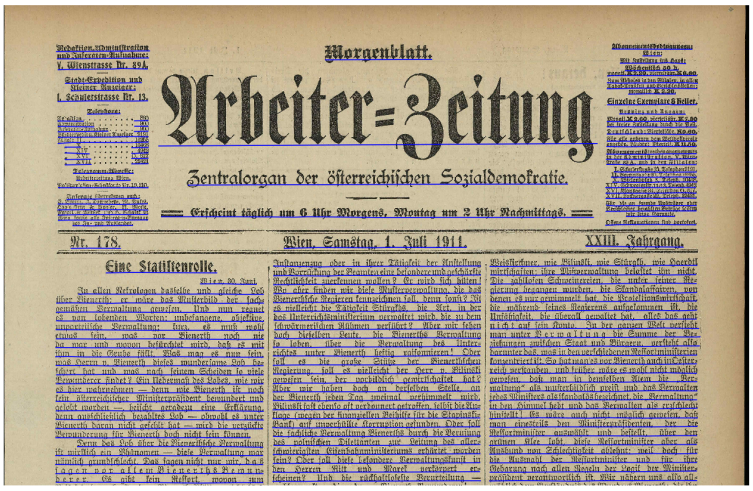
Task 2.1 – Layout Analysis (LA)

In this task we will develop algorithms and methods, as well as implement and deliver tools for automatic segmentation of digitized page images that result in text line images. Furthermore, the tools to be developed will result in first, rather coarse, structuring of the digitized page into text blocks. The work will follow two basic directions: (a) Image processing based methods will be fine-tuned/adapted and used on NewsEye historical newspapers. In collaboration with UIBK (cf. WP1), we will compile a set of basic layouts or layout properties in order to then apply neural models for classifying newspaper pages. (b) Machine Learning approaches, i.e. an end-to-end technology relying on Neural Networks (NNs), will be trained by appropriate Ground Truth (cf. WP1). This task comprises further research on how to extend recently proposed methods for the text line detection problem solely based on Machine Learning techniques like Convolutional or Recurrent Neural Networks under challenging conditions of varying newspaper layouts with a huge number of text lines.

2 Baseline Detection Problem

State-of-the-art systems in fields like ATR (cf. Task 2.2) and keyword spotting (KWS) are based on segmented words or text lines as input, although efforts are made to develop systems working solely on the rough input image without any a-priori segmentation [1, 2, 3]. Hence, a workflow which involves a text line extraction by a following textual information extraction is the widespread standard. Extracting these text lines is based on a baseline detection algorithm where we define a baseline as in [4] and give an example in Figure 1.

[1] ATR = Optical Character Recognition + Handwritten Text Recognition
[2] Österreichische Nationalbibliothek


Morgenblatt.

Arbeiter-Zeitung

Zentralorgan der österreichischen Sozialdemokratie.

Erscheint täglich um 6 Uhr Morgens, Montag um 2 Uhr Nachmittags.

Nr. 178. Wien, Samstag, 1. Juli 1911. XXIII. Jahrgang.

Eine Statistenrolle.

Figure 1: An example from the *Arbeiter Zeitung* newspaper from 1911 with baselines in blue.

Definition 1 (Baseline)**.** A *baseline* is defined in the typographical sense as the virtual line where most of the characters rest upon and descenders extend below.

To determine these baselines automatically, a two-stage method as proposed by Grüning et al. [5] is used. The first stage comprises a supervised pixel labeling approach where each pixel of an image is assigned to a class. In our case these classes are *baseline*, *separator* and *other*. The pixel labeling is done by a deep convolutional neural network (CNN) architecture called ARU-Net, which is introduced in Section 2.1. In a second step, the output of the network serves as input for an image processing based bottom-up clustering approach. This second stage (cf. Section 2.2) allows for an error correction of the network output by incorporating domain knowledge based on assumptions, which hold for text lines in general.

2.1 ARU-Net

Within the last few years, different architectures were proposed for the pixel labeling task, most of them based on CNNs [6], which are also exploited for the ATR task in WP 2. CNNs are a special kind of neural networks (NNs) that incorporate the spatial relationships of an image and are therefore well suited for the baseline detection task. Traditionally, a CNN outputs one single global class for the entire input and suffers from a fixed input dimension. For semantic segmentation [7], the so called fully convolutional networks (FCNs) were introduced, which expand the architecture of a CNN to overcome these limitations. Thus a segmentation task (i.e. predicting a class for each image pixel) is performed instead of a classification task.
The U-Net[3] proposed in [8] furthermore introduces shortcuts between layers of the same spatial dimension. This allows for an easier combination of local low-level features and global higher-level features. Finally, the ARU-Net [5] extends the U-Net by two key concepts, spatial attention (A) and depth (residual structure (R)). The attention mechanism makes it possible to handle various font sizes, especially mixed

[3]The "U" in the name reflects the U-shaped form of the architecture, see Figure 2.

font sizes on a single page. The residual structure helps very deep NNs with error backpropagation, i.e. makes them still trainable and yield state-of-the-art results [9].

In Figure 2 the basic architecture of the U-Net with residual connections and the combination with the attention network is depicted. Specifically, the A-Net is a multi-layer CNN which generates a single output feature map. Both networks are then applied to different scales of the input image where weights are shared across the scales. Trainable deconvolutional layers are applied to the down-sampled images to obtain feature maps of spatial dimension equal to the inputs. A softmax normalization layer is applied to the attention maps, the higher the resulting value at a certain position the more attention is paid to that area at the corresponding scale (higher values are reflected by brighter pixels in Figure 2). A point-wise multiplication of the attention maps with the corresponding feature maps of the RU-Net is the final step before summing up the results to get the pixel-wise classification. For a more detailed description of the architecture we refer to [5].

For the baseline detection problem we introduce three different (per pixel) classes: *baseline*, *separator* and *other*. The separators mark beginning and end of each text line and are mainly introduced to avoid segmentation errors, e.g., for baselines that are close together like in two adjacent columns as we have them in newspaper pages. Ground truth (GT) is produced from XML files in the PAGE format [10] holding the baseline information. PAGE is the main GT input format for all WP 2 tasks, see Deliverable D1.1. An example newspaper image with corresponding pixel GT for baseline and separator is given in Figure 3.

2.2 Baseline estimation

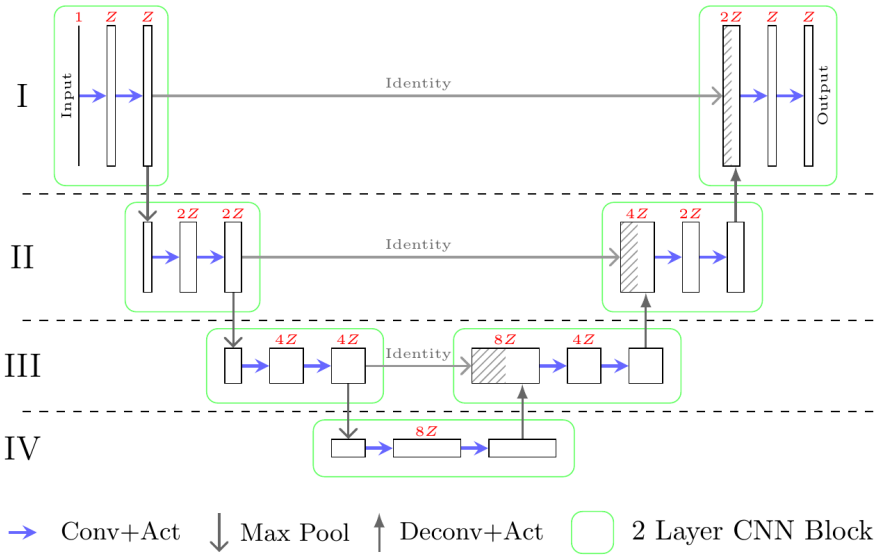
As a second step, the baselines are estimated given the output of the ARU-Net. This can be divided into three subtasks: superpixel (SP) calculation, state estimation and SP clustering. Basically, a SP is a pixel with additional information and of certain importance. It is introduced to reduce the number of pixels to be regarded for the baseline estimation problem since the number of pixels in an image often exceeds several millions.

2.2.1 Superpixel Calculation

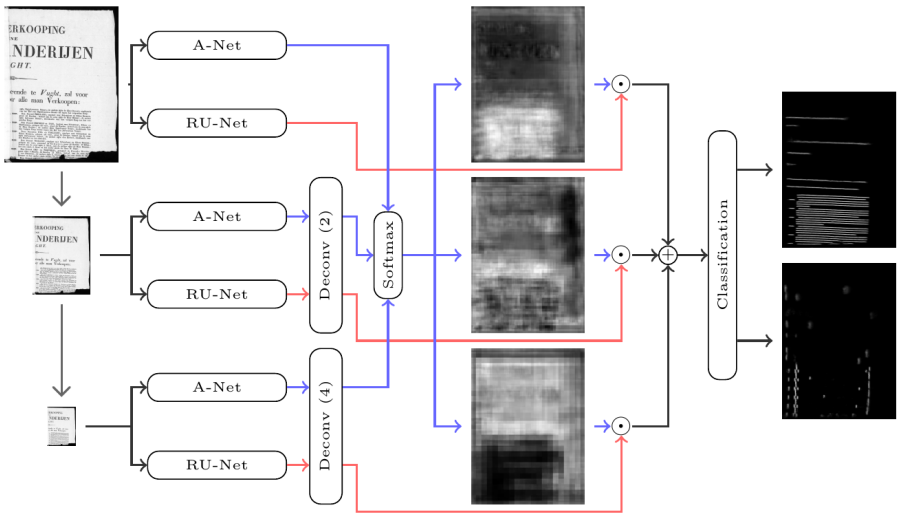
To get an initial set of SPs the baseline map B generated by the ARU-Net is binarized ($B^{(b)}$), with a following morphological skeleton calculation ($B^{(s)} = SKE(B^{(b)})$) based on Lantuéjoul's formula [11]. All foreground pixels (pixels with an intensity of 1) of the skeleton $B^{(s)}$ build an initial set of pixels which are then sorted in descending order w.r.t. their baseline confidences. The final set $\mathcal{S}$ is calculated by iteratively adding pixels from the sorted list, under the constraint that a new pixel $\mathbf{p}$ is added only if its Euclidean distance to all current pixels in $\mathcal{S}$ is bigger than a given threshold d. These SPs build the basis for the following clustering algorithm.

2.2.2 State Estimation

Under the assumption, that we can assign each SP to a certain text line, the state of a SP should encode meaningful characteristics of its text line. These are given by the local text orientation and the interline distance.



(a) RU-Net.



(b) RU-Net and A-Net applied to different scales of the image.

Figure 2: The U-Net architecture with residual connections in the CNN blocks and the combination with an attention network to perform the pixel labeling task as done in [5].

Definition 2 (Local text orientation)**.** The *local text orientation* θ of a SP $\mathbf{p}$ is the slope of its text line's baseline at the coordinates closest (w.r.t. the Euclidean distance) to $\mathbf{p}$.

Definition 3 (Interline distance)**.** The *interline distance* s of a SP $\mathbf{p}$ is the distance of its text line's baseline to the nearest other baseline. Distance means the distance which is orthogonal to the local text direction of $\mathbf{p}$.

The local text orientation can be calculated by using the baseline image B and local information whereas the interline distance combines local information of the text line's periodicity with the assumption that nearby SPs tend to have similar interline distances.

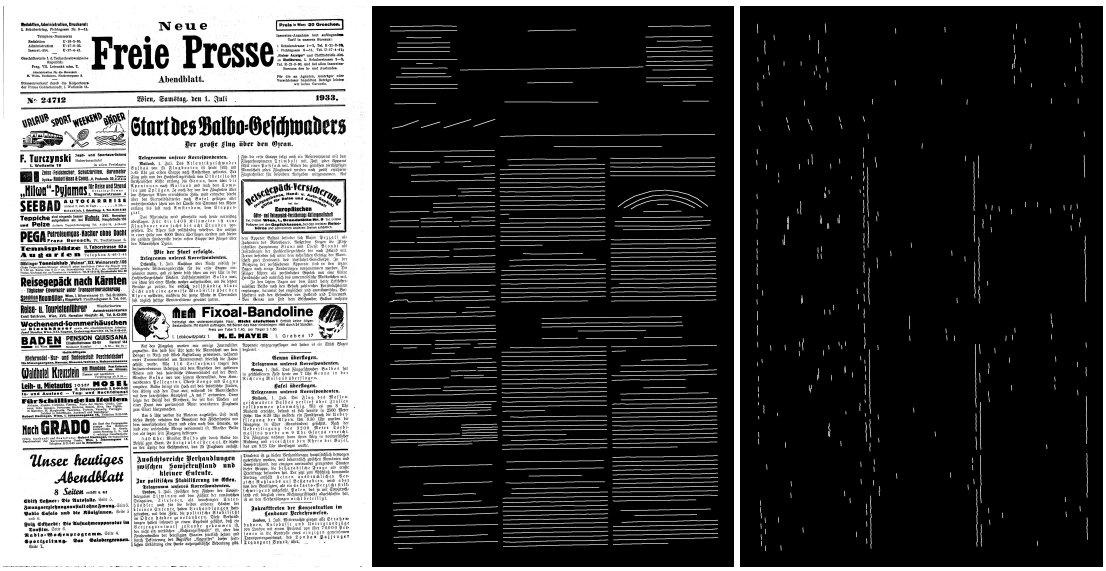


Figure 3: An example from the “Neue Freie Presse” newspaper from 1933 with the original image (left), the baseline pixel GT (middle) and the separator pixel GT (right).

2.2.3 Superpixel Clustering

Finally, the set of SPs $\mathcal{S}$ together with their state information (θ, s) is utilized to cluster the SPs to build baselines. A cluster is a set of SPs where exactly one cluster is assigned to exactly one baseline. The following assumptions must hold while performing the clustering. First, the baselines should not exceed a certain curvilinearity value and second, no other baselines are within the interline distance of a baseline. The first assumption means that a baseline can be approximated by a polynomial function of a certain degree.

2.3 Evaluation

It remains to evaluate the quality of the detected baselines. This is done by using the evaluation scheme introduced in [12], which also forms the basis for the AS measure, cf. Deliverable D2.3. The main advantage of this scheme is that it does not require binarized images. It was used in the *ICDAR2017 Competition on Baseline Detection* [13] and the *ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts* [14].

When measuring the performance of a classifier there are several metrics to choose from: accuracy, precision, recall, specificity, F1-score, etc. If your classification task is imbalanced, i.e. the number of items belonging to one class is significantly lower than those belonging to the other class(es), one often uses the precision and recall values. Precision is the number of items correctly identified as positive out of all items identified as positive, e.g., the number of correct results divided by the number of all results when performing a query in a search engine. Recall is the number of items correctly identified as positive out of all true positive items, e.g., the number of correct results divided by the number of results that should have been returned for the query example. It is easy to achieve a recall value of 100% by mapping every sample to the positive class. Therefore, recall is often combined with precision, resulting in the harmonic mean of both values, the F1-score.

The baseline evaluation scheme is based on these three measures, resulting in the R-value, P-value and

Table 1: ONB dataset comprising four Austrian newspapers from the 19th and 20th century: *Arbeiter Zeitung, Innsbrucker Nachrichten, Illustrierte Kronen Zeitung* and *Neue Freie Presse*.

	year	# pages	# text lines	avg # $\frac{\text{text lines}}{\text{page}}$	binarized
Arbeiter Zeitung	1895	8	3465	433	no
	1911	12	5217	434	no
	1933	10	4311	431	yes
Innsbrucker Nachrichten	1864	12	553	46	yes
	1911	38	8107	213	yes
	1933	20	7622	381	no
Illustrierte Kronen Zeitung	1911	16	3477	217	no
	1933	16	3249	203	no
Neue Freie Presse	1873	16	8609	538	yes
	1895	20	10925	546	yes
	1911	38	18097	476	yes
	1933	26	13199	507	yes
Overall	–	232	86831	374	–

F-value. The R-value indicates how reliably the text on a page is detected, ignoring segmentation/layout errors. As long as all text lines are covered by baselines, we have a high R-value. It doesn't matter if a line is split up into more than one baseline or if two (horizontally) adjacent text lines are covered by one baseline. However, the P-value indicates how reliably the structure of the text lines of the document is detected where splits and merges as stated before are taken into account and get penalized. Since the definition of a baseline (see Definition 1) is very vague, the measure is also invariant to small differences between GT and hypotheses which is realized through threshold values for every GT baseline. Based on the P- and R-value the harmonic mean (F-value) is calculated as

$$F = \frac{2 \cdot R \cdot P}{R + P}.$$

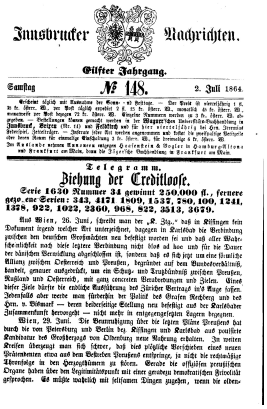
Since $P, R \in [0, 1]$, also $F \in [0, 1]$ follows, where bigger values are better. For a detailed description of R-value, P-value and F-measure we refer to [12].

2.4 Experiments

The proposed workflow [5] was successfully applied to handwritten texts in competitions like the *ICDAR 2015 Competition on Text Line Detection in Historical Documents* [15], *ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts* [14], *cBAD: ICDAR2017 Competition on Baseline Detection* [16] and *cBAD: ICDAR2019 Competition on Baseline Detection* [17] and outperformed all other state-of-the-art methods.
As far as we know there are no official datasets for baseline detection on newspaper articles. However, our project partners ONB and UIBK-DEA provided us with a collection of four Austrian newspapers ranging from 1864 to 1933 as described in Table 1 together with GT files for baseline detection, ATR and AS. One sample from each newspaper is given in Figure 4.

To train and test the ARU-Net described in Section 2.1 we divided the collection into two disjoint subsets: a training set (Train) containing 216 pages and a test set (Test) containing the remaining 16 pages. The test set was selected by hand and contains images from each newspaper, also covering pages with

(a) Arbeiter Zeitung.



(b) Innsbrucker Nachrichten.

(c) Illustrierte Kronen Zeitung.

(d) Neue Freie Presse.

Figure 4: Sample images from the ONB dataset.

Table 2: Evaluation (in %) on the ONB dataset for Train and Test for different Network architectures.

	ONB Train			ONB Test		
Network	P	R	F	P	R	F
RU (ONB)	95.31	99.04	97.14	93.62	97.35	95.36
ARU (ONB)	96.76	97.42	96.89	95.29	97.15	96.17
RU (ONB+G)	92.97	98.65	95.72	93.66	98.08	95.72
ARU (ONB+G)	95.21	97.75	96.33	93.97	97.82	95.76
ARU new (ONB)	96.16	99.03	97.57	94.92	98.63	96.66
ARU (ONBv2)	97.33	98.90	98.06	96.17	97.94	96.94
RU (ONBv2)	98.10	99.20	98.61	96.74	98.70	97.65
ABBYY 11 SDK (ONBv2)	95.85	94.22	94.77	96.90	93.45	93.93

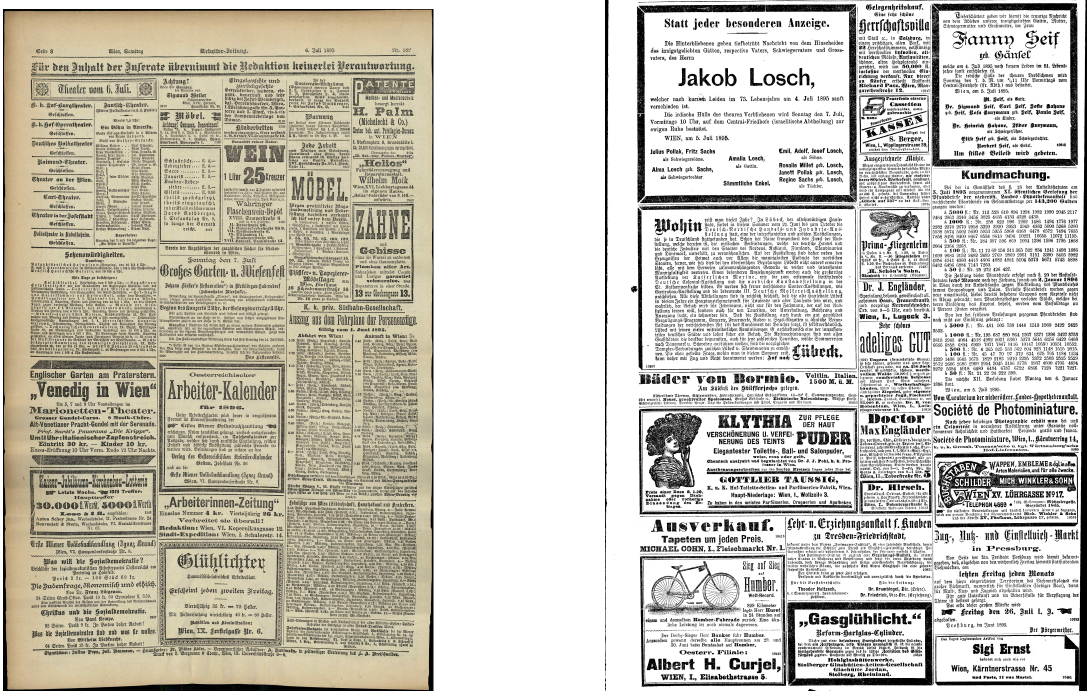
complex layout, cf. Figure 5. We also trained another ARU-Net adding to the train set another 101 pages from a German newspaper called *Germania* provided by the University Library of Rostock.

In year 1 there were four main errors made by the model:

1. complex/graphical writings split up into multiple baselines
2. baselines in tables
3. split of one baseline into multiple smaller ones (mostly two)
4. larger blocks of uncovered baselines

The first three points are mainly segmentation errors, which are reflected by a poor P-value. In the same way the last point is reflected by a poor R-value. We give an example for every error type in Figure 6. First experiments, depicted in Table 2 result in an F-value of 96% on the test set and an F-value of 97% on the train set. There are mainly segmentation errors (points 1-3) made by the model, which is reflected by the worse P-value. As can be seen from the evaluation values, adding extra training data does not add much extra performance on the test set, and present errors should be solved by conventional LA tools.

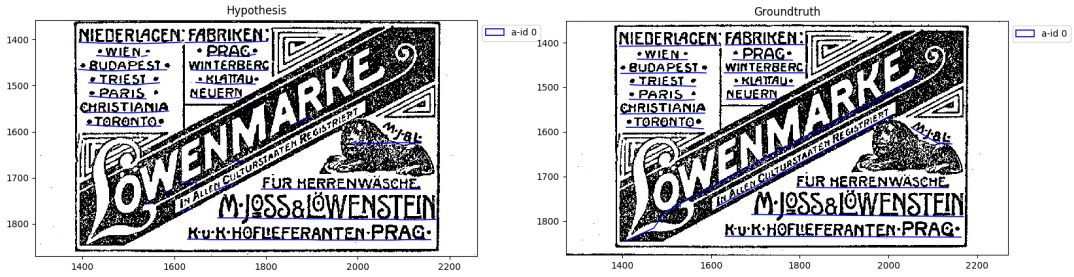
In year 2 we trained another model incorporating additional GT data from other newspaper issues which



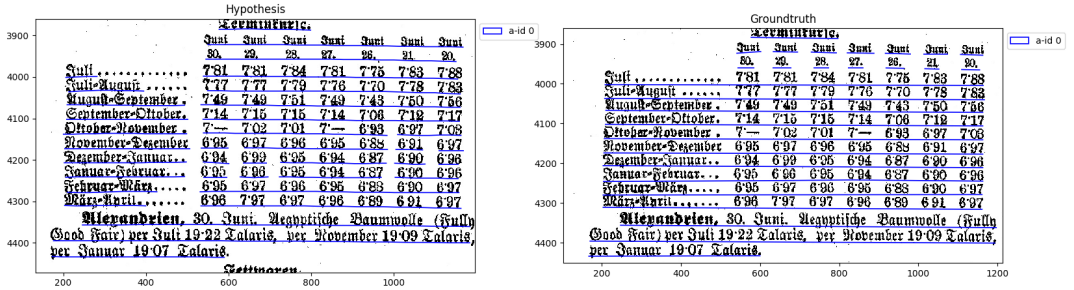
(a) Train (*Arbeiter Zeitung*). (b) Test (*Neue Freie Presse*).

Figure 5: Two example images with a complex layout from each subset.

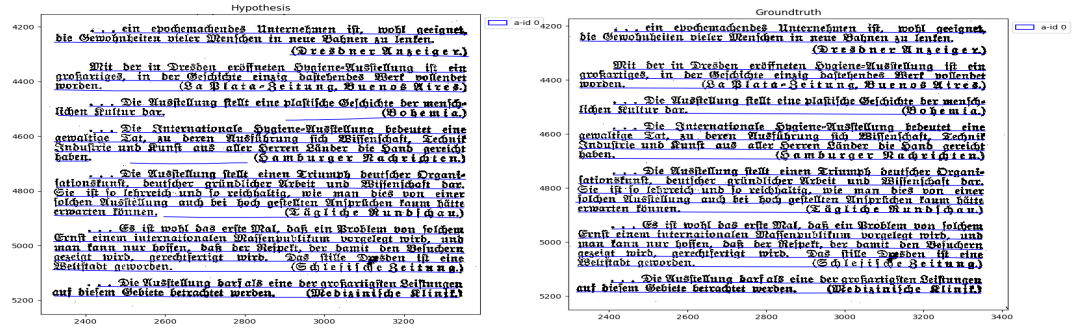
resulted in slightly better results compared to year one. However, after taking a closer look at the original GT data, we found some errors and inconsistencies, which we fixed and trained a new model (ONBv2 in Table 2). On the newer version of the dataset, this resulted in a better P-Value and an overall better F-Value. The fourth error could be resolved completely, split-up baselines only occur in bigger headings or complex writings and baselines in tables are still a problem.



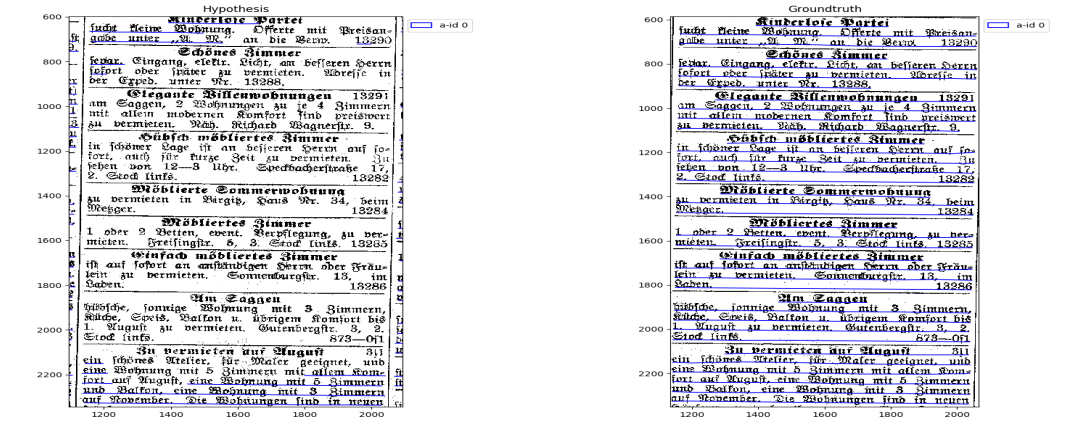
(a) Complex/graphical writing.



(b) Tables.



(c) Baseline split.



(d) Uncovered baselines.

Figure 6: Main errors made by the baseline detection algorithm.

3 Text Block Detection Problem

AS algorithms can be divided into different categories,

- grouping different units like pixels, blocks or baselines to identify articles,
- rule based or machine learning based algorithms,
- using visual or textual features or a hybrid approach.

As described in previous Deliverables D2.1 and D2.3 we focused on clustering baselines using machine learning based methods. In year 2 we started experimenting with detecting text blocks to further improve the quality of the AS output, where we define a text block in two different ways. First, as a set of baselines and second as a physical region defined on pixel bases.

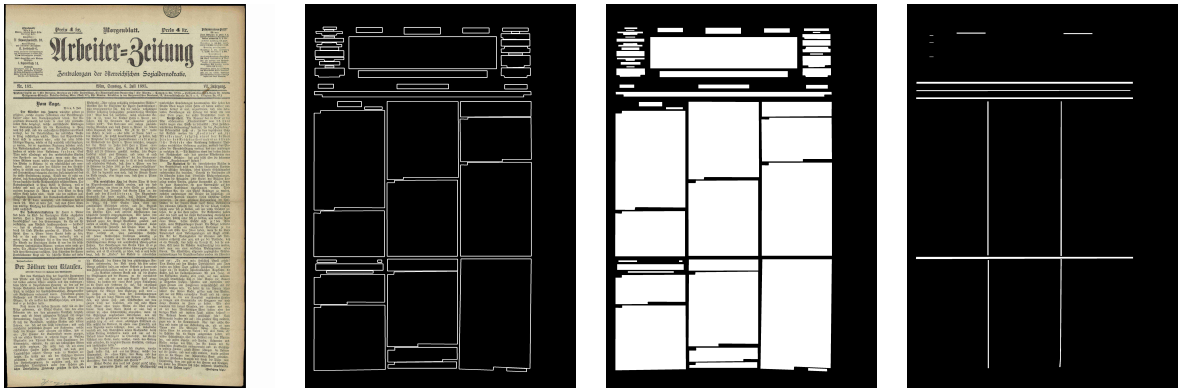
3.1 Pixel-based Approach

Existing methods relying on text block regions take their input from OCR engines like ABBYY, OCRopus or Tesseract [18, 19]. However, for more complex newspaper pages, this becomes a difficult task for these tools. Machine learning algorithms based on a U-Net architecture as introduced in Section 2.1 could be a solution to this problem. In [20] the authors use a FCN to segment scientific papers and articles into text blocks, tables and figures also incorporating contour information of the regions. We applied a similar approach for text block detection on historical newspaper pages. Therefore, we created GT data on a subset of the dataset introduced in Section 2.4 comprising 50 pages for text blocks, text block contours and (visible) separators. An example is given in Figure 7. Since we only have 50 pages, we focused on using all data for training and therefore need to rely on human feedback on the visual outputs to rate the quality of the algorithm rather than evaluating it with a metric like pixel accuracy, intersection over union or mean average precision. For training we used the same model as introduced in Section 2.1. First results show that the quality of the output heavily depends on the spatial dimensions of the newspaper and the ratio of the size of the text to the aspect ratio of the page. In Figure 8 one can see the result of the text block channel output compared on two different scalings on the french newspaper “L’œuvre” that was not used for training. To make the text block detection usable the NN needs to produce reliable outputs independent of the size of the newspaper and a subsequent post-processing needs to be applied to remove uncertainties on the borders of neighboring text blocks. As an additional task the channels that the network should detect could be extended by table regions and image/graphic regions.

3.2 Alpha-shape Approach

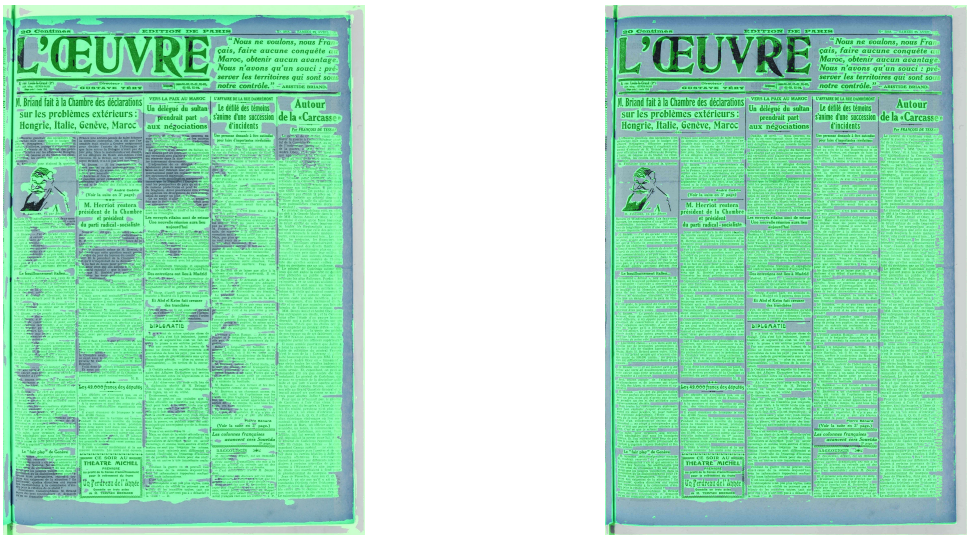
Besides the Pixel-based, i.e. machine learning-based approach, we developed a simple clustering-based, three step method to construct text blocks. In the first step we detect the baselines of a given image. Afterwards, we cluster these lines together with a DBSCAN-based algorithm (for more details see Deliverable D2.6). Therefore, we speak in this context of baseline clusters (generated by rule-based methods) instead of text blocks (generated by machine learning-based methods). Since a baseline is represented by a polygon which is determined by a list of points, we outline in our last step these clusters to get our regions. The last point is done by performing an alpha-shape algorithm on every cluster in the two-dimensional space. The advantage of alpha-shapes (see [21]) in contrast to the convex hull is that we can enclose the text more closely, since these shapes are usually not convex (see Figure 9).

Please note, that a favorable side effect of having regions is that the computation of the reading order of



(a) Original Image. (b) Text Block Contours GT. (c) Text Block GT. (d) Separator GT.

Figure 7: Example GT for the “Arbeiter Zeitung” newspaper from 1895 for the text block detection task.



(a) Page size of 1085x1590 (scaling factor=0.2). (b) Page of size 2895x4240 (scaling factor=0.8).

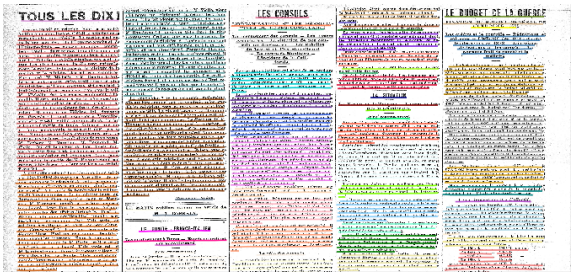
Figure 8: Text block detection NN output on a newspaper page from the issue “L’œuvre” from 1926. The image was not seen in the training.

the single lines within regions is trivial. However, the order between regions is much more complicated and not solved yet.

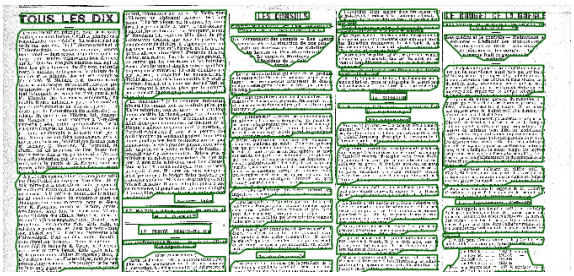
For now we are using the second, i.e. the baseline clustering approach. In the future, we want to further improve the pixel labeling approach and to either use it on its own or combine it with the alpha-shape algorithm.

4 Patch-wise Layout Analysis

Deep learning is a term that has become more important in recent years, mainly due to better computer hardware and the introduction of software libraries like TensorFlow, Theano and PyTorch. These make it easier to build machine learning models and to use the ability of GPUs to quickly perform operations like matrix multiplications. However, the complexity and depth of such a model, or in other words, the number of trainable weights is limited by the amount of memory that is available. One possibility to



(a) Part of a newspaper page with marked baselines (baselines belonging to the same cluster have the same color).



(b) Part of a newspaper page with marked text blocks.

Figure 9: Baseline clusters and corresponding alpha-shape regions.

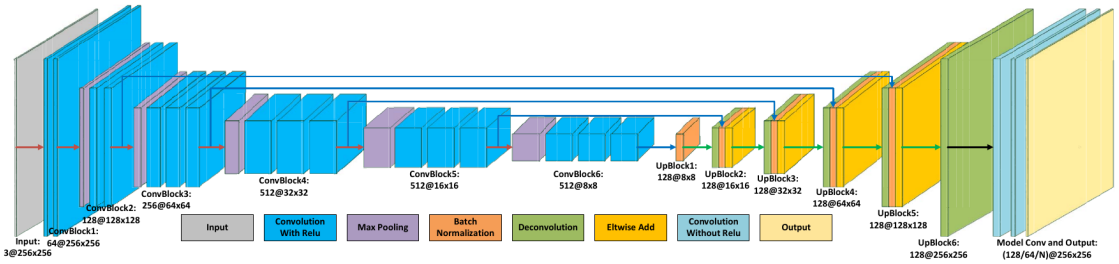


Figure 10: The U-Net architecture used for the patch-wise pixel labeling task as described in [22]. Reprinted from [22].

use deeper models without changing the hardware is to shrunken the size of the input. Therefore, we experimented with a patch-wise processing of the input images for the baseline detection scenario.

We restricted the input to patches of size 256x256 and 512x512. In training a random patch of an image is chosen and forwarded to the NN. For evaluation/inference, the image is divided into sub-images of the same size as the network was trained on with an overlap of $\frac{1}{4}$, i.e. 128 pixels for a 512x512 patch. The outputs are then merged to the original image size and in case of an overlap combined with a maximum operation. For testing, we applied the workflow to the *cBAD: ICDAR2019 Competition on Baseline Detection* [17] which also includes historical newspaper pages. As a first step we trained the same ARU-Net that we used in the previous two sections on the full image size and on patches. We also built a deeper network as described in [22] that is an extension of the VGG16 architecture [23]. Figure 10 depicts the model which we call layer pyramid network (LPN) in the following.

The results in Table 3 show that we can reach the same results when training the NN on patches as if we would train it on the whole image provided the patch size is not too small and holds enough information. The deeper network does not bring major improvements over the other models. Since it needs two times as long as the ARU-Net to train and evaluate we would stick to the flatter architectures. However, the patch-wise processing of images makes it possible to use models that would not work on larger images due to memory restrictions. One of those models is the capsule network which we want to briefly introduce in the following section.

Table 3: Evaluation (in %) on the *cBAD: ICDAR2019 Competition on Baseline Detection* test set for different network architectures. The first row is an RU-Net trained and evaluated on the full image. The other rows are trained on the listed patch size and evaluated on the whole image as described in the text.

		cBAD (2019)		
Network	Patch Size	P	R	F
Baseline	Full Image	91.96	92.65	92.31
Baseline	512x512	90.94	92.78	91.85
ARU	512x512	92.12	92.04	92.08
RU	512x512	91.56	92.59	92.07
LPN	512x512	92.43	92.15	92.29
Baseline	256x256	84.79	92.42	88.44
ARU	256x256	91.14	89.96	90.55
RU	256x256	90.32	91.31	90.81
LPN	256x256	91.73	90.08	90.90

5 Capsule Networks

Recently, a new type of architecture, the capsule network was proposed in [24] by Sabour et al.[4] The basic idea behind capsule networks is to eliminate the drawbacks of CNNs, e.g., loosing spatial information through max pooling layers. To explain capsule networks, we first have a look at computer graphics. In computer graphics one wants to build an abstract representation of a scene containing multiple objects where each object type has different instantiation parameters. Then you want to call a rendering function on this representation which gives an image as ouput. Inverse graphics is the reverse process, i.e. given an image we want to detect the corresponding instantiation parameters. Basically, capsule networks are NNs that try to perform inverse graphics. A capsule is a function that tries to predict the presence and instantiation parameters of a particular object at a given location. Instantiation parameters are e.g., the pose, deformation, albedo, hue or texture of an object. The capsules are inspired by the brain's cortical minicolumn where neurons in a minicolumn encode similar features. They are represented by vectors whose orientation encodes the object's estimated instantiation parameters and whose length reflects the probability that the entity exists. One key-feature of capsule networks is that they preserve detailed information about the object's location and its pose throughout the network. Pooling layers in CNNs tend to lose such information. That means that capsule networks are equivariant instead of invariant like the CNNs. By an algorithm called routing-by-agreement the information between different capsules is shared and passed on. Basically, capsules from the previous layer try to predict the output of the capsules in the actual layer.

The idea of capsule networks was then extended by LaLonde et al. in [25] to object segmentation, i.e. to a pixel labeling task. They built a U-Net architecture using capsules instead of common convolutional layers and applied it to segment pathological lungs from low dose CT scans which have a size of 512x512 pixels. We began re-implementing the model and applying it to our baseline and text block detection task. Sadly, we do not yet have any presentable results at this time. The architecture of the network can be seen in Figure 11.

[4] As a side note, one of the authors is Geoffrey Hinton, a pioneer of deep learning.

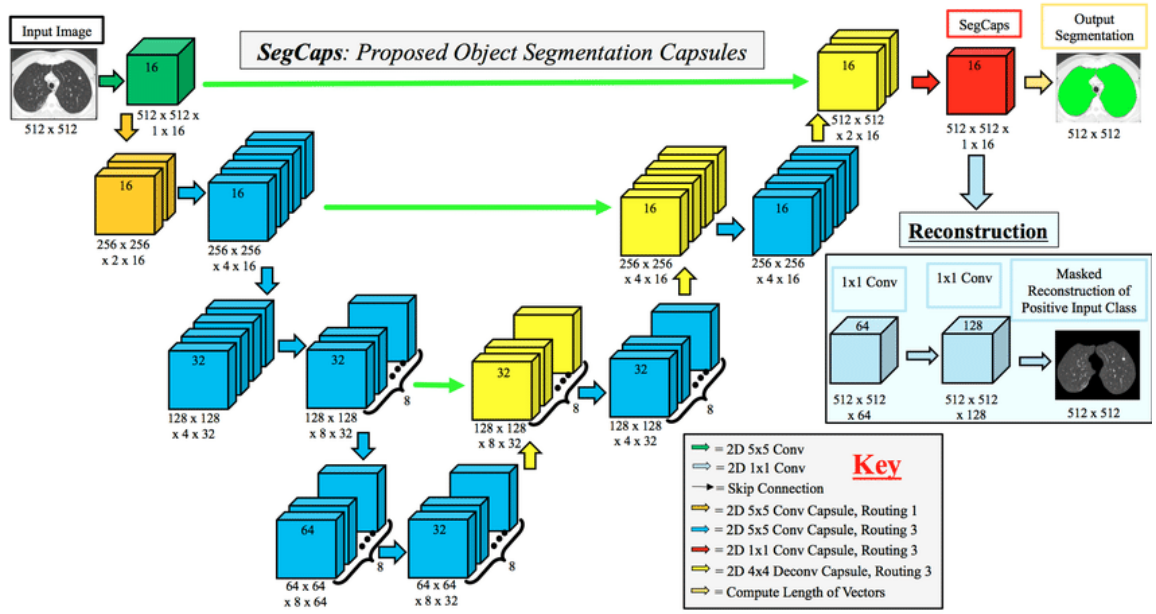


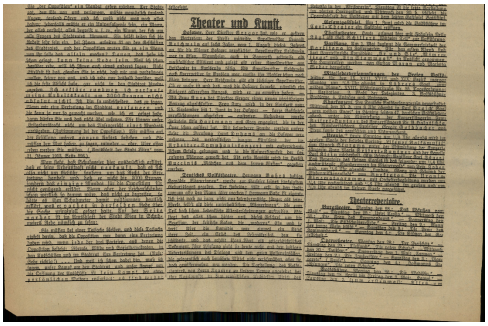
Figure 11: The capsule network architecture used for object segmentation. Reprinted from [25].

6 Comparison to OCR engines

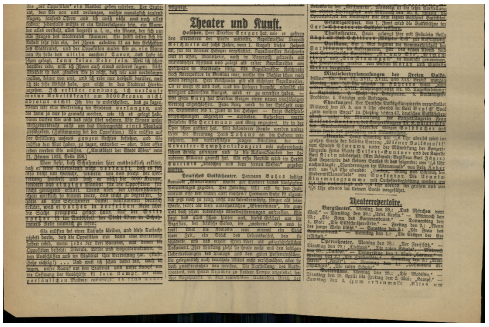
When analysing the layout of a digitized newspaper page one could also use and rely on common OCR tools like ABBYY which perform connected component analysis and use rule based algorithms to extract all the relevant information. However, recent versions are also integrating machine learning models to handle images with bad quality resulting from washed out, too bright or too dark parts, unknown skewing angles etc. In this section we want to compare the baseline and text block detection quality of our algorithms with ABBYY FineReader 11 SDK, which can be used in Transkribus. To our knowledge the machine learning models are not yet integrated in version 11 and are only available from version 12.

For the baseline detection task the quality of the ABBYY output was nearly perfect on simple pages and alright on more complex pages. However, our baseline detection algorithm outperformed the engine on the ONB dataset and has an overall better performance on more complex datasets. In Table 2 we added the P-Value, R-Value and F-Value of ABBYY for the ONBv2 dataset which are worse than our results. Overall, common errors that were made are bigger blocks of undetected baselines, undetected baselines for larger headings, undetected baselines in advertisements and merging of two separate baselines if no separator information is given and the baselines are close together. Two examples are given in Figure 12.

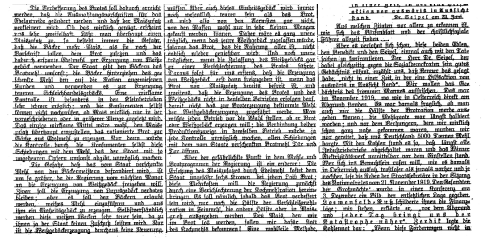
For the text block detection task we compare the output of ABBYY to the alpha-shape based approach from Section 3.2. In general, for the AS task it is better to extract text blocks that are over-segmented, since it is much easier to merge text blocks to articles than to split them into multiple parts. Also the ABBYY text blocks look correct, they are too under-segmented in the most cases. However, in some cases the ABBYY workflow is still performing better than our approach as can be seen in Figure 13. We hope that either the alpha-shape algorithm can be further improved or that with our pixel labeling approach we can find a good intermediate path between the two approaches.



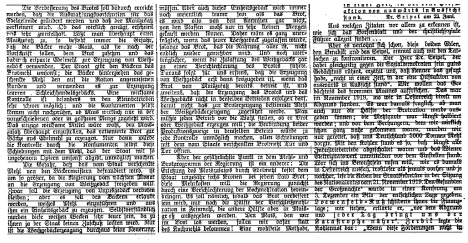
(a) U-Net output – example 1.



(b) ABBYY output, missed baselines – example 1.

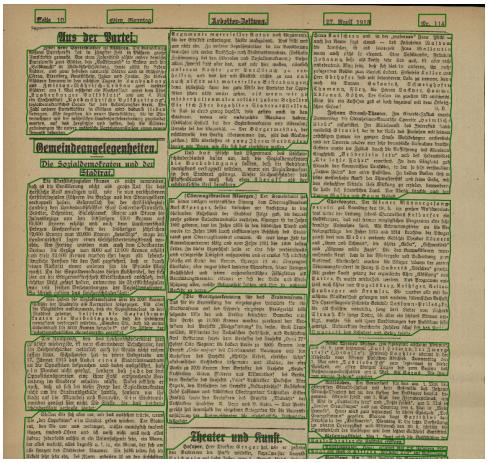


(c) U-Net output, merged baselines – example 2.



(d) ABBYY output, merged baselines – example 2.

Figure 12: Comparison of U-Net and ABBYY for the baseline detection task.



(a) Alpha-shape output – example 1.

(b) ABBYY output – example 1.

(c) Alpha-shape output – example 2.

(d) ABBYY output – example 2.

Figure 13: Comparison of alpha-shape and ABBYY for the text block detection task.

7 Tools and Hardware

To train the aforementioned neural networks we use TensorFlow [26], Google's C++ based open-source software library for numerical computations, mainly used for machine learning applications. TensorFlow has APIs available in several languages, such as Python and Java. Our software is written in Python since the Python API is the most complete one and also the easiest to use. The computations are depicted using dataflow graphs, where edges represent multidimensional data arrays called tensors, that communicate via mathematical operations (ops) represented by nodes. One of the main advantages is the relatively easy building and training of machine learning models such as feedforward NNs, recurrent neural networks (RNNs) and CNNs. TensorFlow also takes advantage of the computing power of GPUs for tasks like matrix multiplications to speed up the overall training process. This proves to be especially useful when using CNNs. The SP/baseline estimation however is done in Java and performed on CPUs only. The baseline detection algorithm can be used via the Transkribus platform and will be updated as soon as the algorithms are specialized for newspaper articles. For the experiments in the previous sections, the models where trained on a single Geforce GTX 1080 Ti GPU with 11 GB RAM and the SP clustering was performed on an Intel(R) Core(TM) i7-8650U CPU with a clock rate of 1.90GHz. On average, for the baseline detection models one training for 250 epochs took 11h for the experiments in Section 2.4, that makes about three samples per second. The clustering takes 50 seconds per image.

8 Conclusions and Future Work

In year 2 we were able to solve some of the baseline detection problems we had in year 1, still there is work to do regarding special cases like tables, more complex writings and advertisements. We presented some first results on the text block detection task and hope that we can use the algorithms in year 3 to further improve the AS of Task 2.3. In addition, we want to test the capsule network architecture on the introduced patch-wise processing of newspaper images. In comparison to OCR engines like ABBYY our algorithms can keep up easily on the baseline detection task and perform well on text block segmentation.

Since this is the last deliverable of Task 2.1 we will focus on using the presented algorithms in Task 2.3. Furthermore, since our evaluation measure for AS includes the baseline detection measure, the evaluation of the LA is implicitly continued in year three within Task T2.3.

References

[1] Marçal Rusiñol, David Aldavert, Ricardo Toledo, and Josep Lladós. "Efficient Segmentation-free Keyword Spotting in Historical Document Collections". In: *Pattern Recogn.* 48.2 (Feb. 2015), pp. 545–555. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.08.021. URL: http://dx.doi.org/10.1016/j.patcog.2014.08.021.

[2] Adam Ghorbel, Nicole Vincent, and Jean-Marc Ogier. "A segmentation free Word Spotting for handwritten documents". In: Aug. 2015. DOI: 10.13140/RG.2.1.1534.7683.

[3] Théodore Bluche. "Joint Line Segmentation and Transcription for End-to-End Handwritten Paragraph Recognition". In: *CoRR* abs/1604.08352 (2016).

[4] Markus Diem, Florian Kleber, Stefan Fiel, Tobias Grüning, and Basilis Gatos. *ScriptNet: ICDAR 2017 Competition on Baseline Detection in Archival Documents (cBAD)*. 2017. DOI: 10.5281/zenodo.257972. URL: https://doi.org/10.5281/zenodo.257972.

[5] Tobias Grüning, Gundram Leifert, Tobias Strauß, and Roger Labahn. "A Two-Stage Method for Text Line Detection in Historical Documents". In: *CoRR* abs/1802.03345 (2018). arXiv: 1802.03345. URL: http://arxiv.org/abs/1802.03345.

[6] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2323. DOI: 10.1109/5.726791.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Proc. of the Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: http://ieeexplore.ieee.org/document/7780459/.

[10] Stefan Pletschacher and Apostolos Antonacopoulos. "The PAGE (Page Analysis and Ground-Truth Elements) Format Framework". In: *2010 20th International Conference on Pattern Recognition*. IEEE, Aug. 2010. DOI: 10.1109/icpr.2010.72.

[11] Jean Serra. *Image Analysis and Mathematical Morphology*. Vol. 1. 2. 1982, p. 610. ISBN: 0126372403.

[12] Tobias Grüning, Roger Labahn, Markus Diem, Florian Kleber, and Stefan Fiel. "READ-BAD: A New Dataset and Evaluation Scheme for Baseline Detection in Archival Documents". In: *CoRR* abs/1705.03311 (2017). arXiv: 1705.03311. URL: http://arxiv.org/abs/1705.03311.

[13] Markus Diem, Florian Kleber, Stefan Fiel, Tobias Grüning, and Basilis Gatos. *ScriptNet: ICDAR 2017 Competition on Baseline Detection in Archival Documents (cBAD)*. 2017. DOI: 10.5281/zenodo.257972. URL: https://doi.org/10.5281/zenodo.257972.

[14] Fotini Simistira, Manuel Bouillon, Mathias Seuret, Marcel Würsch, Michele Alberti, Rolf Ingold, and Marcus Liwicki. "ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts". In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 1361–1370.

[15] Michael Murdock, Shawn Reid, Blaine Hamilton, and Jackson Reese. “ICDAR 2015 competition on text line detection in historical documents”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 1171–1175. DOI: 10.1109/ICDAR.2015.7333945.

[16] Markus Diem, Florian Kleber, Stefan Fiel, Basilis Gatos, and Tobias Grüning. “cBAD: ICDAR2017 Competition on Baseline Detection”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 1355–1360.

[17] Markus Diem, Florian Kleber, Robert Sablatnig, and Basilis Gatos. “cBAD: ICDAR2019 Competition on Baseline Detection”. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. ISSN: 1520-5363. Sept. 2019, pp. 1494–1498. DOI: 10.1109/ICDAR.2019.00240.

[18] Anukriti Bansal, Santanu Chaudhury, Sumantra Dutta Roy, and J.B. Srivastava. “Newspaper Article Extraction Using Hierarchical Fixed Point Model”. In: *2014 11th IAPR International Workshop on Document Analysis Systems*. Apr. 2014, pp. 257–261. DOI: 10.1109/DAS.2014.42.

[19] R. Furmaniak. “Unsupervised Newspaper Segmentation Using Language Context”. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. ISSN: 2379-2140. Sept. 2007, pp. 1263–1267. DOI: 10.1109/ICDAR.2007.4377118.

[20] Dafang He, Scott Cohen, Brian Price, Daniel Kifer, and C. Lee Giles. “Multi-Scale Multi-Task FCN for Semantic Page Segmentation and Table Detection”. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. ISSN: 2379-2140. Nov. 2017, pp. 254–261. DOI: 10.1109/ICDAR.2017.50.

[21] Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. “On the shape of a set of points in the plane”. In: *IEEE Trans. Information Theory*. 1981.

[22] X. Li, F. Yin, T. Xue, L. Liu, J. Ogier, and C. Liu. “Instance Aware Document Image Segmentation using Label Pyramid Networks and Deep Watershed Transformation”. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pp. 514–519. DOI: 10.1109/ICDAR.2019.00088.

[23] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

[24] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. *Dynamic Routing Between Capsules*. 2017. arXiv: 1710.09829 [cs.CV].

[25] Rodney LaLonde and Ulas Bagci. *Capsules for Object Segmentation*. 2018. arXiv: 1804.04241 [stat.ML].

[26] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.