International Journal on Document Analysis and Recognition (IJDAR) (2019) 22:285–302
https://doi.org/10.1007/s10032-019-00332-1

SPECIAL ISSUE PAPER

A two-stage method for text line detection in historical documents

Tobias Grüning[1] · Gundram Leifert[2] · Tobias Strauß[1] · Johannes Michael[2] · Roger Labahn[2]

Received: 16 November 2018 / Revised: 23 April 2019 / Accepted: 28 June 2019 / Published online: 23 July 2019

Abstract
This work presents a two-stage text line detection method for historical documents. Each detected text line is represented by its baseline. In a first stage, a deep neural network called ARU-Net labels pixels to belong to one of the three classes: baseline, separator and other. The separator class marks beginning and end of each text line. The ARU-Net is trainable from scratch with manageably few manually annotated example images ($<$ 50). This is achieved by utilizing data augmentation strategies. The network predictions are used as input for the second stage which performs a bottom-up clustering to build baselines. The developed method is capable of handling complex layouts as well as curved and arbitrarily oriented text lines. It substantially outperforms current state-of-the-art approaches. For example, for the complex track of the cBAD: ICDAR2017 Competition on Baseline Detection the F value is increased from 0.859 to 0.922. The framework to train and run the ARU-Net is open source.

Keywords Baseline detection · Text line detection · Layout analysis · Historical documents · U-Net · Pixel labeling

1 Introduction

Accessibility of the valuable cultural heritage of historical documents is an important concern of archives, libraries as well as certain companies, e.g., those specialized in genealogy. After years of digitization at an industrial scale to protect and preserve these valuable goods, millions over millions of scanned pages are stored at servers all over the world [1]. The generic next step is to make the enormous amount of content of these document images accessible and enable humanists, historians, genealogists as well as ordinary people to efficiently work with these documents. Besides the cost- and time-consuming process of manually annotating volumes [2], it is subject to current research and scientific discussion how to automate this process [3].

Since 2009, tremendous progress in the field of Automated Text Recognition[1] (ATR) [4,5] as well as Keyword Spotting (KWS) [6–8] was achieved. The performance of state-of-the-art systems reaches character error rates below 10% for ATR [9] and mean average precisions above 0.9 for KWS [10] for complex handwritten documents. Although efforts are made to develop systems working solely on the rough input image without any a-priori segmentation [11–13], the best performing recognition systems—with reference to recently hosted competitions—rely on segmented words or text lines as input. Entirely segmentation-free approaches suffer either from an enormous training/inference time and/or, up to now, did not demonstrate its applicability with competitive quality on challenging datasets [10]. Hence, a workflow which involves a text line extraction followed by the transformation of pixel information into textual information (ATR/KWS) is the widely used standard. This work deals with the first step of the information retrieval pipeline, namely the text line extraction. This is a mandatory step since errors directly effect the performance of the overall information retrieval process. The text line extraction is still unsolved to a certain extent for historical documents due to difficulties such as physical degradations (e.g., bleed-through, faded away characters, heterogeneous stroke intensity), image capture conditions (e.g., scan curve, illumination issues), complex layouts (e.g.,

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10032-019-00332-1) contains supplementary material, which is available to authorized users.

✉ Tobias Grüning
tobias.gruening@planet.de

[1] Planet AI, Warnowufer 60, 18057 Rostock, Germany

[2] Computational Intelligence Technology Lab, Department of Mathematics, University of Rostock, 18057 Rostock, Germany

[1] Optical Character Recognition + Handwritten Text Recognition.

structured documents, marginalia, multi-column layouts, varying font sizes), arbitrary orientations and curved text lines.

The results achieved by state-of-the-art approaches are not satisfying [14], especially if dealing with heterogeneous data. Therefore, this work focuses on the extraction of text lines in arbitrary historical documents. Since different ATR/KWS systems necessitate different text line representations, e.g., bounding boxes [15], x-height areas [16] or more precise polygonal representations following all ascenders and descenders [7], there is not the **one** correct text line representation. Therefore, we limit ourselves toward the text line detection task by representing each text line by its baseline. The detected baselines allow for an extraction of the text lines in an appropriate—with respect to the following method way. The problem of extracting a text line given its baseline has to handle problems like touching or overlapping components and can be tackled by applying, e.g., histogram approaches to estimate the x-height [16] or by utilizing dynamic programming to calculate separating seams [17] However, this is not within the scope of this work.

Besides the classical image processing-based approaches, deep learning-based methods became omnipresent in the document analysis community within the last years. Such techniques were recently used to solve several different problems such as binarization [18], page boundary extraction [19], page segmentation [20] or text line detection [16]. The presented work to our knowledge is the first which uses a two-stage method, combining deep learning strategies and state-of-the-art image processing-based techniques. We propose an extension of the U-Net [21], the so-called ARU-Net. The fully convolutional U-Net is extended by incorporating residual blocks [22] to increase its representative power. Furthermore, a spatial attention mechanism is developed which allows the ARU-Net to focus on image content at different positions and scales. The network is designed to processes the entire, arbitrarily-sized image at once to take account of all spatial context. The ARU-Net is universal in a way that it could be used to tackle any pixel labeling task. In this work, it is trained in a fully supervised fashion to classify each pixel to belong to one of the following classes: baseline, separator and other. The separator class is introduced to explicitly predict beginning and end of each text line and not just rely on the information implicitly given by the baseline class. This is advantageous for text lines which are close together but have to be separated, e.g., those belonging to different columns. The network output serves as input for an image processing-based bottom-up clustering approach. This approach utilizes so-called states of superpixels [23], which encode local text orientation and interline distances. This second stage allows for an error correction of the network output by incorporating domain knowledge based on assumptions, which hold for text lines in general, see Sect. 3.3.3. Additionally, it is easily possible to incorporate the separator information, which allows for a handling of documents with complex layouts, e.g., images containing tables or marginalia.

Each method relying on supervised deep learning and therefore relying on training data can suffer from the need of an enormous amount of labeled training data. We demonstrate that the presented approach achieves high-quality results on the Bozen dataset [24] with less than 50 full-page training samples by using data augmentation strategies. Along with an annotating effort of just a few minutes per page, the adaptation of the proposed method is easy and cheap. We demonstrate the applicability of the proposed method for images with arbitrarily oriented as well as curved text lines by achieving nearly as good results as for straight 0° oriented text lines. Finally, we show that the presented approach outperforms state-of-the-art methods on three different datasets. A relative F value [25] error (the gap to 1.0) reduction of at least 24% is achieved for the cBAD dataset [26]. This dataset is composed of 2036 historical images with annotated baselines of nine different archives and libraries from all over Europe and is therefore—in the opinion of the authors—the most representative and heterogeneous freely available dataset. Especially, for the complex track, which contains mostly documents with complex layouts, the average F value is increased from 0.859 to 0.922.

The main contributions of this work are:

- Introduction of a newly designed deep neural network (ARU-Net) for pixel labeling along with a meaningful parametrization—the ARU-Net and its training framework are open source,[2]
- Introduction of the new concept of learned separators to handle complex layouts instead of an a-priori page segmentation or white-/blackrun calculation
- Introduction of a state-of-the-art two-stage workflow which combines state-of-the-art deep learning and image processing techniques—the entire workflow is freely usable via the Transkribus platform.[3]

2 Related work

A comprehensive survey of approaches for text line extraction in historical documents is given in [27,28]. In this section, we will focus on approaches relevant for this work.

In [17,29,30], the principle of dynamic programming is utilized to calculate cost optimal paths passing the image from left to right to separate different text lines from each other. These methods basically differ in the way the images are pre-processed and in the definition of the cost function.

[2] https://github.com/TobiasGruening/ARU-Net.

[3] https://transkribus.eu.

Garz et al. [31] propose a method based on clustering of interest points (this is just another name for what we call superpixel). Using a standard clustering technique, interest points in an area which exceeds a certain density are clustered to form word clusters. Word clusters are separated to sub-word segments, and these are finally grouped to build text lines. Ryu et al. [23] propose an algorithm which uses certain characteristics (so-called states) of extracted connected components to assign costs to certain clustering results. These states encode local text orientation and interline distances and are introduced in Definition 3.13. Subsequently, using four different operations (merge, split, merge–split, merge–merge–split) on an initial coarse clustering, the costs are minimized to obtain an optimal clustering, which leads to the final text line segmentation. Ahn et al. [32] improve this approach by the introduction of a newly developed binarization method and an improved clustering process. Grüning et al. [33] extended the approach of Ryu et al. so that it is applicable for more general superpixels with a newly introduced clustering procedure which does not rely on a coarse initial clustering. Besides these "classical" approaches, which are based on image processing techniques, methods based on machine learning gained importance within the last two years. Moysset et al. [34] propose a method based on a recurrent neural network. The network is trained given only the number of lines in the image utilizing connectionist temporal classification which was introduced to train networks for handwriting text recognition and allows for ground truth data without any alignment. The trained neural network predicts confidences for the vertical coordinates of the image to belong either to the classes line or interline. Further post-processing of the neural network output is performed to detect the text lines. In follow-up works, they formulated the problem as a regression problem [35]. The recurrent neural network directly predicts bounding boxes as well as the start of each text line, respectively. Besides this regression-based approach, classification-based approaches were proposed most recently. In contrast to the approach of Moysset et al., these methods perform a pixel labeling to classify each image pixel (instead of classifying rows of pixels, only). For instance, Renton et al. [16] propose a fully convolutional network (FCN) based on dilated (or atrous) convolutions to classify pixels as text line main body or not. The classification results are utilized to extract the text line information. These techniques are currently very popular, e.g., four of the five participants of the cBAD: ICDAR2017 Competition on Baseline Detection [36] use methods relying on FCNs. For example, the methods presented in [16,52–54] tackle the problem of baseline detection with fully convolutional neural networks. However, these methods either rely on a patch-wise processing of the input image (which results in a cumbersome reconstruction of the baseline hypothesis for the entire image), on massive pre-trained encoder structures (which significantly increases the number of model parameter and slows down the system) or on elementary post-processing steps (which deteriorates the system's performance). The presented method overcomes this limitations and shows its superiority on the challenging cBAD dataset.

3 Methodology

In this section, we introduce the two-stage method for baseline detection, see Fig. 1. The first stage relies on a deep neural network—the ARU-Net—and performs a pixel labeling. The pixel labeling can be seen as some kind of goal-oriented binarization. Instead of detecting all foreground elements, it restricts itself to those elements which are of interest for the specific task. The second stage performs a superpixel (SP) extraction on the first stage's output. These SPs are further clustered to build baselines. In the following, the problem of baseline detection is formulated. Afterward, a detailed description of the proposed ARU-Net is given. Finally, the SP extraction and clustering approach are described.

3.1 Problem statement

We will introduce the problem of baseline detection in a formal way by defining all necessary termini and notation. Within this work, we follow the definition of a baseline given in [26]:

Definition 3.1 (*Baseline*) A *baseline* is defined in the typographical sense as the virtual line where most characters rest upon and descenders extend below.

Hence, each baseline can be represented by a polygonal chain. The infinite set $\mathcal{P}$ of all possible polygonal chains is called *polygonal chain space*. Within this work, we limit ourselves toward gray-scale images. The set of all possible (gray-scale) images $\mathcal{I} = \bigcup_{h,w\in\mathbb{N}} [0,1]^{h\times w}$ is called *image space*. If the colored image is available, we usually use this one for visualization even though it is converted to its gray-scale version for calculations. For visualization purposes, a pixel intensity value of 1 means white and 0 means black. I_h denotes the height of image I, I_w denotes the width, analogously.

Definition 3.2 (*Baseline detector, baseline hypothesis*) We call a function $b : \mathcal{I} \to \mathfrak{P}(\mathcal{P})$ which maps each image to a subset of $\mathcal{P}$ a *baseline detector*. The set of all baseline detectors is denoted by $\mathcal{B}$. The output of b for a certain image I is called *baseline hypothesis*.

Definition 3.3 (*Baseline ground truth*) The set $\mathcal{G}_I \subset \mathcal{P}$ of polygonal chains representing the baselines of an image I

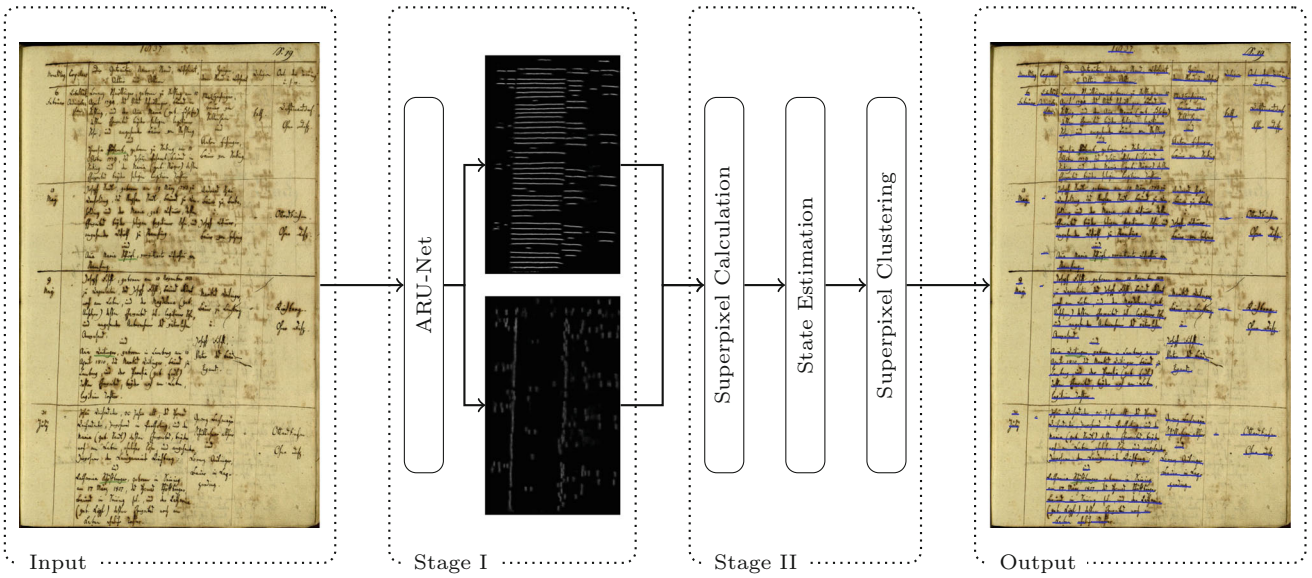


Fig. 1 Two-stage workflow to detect baselines—the first stage utilizes a deep hierarchical neural network to perform a pixel labeling. The result of Stage I is the input for an image processing-based method in Stage II. This method clusters superpixel to build baselines. The image is sampled from the cBad complex test set [25]

(possibly annotated by a human operator) is called *baseline ground truth* (for image I).

Definition 3.1 allows for some baseline variety. Hence, there is not the one unique and correct ground truth for an image. Therefore, ground truth information is always biased by its creator. This has to be taken into account for the evaluation process as well as for the baseline detector design.

Definition 3.4 (*Similarity score*) A function $\langle \cdot, \cdot \rangle_\mu : \mathcal{P} \times \mathcal{P} \to [0, 1]$ assigning a scalar value to each pair of baseline ground truth and baseline hypothesis polygonal chain sets is called *similarity score*.

A value of 1.0 indicates that two polygonal chains are regarded as equal. Within this work, we follow the similarity score introduced in [25]: We measure the accuracy of a baseline detector in terms of the F value, see [25] for a detailed introduction.

The problem tackled in this work can now be formulated as follows: Suppose there are two sets of images along with their baseline ground truth information $\mathcal{T}_{\text{train}} = \{(I, \mathcal{G}_I)_i \mid i = 1, \ldots, n\}$ and $\mathcal{T}_{\text{test}} = \{(I, \mathcal{G}_I)_i \mid i = 1, \ldots, m\}$. We aim for a design of a baseline detector b^* given $\mathcal{T}_{\text{train}}$ which solves

$$b^* = \arg\max_{b \in \mathcal{B}} \sum_{(I, \mathcal{G}_I) \in \mathcal{T}_{\text{test}}} \langle \mathcal{G}_I, b(I) \rangle_\mu .$$

In the design phase of b^*, the set $\mathcal{T}_{\text{test}}$ is unknown and one is allowed to use solely $\mathcal{T}_{\text{train}}$. Hence, one has to ensure that b^* generalizes well from $\mathcal{T}_{\text{train}}$ to $\mathcal{T}_{\text{test}}$.

Since the proposed design consists of two stages and the first stage relies on deep learning techniques, an adaptation to a differently biased ground truth (produced by a different annotator) can be done easily by retraining the first stage without any fine tuning done by experts.

3.2 Stage I: ARU-Net

Typically, layout analysis algorithms directly work on the input image I or on a binarized version of it [17,23,29–31,33]. Instead, we employ a more goal-oriented transformation of the input image utilizing a neural network, which is trained in a supervised manner to assign a certain class to each pixel like in [21,37,38]. This is often referred to as pixel labeling or semantic segmentation. We will introduce the problem of pixel labeling utilizing hierarchical neural networks, followed by a description of the proposed ARU-Net architecture.

3.2.1 Pixel labeling: problem formulation

Definition 3.5 (*Neural pixel labeler*) A *neural pixel labeler* (NPL) for the classes $\mathcal{C} = \{c_1, \ldots, c_n\}$ is a hierarchical neural network $\Phi(\,\cdot\,; \boldsymbol{w}) : \mathcal{I} \to \mathcal{I}^{|\mathcal{C}|}$. The NPL is parametrized by $\boldsymbol{w} \in \mathbb{R}^N$. For $I \in \mathcal{I}$, it performs a prediction over all pixels and all possible classes $\Phi(I; \boldsymbol{w}) = C \in [0, 1]^{I_h \times I_w \times |\mathcal{C}|}$, where C sums to one over all classes and for all coordinates.

$C(:, :, c) = C_{:,:,c} \in \mathcal{I}$ denotes the image which encodes the pixel-wise prediction (confidence) for the cth class.

Definition 3.6 (*Pixel ground truth*) A Cartesian product $G_I \in \mathcal{I}^{|\mathcal{C}|}$ is called *pixel ground truth* (for image I) if it assigns exactly one class (one-hot-encoding) to each pixel.

Following the problem formulation of Sect. 3.1, we aim for an NPL, which was tuned on a training set and optimally performs on a test set. Assume there are training and test sets as stated above, but with pixel ground truth information instead of baseline ground truth information, which are denoted by $\widetilde{\mathcal{T}}_{\text{train}}$ and $\widetilde{\mathcal{T}}_{\text{test}}$. The performance of an NPL is evaluated in terms of the cross-entropy between the predicted and the ground truth distributions. The cross-entropy can also be motivated by a maximum likelihood estimation. This results in the cross-entropy loss function.

Definition 3.7 (*Loss function*) Let $\widetilde{\mathcal{T}}$ be a set of images along with their pixel ground truth and $\Phi(\,\cdot\,;\boldsymbol{w})$ is an NPL. The performance of Φ on $\widetilde{\mathcal{T}}$ is evaluated in terms of the (cross-entropy) *loss function*

$$-\sum_{(I,G_I)\in\widetilde{\mathcal{T}}}\sum_{y=1}^{I_h}\sum_{x=1}^{I_w}\sum_{c=1}^{|\mathcal{C}|} G_I(y,x,c)\ln\Phi(I;\boldsymbol{w})_{y,x,c}.$$

To improve the performance of the NPL on the training set, one can calculate the loss function's gradient with respect to the model parameters using the well-known technique of backpropagation [39]. The gradient is used to update the model parameters by gradient descent: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \tau \cdot \frac{\partial L}{\partial \boldsymbol{w}}(\Phi, \widetilde{\mathcal{T}}_{\text{train}})$ with a *learning rate* τ. This is repeated to successively adapt the NPL. The process of adapting the model by minimizing its loss is called *training*. Since one does not aim for a minimization of the loss on the training set, the system has to generalize to achieve high-quality results on the test set as well. To stabilize training, avoid over-fitting, and improve generalization, etc., dozens of techniques to improve the simple update rule which is stated above were introduced within the last years. Since the introduction of these is beyond the scope of this work, we refer to [40]. Details on techniques used within this work are given in Sect. 4.

3.2.2 ARU-Net: architecture

The ARU-Net is a special form of an NPL and is described in this section. We omit a formal introduction of the used neural network components and concepts and refer to the above-mentioned literature. Within the last few years, different architectures were proposed for the pixel labeling task. Most of them are based on convolutional neural networks (CNNs) [41]. A direct application of CNNs for semantic segmentation is presented in [37]. The presented fully convolutional network (FCN) combines local features to produce more meaningful high level features using pooling layers. Pooling reduces the spatial dimension. Thus, the result suffers from a coarse resolution. Noh et al. [38] tackle this problem by applying a deconvolutional network on the subsampled output of the FCN. The U-Net proposed in [21] 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. Additionally, error propagation for deep structures is facilitated, and the so-called vanishing gradient problems [42] are reduced. The U-Net is the basis for the proposed ARU-Net. We extend the U-Net by two more key concepts—spatial attention (A) and depth (residual structure (R)) to be described below. Remarkably, in contrast to the U-Net proposed in [21], we perform border padding. Hence, the spatial dimensions in each scale space of the U-Net are all the same, see Fig. 2 for a schematic representation of an U-Net. The output of the U-Net thus is a feature map (Z features in Fig. 2) of the same spatial dimension as the input. Hence, the U-Net becomes an NPL as defined in Definition 3.5 by adding a convolutional (to get pixel-wise predictions) softmax classifier on top which distinguishes between the different classes of $\mathcal{C}$.

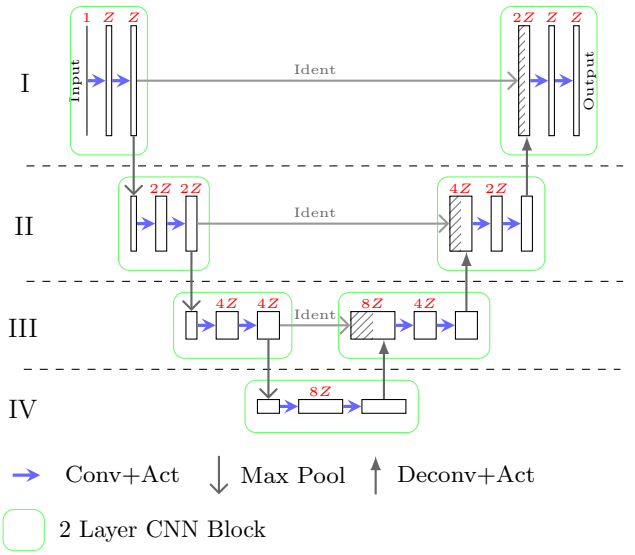


Fig. 2 U-Net—the input is an image of arbitrary spatial dimension. "Act" is the activation function thus the rectangles represent sets of activation maps. Each rectangle represents a 3-dim array ($\in \mathbb{R}^{h\times w\times z}$). Within each scale space (roman numbers), the feature map widths and heights are constant (encoded by the height of the rectangles). The number of feature maps Z is pictured by the width of the rectangles. Between adjacent scale spaces the spatial dimension decreases by a certain factor (2 in the figure) and the representative depth (number of feature maps) increases by the same factor

Remark 3.1 If the presented architectures are used for the pixel labeling task, it is implicitly assumed that such a classifier is always added to generate per class confidences at pixel level.

He et al. [22] introduce very deep neural networks which are still trainable and yield state-of-the-art results. This is

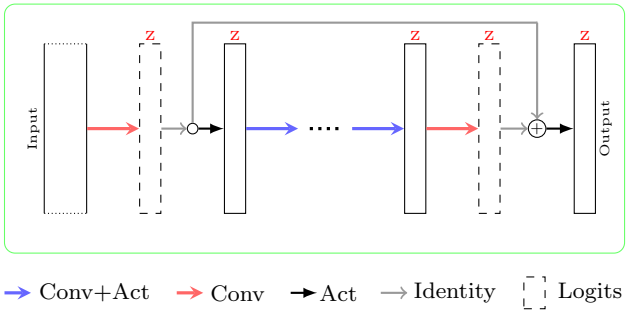


Fig. 3 Residual Block—the input is convolved and the resulting 3-dim array (the maps before passed through an activation function are referred to as logits) is used twice. At the first branch, it is passed through the activation function and further processed by several convolution layers. At the second branch, it is directly fed into a summation node. After a point-wise summation of the two logit maps, an activation function is applied. The shortcut enables for an easy identity propagation and error backpropagation. Arbitrarily many inner layers are possible

achieved using so-called residual blocks. Residual blocks introduce shortcuts, which enable the error backpropagation and identity propagation even for very deep structures. Hence, the vanishing gradient problems are reduced [22]. There are various different forms of residual blocks. The one used within this work is depicted in Fig. 3.

Definition 3.8 (*RU-Net*) An *RU-Net* is an U-Net with residual blocks.

That means, each of the 2 layer CNN blocks in Fig. 2 is replaced by a residual block as in Fig. 3.

To explicitly incorporate the potential to handle various font sizes, especially mixed font sizes on a single page, we introduce a pixel-wise (spatial) attention mechanism. For this purpose, we introduce an attention network (A-Net). The A-Net is a multilayer CNN which generates a single output feature map. The A-Net will be applied along with the RU-Net at different scales, and the same network weights are used on all scales (weight sharing). Specially, a scale pyramid is built by downscaling the input image $I = I_1$ several times. The resulting (scaled) images $I_1,\ I_2,\ I_4,\ I_8, \ldots,\ I_s$ (subscripts denote the scaling factors) are fed into the RU-Net and the A-Net. Trainable deconvolutional layers (of corresponding scales) are applied on the outputs of the RU- and the A-Net to obtain feature maps of spatial dimensions equal to the inputs. $A_1, \ldots, A_s$ denote the up-sampled feature maps of the A-Net, $\mathrm{RU}_1, \ldots, \mathrm{RU}_s$ of the RU-Net, respectively. After applying a pixel-wise softmax normalization for the attention maps

$$\widehat{A}_i(y,x) = \frac{\exp(A_i(y,x))}{\sum_{j\in\{1,2,\ldots,s\}} \exp(A_j(y,x))}$$

the normalized attention maps $\widehat{A}_i$ sum to one (pixel-wise). The feature maps RU_i are combined following

$$ARU = \sum_{i\in\{1,2,\ldots,s\}} \mathrm{RU}_i \odot \widehat{A}_i,$$

where $\odot$ is the Hadamard product. ARU is the input for the classifier to build a NPL, see Remark 3.1.

Definition 3.9 (*ARU-Net*) An RU-Net incorporating the described spatial attention mechanism is called *ARU-Net*, see Fig. 4.

The point-wise multiplication combined with the pixel-wise attention maps allow the ARU-Net to pay attention in different scales at different positions of the image. In Fig. 4, one can see that this behavior was indeed learned by the network. It seems like the RU-Net is specialized on a certain font size and the A-Net distinguishes between areas of different font sizes (bright and dark areas).

The ARU-Net as introduced can be used for any pixel labeling task, e.g., binarization, page detection and page segmentation. The purpose of the ARU-Net is defined and fixed by the number of classes and the ground truth data provided for training. In this work, we limit ourselves to the baseline detection problem introduced in Sect. 3.1. For this purpose, we introduce three different classes: baseline (bl), separator (sep) and other ($\varnothing$). The separators mark beginning and end of each text line. Although the separator information is implicitly encoded by the baselines, it is advantageous to explicitly introduce it as possible classification result. Especially, for baselines which are close together, e.g., such belonging to two adjacent columns, this approach helps to avoid segmentation errors. Pixel ground truth for the classes $\mathcal{C} = \{\mathrm{bl}, \mathrm{sep}, \varnothing\}$ could be automatically generated by Algorithm S.1 (supplements) given the baseline ground truth.

A sample image with baseline ground truth along with its generated pixel ground truth is depicted in Fig. 5. The prediction of a trained ARU-Net for this sample image is shown in Fig. 6a.

3.3 Stage II: baseline estimation

This subsection describes the second stage of the proposed approach. Baselines are estimated given the output of the ARU-Net. This task consists of three steps: superpixel calculation, state estimation and superpixel clustering, which are described in the following.

The trained ARU-Net generates an output $C \in [0,1]^{I_h \times I_w \times 3}$ for each image $I \in \mathcal{I}$. In the following, $B = C_{:,:,1}$ denotes the image encoding the confidence of each pixel belonging to a baseline and $S = C_{:,:,2}$ is the separator image, see Fig. 6a

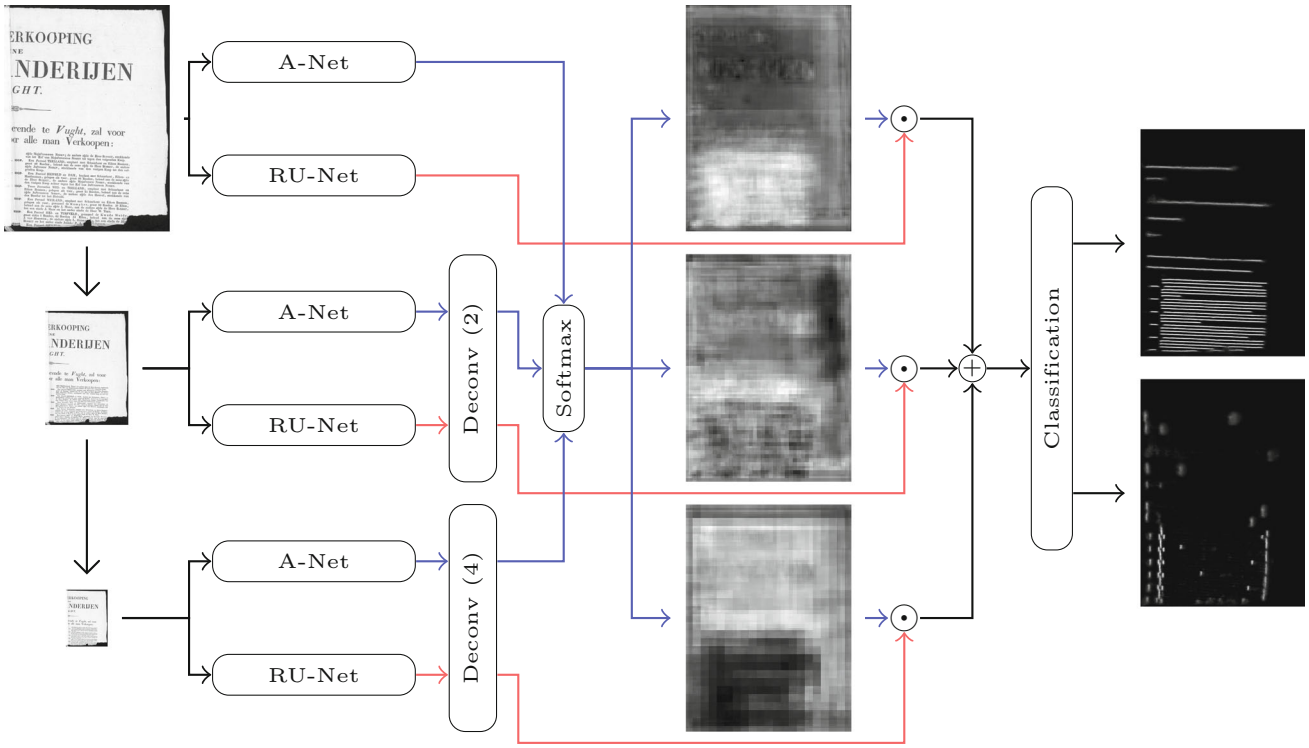
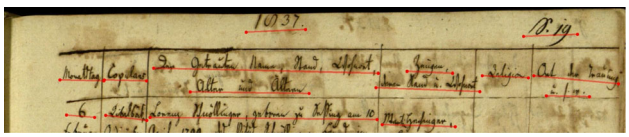
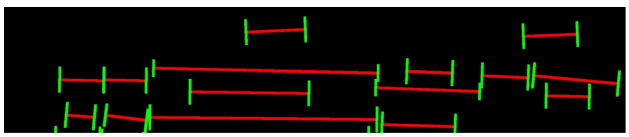


Fig. 4 ARU-Net—the input image and its downscaled versions are fed into the A-Net and R-U-Net (weight sharing across different scales). The results for the lower resolutions are deconvolved. The attention maps are passed through a softmax normalization. The brighter the map at a certain position, the more attention is paid to that position at the corresponding scale. The attention maps are point-wise multiplied with the feature maps of the RU-Net. The results are summed, and a classification is performed

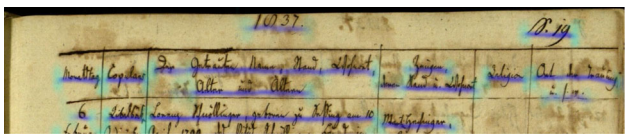


(a) Baseline ground truth – The baselines are defined by the red dots. Dots of the same baseline are connected.

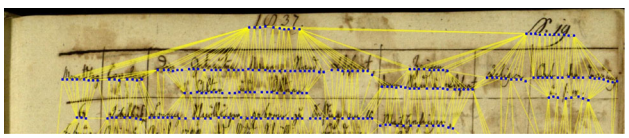


(b) Pixel ground truth – Green encodes the separator class, red the baseline class and black the "other" class. See supplements for an algorithm to automatically generate such GT.

Fig. 5 Baseline and pixel ground truth—these are shown for the top snippet of the image of Fig. 1 (color figure online)



(a) ARU-Net output – The estimated baselines B (blue) and separators S (cyan) are shown.



(b) Superpixel and neighborhood system – The calculated SPs (blue) are shown along with the resulting Delaunay neighborhood system $\mathcal{N}$ (yellow).

Fig. 6 Baseline detection process—two intermediate steps are shown for the top snippet of the image of Fig. 1 (color figure online)

3.3.1 Superpixel calculation

The number of all pixels in an image often exceeds several millions. To reduce the dimensionality of the problem (the number of pixels to be regarded for the baseline estimation), we limit ourselves to a subset of all pixels.

Definition 3.10 (*Superpixel*) Let $\mathcal{S} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\}$ be a subset of the image pixels of I (typically, $N \ll I_h \cdot I_w$ holds). An element of $\mathcal{S}$ is called *superpixel (SP)*.

Basically, the definition of a superpixel does not introduce any new concept. A SP is just a normal pixel which is somehow regarded to be of certain importance. Since it is a

frequently used term, we decided to introduce it via a definition. It is easy to see that the choice of the set of SPs is crucial for the overall performance. If there are no SPs for a baseline at all, this baseline will be missed. To calculate a suitable set of SPs, we utilize the baseline map B generated by the ARU-Net.

In a first step, B is binarized $B_b = B > b$ by an element-wise comparison of B with a confidence threshold b. The morphological skeleton $B_s = \mathrm{SKE}(B_b)$ is calculated for B_b following Lantuéjoul's formula [43]. All foreground pixels (pixels with an intensity of 1) of B_s build an initial set of pixels $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_M\}$. Its elements are sorted in descending order w.r.t. their baseline confidences. Finally, $\mathcal{S}$ is set up by iteratively adding pixels of this sorted list(beginning with the first pixel). To keep the number of SPs small, a new pixel $\boldsymbol{p}$ is added to $\mathcal{S}$ only if its distance to all other SPs exceeds a certain threshold $\|\boldsymbol{p} - \boldsymbol{q}\|_2 > d \quad \forall \boldsymbol{q} \in \mathcal{S}$, otherwise it is skipped. In Fig. 6b, the set of resulting SPs is shown. These SPs build the basis for the further clustering.

Remark 3.2 For all experiments, we have chosen fixed values of $b = 0.2$ (binarization threshold) and $d = 10$ (distance threshold). These demonstrated to be well suited for a wide range of different scenarios. Hence, they are not regarded as free parameters of the system which have to be further tuned. This also holds for the parameters which are fixed in Remarks 3.5 and 3.9.

3.3.2 Superpixel state estimation

Assume we can assign each SP to a certain text line. The state of an SP should encode meaningful characteristics of its text line. These characteristics will be defined and combined to build the state. This work is based on the previous work of [23, 33], but adapted to the characteristics of SPs extracted given the ARU-Net output, e.g., easier calculation of the local text orientation as well as a different smoothing cost formulation.

Definition 3.11 (*Local text orientation*) The *local text orientation* θ of an SP $\boldsymbol{p}$ is the slope of its text line's baseline at the coordinates closest (w.r.t. the Euclidean distance) to $\boldsymbol{p}$.

Definition 3.12 (*Interline distance*) The *interline distance* s of an SP $\boldsymbol{p}$ 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 $\boldsymbol{p}$.

Definition 3.13 (*State*) The state of an SP is the pair (θ, s) of its local text orientation and its interline distance.

In the following, we will describe a method to estimate the states of all SPs. The local text orientation will be calculated in a straightforward way utilizing solely the baseline image B and local information. On the other hand, the estimation of the interline distances combines local information of the text line's periodicity with the more global assumption that nearby SPs tend to have similar interline distances. For these approaches, the concepts of neighborhood and connectivity are mandatory and will be introduced.

Definition 3.14 (*Neighborhood system, edge, adjacent*) We call a subset $\mathcal{N} \subset \mathcal{S} \times \mathcal{S}$ *neighborhood system*. An element of $\mathcal{N}$ is called *edge* and denoted by $\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}$. $\mathcal{N}$ is not directed ($\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}} = \boldsymbol{e}_{\boldsymbol{q},\boldsymbol{p}}$). Two SPs $\boldsymbol{p}$, $\boldsymbol{q}$ are *adjacent* if $\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}} \in \mathcal{N}$. $\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}} \backslash \boldsymbol{p} \in \mathcal{S}$ denotes the SP $\boldsymbol{q}$.

Remark 3.3 In the following, the neighborhood system $\mathcal{N}$ for a set of SPs is always calculated by Delaunay's triangulation [44].

Definition 3.15 (*Connectivity function*) The line segment $g(\,\cdot\,; \boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}) : [0,1] \to \mathbb{R}^2$ defined by $g(\tau; \boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}) := \boldsymbol{p} + \tau\,(\boldsymbol{q} - \boldsymbol{p})$ connects the two pixels $\boldsymbol{p}, \boldsymbol{q}$ of the edge $\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}$. The function $\Gamma : \mathcal{N} \times \mathcal{I} \to [0,1]$ defined by

$$\Gamma(\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}, I) = \frac{\int_0^1 I(g(\tau; \boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}))d\tau}{\|\boldsymbol{p} - \boldsymbol{q}\|_2}$$

is called *connectivity function*. $I(g(\tau; \boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}))$ denotes the intensity of the pixel in I closest (w.r.t the Euclidean distance) to the real-valued coordinates $g(\tau; \boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}})$.

The connectivity function calculates the average intensity for a given image along the shortest path connecting two pixels. The local text orientation of each SP is estimated by $\theta_{\boldsymbol{p}} = \mathrm{LTO}(\boldsymbol{p}; \mathcal{N}, B)$ utilizing $\mathcal{N}$ and the baseline image B, see Algorithm 1. The LTO algorithm picks the two neighbors of an SP $\boldsymbol{p}$ with the largest baseline connectivity to $\boldsymbol{p}$ and determines the slope of the line passing through these neighbors.

Algorithm 1: Local Text Orientation of $\boldsymbol{p}$

input : SP $\boldsymbol{p}$, neighborhood system $\mathcal{N}$, baseline image B
output: local text orientation θ of $\boldsymbol{p}$

$\mathcal{M} \leftarrow \{\boldsymbol{e}_{\boldsymbol{q},\boldsymbol{r}} \in \mathcal{N} \mid \boldsymbol{q} = \boldsymbol{p} \vee \boldsymbol{r} = \boldsymbol{p}\}$
$\boldsymbol{L} \leftarrow$ sorted list of $\mathcal{M}$ ▷ sorted by $\Gamma(\boldsymbol{e}_{\boldsymbol{q},\boldsymbol{r}}, B)$
if $|\boldsymbol{L}| == 1$ **then**
  $\boldsymbol{e}_{\boldsymbol{q},\boldsymbol{r}} \leftarrow \boldsymbol{L}_1$ ▷ $\boldsymbol{L}_k$ is the k-th element
else
  $\boldsymbol{e}_{\boldsymbol{q},\boldsymbol{r}} \leftarrow (\boldsymbol{L}_1 \setminus \boldsymbol{p}, \boldsymbol{L}_2 \setminus \boldsymbol{p})$
return: $\theta \leftarrow \arctan\left(\frac{r_y - q_y}{r_x - q_x}\right)$

The periodicity of text lines in document images is utilized to calculate the interline distances. We determine the interline distance of an SP $\boldsymbol{p}$ by evaluating the regional text line periodicity around $\boldsymbol{p}$ as follows. For an SP $\boldsymbol{p}$, a circular region of diameter $d \in \mathbb{N}$ around $\boldsymbol{p}$, and a projection direction determined by the local text orientation $\theta_{\boldsymbol{p}}$, let

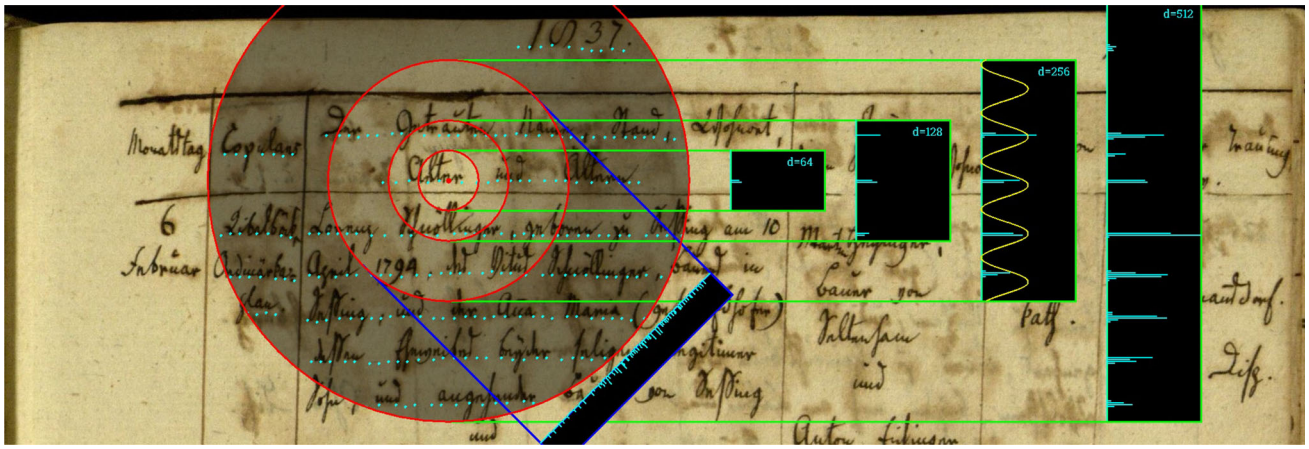


Fig. 7 Interline distance estimation—illustration of several projection profiles for a certain SP (red point). The profiles for different diameters $d \in \{64, 128, 256, 512\}$ and an orientation of 0° are shown in green. The winning period (interline distance) is drawn as yellow curve. In blue a histogram for a wrong orientation (45°) is shown (color figure online)

$\boldsymbol{h}^{\boldsymbol{p},d} = (h_1^{\boldsymbol{p},d}, \ldots, h_d^{\boldsymbol{p},d}) \in \mathbb{N}^d$ be the projection profile with respect to $\mathcal{S}$, see Fig. 7. For the calculation of $\boldsymbol{h}^{\boldsymbol{p},d}$, only SPs with a distance to $\boldsymbol{p}$ of less than $\frac{d}{2}$ are taken into account.

Remark 3.4 The projection profile $\boldsymbol{h}^{\boldsymbol{p},d}$ can be calculated very efficiently by utilizing the cross-product of the orientation vector $\boldsymbol{o} = (\cos(\theta_{\boldsymbol{p}}), \sin(\theta_{\boldsymbol{p}}))^T$ and the vectors $\overrightarrow{\boldsymbol{pq}}$ for $\boldsymbol{q} \in \mathcal{S}$ with $\|\boldsymbol{p} - \boldsymbol{q}\|_2 \leq \frac{d}{2}$.

To extract the regional periodicity inherent in the projection profile $\boldsymbol{h}^{\boldsymbol{p},d}$, a Discrete Fourier Transformation (DFT) is applied to $\boldsymbol{h}^{\boldsymbol{p},d}$ with resulting coefficients $\boldsymbol{H}^{\boldsymbol{p},d} = (H_1^{\boldsymbol{p},d}, \ldots, H_d^{\boldsymbol{p},d}) \in \mathbb{C}^d$. A coefficient $H_k^{\boldsymbol{p},d}$, $k \in \{1, \ldots, d\}$ corresponds to the portion of the signal with a period of $\frac{d}{k}$ to the entire signal $\boldsymbol{h}^{\boldsymbol{p},d}$. In the simplest case, the index k' of the dominant coefficient of $\boldsymbol{H}^{\boldsymbol{p},d}$ determines the interline distance s of $\boldsymbol{p}$ as $s = \frac{d}{k'}$. However, we may be forced to assign a different value to $\boldsymbol{s}$ due to additional constraints to be discussed in a moment. Therefore, we introduce a data energy value for each possible value $\frac{d}{k}$ of the interline distance s of $\boldsymbol{p}$. From energy, we then derive a data cost to be used within a cost minimization framework for finding the optimal interline distance.

Definition 3.16 (*Data energy, data cost*) The *data energy* of SP $\boldsymbol{p}$ and interline distance $\frac{d}{k}$ is given by $E_{\boldsymbol{p}}\left(\frac{d}{k}\right) = \frac{\left|H_k^{\boldsymbol{p},d}\right|^2}{\|\boldsymbol{H}^{\boldsymbol{p},d}\|_2^2}$. The *data cost* is calculated by $D_{\boldsymbol{p}}\left(\frac{d}{k}\right) = -\log\left(E_{\boldsymbol{p}}\left(\frac{d}{k}\right)\right)$.

Remarkably, the data energy is normalized such that it sums (over k) up to 1.0 for arbitrary $d \in \mathbb{N}$. To cover a suitable range of different interline distances as well as to be robust against disturbances due to close-by text regions of a different style, the projection profiles and DFTs are calculated for different diameters $d \in \{64, 128, 256, 512\}$ and $k \in \{3, 4, 5\}$. The choice of the values for d and k is application driven and results in reasonable interline distances (sorted list) of $\boldsymbol{S} := (170.7, 128.0, 102.4, 85.3, 64.0, 51.2, 42.7, 32.0, 25.6, 21.3, 16.0, 12.8)$.

In the following, we write $s_{\boldsymbol{p}}$ for the assigned interline distance $s = \frac{d}{k} \in \boldsymbol{S}$ of SP $\boldsymbol{p}$ and say $\boldsymbol{p}$ is labeled with $s_{\boldsymbol{p}}$. A labeling $\{s_{\boldsymbol{p}}\}_{\boldsymbol{p}\in\mathcal{S}}$ of $\mathcal{S}$ assigns an interline distance to each SP of $\mathcal{S}$. Following a greedy labeling strategy by assigning the interline distance with the highest data energy to each SP leads to a noisy result, see Fig. 8a. To reduce the noise effects, the influence of close-by SPs is taken into account. It is reasonable to expect that neighboring SPs tend to have similar interline distances. This expectation is encoded via a smoothing cost defined for adjacent SPs.

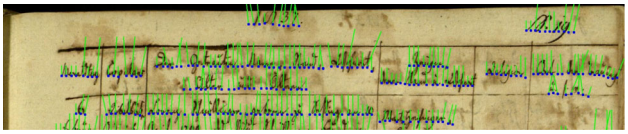
Definition 3.17 (*Smoothing cost*) For each $\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}} \in \mathcal{N}$ (and assigned interline distances $s_{\boldsymbol{p}}, s_{\boldsymbol{q}}$) the *smoothing cost* is defined by

$$V_{\boldsymbol{p},\boldsymbol{q}}(s_{\boldsymbol{p}}, s_{\boldsymbol{q}}) = \begin{cases} \sigma, & \langle s_{\boldsymbol{p}}, s_{\boldsymbol{q}}\rangle_{\boldsymbol{S}} \geq 4, \\ \langle s_{\boldsymbol{p}}, s_{\boldsymbol{q}}\rangle_{\boldsymbol{S}}, & \text{else.} \end{cases}$$

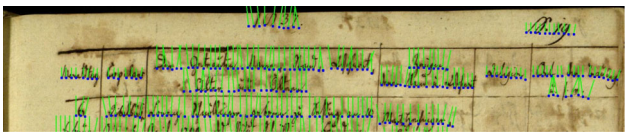
$\langle s_{\boldsymbol{p}}, s_{\boldsymbol{q}}\rangle_{\boldsymbol{S}}$ is the index difference of $s_{\boldsymbol{p}}$ and $s_{\boldsymbol{q}}$ in the sorted list $\boldsymbol{S}$ of possible interline distances, e.g., $\langle 16.0, 42.7\rangle_{\boldsymbol{S}} = 4$. Thus, the smoothing cost $V_{\boldsymbol{p},\boldsymbol{q}}(s_{\boldsymbol{p}}, s_{\boldsymbol{q}})$ becomes large if interline distances of different sizes are assigned to adjacent SPs. A maximum cost value of σ is used for huge differences in the interline distances. Setting σ to a large value prevents neighboring SPs to differ to much in their interline distances.

Definition 3.18 (*Labeling cost*) The *labeling cost* is given by

$$C\left(\{s_{\boldsymbol{p}}\}_{\boldsymbol{p}\in\mathcal{S}}\right) = \alpha \sum_{\boldsymbol{p}\in\mathcal{S}} D_{\boldsymbol{p}}\left(s_{\boldsymbol{p}}\right) + \beta \sum_{\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}\in\mathcal{N}} V_{\boldsymbol{p},\boldsymbol{q}}(s_{\boldsymbol{p}}, s_{\boldsymbol{q}}).$$



(a) Greedy states – The SP states for a greedy labeling using the highest data energy are shown.



(b) Smoothed states – The SP states for a cost optimal labeling are shown.

Fig. 8 SPs with their assigned states—the local text orientation of each SP is visualized by the orientation of the green lines (rotated by 90°). The length of the lines encodes the interline distance of the corresponding SP (color figure online)

The data cost and the smoothing costs are weighted by α and β, respectively, to form the labeling cost. The graph cut algorithm [45] is utilized to minimize the labeling cost. The final labeling is shown in Fig. 8b.

Remark 3.5 For all experiments, we have chosen fixed values of $\sigma = 25$, $\alpha = 1$ and $\beta = 1$.

3.3.3 Superpixel clustering

In the previous subsections, the calculation of SPs and their enrichment with state information was described. In a final step, this state information is utilized to cluster the SPs to build baselines. There will be a one-to-one assignment between clusters and baselines. In the following, we call a set of SPs *cluster*.

In this subsection, we formulate the clustering problem and introduce a greedy clustering procedure to solve the problem. Two assumptions which hold for baselines in general constitute the conditions for the clustering problem:

(I) Baselines should not exceed a certain curvilinearity value.
(II) Within the interline distance of a baseline, there are no other baselines.

Basically, assumption (I) claims that a baseline can be approximated by a polynomial function of a certain degree, see [23]. Assumption (II) is self-explanatory.

Remark 3.6 In the following, $\theta(\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\})$ denotes the average orientation and $s(\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\})$ the average interline distance of all SPs in $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$.

Definition 3.19 (*Curvilinearity value*) Let $\text{deg} \in \mathbb{N}$ and $\mathcal{S}$ be a set of SPs. Assume $p_{\mathcal{S},\text{deg}}(t) \in \mathbb{P}[t]$ is the polynomial which solves the linear regression problem in the monomials $t^0, t^1, \ldots, t^{\text{deg}}$ for $\mathcal{S}'$ which results from $\mathcal{S}$ by rotating all pixels by $-\theta(\mathcal{S})$. The root-mean-square regression error normalized by $s(\mathcal{S})$ is called *curvilinearity value* of $\mathcal{S}$ and is denoted by $cur(\mathcal{S}, \text{deg})$.

Remark 3.7 We fix $\text{deg} = 3$ and omit it in the following.

Definition 3.19 allows for an easy evaluation of (I). To test for (II) we will introduce the distance of two clusters. Remarkably, only distances orthogonal to the text orientation should be taken into account. First, the orthogonal component of the distance between two SPs is introduced. Afterward, this is generalized for two clusters of SPs.

Definition 3.20 (*Off-text distance*) Given two SPs $\boldsymbol{p}, \boldsymbol{q}$ and an orientation θ, the *off-text distance* of $\boldsymbol{p}$ and $\boldsymbol{q}$ is the length of the component of $\boldsymbol{p} - \boldsymbol{q} \in \mathbb{R}^2$ which is orthogonal to θ. It is denoted by $\|\boldsymbol{p} - \boldsymbol{q}\|_\theta$.

Remark 3.8 The off-text distance can be efficiently calculated by $\|\boldsymbol{p} - \boldsymbol{q}\|_\theta = \left|\left(\boldsymbol{p}_x - \boldsymbol{q}_x\right)\sin(\theta) - \left(\boldsymbol{p}_y - \boldsymbol{q}_y\right)\cos(\theta)\right|$.

Calculating the minimal pairwise off-text distance of all SPs of two clusters could result in a cluster distance distorted by SP outliers. Therefore, SPs in each cluster will be projected onto the corresponding regression curve obtained by the regression problem in Definition 3.19, before taking pairwise distances.

Definition 3.21 (*Regression curve*) Let $\mathcal{S}$, $\mathcal{S}'$ and $p_{\mathcal{S}}(t)$ be of Definition 3.19. The spatial t-range of $\mathcal{S}'$ is given by $t_{\min} = \min\{\boldsymbol{p}_x \mid \boldsymbol{p} \in \mathcal{S}'\}$ and $t_{\max} = \max\{\boldsymbol{p}_x \mid \boldsymbol{p} \in \mathcal{S}'\}$. A curve $c_{\mathcal{S}} : [0, 1] \to \mathbb{R}^2$ which results from rotating the graph of $p_{\mathcal{S}}(t)$ for $t \in [t_{\min}, t_{\max}]$ by $\theta(\mathcal{S})$ is called *regression curve* of $\mathcal{S}$.

The SPs in $\mathcal{S}$ are projected onto $c_{\mathcal{S}}$ (in direction $\theta(\mathcal{S}) + \frac{\pi}{2}$). The resulting projected SPs are denoted by $\mathcal{S}^c$. To achieve robust distance estimates even for curved and differently slanted text lines, we focus on SPs of the different clusters which are quite close to each other and furthermore take into account the slope of the regression curve at the specific SP positions instead of averaging over the entire text line.

Definition 3.22 (*Cluster distance*) Assume two clusters $\mathcal{S}_1$, $\mathcal{S}_2$ with regression curves $c_{\mathcal{S}_1}(t)$, $c_{\mathcal{S}_2}(t)$ and projected SPs $\mathcal{S}_1^c$, $\mathcal{S}_2^c$. The *cluster distance* is defined as

$$d(\mathcal{S}_1, \mathcal{S}_2) = \min_{\substack{\boldsymbol{p} \in \mathcal{S}_1^c, \boldsymbol{q} \in \mathcal{S}_2^c: \\ \|\boldsymbol{p} - \boldsymbol{q}\|_2 < 4 \cdot s(\mathcal{S}_1 \cup \mathcal{S}_2)}} \|\boldsymbol{p} - \boldsymbol{q}\|_{\theta^c(\boldsymbol{p}, \boldsymbol{q})},$$

where $\theta^c(\boldsymbol{p}, \boldsymbol{q})$ is the average slope of the corresponding regression curves at $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively.

Since, it is now possible to evaluate conditions (I) & (II), we will use this to introduce feasible sets of clusters. For this purpose, we will limit ourselves to partitions (a special kind of cluster sets) and require the baseline clusters to be $\mathcal{N}$-linked. The set of all partitions of a set $\mathcal{M}$ is denoted by $par(\mathcal{M})$.

Definition 3.23 (*$\mathcal{N}$-linked*) Let $\mathcal{S}$ be a cluster and $\mathcal{N}$ be a neighborhood system. $\mathcal{S}$ is *$\mathcal{N}$-linked* iff $\forall \boldsymbol{p}, \boldsymbol{q} \in \mathcal{S}\ \exists \boldsymbol{p}_0, \ldots, \boldsymbol{p}_N \in \mathcal{S} : \boldsymbol{p}_0 = \boldsymbol{p} \wedge \boldsymbol{p}_N = \boldsymbol{q} \wedge \boldsymbol{e}_{\boldsymbol{p}_i, \boldsymbol{p}_{i+1}} \in \mathcal{N}\ (0 \leq i \leq N-1)$ holds.

That means, for all pairs of SPs there are edges in $\mathcal{S}$ which connect these respective SPs.

Definition 3.24 (*Feasible*) For $\gamma, \delta \in \mathbb{R}_+$, $L \in \mathbb{N}$, a set of SPs $\mathcal{S}$ and a neighborhood system $\mathcal{N}$, we call a set of clusters $\mathscr{P} = \{\mathcal{S}_0, \ldots, \mathcal{S}_L\}$ *feasible* iff

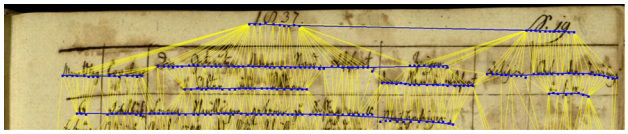
1. $\mathscr{P} \in par(\mathcal{S})$
2. $\forall i > 0 :\ \mathcal{S}_i$ is $\mathcal{N}$-linked
3. Conditions (I) and (II) hold:
   - $cur(\mathcal{S}_i) < \gamma \quad \forall i > 0$
   - $d(\mathcal{S}_i, \mathcal{S}_j) > \delta \cdot \max\{s(\mathcal{S}_i), s(\mathcal{S}_j)\} \quad \forall i, j > 0, i \neq j.$

The set of feasible sets of clusters is denoted by $feas_{\mathcal{N}}(\mathcal{S})$.

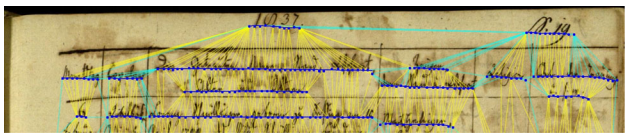
The clusters $\mathcal{S}_i,\ i > 0$ identify the baselines, $\mathcal{S}_0$ constitutes the clutter cluster containing SPs not belonging to any baseline. We identify the baseline corresponding to $\mathcal{S}_i$ with the polygonal chain of the projected SPs $\mathcal{S}_i^c$ which follow the regression curve $c_{\mathcal{S}_i}(t)$, see Fig. 9. The number $L \in \mathbb{N}$ of baselines is (*a* priori) unknown. In the following, we will incorporate domain knowledge to promote SPs belonging to different baselines not to be $\mathcal{N}$-linked. Hence, clusterings with erroneously connected baselines are not feasible anymore. This is done by a modification of the neighborhood system $\mathcal{N}$.

Since baselines of different text orientations should not contribute to the same cluster, we adjust the initial neighborhood system $\mathcal{N}$ by removing edges $\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}$ of SPs with substantially different local orientations: $\left|\theta_{\boldsymbol{p}} - \theta_{\boldsymbol{q}}\right| \mod \pi > \frac{\pi}{4}$. In addition, it is an ease to incorporate layout information by further adjusting $\mathcal{N}$. The layout information encoded by the separator image S (Fig. 6a) can be incorporated by taking into account the connectivity of SPs in S. All edges $\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}} \in \mathcal{N}$ for which a separator is crossed, i.e., $\Gamma(\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}, S) > \eta$ or $\max_\tau S(g(\tau; \boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}})) > 2 \cdot \eta$ (g of Definition 3.15) holds, are removed, see Fig. 9b.

Finally, a common scenario is the baseline detection with given text regions. We assume that the text regions are represented by closed polygonal chains $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_N$. This additional layout information (if available) is easy to integrate. All edges for which $\nexists \boldsymbol{R}_i :\ \boldsymbol{R}_i$ contains $\boldsymbol{p}, \boldsymbol{q}$ holds



(a) Without separator information – The entire neighborhood system (yellow) is shown.



(b) With separator information – The neighborhood system was reduced by removing edges (cyan) with high separator connectivity. The corresponding separator information is illustrated in Fig. 6a.

Fig. 9 Influence of the separator information—the resulting baselines (blue lines) with and without taking into account the separator information are shown (color figure online)

are removed. Roughly speaking, a closed polygonal chain *contains* an SP if for all "ways" from the SP to the image border one have to cross the polygonal chain. Hence, SPs which are part of different non-overlapping text regions are not $\mathcal{N}$-linked any more. Thus, each baseline $\mathcal{S}_i,\ i > 0$ is entirely contained in one text region for all feasible sets. The resulting neighborhood system is still denoted by $\mathcal{N}$.

Remark 3.9 For all experiments, we have chosen fixed values of $\gamma = 0.3$, $\delta = 0.5$ (Definition 3.24) and $\eta = 0.125$.

After reducing the neighborhood system, we now introduce the total baseline energy. We will assign an energy to all feasible sets and aim for an optimal one. This allows for the formulation of the clustering problem to be solved.

Definition 3.25 (*Total baseline energy*) Let B be a baseline image, $\mathcal{N}$ a neighborhood system and $\mathscr{P} = \{\mathcal{S}_0, \ldots, \mathcal{S}_L\}$ a set of clusters over $\mathcal{S}$. With $\mathcal{N}(\mathcal{S}_i) = \{\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}} \in \mathcal{N} \mid \boldsymbol{p}, \boldsymbol{q} \in \mathcal{S}_i\} \subset \mathcal{N}$ the *total baseline energy* is defined by

$$b(\mathscr{P}) = \sum_{i=1}^{L} \sum_{\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}} \in \mathcal{N}(\mathcal{S}_i)} \Gamma(\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}, B).$$

Finally, the clustering problem can be formulated as

$$\mathscr{P}^* = \operatorname*{arg\,max}_{\mathscr{P} \in feas_{\mathcal{N}}(\mathcal{S})} b(\mathscr{P}).$$

Because there could be a huge number of feasible sets of clusters for large $\mathcal{S}$, we introduce a greedy clustering algorithm. The proposed algorithm clusters edges of $\mathcal{N}$ instead of clustering SPs. If an edge is assigned to a cluster (set) of edges, we assign both corresponding SPs to the corresponding cluster of SPs. In a first step, the set of edges in $\mathcal{N}$ is

sorted in decreasing order w.r.t.

$$\left(1 - \frac{\|\boldsymbol{p} - \boldsymbol{q}\|_{\theta(\{\boldsymbol{p},\boldsymbol{q}\})}}{\|\boldsymbol{p} - \boldsymbol{q}\|_2}\right) \cdot \Gamma(\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}, B).$$

Hence, the sorting takes into account the B-connectivity value of an edge and discounts it if $\boldsymbol{e}_{\boldsymbol{p},\boldsymbol{q}}$ is rather orthogonal to $\theta(\{\boldsymbol{p}, \boldsymbol{q}\})$. Discounted edges are less likely part of a baseline and are therefore sorted to the end of the list. The sorted list is denoted by $\boldsymbol{N}$. This avoids that these edges are falsely assigned to baseline clusters which are composed of just a few correct edges (statistics of the cluster are not reliable, yet). Given $\mathcal{S}$ and $\boldsymbol{N}$, the proposed clustering algorithm assigns all edges to the clutter cluster ($\mathcal{S}_0$). It iteratively moves edges to baseline clusters such that the resulting set of clusters remains feasible and the total baseline energy increases. The algorithm is shown in detail in the supplements (Algorithm S.2).

4 Experiments

The experiment section is divided into 4 subsections. First, we investigate the influence of the training set size as well as the influence of different data augmentation strategies. This is followed by an investigation of the performance of the proposed method if it is applied to images with curved or arbitrarily oriented text lines. The third subsection presents and compares results of different versions of our proposed NPL architectures on the very heterogeneous and challenging cBAD dataset [25,26]. We perform statistical tests to show the statistical significance of the stated conclusion—the superiority of the proposed ARU-Net in a two-stage workflow over other architectures and a single-stage workflow. Finally, we compare the proposed method against other state-of-the-art methods on the datasets of 3 recently hosted competitions. As mentioned in Sect. 3, we will follow the similarity score of [25] (F value) to measure the quality of the baseline detection. The configuration for all experiments including the hyperparameters of the network architecture as well as the training is summarized in Table 1. This configuration is the result of an extensive search in the hyperparameter space and results in impressive results for various scenarios/datasets.

Since no early stopping based on the loss for any validation set is used, we train on the entire training set. The ARU-Net workflow for training and inference (Tensorflow code) as well as a trained network are freely available.[4] The ARU-Net training takes 3–24 h from scratch (dependent on the number of epochs and samples per epoch) on a Titan X GPU. The inference time per image ranges from 2 to 12 s per image on a dual-core laptop (Intel Core i7-6600U with 16GiB RAM), this reduces to 0.5 to 2 s running the ARU-Net on the Titan X.

4.1 Influence of training sample number and data augmentation

A major drawback of state-of-the-art approaches (Sect. 2) is the need for an extensive expert tuning if confronted with scenarios which are not already covered. But the eligibility for an usage at industrial scale depends on the possibility to easily adapt at reasonable cost. For approaches relying on machine learning, this reduces to two questions:

- What about the amount of ground truth needed?
- What about the effort of ground truth production?

Concerning the second question, we refer to the automatic generation of pixel ground truth given the baseline ground truth. The annotation of (polygonal) baselines for a document image is quite easy and does not need remarkable expert knowledge compared to, e.g., ground truth production for ATR systems for historical handwritings or even the text line annotation at surrounding polygon level. The effort is reduced to several minutes per page by using platforms such as Transkribus.[5] In the following, we want to examine the first question.

The influence of training dataset size along with different data augmentation strategies is investigated for the freely available Bozen dataset[6] [24], see Fig. S.1 (supplements). This dataset is a subset of documents from the Ratsprotokolle collection of Bozen composed of minutes of the council meetings held from 1470 to 1805 and consists of 400 pages. It is written in Early Modern German. Baseline ground truth information is available in form of PAGE[7] XML. The dataset is quite challenging concerning layout analysis issues. Most of the pages consist of a single main text region with many difficulties for line detection and extraction, e.g., bleed-through, touching text lines and marginalia. For the following experiments, we have randomly divided the Bozen set in a set of training samples $\mathcal{T}$ of size 350 and a test set of size 50. In a first step, we randomly set up a chain of subsets of $\mathcal{T}$

$$\mathcal{T}_1 \subset \mathcal{T}_3 \subset \mathcal{T}_5 \subset \mathcal{T}_{10} \subset \mathcal{T}_{30} \subset \mathcal{T}_{50} \subset \mathcal{T}_{100} \subset \mathcal{T}_{200} \subset \mathcal{T}_{350},$$

where $\mathcal{T}_i$ contains i training samples (pages and pixel ground truth). Since we expect an influence of the choice of training samples (= sorting of $\mathcal{T}$), we repeat the mentioned procedure 4 times. Notably, the test set remains untouched. Finally,

[4] https://github.com/TobiasGruening/ARU-Net.

[5] https://transkribus.eu.

[6] https://zenodo.org/record/218236.

[7] http://www.primaresearch.org/tools.

Table 1 Hyperparameters: the architecture and training configuration which were used in this work are described

Image pre-processing: input image I is downscaled by a factor of 2 for $\max\{I_h, I_w\} < 2000$, 3 for $2000 \leq \max\{I_h, I_w\} < 4800$ or 4 followed by a normalization to mean 0 and variance 1 (on pixel intensity level)
RU-Net architecture, see Figs. 2 and 3: number of scale spaces: 6, initial feature depth: 8, residual depth (activated layers in a residual block): 3, feature increasing and spatial decreasing factor: 2, activation function: ReLu, kernel size: 3×3, stride: 1
A-Net architecture: 4 layer CNN, activation function: ReLu, kernel size: 4×4, stride: 1, maxpooling of size 2×2 after each convolution, feature number: 12, 16, 32, 1
ARU-Net architecture, see Fig. 4: number of image scales: 5, classifier: 4×4 convolution layer with softmax activation
Training: weight initialization: Xavier, optimizer: RMSprop, learning rate: 0.001, learning rate decay per epoch: 0.985, weight decay on the L_2 norm: 0.0005, exponential moving average on the model weights: 0.9995, mini batch size: 1 (due to memory limitations of the GPU), early stopping: none (trained for a fixed number of epochs)

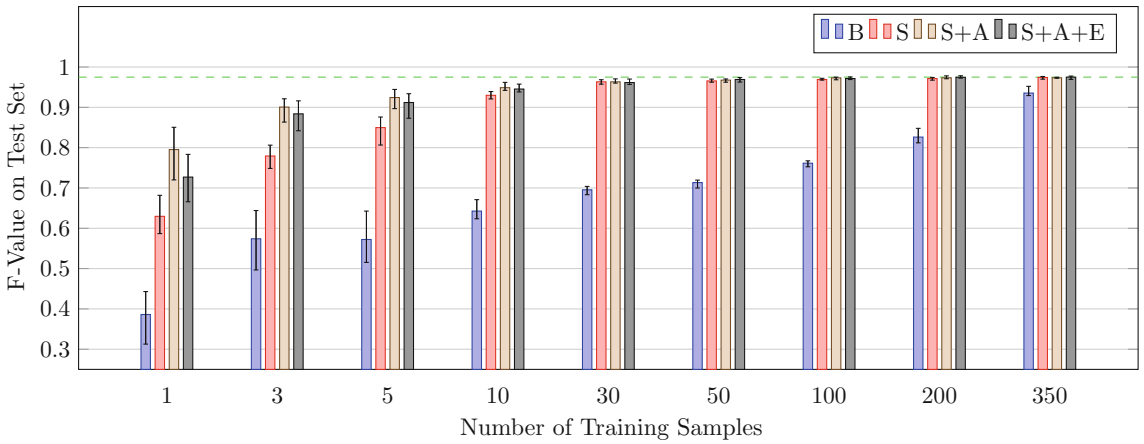


Fig. 10 Influence of the number of training samples and of different data augmentation strategies—the bar height represents the mean F value. The error bars encode min–max values of the 5 experiments (not the standard deviation). The dashed green line marks the maximum mean value of 0.975 achieved for 350 trainings samples. For a description of the different augmentation strategies: B, S, $S + A$ and $S + A + E$, see main text (color figure online)

we got 45 training sets—five of each quantity. For each set, we trained the RU-Net for 100 epochs with 256 images per epoch. Therefore, we randomly choose samples of the training set and remove them from the set. If each element of the training set was used for training once, we start again with the initial training set. Hence, it does not matter whether the number of training samples per epoch exceeds the size of the training set or not. This procedure guarantees the same amount of training samples shown to the networks in training independent of the size of the training set. The RU-Net was chosen instead of the ARU-Net, because of the homogeneity of the Bozen dataset concerning font size and resolution. We trained the RU-Net from scratch on all 45 sets in 4 different scenarios. For training purposes the image pre-processing mentioned in Table 1 is disabled. Instead, the training samples $(I, \mathcal{G}_I)_i$ are pre-processed following one of the four strategies:

1. Subsampled by a constant factor of 3 (no further data augmentation—one training sample per element of the training set)—B
2. Randomly subsampled by a factor $s \in [2, 5]$—S
3. $S+$ random affine transformation (three corner points of the image are randomly shifted within a circle of diameter $0.025 \cdot \max(\mathcal{I}_h, \mathcal{I}_w)$ around there original position)—$S + A$
4. $S + A+$elastic transformation [46]—$S + A + E$.

For the test set, the images were subsampled by the constant factor of 3 in all scenarios. The results of these 180 experiments are shown in Fig. 10. One can see that all 3 data augmentation strategies significantly improve the performance compared to the base (B) strategy. Notably, for small numbers of training samples the min–max difference is much larger than for higher number of training samples. Hence, if

just a few training samples are available, the choice of these is of importance. The best mean F value (0.975) is achieved for all 350 training samples with the $S+A+E$ strategy. Nevertheless, there only is a negligible loss in performance for 200 or 100 training samples. Even for 30 training samples, a F value of 0.963 is achieved for the $S+A$ strategy, which is sufficient for most applications, see Fig. S.1 (supplements). This results in a quite acceptable effort for ground truth production making the presented approach interesting even for industrial production. The $S+A$ data augmentation strategy will be the default for the rest of this work.

Of course, the presented numbers are not directly transferable to collections with pages of entirely different scenarios, e.g., census tables mixed with postal cards mixed with …One would expect that more than 30 training samples are necessary for this kind of scenario. Nevertheless, the presented experiment reflects a common situation: One has a robust baseline detector which was trained on very heterogeneous data (see Sect. 4.4.3), but this detector does not work satisfyingly well for a certain (in most cases quite homogeneous) collection. The numbers presented here give a hint concerning the effort of ground truth production necessary in this scenario.

4.2 Curved and oriented text lines

In this subsection, we demonstrate the ability of the introduced approach to handle curved or arbitrarily oriented text lines. In a first experiment, the test set of the Bozen dataset was deformed to contain arbitrarily curved text lines. For this purpose, we utilized trigonometric functions with random period to simulate curved text lines in the test phase. The RU-Net was trained (5 times) for 100 epochs with 256 samples per epoch on the Bozen training set using the $S+A+E$ augmentation strategy with strong elastic deformations. We choose elastic transformations in training, because they simulate curves of different amplitudes and frequencies in the same image. Furthermore, we increased the polynomial degree (Definition 3.19) to 5 to enable the system to handle the curvatures present in the test set.

Remark 4.1 Different methods were used to deform the images during training and test phases. Hence, the system had to learn the concept of curved text lines instead of an inversion of the image degradation method used in the training phase.

In a second experiment, we have trained an RU-Net (5 times) on arbitrarily oriented samples of the Bozen training set and evaluated the resulting networks on oriented pages of the test set. The results are shown in Table 2, and a few sample images are shown in Fig. S.2 (supplements). For the curved scenario, the results are as good as for the base scenario. In case of the oriented scenario, the results are slightly worse, but still excellent. This demonstrates the applicability for images with curved or oriented text lines without remarkable adaptation of the workflow. Finally, we have trained five models with all degradations (affine, elastic, rotation) and evaluated this model on the three different scenarios. The corresponding F values are depicted in Table 2. The system is worse than the experts for the base and curved scenarios, but for the oriented scenario it even benefits from the additional elastic transformations.

Table 2 Results for the Bozen test set: the results in the base (B), curved (C) and oriented (O) scenarios are depicted. Finally, the F-val for a single system trained with all degradations is shown for the different test lists (A)

	∅ P-val	∅ R-val	∅ F-val [min, max]
B	**0.977**	**0.973**	**0.975** [0.969, 0.977]
C	**0.980**	**0.969**	**0.975** [0.973, 0.976]
O	**0.963**	**0.966**	**0.964** [0.958, 0.967]
	F-val (B)	F-val (C)	F-val (O)
A	**0.953**	**0.957**	**0.968**

The P-, R- and F values are strongly related to the well-known precision and recall measures, see [25]
Bold values indicate that the number is the mean for several experimental runs instead of a single run result

4.3 U-Net versus ARU-Net versus single-stage workflow

In Sect. 3, we have introduced the ARU-Net in a two-stage workflow. In this section, we will investigate its superiority over the classical U-Net as well as over a "single-stage" workflow. For this purpose, we have trained the U-, RU-, ARU- and LARU-Net (each 5 times—random weight initialization and random training sample order) on the recently introduced cBAD dataset[8] [26]. The LARU-Net is an ARU-Net with a separable MDLSTM[9] layer at the lowest resolution to incorporate full spatial context. The details of the dataset are described in [25]. In our opinion, this is the most challenging freely available dataset at the moment. We have trained each network for 250 epochs, 1024 training samples each epoch using the $S+A$ data augmentation strategy. To assure the statistical significance of the posed superiority of the newly introduced architecture, we follow [47] and provide the results of a statistical analysis. The choice of appropriate statistical tests is quite limited since we cannot make any assumptions regarding the underlying distribution. We utilize 95% confidence intervals (CI) provided by nonpara-

[8] https://zenodo.org/record/257972.

[9] A separable MDLSTM layer is a concatenation of two (x- and y-direction) BLSTM layers.

Table 3 Results for the cBAD test set: the results for different neural network architectures and the workflow without Stage II (for the ARU-Net) are shown

	∅ F-val [95% CI]		CI	T–D
	Simple track	Complex track		
ARU I[a]	**0.9627** [0.9615, 0.9636]	**0.9081** [0.9071, 0.9095]		
U	**0.9714** [0.9701, 0.9721]	**0.9114** [0.9107, 0.9122]	✓	✓
RU	**0.9756** [0.9744, 0.9766]	**0.9182** [0.9165, 0.9203]	✓	✓
ARU	**0.9781** [0.9772, 0.9789]	**0.9223** [0.9214, 0.9230]	✓	✓
LARU	**0.9772** [0.9765, 0.9780]	**0.9233** [0.9217, 0.9249]	✗	✗

Each architecture is trained 5 times on the cBAD train set. The results are sorted with respect to computational effort. The last two columns indicate whether an architecture is superior to all before mentioned ones in terms of disjunct confidence intervals and the Tukey–Duckworth test

Bold values indicate that the number is the mean for several experimental runs instead of a single run result

[a]Single-stage workflow—baseline estimation by basic image processing methods (binarization of B followed by a CC analysis, no usage of S)

metric bootstrapping [48] as well as the Tukey–Duckworth test (level of significance: 5%) [49]. The results obtained are summarized in Table 3. The ARU-Net performs significantly (last two columns) better than all architectures with less computational effort. The LARU-Net could not prove its superiority and is therefore dismissed. Furthermore, the results show that the introduction of the second stage is beneficial for the overall performance. It has to be mentioned that the above comparison is not fair concerning the number of trainable parameters—U-2.16, RU-4.13, ARU-4.14, LARU-6.25 (in millions)—nor concerning the training or even inference time. The comparison is about different architectures which, theoretically, have different capabilities, and whether they make good use of them or not. For instance, the LARU-Net should be capable of incorporating a more detailed spatial context, but in fact it does not benefit (in our settings) from this capability.

4.4 Comparison against the State of the Art

In this subsection, we compare the proposed framework against the state of the art. We have chosen the 3 most recent competitions on text line detection for historical documents, namely ICDAR 2015 competition on text line detection in historical documents [14], ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts (Task 2) [50] and cBAD: ICDAR2017 Competition on Baseline Detection [36]. We will not further introduce the datasets or metrics used and refer to the competition papers.

4.4.1 ICDAR 2015 competition on text line detection in historical documents (ANDAR-TL)

The ARU-Net was trained on the cBAD training set.[10] This competition aims at the origin point (OP) detection. An OP is roughly spoken the lower left "corner" of a text line. Hence, we calculate the left most point of each detected baseline. This is the output of our system for this competition. The achieved results are shown in Table 4. Since the ARU-Net was not trained on the original training data, it is hard to compare its results to the other ones. Nevertheless, we would like to stress the fact that trained systems usually perform better if training set and test set are sampled from the same distribution. For example, the ARU-Net trained on the cBAD training set achieves an average F value of 0.9605 for the Bozen test set, which is worse than the F value of 0.9750 of the system trained solely on the Bozen training set, see Table 2. This indicates (but does not prove) the superiority of the presented method over the other methods in Table 4.

Table 4 Origin point (OP) detection results for the ANDAR-TL test set: results for the dataset of [14] are shown

	#COR	#DF	#DM	#FP	cost
UNIFR	2578	3022	6456	267	19.00
IA-2	5655	407	6032	102	14.51
A2iA-3[a]	6523	2490	2263	181	13.20
SNU [32]	7741	948	2700	25	9.77
[33]	8015	517	2860	21	8.19
Ours	9610	358	1942	83	<u>5.39</u>

Underlined value indicates the best result

[a]According to [28] this is an extension of [34]

#COR means the number of correctly detected OPs, #DF means the number of detection failures (no OP detected by the system), #DM means the number of detection misses (detected OP far away from the ground truth OP) and #FP means the number of false positives

4.4.2 ICDAR2017 competition on layout analysis for challenging medieval manuscripts (Task 2)

The DIVA-HisDB dataset consists of 150 annotated pages of three different medieval manuscripts with challenging layouts, see Fig. 11. The ARU-Net was trained for 250 epochs 1024 samples per epoch on the competition training

[10] The competition training data were not available to the authors.

Fig. 11 Results for an image of the CSG18 subset of the test set—the original image (only the main text lines were ground-truthed), the baseline image generated by the trained ARU-Net and the baselines detected by the proposed method are shown (from left to right)

Table 5 Results for the ICDAR2017 competition on layout analysis for challenging medieval manuscripts: the F values for Task 2 of all participants and the proposed method are shown for the different subsets of the test set

	CB55	CSG18	CSG863	Overall
CVML	0.9534	0.8734	0.9751	0.9340
BYU	0.9597	0.9879	0.9830	0.9768
CITlab	0.9896	0.9853	0.9716	0.9822
Ours	0.9980	0.9828	0.9889	0.9899

Underlined value indicates the best result

data[11] provided by the competition organizers. This allows an entirely fair comparison to the participant's results, see Table 5. The proposed method substantially outperforms the winning one and reduces the error (the gap to 1.0) by 43.26% (relatively). The specialty of this competition was that the methods should focus on a special kind of text, e.g., comments were not annotated as text. Hence, the ARU-Net had to learn to distinguish between different types of text. The output of the ARU-Net and the detected baselines for a sample image of the CSG18 subset of the test set are shown in Fig. 11. One can see that the ARU-Net entirely ignores all text entities not regarded (in this competition) as main text. Remarkably, no further information besides the input image is provided to the ARU-Net.

[11] http://diuf.unifr.ch/main/hisdoc/diva-hisdb.

Table 6 Results for the cBAD test set: the P-, R- and F values of all participants and of the proposed method for the simple and complex track of the cBAD: ICDAR2017 Competition on Baseline Detection are shown

	Simple track		Complex track	
	P/R-val	F-val	P/R-val	F-val
LITIS	0.78/0.84	0.807	–/–	–
[51]	0.75/0.93	0.827	–/–	–
UPVL	0.94/0.86	0.894	0.83/0.61	0.702
[16]	0.88/0.88	0.880	0.69/0.77	0.730
BYU	0.88/0.91	0.892	0.77/0.82	0.796
[52]	–/–	–	0.85/0.85	0.851
[53]	0.97/0.97	0.971	0.85/0.86	0.859
[54]	0.88/0.97	0.920	0.79/0.95	0.860
Ours	**0.98/0.98**	0.978	**0.93/0.92**	0.922

Bold values indicate that the number is the mean for several experimental runs instead of a single run result
Underlined value indicates the best result

4.4.3 cBAD: ICDAR2017 Competition on Baseline Detection

We compare our average result for the ARU-Net (see Table 3) to the results presented in [36], see Table 6. Our method performs considerably better in both tracks compared to all submissions. Especially, the increase in performance for the complex track is massive. Remarkably, the winning team uses an U-Net-based system with task specific pre- and post-processing. This indicates that the newly introduced concepts and parametrization, which are presented in this work, signif-

icantly improve the capability of the classical U-Net. Some results on chosen images of the cBAD test set are shown in Figs. S.3–S.5 (supplements). Notably, no further information besides the input image (and the text region information in the simple track) is provided to the ARU-Net nor to the second stage of the workflow during inference.

5 Conclusion

In this work, we presented a machine learning-based method for text line detection in historical documents. The text lines are represented by their baselines. The problem and the proposed method were introduced thoroughly. The proposed ARU-Net, which is a universal pixel labeling approach, was trained to predict the baseline position and the beginning and end of each text line. This enables the system to handle documents with complex layouts, e.g., tables, marginalia, multi-column layouts. We have shown that the system can be trained from scratch with manageably few training samples for a complex but homogeneous collection. Remarkably, ground truth production is quite cheap. A ground truth sample is just a page with annotated baselines, which can be done in a few minutes per page. Notably, this annotation process is possible without any expert knowledge. This is a big advantage compared to classical image processing-based methods, which typically demand for expert knowledge in the adaptation phase. Therefore, one can expect that an adaptation on collections, which are not covered by the neural network, is possible by a wide audience of users (not only computer scientists) with reasonable ground-truthing effort. The applicability of the proposed method was shown for straight, curved and oriented text lines as well as for a combined scenario. The superiority of the proposed ARU-Net in the two-stage workflow over the classical U-Net and over a simplified workflow was shown and statistically verified. Finally, we showed that the proposed method substantially outperforms the previous state of the art. Nevertheless, as one can see in Figs. S.3–S.5 (supplements) there are still errors made by the system, e.g., missed baselines (see Fig. S.4—bottom right), segmentation errors (see Fig. S.5—bottom left), false positives (see Fig. S.3—top left) or problems with strongly degraded documents (see Fig. S.4—top left). But these errors do not seem to follow a certain deterministic principle, which is not surprising for a method based on machine learning. However, we plan to test newly introduced concepts like capsules, memory augmentation and deeply supervised networks to further improve the system's performance.

Acknowledgements NVIDIA Corporation kindly donated a Titan X GPU used for this research. This work was partially funded by the European Unions Horizon 2020 research and innovation programme under Grant Agreement No. 674943 (READ Recognition and Enrichment of Archival Documents). Finally, we would like to thank Udo Siewert for his valuable comments and suggestions.

References

1. Isaac, A., Clayphan, R., Haslhofer, B.: Europeana: moving to linked open data. Inf. Stand. Q. **24**(2/3)
2. Causer, T., Wallace, V.: Building a volunteer community: results and findings from transcribe bentham. Digit. Humanit. Q. **6**(2), 1–28 (2012)
3. Sánchez, J.A., Mühlberger, G., Gatos, B., Schofield, P., Depuydt, K., Davis, R.M., Vidal, E., de Does, J.: TranScriptorium: a European project on handwritten text recognition. In: Proceedings of the 2013 ACM Symposium on Document Engineering, pp. 227–228. ACM, (2013)
4. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in Neural Information Processing Systems 21, NIPS'21, pp. 545–552. (2008)
5. Leifert, G., Strauß, T., Grüning, T., Wustlich, W., Labahn, R.: Cells in multidimensional recurrent neural networks. J. Mach. Learn. Res. **17**(1), 3313–3349 (2016)
6. Puigcerver, J., Toselli, A.H., Vidal, E.: Word-graph and character-lattice combination for KWS in handwritten documents. In: 2014 14th International Conference on Frontiers in Handwriting Recognition, pp. 181–186. IEEE, (2014)
7. Strauß, T., Grüning, T., Leifert, G., Labahn, R.: CITlab ARGUS for Keyword Search in Historical Handwritten Documents: Description of CITlab's System for the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task, CEUR Workshop Proceedings, Évora, Portugal, (2016)
8. Strauß, T., Leifert, G., Grüning, T., Labahn, R.: Regular expressions for decoding of neural network outputs. Neural Netw. **79**, 1–11 (2016)
9. Sanchez, J.A., Romero, V., Toselli, A.H., Vidal, E.: ICFHR2014 competition on handwritten text recognition on Transcriptorium datasets (HTRtS). In: Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR, vol. 2014-Decem, pp. 785–790. IEEE, (2014)
10. Pratikakis, I., Zagoris, K., Puigcerver, J., Toselli, A.H., Vidal, E.: ICFHR2016 handwritten keyword spotting competition (H-KWS 2016), In: Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR, pp. 613–618. IEEE, (2016)
11. Rusiñol, M., Aldavert, D., Toledo, R., Lladós, J.: Efficient segmentation-free keyword spotting in historical document collections. Pattern Recognit. **48**(2), 545–555 (2015)
12. Bluche, T.: Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In: Advances in Neural Information Processing Systems, pp. 838–846. (2016)
13. Konidaris, T., Kesidis, A.L., Gatos, B.: A segmentation-free word spotting method for historical printed documents. Pattern Anal. Appl. **19**(4), 963–976 (2016)
14. Murdock, M., Reid, S., Hamilton, B., Reese, J.: ICDAR 2015 competition on text line detection in historical documents. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, vol. 2015-Novem, pp. 1171–1175. IEEE, (2015)
15. Sudholt, S., Fink, G.A.: Phocnet : a deep convolutional neural network for word spotting in handwritten documents. In: Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR, pp. 1–6. (2016)
16. Renton, G., Soullard, Y., Chatelain, C., Adam, S., Kermorvant, C., Paquet, T.: Fully convolutional network with dilated convolutions for handwritten text line segmentation. Int. J. Doc. Anal. Recognit. (IJDAR) **21**, 1–10 (2018)
17. Arvanitopoulos, N., Süsstrunk, S.: Seam carving for text line extraction on color and grayscale historical manuscripts. In: Inter-

national Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 726–731. (2014)
18. Vo, Q.N., Kim, S.H., Yang, H.J., Lee, G.: Binarization of degraded document images based on hierarchical deep supervised network. Pattern Recognit. **74**, 568–586 (2018)
19. Tensmeyer, C., Davis, B., Wigington, C., Lee, I., Barrett, B.: PageNet: page boundary extraction in historical handwritten documents. In: Proceedings of the 4th International Workshop on Historical Document Imaging and Processing, HIP '1, pp. 59–64. ACM, New York, USA, (2017)
20. Chen, K., Seuret, M., Hennebert, J., Ingold, R.: Convolutional neural networks for page segmentation of historical document images. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, pp. 965–970. (2017)
21. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: MICCAI, pp. 234–241. (2015)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. (2016)
23. Ryu, J., Koo, H.I., Cho, N.I.: Language-independent text-line extraction algorithm for handwritten documents. IEEE Signal Process. Lett. **21**(9), 1115–1119 (2014)
24. Sánchez, J.A., Romero, V., Toselli, A.H., Vidal, E.: READ dataset Bozen (2016). https://doi.org/10.5281/zenodo.218236
25. Grüning, T., Labahn, R., Diem, M., Kleber, F., Fiel, S.: READ-BAD: A New Dataset and Evaluation Scheme for Baseline Detection in Archival Documents. arXiv preprint arXiv:1705.03311
26. Diem, M., Kleber, F., Fiel, S., Grüning, T., Gatos, B.: ScriptNet: ICDAR 2017 Competition on Baseline Detection in Archival Documents (cBAD) (2017). https://doi.org/10.5281/zenodo.257972.
27. Zahour, A., Likforman-Sulem, L., Boussalaa, W., Taconet, B.: Text Line segmentation of historical Arabic documents. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, vol. 1(2–4), pp. 138–142. (2007)
28. Eskenazi, S., Gomez-Krämer, P., Ogier, J.-M.: A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognit. **64**, 1–14 (2017)
29. Nicolaou, A., Gatos, B.: Handwritten text line segmentation by shredding text into its lines. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, pp. 626–630. (2009)
30. Saabni, R., Asi, A., El-Sana, J.: Text line extraction for historical document images. Pattern Recognit. Lett. **35**(1), 23–33 (2014)
31. Garz, A., Fischer, A., Sablatnig, R., Bunke, H.: Binarization-free text line segmentation for historical documents based on interest point clustering. In: Proceedings of 10th IAPR International Workshop on Document Analysis Systems, DAS 2012, pp. 95–99. IEEE, (2012)
32. Ahn, B., Ryu, J., Koo, H.I., Cho, N.I.: Textline detection in degraded historical document images. EURASIP J. Image Video Process. **2017**(1), 82 (2017)
33. Grüning, T., Leifert, G., Strauß, T., Labahn, R.: A robust and Binarization-free approach for text line detection in historical documents. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, pp. 236–241. (2017)
34. Moysset, B., Kermorvant, C., Wolf, C., Louradour, J.: Paragraph text segmentation into lines with Recurrent Neural Networks. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR 2015-Novem, pp. 456–460. (2015)
35. Moysset, B., Kermorvant, C., Wolf, C.: Learning to detect, localize and recognize many text objects in document images from few examples. Int. J. Doc. Anal. Recognit. (IJDAR) **21**, 1–15 (2018)
36. Diem, M., Kleber, F., Fiel, S., Gatos, B., Grüning, T.: cBAD: ICDAR2017 competition on baseline detection. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, pp. 1355–1360. (2017)
37. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 3431–3440. (2015)
38. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, vol. 2015 Inter, pp. 1520–1528. (2015)
39. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
40. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
41. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, vol. 86(11), pp. 2278–2323. (1998)
42. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, PMLR, vol. 9, pp. 249–256. (2010)
43. Serra, J.: Image Analysis and Mathematical Morphology, vol. 1. Academic Press Inc., New York (1982)
44. Delaunay, B.: Sur la sphere vide. Bulletin de l'Académie des Sciences de l'URSS **6**, 793–800 (1934)
45. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
46. Simard, P., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: Proceedings of the 7th International Conference on Document Analysis and Recognition, pp. 958–963. (2003)
47. Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, pp. 67–72. (2017)
48. Efron, B.: Better bootstrap confidence intervals. J. Am. Stat. Assoc. **82**(397), 171–185 (1987)
49. Tukey, J.W.: A quick compact two sample test to Duckworth's specifications. Technometrics **1**(1), 31–48 (1959)
50. Simistira, F., Bouillon, M., Seuret, M., Würsch, M., Alberti, M., Ingold, R., Liwicki, M.: ICDAR2017 competition on layout analysis for challenging medieval manuscripts. In: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, pp. 1361–1370. (2017)
51. Aldavert, D., Rusiñol, M., Manuscript text line detection and segmentation using second-order derivatives. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 293–298. IEEE, (2018)
52. Quirós, L.: Multi-task Handwritten Document Layout Analysis. arXiv preprint arXiv:1806.08852
53. Fink, M., Layer, T., Mackenbrock, G., Sprinzl, M.: Baseline detection in historical documents using convolutional u-nets. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 37–42. IEEE, (2018)
54. Oliveira, S.A., Seguin, B., Kaplan, F.: Dhsegment: a generic deep-learning approach for document segmentation. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 7–12. IEEE, (2018)