Heidelberg University

Final Project for the Lecture
Fundamentals of Machine Learning

Optical Character Recognition

Authors:
Lucas-Raphael Mueller
Roman Remme
Lucas Moeller

Supervisors:
Prof. Dr. Ulrich Koethe
Lorenzo Cerrone

May 23, 2018

Abstract

We present a document reading system that takes as input smartphone-quality images of machine printed text. We aim at producing a string holding the text shown in the image and to further extract IBANs from it.
The final system consists of maximally stable extremal region extraction to propose character candidates that are individually classified using a convolutional neural network.
In a test set of 39 images we cannot achieve great reliability, however the results are acceptable for good quality images and we are able to extract 69% of the contained IBANs correctly.

Contents

1 Introduction

This is the final report on a project required to complete the lecture "Fundamentals of Machine Learning" at Heidelberg University in the winter semester 2017/18 led by Prof. Dr. Ulrich Koethe.
We propose a simple document recognition system, that uses images from a smartphone camera as input.
Historically, the interest for optical document and thus character recognition (OCR) arose from applications like postal code and bank check evaluation. These early systems used calibrated scanning hardware. Therefore, it was sufficient to binarize the image using a threshold and to apply connected component analysis (CCA) in order to extract character candidates, which can ultimately be fed through a classifier. Such systems are still at use in many applications [4].
Today cameras in mobile devices offer a whole new range of potential applications for OCR. However, such images are obviously not as well calibrated and are taken in a brought variety of conditions. Potential applications are not limited to document images but may also apply to complicated scene images containing various kinds of text. Classical OCR systems perform poorly on such images and more sophisticated methods are needed.
Scene text recognition is a challenging task and still an open research question. Today's methods range from improvements of the classical approach with independent recognition and classification systems [7, 5, 6] to fully integrated end-to-end systems [16, 1, 17, 18]. A good review is provided by Zhu *et al.* [19]. As in many cases, especially, the rise of artificial neural networks has led to great improvement in the field.
In this project we restrict ourselves to document recognition with an extension to the use-case of IBAN recognition. The presented system consists of independent segmentation and classification algorithms. For segmentation we compare a simple CCA algorithm and a maximally-stable-extremal-regions identifier. The classification task is done by a convolutional neural network (CNN). Prior to classification potential character candidates are additionally filtered on a heuristic bases and clustered into lines and words. Two different algorithms to correct for rotation of the text content are evaluated.
The rest of the report is organized as follows: section 2 provides the theoretical background on the used algorithms and the details on the heuristic filtering. In section 3 we evaluate the individual components of the system and compare algorithms. Finally, in subsection 3.4 the application to the

recognition of IBANs is tested.

2 Methods

2.1 Artificially Generated Images

To test our algorithm and to train the neural network we generate artificial images. They consist of black text on a white background. The text itself, its font style and the number of lines as well as its position in the image and rotation are chosen randomly. To emulate potentially typical artifacts resulting from compression we store the images as RGB jpg-files before utilizing them. For examples see Figure 7 or Figure 8. We used a selection of 118 copyright free fonts, from the GitHub repository [11].

2.2 Simple Connected Components

One way of extracting characters from an image is through "connected components analysis (CCA)" which typically consists of three steps. First, the image needs to be binarized, which in a gray scale image can be done by thresholding the image. A color image needs to be transformed to gray scales previously. The threshold can be a simple global one for good quality images without major intensity variation. Alternatively, a dynamic threshold can be used as for instance Niblack's or Sauvola's methods [12].
Subsequently, connected components are found by iterating a 3×3 kernel through the resulting binary image from top left to bottom right. If the center pixel is a foreground pixel its top left four neighbors are checked for labels. If none of them are labeled a new label is introduced. If only a single label occurs among them the center pixel is given the same label. If several labels occur they are reported to be equivalent.
The equivalence classes of labels are efficiently managed by a disjoint set data structure. In a second iteration through the image equivalent labels are replaced by the smallest label of the respective equivalence class.

2.3 Maximally Stable Extremal Regions

Another approach to extracting characters is to identify maximally stable extremal regions (MSERs) which has been identified as one of the best region

detectors [10]. A region R_i^g is again found by applying connected component analysis to a binary image I^g that results from applying the threshold g to the original image. However, not a single threshold is used but the process is repeated for a number of thresholds $g \in G$. For each resulting region R_i^g a stability value $\phi(R_i^g)$ is calculated as follows:

$$\phi(R_i^g) = -\frac{|R_i^g| - |R_j^{g-\Delta}|}{|R_j^{g-\Delta}|} \tag{1}$$

$|.|$ denotes cardinality, i.e. the number of pixels belonging to a region, Δ is a model parameter and refers to the variation of the threshold value [3]. Note that $R_j^{g-\Delta}$ may not be distinct. In this case $R_j^{g-\Delta}$ must be the union of all regions $R_c^{g-\Delta}$ that are subsets of R_i^g.
A region whose stability value is locally maximal is called maximally stable. Intuitively, these are regions that change little upon a variation of the threshold.
MSERs can be found efficiently in linear time as was shown by Nistér and Stewénius [15]. As MSERs are determined in a local fashion they need not be the desired characters but may only be parts of them. Therefore, the extracted regions are filtered according to their bounding boxes. If the bounding box of region R_i lies within the one of region R_j they are considered equivalent and R_i is omitted. This can again be efficiently handled by the disjoint set data structure.

2.4 False Positive Elimination

Clearly, not all letter candidates proposed by connected component analysis or MSER extraction are actually characters.
The elimination of MSERs that are subsets of others was described in subsection 2.3. Other false positive components that may result from either CCA or MSER are evaluated according to the following heuristic, that is a much simplified and slightly different version of the method proposed by Shi *et al.* [14].
Two components are considered neighbors if their euclidean distance d_E is smaller than a dynamic threshold value and their mean intensity difference d_C as well as their ratio of area and aspect ratio are smaller than respective hard thresholds T_C, T_A and T_r:

$$
\begin{aligned}
& d_E(R_i, R_j) && < min[max(w_i, h_i), max(w_j, h_j)] && (2)\\
\wedge\, & d_C(R_i, R_j) && < T_C && (3)\\
\wedge\, & max(A_i, A_j)/min(A_i, A_j) && < T_A && (4)\\
\wedge\, & max(r_i, r_j)/min(r_i, r_j) && < T_r && (5)
\end{aligned}
$$

A_i is the area of component R_i's bounding box and $r_i = w_i/h_i$ is its aspect ratio.
Components that do not have neighbors according to these conditions are omitted.

2.5 Pixel-Based Rotation and Line Segmentation

Without further information the correct orientation of an image containing a block of text can be determined from projections of the image [13]. For this the image is rotated by a number of angles $\Theta \in A$. At each rotation its normalized projection onto the vertical axis P_Θ is calculated as well as the according information entropy

$$H(P_\Theta) = -\sum_i^B p_i \log(p_i). \tag{6}$$

Here i is the bin index in P_Θ and B is the number of bins. p_i is the relative intensity contribution of bin i to the rotated image. In other words p_i is the probability for an intensity contribution to fall into a pixel that itself falls into bin i. $H(P_\Theta)$ can then be understood as the expectation value for the information content of an intensity contribution in P_Θ:

$$H(P_\Theta) = E(I(p_i)) = E(-\log(p_i)) \tag{7}$$

A text block with horizontal line orientation minimizes entropy, because as intensity contributions fall into few bins their expected information content is little. They are "maximally ordered". Hence, the optimal rotation angle $\Theta^* \in A$ can be found from

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, H(P_\Theta). \tag{8}$$

An intensity contribution is to be seen as the minimum value by how much intensity can change. Typically, this is one on a scale from zero to 255. Note that the bin width needs to be significantly smaller than line height.
Subsequent to the correction for rotation, however, still without further knowledge about potential characters a line segmentation can be done from a projection onto the vertical axis. To extract minimum and maximum vertical coordinates for each line a one dimensional connected component analysis is performed on the resulting normalized histogram using a predefined threshold. In order to find horizontal boundaries of each text line the process can be repeated independently for each line in the other direction.
Obviously, this approach including both rotation and line segmentation is computationally expensive, because every pixel value needs to be processed. Therefore, we propose another approach to correct for rotation, that makes use of additional information in the next section.

2.6 Component-Based Rotation

As opposed to the approach for rotation correction described in the previous section, in this section we propose an alternative that makes use of the positions of character candidates. Given n bounding boxes of proposed characters, first their centers $m_i, i = 1..n$ are computed. Those take the position of all pixels in the previous approach: For a range of angles $\Theta \in A$, their projections y_i onto the vertical axis is computed. Next, a heuristic approach to find the correct rotation angle is used: If the image is oriented correctly, the projections of the centers m_i can be expected to fall into tight clusters, corresponding to the lines of text. To find these clusters, the mean-shift algorithm supplied by skikit-learn is used. It is well suited, as it does not reqire the number of clusters as input, but only a bandwith parameter corresponding to typical cluster width. This parameter is estimated as a fifth of the average bounding box height, which worked well in all cases tested by us.

The Mean Shift algorithm is a controid based clustering algorithm. In each step, a candidate centroid y_i is moved towards the centroid of its neighbourhood $N(y_i)$, defined as the set of data points with a distance to y_i smaller

than the previously defined bandwidth.

$$z_i^{t+1} = y_i^t + m(y_i^t), \tag{9}$$

$$m(y_i^t) = \frac{\Sigma_{y_j \in N(y_i^t)} K(y_j - y_i^t) y_j}{\Sigma_{y_j \in N(y_i^t)} K(y_j - y_i^t)} = \frac{\Sigma_{y_j \in N(y_i^t)} y_j}{|N(y_i^t)|} \tag{10}$$

This is repeated until some convergence criterion is met. In our case, all data points are chosen as initial cluster centers, and are clustered based on where they end up after the iterative process.

A loss is computed to judge the quality of the clustering: The mean squared distance of the y_i to their respective cluster centers $\hat{y}_i$ is multiplied with the number of clusters n_Θ to get the loss L_Θ:

$$L_\Theta = n_\Theta \sum_i ||y_i - \hat{y}_i||^2 \tag{11}$$

Finally, we use the angle $\Theta^* = \text{argmin}_\Theta(L_\Theta)$. A visualization of the method, parallel to the one for the pixel-based method in Figure 7 can be found in Figure 8. With the clustering corresponding to Θ^*, the character proposals are additionally partitioned into lines of text. The lines are sorted by their average vertical position, and the characters in each line by their centers horizontal position. The only step left to generate the desired output is to classify the characters.

2.7 Character Classification

To finally classify the character proposals, we employ a small convolutional neural network (CNN). For the character proposals to be fed into the CNN, some preprocessing is necessary: For each character proposal, the image region in the bounding box is cut out. In order to make it possible to distinguish lower-case characters from their upper-case versions, the bounding box height is first normalized per line of text: All boxes are enlarged at the top to match the tallest box in their corresponding line. Next, the cut-outs are re-scaled to 32 by 32 pixels after padding with zeros if necessary. They are individually re-scaled such that the average value of pixels belonging to characters (according to the segmentation algorithm) is 1, and the average value of background pixels is approximately 0.

For the neural net, a modified LeNet [9] is used. The Network consists of a feature extractor and a classifier. The former is made up of two convolutional

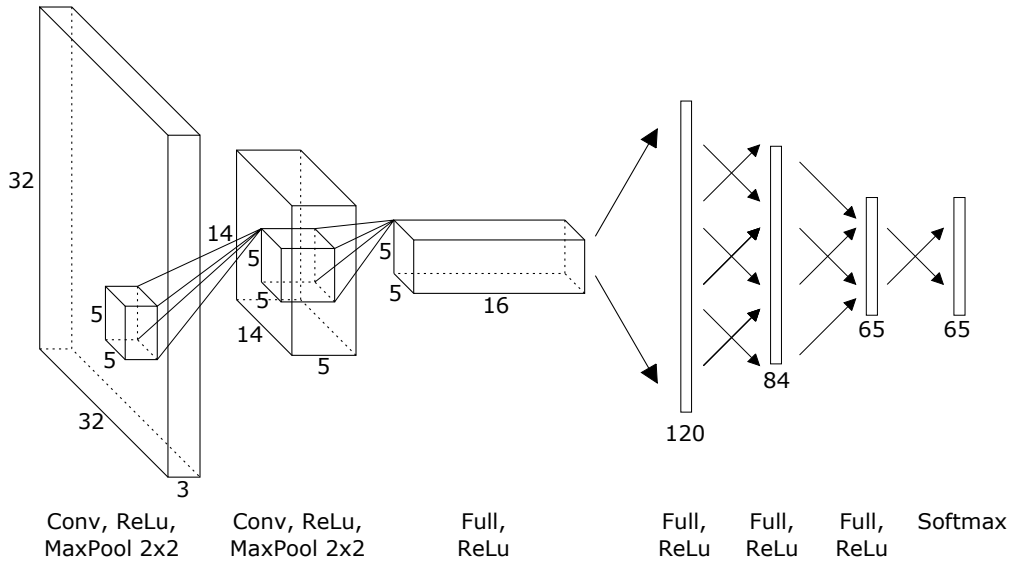


Figure 1: Schematic view of the neural networks architecture.

layers, each followed by a ReLU activation function and 2x2 max pooling. The latter has four fully connected layers. A final softmax layer gives the class probabilities. See Figure 1 for a schematic view of the model.

3 Results and Discussion

3.1 Character segmentation

Figure 2 shows the proceeding of the CCA algorithm. Image (b) is the result of the kernel's first pass through the image. The found connected components are given the same colour. It can be seen that U-shaped structures are not recognized as being connected, because the kernel is iterated from top left to bottom right of the image. Only after eliminating equivalent labels the correct connected components are found as seen in image (c). (a) shows the original image with the resulting bounding boxes for connected components in red.

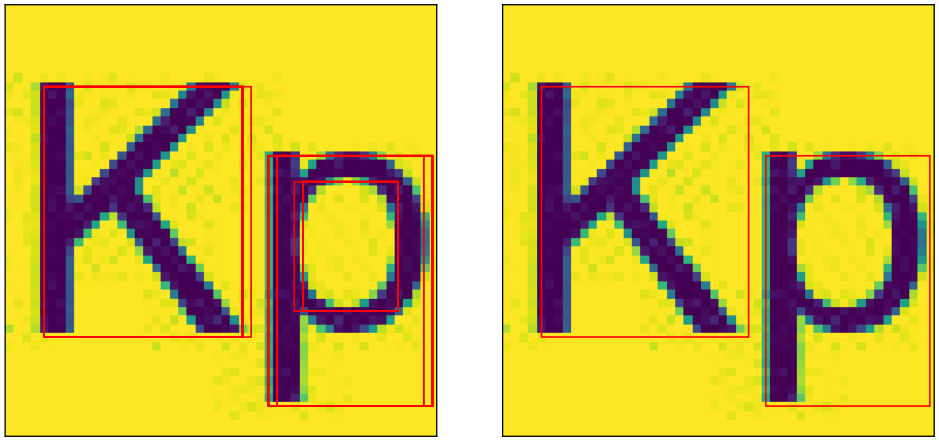
Figure 3 shows the results of the MSER extraction. In (a) all components that result from local maxima in the stability value ϕ as defined in Equation 1 are shown. Due to the locality of this criterion several MSERs can be found

(a) (b) (c)

Figure 2: Connected component analysis: (a) shows the original image with the resulting bounding boxes of the characters, (b) depicts the extracted components after the first pass through the image and in (c) the final components after eliminating equivalences are seen.

for a single character. Further, the algorithm is run twice to find both light MSERs in a dark environment and vice versa. This can be seen from the fact that the inner part of the "p" is identified as a component itself. In (a) the filtering as described in subsection 2.4 has been applied and only one component per character remains.
In Figure 4 an example for a more complex real image is shown. Again in (a) the unfiltered extracted MSERs are visualized. (b) demonstrates that the filtering successfully eliminates false positives in the image's background and reliably finds a single component for each character. The only exception are letters containing dots.

As a comparison Figure 6 shows the performance of the CCA algorithm on the same real image. Clearly, the algorithm struggles finding the characters with a standard threshold of 0.5 in the normalized image as is shown in (a). Especially, the intensity gradient in the background resulting from the reflecting book cover is a problem. Using an optimized threshold plus the filtering as described in subsection 2.4 the CCA is actually able to identify the characters correctly. However, manual adjustment of the threshold is obviously impractical.
Figure 5 shows an example for an image that is problematic for the MSER algorithm, because the writing is very blurry. We also use this example to demonstrate the effect of the parameter Δ (see Equation 1). For a small value of Δ as in subfigure (a) the algorithm finds more local stability maxima and

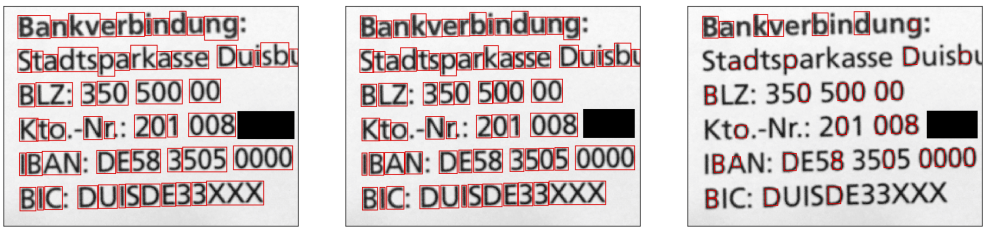


(a) (b)

Figure 3: MSER extraction: (a) shows the extracted components' bounding boxes before filtering them and in (b) they have been filtered as described in subsection 2.4.

(a) (b)

Figure 4: MSER extraction on a real image with a more complex background: (a) shows the bounding boxes before filtering them and in (b) they have been filtered as described in subsection 2.4.



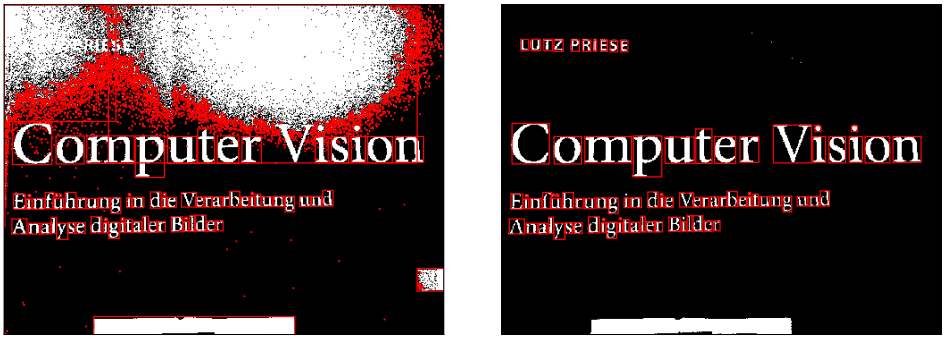
(a) $\Delta = 5$ (b) $\Delta = 10$ (c) $\Delta = 15$

Figure 5: MSER extraction with different values for Δ (see Equation 1) in a low quality image.

due to the filtering (subsection 2.4) MSERs containing single characters are often eliminated in favour of larger ones combining several characters. The unfiltered image is not shown. For larger values of Δ an increasing fraction of characters is not identified as MSERs anymore. A good compromise can hardly be found for this image. Subfigure (b) shows the optimal solution with $\Delta = 10$, but it still contains errors. We find that $\Delta = 15$ works well on good quality images.
Generally, blur is a known issue with MSER algorithms. One interesting approach of addressing it is proposed by Chen *et al.* who combine MSER with Canny edge detection [2]. However, from this paper it is not clear to us how the Canny edges, that are generally not enclosed boundaries are utilized exactly to crop the proposed MSERs.

3.2 Rotation and Line Segmentation

Our first approach towards rotation and line segmentation was the pixel based version described in subsection 2.5. Figure 7 shows an example of the algorithm applied to an artificial image. For the rotation 50 angles in the range $[-20°, 20°]$ were evaluated. The original image is seen in (a). Entropy as a function of the rotation angle (Equation 6) for this image is seen in (d) and is minimal at an angle of approximately $-15°$. Subfigures (b) and (e) show the rotated image with two different line segmentations. The respective histograms are shown in (c) and (f). The first segmentation is correct, but in the second one a common error occurs. Due to a slight shift of the threshold a false line between the second and third one is identified from the tail of the "g". The chance for such error decreases for longer text lines. However,



(a) (b)

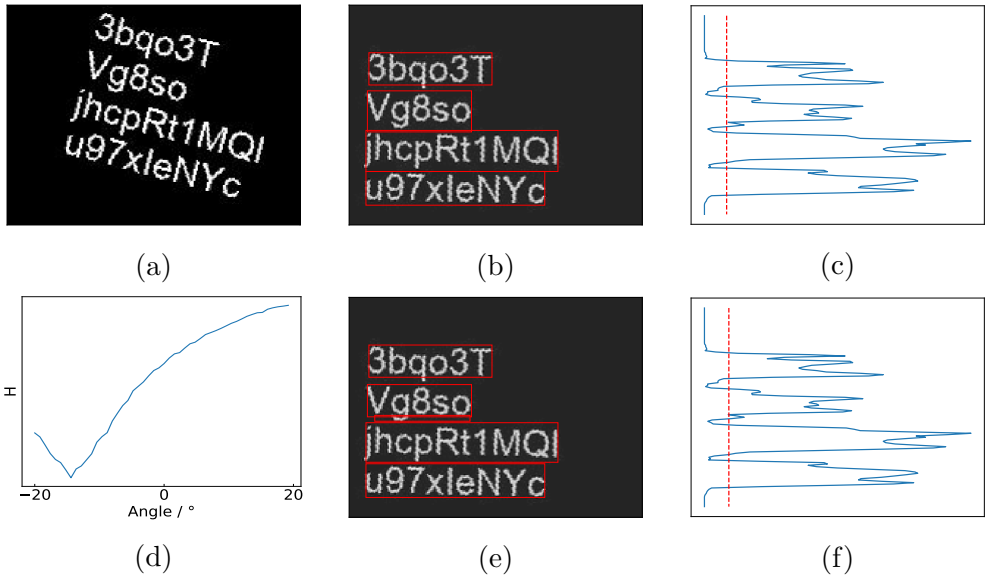
Figure 6: CCA on a real image with a more complex background: (a) shows the bounding boxes of components found with a standard threshold of 0.5 and no filtering, where (b) is the result of an optimized threshold plus filtering.

choosing the right threshold value remains an issue.

This is no issue when using our second algorithm based only on bounding box positions (subsection 2.6). Figure 8 shows an exemplary application. 100 angles in the interval $[-45°, 45°]$ are sampled. Although using a higher number of angles, this algorithm is approximately one magnitude faster than the first one, the reason being the much lower number of character proposals in comparison to pixels. In Figure 8 (d), the projections of the bounding box centers as a function of the angle are shown. Well established clusters are observed in the vicinity of the correct rotation angle of about 17°. This is quantified by the clustering loss displayed in subfigure (c), by which the rotation angle is chosen. Overall, this algorithm yields very reliable results, given that the characters are segmented correctly. It is also much faster and additionaly reliably partitions the text block into lines, hence we exclusivly use it in all following experiments.

3.3 Network Training

57623 characters are extracted from 3000 generated images (see section 2.1) and split into a training (80%) and validation (20%) set. The network is meant to distinguish between lower and upper case Latin letters, numbers between zero and nine as well as '(', ')' and '@'.
It is trained using the Adam optimizer [8] with a learning rate of 3×10^{-3}.



(a) (b) (c)

(d) (e) (f)

Figure 7: Application of the pixel-based rotation and line segmentation algorithm on a randomly generated artificial image: (a) is the original image. (d) shows entropy as a function rotation angle. (b) and (e) are the outcomes with red bounding boxes for the lines using two slightly different thresholds for line segmentation. (c) and (f) are the respective histograms with the used thresholds in red.

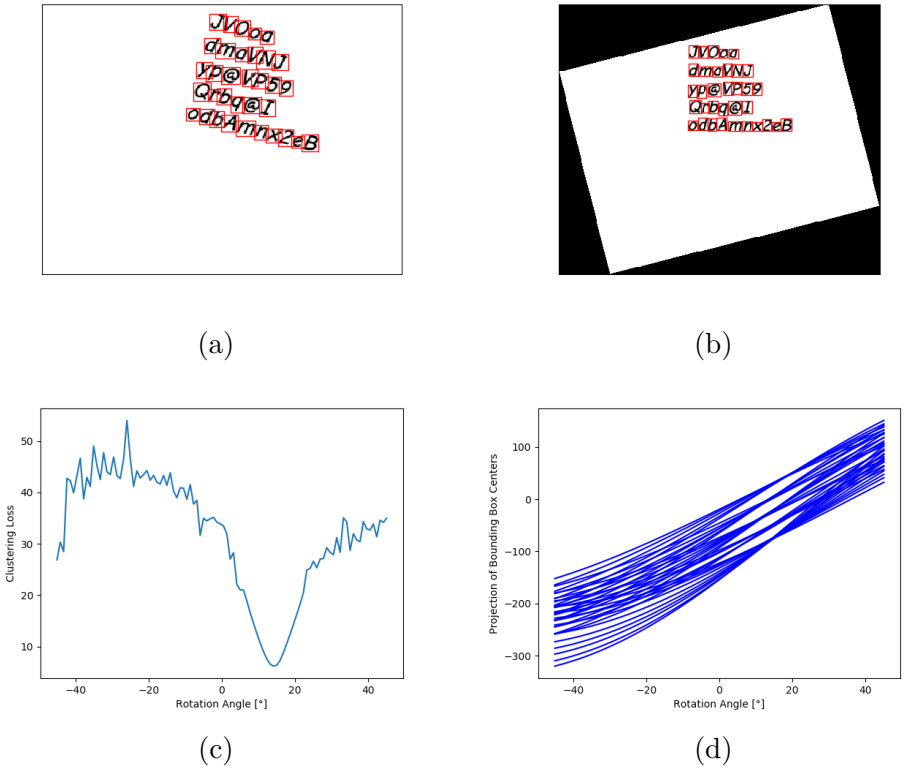


Figure 8: Application of the component-based rotation and line segmentation algorithm on a randomly generated artificial image: (a) is the original image with bounding boxes (b) shows the result of the rotation correction. (c) is the clustering loss L_Θ depending on rotation angle and (d) shows the projections of the bounding box centers for the different angles.

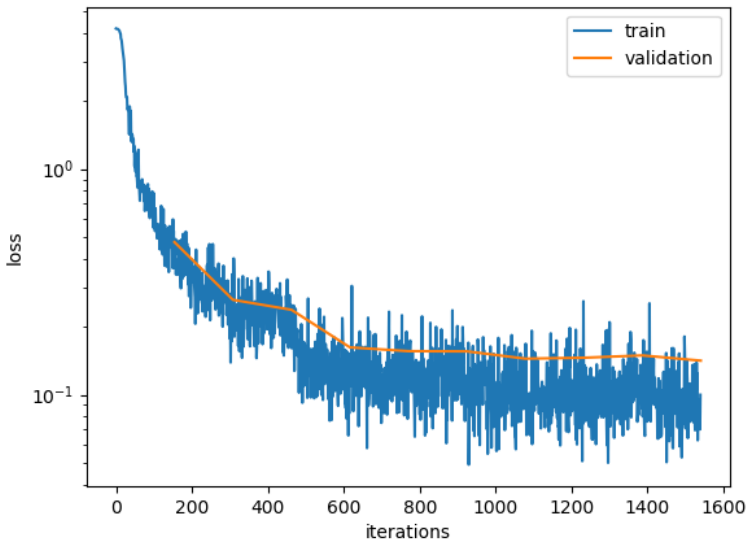


Figure 9: Loss on training examples and average loss on the validation set during training.

The learning rate is reduced by a factor of 5 every three epochs, and training is stopped after 10 epochs. Loss curves for training and validation can be seen in Figure 9. The final classification accuracy on the validation set is 95.8%. This is lower than one would expect considering the relatively simple task. However, it can be explained by the fact that some characters are not extracted correctly from the generated images resulting in bad training examples. The accuracy is still sufficiently high not to be the bottleneck of our algorithm's performance, see the results on segmentation in subsection 3.1

3.4 IBAN detection

In order to evaluate the end-to-end performance of our system and to test its applicability to IBAN recognition it is tested on 39 photos of bank account information taken with a smartphone camera. The photos were taken in different lighting conditions and in a natural way without paying much attention to achieving good quality. Before applying the algorithm, however, they were cropped to contain only a single block of text. We analyze both

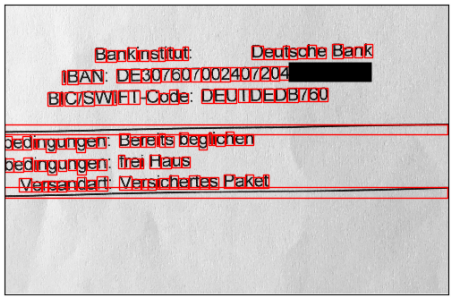
character:	0	5	1	4
mistaken as:	O	S	I	a

Table 1: Typical classification mistakes

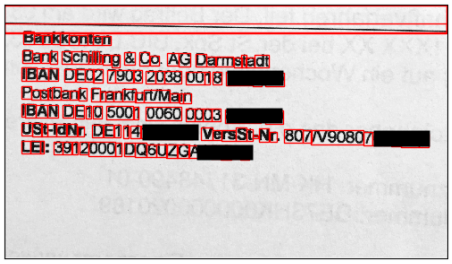
the detection of characters with the MSER algorithm and the subsequent classification of the contained IBAN.
Perfect detection of all characters in the image was achieved in 33% of the images. In 26% of all images at least two characters were mistaken as one and in 41% of the images at least one character was (additionally) not recognized. Despite this characters belonging to IBANs are more often well detected, which leads to the fact that they can often be classified although detection errors occur somewhere else in the image. 23% of the IBANs were classified without any mistake and another 46% contain minor classification mistakes. All mistakes that are considered minor are shown in Table 1. By far the most frequent classification mistake is "0" being mistaken as "O". Correction for these mistakes after classification leads to an overall detection of 69% of all IBANs in the images. Figure 10 shows some example images including the bounding boxes of detected characters, the classification output and recognized IBANs.

4 Conclusion

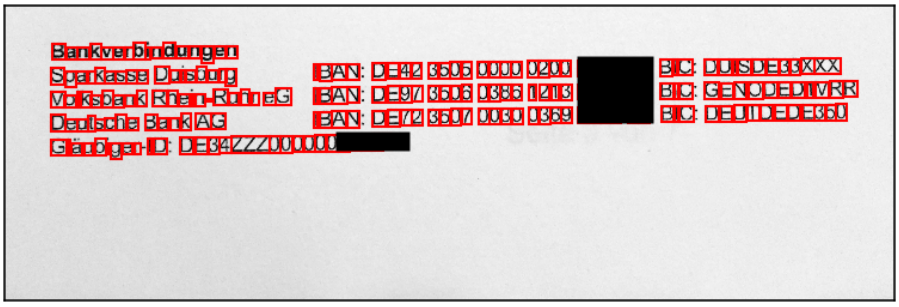
The here proposed system is a rather classical approach towards optical character recognition consisting of separate extraction of character candidates and their subsequent classification with additional correction for rotation.
For the extraction of characters we compared simple connected component analysis (CCA) in a binary version of the input image to the extraction of maximally stable extremal regions (MSER). Due to the binarisation's sensitivity to different lighting conditions and complex backgrounds we found the MSER variant to be more applicable and robust. Nevertheless, MSER extraction struggles with blur, which may be due to bad printer quality or the text being out of focus. For the extraction to become reliable this would need to be improved.
Two rotation algorithms were compared. The fist one needs to process all pixel values and is applied prior to character extraction. It is, therefore, slower than the second one, which is based on character candidates. In addi-



(a)
Bankinstitut Deutsche Bank
lBAN DE30760700240720*
BlCISWlFT Code DEUTDEDB760
2
Dedingungen Bereits beglichen
bedingungen frei Haus
Versandad Versichedes Paket
2
(DE307607002407204*)



(b)
2
Bankkonten
BankSchilling5Co AG Darmstadt
IBAN DEO279O32OMOO18*
PostbankFrandumMain
IBAN DE1O5@1 OmOOO3*
USt2IdNr DE1141O* VemStQNr 8O7N9O7*
LEI 3912O@1DQ6UZG*
(None)



(c)
Bankverbindungen
Sparkasse Duisburg lBAN DE42 3505 OO0O 020O * BlC DUISDE33XXX
Volksbank Rhein2RuhreG lBAN DE97 35O6 O386 1213 * BlC GENODED1VRR
Deutsche BankAG lBAN DE72 3507 0030 0369 * BlC DEUTDEDE350
Glaubiger lD DE34ZZZ00O0006*
(DE42350500000200*, DE97350603861213*, DE72350700300369*)

Figure 10: Some example images of bank account information with bounding boxes of MSERs in red. The corresponding output is printed in the subcaptions with IBANs in brackets if any were detected. The images and outputs are censored.

tion, the latter algorithm already clusters characters into lines and is, hence, superior to the former.
The extracted and rotated character candidates are classified with a convolutional neural network, that was trained on artificial images of text inlcuding different font styles. We were able to achieve a classification accuracy of 95.8%.
The system consisting of MSER extraction, the second rotation algorithm, the CNN and additional heuristic filtering of the character candidates was applied to a set of 39 images of bank account information. Perfect classification of the entire text was very rarely achieved. However, 23% of the contained IBANs were correctly classified. By a simple replacement of the most common classification errors this could be improved to 69%, which is not enough for a satisfactory application but a reasonable result for this project.

5 Appendix

5.1 Code

The code for this project is available on GitHub. A documentation is also provided.

References

[1] Alessandro Bissacco, Mark Cummins, Yuval Netzer, and Hartmut Neven. PhotoOCR: Reading text in uncontrolled conditions. *Proceedings of the IEEE International Conference on Computer Vision*, pages 785–792, 2013.

[2] Huizhong Chen, Sam S Tsai, Georg Schroth, David M Chen, Radek Grzeszczuk, and Bernd Girod. Edge-Enhanced Maximally Stable Extremal Regions. pages 3–6, 2011.

[3] Michael Donoser, Hayko Riemenschneider, and Horst Bischof. Shape guided Maximally Stable Extremal Region (MSER) tracking. *Proceedings - International Conference on Pattern Recognition*, pages 1800–1803, 2010.

[4] Hiromichi Fujisawa. Forty years of research in character and document recognition-an industrial perspective. *Pattern Recognition*, 41(8):2435–2446, 2008.

[5] Pan He, Weilin Huang, Tong He, Qile Zhu, Yu Qiao, and Xiaolin Li. Single Shot Text Detector with Regional Attention. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:3066–3074, 2017.

[6] Weilin Huang, Yu Qiao, and Xiaoou Tang. Robust scene text detection with convolution neural network induced MSER trees. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8692 LNCS(PART 4):497–511, 2014.

[7] Fan Jiang, Zhihui Hao, and Xinran Liu. Deep Scene Text Detection with Connected Component Proposals. pages 1–10, 2017.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[10] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[11] Brennan Novak. opensourcedesign/fonts. https://github.com/opensourcedesign/fonts, 2014.

[12] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[13] Joachim Schenk and Gerhard Rigoll. *Mensch-Maschine-Kommunikation*. 2010.

[14] Cunzhao Shi, Chunheng Wang, Baihua Xiao, Yang Zhang, and Song Gao. Scene text detection using graph model built upon maximally stable extremal regions. *Pattern Recognition Letters*, 34(2):107–116, 2013.

[15] H Stewénius Nistér D. Linear Time Maximally Stable Extremal Regions. pages 183–196, 5303.

[16] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. *Proceedings of the IEEE International Conference on Computer Vision*, (4):1457–1464, 2011.

[17] Tao Wang, D J Wu, a Coates, and a Y Ng. End-to-end text recognition with convolutional neural networks. *21st International Conference on Pattern Recognition,2012 (ICPR2012)*, (Icpr):3304–3308, 2012.

[18] Peter Wide. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition Baoguang. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 39(11):2298–2304, 2017.

[19] Yingying Zhu, Cong Yao, and Xiang Bai. Scene text detection and recognition: recent advances and future trends. *Frontiers of Computer Science*, 10(1):1–18, 2015.