CNN-based Ruled Line Removal in Handwritten Documents

Christian Gold and Torsten Zesch
CATALPA - Center of Advanced Technology
for Assisted Learning and Predictive Analysis
FernUniversität in Hagen, Germany
firstname.lastname@uni-due.de

Abstract

Even state-of-the-art neural approaches to handwriting recognition struggle when the handwriting is on ruled paper. We thus explore CNN-based methods to remove ruled lines and at the same time retain the parts of the writing overlapping with the ruled line. For that purpose, we devise a method to create a large synthetic dataset for training and evaluation of our models. We show that our best model variants are capable of reconstructing characters that are overlapping with the line to be removed, which is a problem that simpler approaches often fail to solve. On a dataset of children handwriting, we show that removing the ruled lines improves character recognition. We made our synthetic dataset and all experimental code available to foster further research in this area.

1 Introduction

A frequently addressed issue in handwriting recognition (HWR) is the recognition itself, although it is preceeded by many steps such as word- or line-level segmentation. In addition, contrast enhancement, noise removal or struck-out word detection can further improve the recognition performance. The main goal of these preprocessing steps is to separate the handwritten text referred to as foreground from the background in the best possible way. Focusing on HWR in the educational domain, handwritten texts challenge this separation as printed forms (exams, school workbooks, writing pads, ...) where ruled lines help guiding the orientation and size of writing are commonly used. Although in most cases only a single ruled line is used, especially younger children use paper with several lines (see Figure 1).

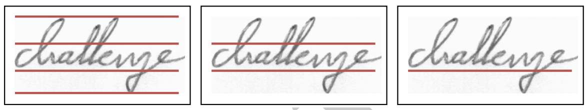


Figure 1: The word ‘challenge’ from the IAM dataset with 4 ruled lines, 2 ruled lines and one ruled line.

Furthermore, in textual answers of exams, ruled lines still exist to indicate that a textual answer must be given.

Ruled lines pose a challenge for several levels of HWR. First, for word-segmentation, where the ruled lines cause a problem as they connect multiple words together making it harder to determine the beginning and end of a word. Second, as character recognition has to interpret stroke-based structures into characters it additionally have to deal with a ruled line, which is sometimes very dominant. The removal process of ruled lines faces the challenge of the retention of character-pixels, while pixels from the ruled line shall be removed. We address this challenge in our work using a neural network.

Although neural networks have recently been successfully used for a variety of image processing tasks, their use for ruled line removal has - to the best of our knowledge - not yet been explored. In this paper, we therefore propose a simple and effective convolutional neural network for line removal while retaining handwritten characters and brightness. Furthermore, the proposed model recreates the shape of characters which align along the ruled line. For training and evaluation, we first create a synthetic dataset which can be adapted to any use case (e.g. number of ruled lines).

2 Related Work

Removing ruled lines with standard image processing algorithms usually follows three stages: *detection*, *removal*, and *text enhancement* (Refaey, 2015). In the *detection* stage, the ruled lines are extracted from the background and handwritten text. At the *removal* stage, the ruled lines are set to background color. In the *text enhancement* stage, the characters are usually re-connected and background noise is revised. Two stage solutions use connected component analysis during the *removal* stage to distinguish between ruled line pixels and text pixels and thus already perform the re-connection. An enhancement stage would then only remove noisy leftovers.

Rani and Vasudev use a canny filtering for the detection stage and a local intensity of black pixels with a morphological above-threshold filtering for the detection of connected components. The enhancement stage deals with the removal of dotted and broken lines and clearing of noise.

The method proposed by Refaey (Refaey, 2014, 2015), use a windowed Hough-Transformation at the detection stage. The removal stage includes an intensity histogram (hue) with a local entropy to detect the connected components. A subsequent morphological operation enhances the text isolation from the background.

Imtiaz et al. make use of prior knowledge (e.g. average width of characters) and determined parameters such as the total number of characters and the total number of lines. With these determined, the detection and removal stage use structuring filters and merging (AND, OR and subtraction) of these results. At the text enhancement stage, the local entropy is calculated to re-connect disconnected characters. Finally, the noise is removed with a median filter.

In (Chen and Lopresti, 2014), the authors present a linear regression model which is capable of detecting ruled lines and especially dealing with broken lines. However, they do not propose a removal method after detection. Some classical approaches, e.g. (Imtiaz et al., 2014) and (Refaey, 2014), are dependant on predefined parameters (e.g. the average width of a character) or on the page layout. All classical approaches cannot deal with characters that overlay ruled lines (see example in Figure 7). Thus, in this paper we leverage the power of neural networks to tackle the task of ruled line removal.

3 Synthetic Dataset Creation

To our knowledge, no freely available dataset containing ruled lines documents with and without the ruled line are published.[1] Thus, we create a synthetic dataset where ruled lines are generated and placed into sequences of concatenated handwritten word images, as shown in Figure 3. The resulting dataset contains two version of text images with and without ruled lines.

3.1 Word Concatenation

The word images for the dataset were taken from IAM (Marti and Bunke, 2002). These word images come from various writers written without any constraints on size, alignment or pen type. Therefore, to create a realistic text line, we need to align these word images vertically along a virtual bottom body line. The virtual lines can be seen in Figure 2. We categorize words into *bottom*, *top*, *body* or *full* (all) according to the characters' extension within each word. The characters which use the space below the bottom body line are: [f, g, j, p, q, y]. Characters using the top space are all upper case characters as well as [b, d, f, h, i, k, l, t]. The word images can then be vertically arranged according to their category (e.g. *bottom* categorized words will be aligned at the bottom, while *body* words will be aligned onto the bottom body line).

Some parameters, e.g. the number of text lines per page or the number of words per text line, are varied. However, for a consistent layout across the page, we assign a random base value to some parameters (e.g. constant base gap between text lines). Each time such a parameter is used, the parameter will then be varied slightly from this base value.

3.2 Adding Ruled Lines

Simultaneously to the word concatenation, a copy of the image with inserted ruled lines will be created. While keeping the page layout consistent, we create pages with one, two or four ruled lines each text line (see Figure 1). To further diversify our training data, we decided to add another type in which we place one ruled line somewhere over the handwritten text line. To make the inserted ruled lines even more realistic, we randomly add several defects. These include noise, differing grey level,

[1]A potentially useful dataset (Kumar and Doermann, 2011) was not available under the published URL.

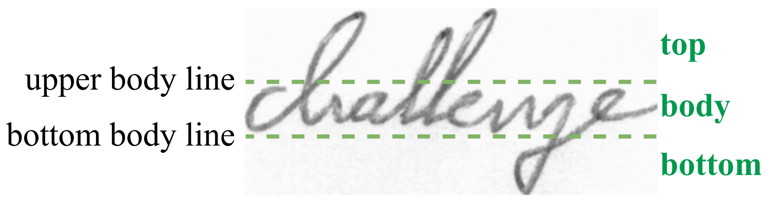


Figure 2: The word ‘challenge’ from the IAM dataset divided into a top, body and bottom part. The word is categorized as *full* as its characters span across all 3 parts.

small rotation as well as other techniques. An example of a final version with and without one ruled line each text line can be found in Figure 3.

In summary, our proposed method is capable of generating unlimited training images while the specific configuration of lines (e.g. number of lines, line thickness, line length, or adding vertical lines) can be adapted to the requirements of the task. Despite the benefits of this synthetic dataset, we want to note that models solely trained on synthetic data always perform worse on real data. For our dataset, we did not specialize on one particular use case only, but rather aim at a widely applicable model by using a variety of line types.

4 Proposed Architectures

To remove ruled lines, we experiment with two modeling strategies: First, a straightforward all convolutional model and second, an autoencoder model. We explore several setups with different hyperparameters of each model to further increase the performance. As a simple baseline, we implement a naive morphological method based on conventional image processing.

4.1 Morphological Baseline

With our morphological filter approach we detect line pixels and set them to white (background color). First, the input is thresholded with Otsu (Otsu, 1979) and followed by an opening and closing morphological filtering with a rectangular (horizontal) structuring element. Thus, we retrieve the ruled lines as white areas which can be extracted with object detection while filtering small areas out. Afterwards, all pixels within the objects are set to the background color, including pixels which intersect with character strokes. With this approach, we can analyze whether the recognition performance decreases if the character shape is not restored after removing the line.

4.2 ALL-CNN

As described in Section 2, earlier, ruled line removal has so far been done using regular image processing techniques, similar to the morphological approach. One basic technique is to apply rectangular filters on the image (e.g. opening and closing). In neural networks, convolutional layers (conv) can be seen as filters, although the filter values are trainable parameters. Instead of applying filters consecutively, several filters are trained in parallel, together with a bias value.

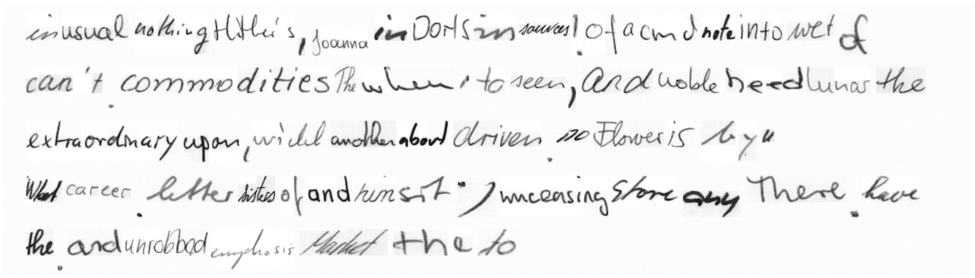
Our model consists of only three conv layers and thus we name our model ALL-CNN following (Springenberg et al., 2014). We set the filter dimension of the first layer to 21x21 followed by two layers of dimension 3x3 each. By choosing a rather large filter dimension at the first layer, the network can gather more information around the centered line pixel. Therefore, the network is able to recognize dependencies of other character pixels around the ruled line (e.g. above and below) and thus to retain the shape of the characters. The number of filters is 32, which is later on varied (see Section 4.4). Additionally, we apply LeakyRelu (Xu et al., 2015) and Batch-Normalization (Ioffe and Szegedy, 2015) after each conv layer. An exemplary architecture of ALL-CNN can be seen in Figure 4.

4.3 Autoencoder

Inspired by the various fields of application of autoencoders (Hinton and Salakhutdinov, 2006) such as cleaning (Yin, 2019) and denoising Yasenko et al., we decided to test a CNN-based autoencoder architecture for the removal of ruled lines. The idea is to have an encoder at the beginning of the bottleneck and a decoding afterward, while filtering out the ruled lines in the taper.

The basic architecture consists of 5 conv layers. A downsampling is applied after the second conv layer and an upsampling after another conv layer (see Figure 5). The number of filters and their di-

(a) Text Lines



(b) Text Lines With Ruled Lines

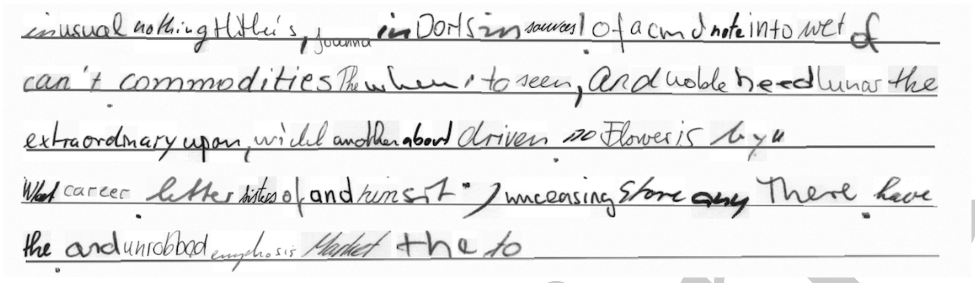


Figure 3: An example page of the synthetic dataset displaying the text setting (a) and the ruled line placement (b).

mension of the first conv layer remain identical to the ALL-CNN model, as the purpose of the first layer remains the same. The remaining conv layers keep the number of filters, but the filter dimension is decreased to 5x5 (conv2) and 3x3 (conv3-5).

4.4 Hyperparameters

We experimented with several model parameters.

DNF - Decay of Number of Filters Initially, we weight all conv layers alike and thus assign the same number of filters, a setup which we denote with *SNF* (Same-Number-of-Filters). We believe that the large filter (21x21) at the beginning will have the most influence for finding the ruled lines and the connection between characters intersecting the ruled line. We assume, that too many filters afterwards would only hold redundant information. Thus, we decay the number of filters after the first conv layer by cutting the number in half starting with 32 filters at conv layer 1, 16 at layer 2 and 8 for layer 3. This setup is called *DNF* (Decay-Number-of-Filters) and is only applied on the ALL-CNN model.

Subtraction In our first results of the basic models (Autoencoder and ALL-CNN), both predicted results were darker than the input images presented.[2] We assumed that this is due to the fact that the deeper the network was, the less information was still present from the input image. Furthermore, all pixels are influenced by the network although only the pixels of the ruled line should be affected. As only a few pixels must be changed, we referred to image processing techniques where only areas of attention (masks) are changed. We decided to use the output of the network to function as such a mask by subtracting the former output from the input layer and thereby present the input layer anew at the end of the network (see Figure 4).

Rectangular Filter Shape While we use the conv layers with a quadratic filter dimensions, the line to be reduced is of vertical shape. Thus, we adapt the filter dimension of 21x9 (width x height) which we denotes as *RectFilter*. While the ruled lines are horizontal, the shape of the characters crossing the ruled line [f, g, j, p, q, y] are rectangular with the longer side on the vertical axis. Therefore, we transpose the dimensions 9x21 as *RectFilter Transpose*.

5 Evaluation Setup

We describe how we evaluate results on synthetic data pixel-wise and on real data using CER as a derived extrinsic metric.

[2]Although the contrast in visualization might differ, we want to note that it can be seen in Figure 6.

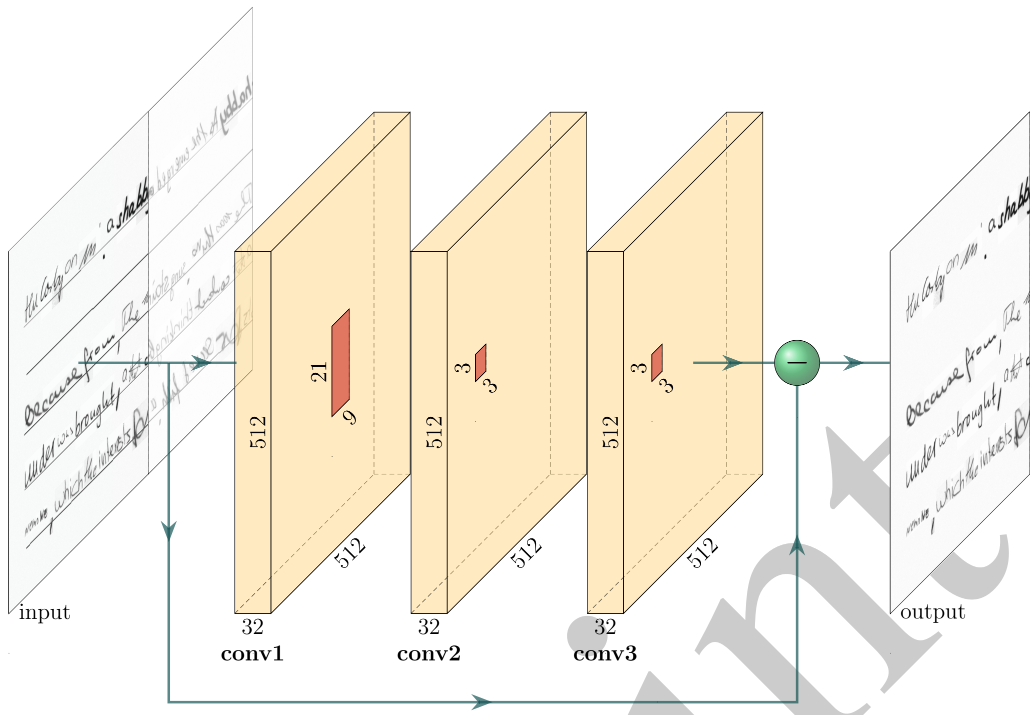


Figure 4: Our proposed ALL-CNN (a variant with rectangle 21x9 filter) network with 3 conv layers, subtraction and with a Rectangular Filter (see Section 4.4). The Batch-Normalization and LeakyRelu after each conv layer are not visualized.

5.1 Synthetic Dataset

Although the synthetic data pages are created as full pages, they are divided into tiles of size 512x512 pixels. We choose this size as a tradeoff between computation costs while covering several (∼5) text lines. Furthermore, due to the varying number of text lines and varying text height, white tiles are common at the bottom of the page and thus are not ignored. We analyze the performance of our models on 100 synthetic images.

Evaluation Metric for Synthetic Data With the synthetic dataset created, a pixel-wise evaluation can be performed. First, we calculate the root mean squared error (RMSE), which compares the delta pixel-wise and eliminates negative values by squaring. [3] Furthermore, we compute the delta between the predicted gray value (after the removal process) and the target gray value. Additionally, we assume that small deviations from the target value will not influence the handwriting recognition and thus set up a gray value threshold of ± 5. In this way, we can differentiate between three cases: no deviations, small deviations, and deviations of a greater extent.

5.2 Real Dataset

To test our methods on a use-case, children's handwritings are a particularly good example as they mainly include ruled lines. We decided to use the FD-LEX dataset (Becker-Mrotzek and Grabowski, 2018) due to its clear structure (white pages with ruled lines) and availability. The dataset comprises freely-written texts with about 370,000 words from 938 learners in the 5th (age 9-11) and 9th (age 14-16) grade from the German school system written in German. For demonstration purposes, we took the first 50 pages of the set GYM 5.1 which are written from children of the 5th grade. In total, we consider 522 text lines. Unfortunately, the transcribed text was spellchecked. Thus, we had to manually transcribe the texts again.

Evaluation Metric for Real Data As the FD-LEX dataset does not provide a ground truth for ruled line removal, the RMSE metric as used before is not applicable. Thus, we follow Cao et al.

[3] It shall benoted that the RMSE is also used for the loss function during training.

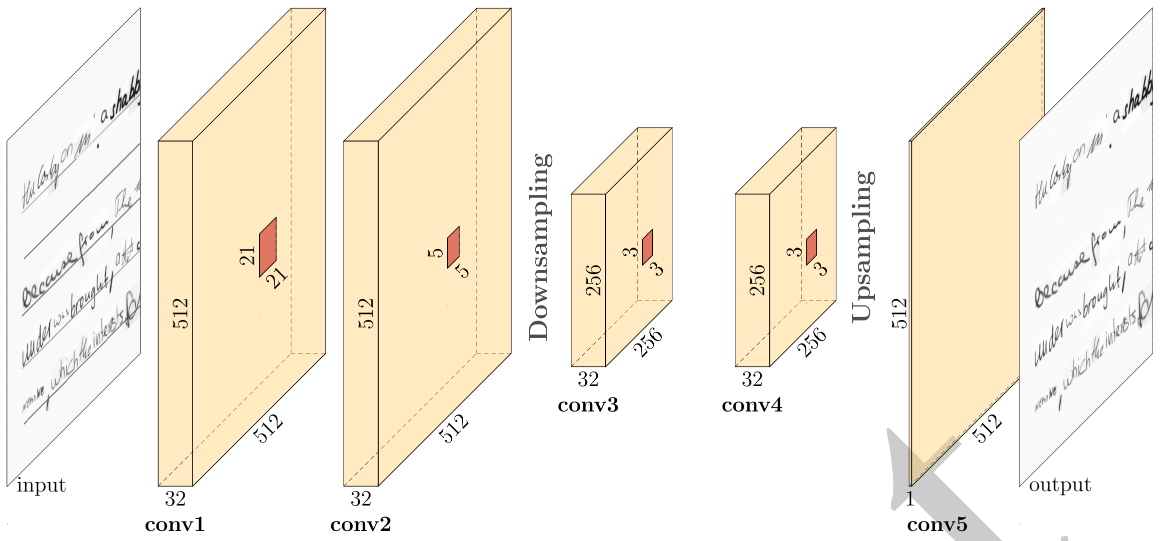


Figure 5: Proposed CNN Autoencoder architecture with 5 convolutional layers.

and use an object-level metric, the recognition rate, as an extrinsic evaluation across all models, which makes it necessary to implement a recognizer. Although Cao et al. use the word error rate, we use the character error rate (CER) as calculated with the edit distance. The CER can be seen as the inverse character accuracy, more precisely, the percentage of incorrectly predicted characters compared to the ground truth text. This way, we are able to analyze mispredicted characters in a more fine-grained way, especially those intersecting with the ruled lines.

Handwriting Recognizer We use a straightforward text line handwriting recognizer with a CNN architecture and a Connectionist Temporal Classification (CTC (Graves et al., 2006)) at the end. The model is trained using the text line images from the IAM dataset while taking the split as defined in "Large Writer Independent Text Line Recognition Task"[4] for training, evaluation, and testing. It should be noted that we chose not to use a language model as it would correct spelling errors.

The handwriting recognizer requires a text line segmentation as a preprocessing step. We use a straightforward segmentation with the 1 A^* path finding algorithm. Afterwards, we receive a pixel-wise mask of the individual text lines. As this segmentation process should not influence the recognition performance across all models, we apply the segmentation on one of the visually best methods (ALL-CNN with Subtraction) and transfer the segmentation masks to all other methods. In this way, we always segment the same part of the images, while having the individual results from the ruled line removal methods. After training, the handwriting recognizer has a CER of 11.52% on the IAM test set.[5]

6 Results & Discussion

Table 1 shows the results of our experiments. The character error rate (CER) on the real handwriting data with ruled lines is rather high (see the next subsection for an analysis of the reasons). However, as we hypothesized, removing ruled lines has a massive effect on recognition performance. Already the baseline removal method reduces CER from 66.3 to 35.4. Our neural models decrease CER even more. The best autoencoder version yields 31.1 while the best All-CNN models yields 28.5 which is a 6.9 percent point improvement over the baseline.

From those numbers alone it is hard to further analyze the individual performance of model variants. We thus now have a look at the synthetic test data where we know the true gold standard without the ruled lines and can perform a pixel-wise scoring.

[4] http://www.fki.inf.unibe.ch/databases/iam-handwriting-database

[5] Note that CER values for the IAM dataset are typically computed per word, while we tackle the much harder task of recognizing whole lines. However, as we are only interested in the relative impact of ruled line removal, the performance of our recognizer is sufficient.

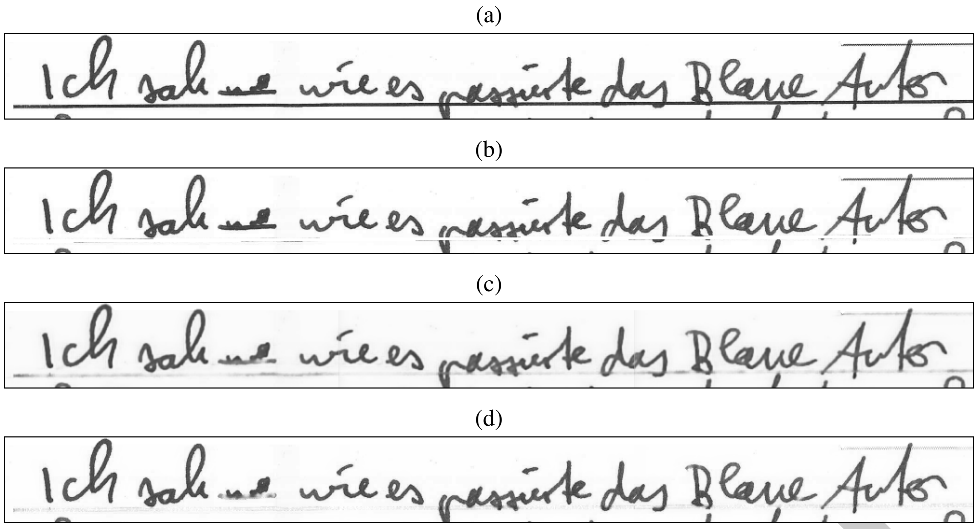


Figure 6: Exemplary results of the first text line from the real dataset. (a) Original (b) Morphological Baseline, (c) Autoencoder and (d) ALL-CNN

Our Morphological method has an RMSE of 6.6 on the synthetic data. This denotes that on average, every pixel differs by 6.6 gray values from the ground truth. In addition, only 2.3% of all pixels were changed and 1.8% of these were above the threshold. This shows that those 1.8% have a bigger distance (e.g. a pixel that should be black was incorrectly changed to white) as they are able to increase the RMSE above the threshold of 5. The CER on the children's data is worse compared to our other methods but better than the original images. This proves our assumption that roughly removing the ruled lines increases the recognition performance. As this approach was naive - since only relevant pixels were changed - we use this as the baseline.

The best Autoencoder setup can reduce the RMSE value to 6.3. In general, however, the resulting images from the Autoencoder change nearly all pixels (above 92%). Furthermore, about one third of all pixels in the image have a difference above the threshold.

The best ALL-CNN architecture achieves the closest image in terms of the RMSE with only 2.6%. Although the amount of changed pixels is high, the amount of pixels exceeding the threshold is low with approx. 2-5%. This - together with the small RMSE - indicates that the delta is small contrary to the Baseline result where a few pixels increased the RMSE. In our results, the rather basic architecture with only 3 conv layers and the renunciation of down and up sampling resulted in our best recognition performance with a CER of 28.5% beating baseline by about 7 percent points. This is a strong result, as CER performance is also influenced by other factors (e.g. crossed-out words, tally marks) and our method focuses on ruled lines only.

6.1 CER Performance Level

While ruled line removal is able to improve recognition quality, the CER is still rather high. We now discuss some factors that additionally influence the results.

A major limitation is that training and application dataset do not match well (but there are not other datasets available). In particular, IAM texts were written by native English adults, while the FD-LEX texts were written by German children. Therefore, this is already an influence of two factors: the delta between adults and children, and between English and German. The latter might cause a bigger issue for the recognizer than is visible at first sight. One difference is the alphabet between German and English. The umlauts ä, ö, ü and the special character ß are not existing in English and thus are not present in the training data. To address this issue partly, we replaced those umlauts with a, o, u and s, respectively, in the ground truth text (including their upper-case variants).

Another, not so obvious gap arose from the different character bi-gram distributions which are learnt from the recognizer due to its temporal structure. For instance, 'th' is the most common

Table 1: Performance of all models on synthetic data with RMSE, amount of pixels which were changed !=0, amount of pixels with a greater difference then the threshold and on real data FD-LEX with CER.

		Synthetic			Real
			Error		
		RMSE	$\neq 0$	$> \pm 5$	CER
Original		-	-	-	66.3
Morphological Baseline		6.6	2.3	1.8	35.4
Autoencoder	SNF	6.6	94.5	38.1	**31.1**
	SNF Subtraction	17.0	92.9	**7.5**	65.9
	SNF RectFilter	**6.3**	93.3	32.5	39.2
	SNF Subtraction RectFilter	11.1	92.8	19.4	55.4
	SNF RectFilter Transpose	6.7	93.8	36.7	31.3
	SNF Subtraction RectFilter Transpose	10.9	92.1	20.3	54.9
All-CNN	DNF 2L	4.7	85.6	5.6	32.3
	DNF 3L	4.5	84.9	6.0	31.8
	DNF 3L Subtraction	3.6	71.7	2.4	31.0
	SNF 2L	4.0	72.8	5.8	30.1
	SNF 3L	4.1	80.7	5.1	31.7
	SNF 3L Subtration	**2.6**	47.4	2.4	**28.5**
	SNF 3L RectFilter	3.6	68.0	4.6	33.8
	SNF 3L Subtraction RectFilter	2.9	63.5	3.4	29.4
	SNF 3L Subtraction RectFilter Transpose	3.0	**44.5**	**1.9**	29.5

bi-gram amongst English words whereas in German it is 'er' or 'en'.[6] In particular, frequencies of double consonants differ between both languages, too. As another difference, in German, three repeated consonants like 'Schifffahrt' (naut. 'shipping') exist. Therefore, the recognizer is biased towards an English dataset.

A further negative influence on the recognizer is, that tally marks are used in some of the FD-LEX images after every 10 words. Those are predicated as |, (or) most of the times. Unfortunately, the prediction of these vertical lines can sometimes influence neighboring characters (e.g. a| = d), too. This makes it difficult to filter these errors out or change the ground truth data to |.

Crossed-out words are another influencing factor as they are represented by a # sign in the transcription but can be interpreted by the recognizer as several characters leading to many errors. This further decreases the performance as children are presumably more likely to cross-out words. Ultimately, we decided to ignore all these issues as we focus on the ruled line removal itself. Therefore, we analyze the model performances with different metrics by looking at the improvement in relative performance rather than the overall performance.

Furthermore, some digitalization artefacts (e.g. gray border from the page during scanning) additionally decrease the performance.

6.2 Hyperparameters

SNF vs. DNF Our hypothesis was that too many filters at the latter layers would only hold redundant information and thus do not influence the recognition performance. As the DNF version is always worse than the SNF version with all other parameters held equal, we can conclude that in the deeper layers, the amount of filters plays an important role.

Subtraction For the ALL-CNN model, adding subtraction was a crucial step that decreased CER by 3 percent points. Subtraction is only beneficial for ALL-CNN, while it strongly decreases performance of the Autoencoder models.

Rectangular Filter Focusing on the recognition results, neither the horizontal nor vertical rectangular filter shape resulted in an improvement compared to the quadratic shape. Only for one model using a rectangular shape came close to beat the best performance on the autoencoder models.

6.3 Remodeling of Overlapping Characters

A special problem for earlier approaches are characters overlapping with ruled lines as those cannot be reconstructed with filtering. In Figure 7, the character 'e' of the word 'modified' is covered with a ruled line. Although the bottom stroke

[6]https://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/8_Transpos/BigramsE1.html and *BigramsD1.html

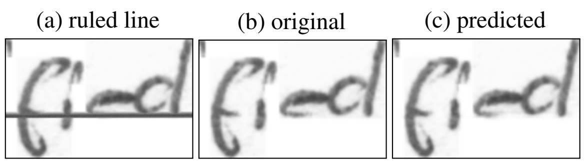


Figure 7: Image segment as example of the model (best performing All-CNN variant) capability to remodel the letter ‘e’ as the bottom stroke overlaps with the ruled line.

of the character ‘e’ overlaps with the ruled line, the CNN model was able to recreate the stroke partly (shorter in length). We assume, that this was possible due to the darkness information still being present in the ruled line as both, the character strokes’ darkness and those of the ruled line interfere with each other. As these images were taken from the synthetic dataset and thus can’t be tested on real data due to the loss of a real ground truth, we assume that our model should be capable to remodel characters likewise, if the darkness in the intersection area is superimposed.

6.4 Prediction of Characters with Restoration

We assumed that the Baseline approach would lead the recognizer to mispredict bottom characters as body characters as those characters cross the ruled line and are thus cut in half when the line is removed. E.g. ‘g’ would be predicted as ‘a’ or ‘o’ and a ‘y’ as ‘u’ or ‘v’. Hence, we analyzed the misprediction of those bottom characters by using the Levenshtein Library[7] and return all edit operations necessary to transform the predicted text line into the ground truth. The numbers in Table 2 correspond to the frequency of the operations ‘equal’ (correctly predicted characters) and ‘replace’ (incorrectly predicted characters). We do not consider ‘insert’ and ‘delete’ operations in this analysis.[8] However, using these numbers across all models makes them comparable again and at least indicate trends which can be seen in Table 2.

For instance, ‘g’ was incorrectly predicted as an ‘s’, ‘a’ or ‘e’ in 8.5% for the Baseline method while for the Autoencoder and ALL-CNN the recognizer mispredicted them as bottom characters like ‘y’ or ‘q’, too. In general, the mispredicted characters are identical across all models except for the character ‘g’.

6.5 Limitations

Both models were trained to detect and remove horizontal ruled lines. Unfortunately, some characters e.g. ‘T’ are build of likewise horizontal lines and thus are removed partly, too. Although these errors occur, we were unable to find upper characters (further examples: ‘E’, ‘t’, ‘F’, ‘H’) being incorrectly predicted due to this result. We thus calculated the accuracy of correctly recognized upper characters (see Table 2).[9] We can see, that the recognition performance of upper characters increased, in relation to the Baseline method, which leaves those characters untouched.

Furthermore, as all models are trained on an English dataset, we assume that the model will perform worse on scripts like Arabic which align strongly on ruled lines. For various reasons, it was not possible for us to set our results in relation to those procedures mentioned in Section 2. In many cases, the datasets and code used in the experiments were not freely available, whereas some links to the datasets were offline and the authors did not respond. For other approaches, the writing script (e.g. Arabic) differed too much. In addition, a large variety of different metrics were used, which often related to characteristics of the line (e.g. distance of lines, skew, position, ...), rather than the accuracy of the removal.

7 Summary

We evaluated a dataset of children handwriting and showed that ruled lines greatly affect character recognition. To this end, we trained different CNN models, namely an autoencoder and an ALL-CNN model, to remove the ruled lines. To do this, we created a flexible synthetic dataset in which we automatically place ruled lines onto handwritten text lines. This dataset is then used to train and evaluate the models. Finally, we tested the models on the children handwritings and showed that the error rate made by the recognizer was cut in half. In addition, we varied hyperparameters to further improve the model performance.

[7] https://pypi.org/project/python-Levenshtein/

[8] For example, a ‘y’ could be recognized as ‘ii’ with two characters instead of one, which would lead to a ‘replace’ and ‘delete’ operation. However, the ‘delete’ operation could be assigned to the next character in the ground truth string and thus makes it invisible for us to analyze.

[9] We want to note, that again, only the operations ‘replace’ and ‘equal’ were considered from the Levensthein output and thus the true value might differ.

Table 2: On the left: the accuracy of the predicted *bottom* characters and their three most common mispredictions. 'y' is left out as it does not occur in the ground truth data and 'q' as it was not predicted by all methods. On the right: the accuracy of the character classes *upper*, *body*, *bottom*.

	'f' (247)		'g' (330)		'j' (330)		'p' (330)		Accuracy in %		
Model	%	pred	%	pred	%	pred	%	pred	up.	body	bot.
Original	33.2	f	51.2	g	57.1	j	30.9	p	42	39	46
	8.5	t	11.5	s	9.5	b	14.4	e			
	5.3	e	2.4	t	4.8	i	3.1	b			
	3.2	l	2.4	l	4.8	s	3.1	s			
Morphological Baseline	61.9	f	84.2	g	76.2	j	66.0	p	73	75	72
	17.4	t	5.5	s	14.3	g	7.2	s			
	1.6	e	1.5	a	4.8	J	4.1	e			
	1.2	l	1.5	e	-	-	2.1	r			
Autoencoder	67.2	f	88.8	g	76.2	j	82.5	p	80	75	71
	13.4	t	2.7	s	9.5	g	3.1	e			
	2.8	l	1.2	y	4.8	o	2.1	r			
	2.0	e	0.9	a	4.8	F	1.0	f			
ALL-CNN	72.1	f	90.0	g	71.4	j	87.6	p	83	80	74
	11.7	t	1.8	s	14.3	g	2.1	o			
	1.6	l	0.9	y	4.8	f	1.0	k			
	1.6	e	0.9	q	4.8	o	1.0	j			

In conclusion, we found that a 3-layer CNN was already enough to solve the removal task. When presenting the input image anew, we were able to reduce the character error rate even further. In comparison, a morphological approach, where ruled lines where removed roughly, was outperformed by 7% by our best performing model (3 Layer ALL-CNN with subtraction) in terms of CER. Not only was the model capable of retaining the connections of characters intersecting with the ruled line, the model was also able to reconstruct strokes aligning with the ruled line. To foster further research, we make the model publicly available.[10]

In future work, we want to extend our training set by using the cvl-database which includes German words (Kleber et al., 2013). Being able to remove ruled lines is a prerequisite for training handwriting recognition models for children where almost all dataset have ruled lines. We plan to train a German children handwriting recognition model based on the cleaned FD-LEX data. However, children handwriting data remains challenging as it often contains other confounding factors like drawings, cross-outs, or tally marks.

[10]https://github.com/catalpa-cl/ruled-line-removal-in-handwritten-documents

References

Michael Becker-Mrotzek and Joachim Grabowski. 2018. FD-LEX (Forschungsdatenbank Lernertexte). Textkorpus Scriptoria. Köln: Mercator-Institut für Sprachförderung und Deutsch als Zweitsprache. Available at: https://fd-lex.uni-koeln.de, DOI: 10.18716/FD-LEX/861.

Huaigu Cao, Rohit Prasad, and Prem Natarajan. 2009. A stroke regeneration method for cleaning rule-lines in handwritten document images. In *Proceedings of the International Workshop on Multilingual OCR*, pages 1–10.

Jin Chen and Daniel Lopresti. 2014. Model-based ruling line detection in noisy handwritten documents. *Pattern Recognition Letters*, 35:34–45.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *23rd ICML*, pages 369–376.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Syed Imtiaz, P Nagabhushan, and Sahana D Gowda. 2014. Rule line detection and removal in handwritten text images. In *2014 ICSIP*, pages 310–315. IEEE.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456. PMLR.

Florian Kleber, Stefan Fiel, Markus Diem, and Robert Sablatnig. 2013. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. In *ICDAR 2013*, pages 560–564. IEEE.

Jayant Kumar and David Doermann. 2011. Fast rule-line removal using integral images and support vector machines. In *2011th ICDAR*, pages 584–588. IEEE.

U-V Marti and Horst Bunke. 2002. The iam-database: an english sentence database for offline handwriting recognition. *IJDAR*, 5(1):39–46.

Nobuyuki Otsu. 1979. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66.

N Shobha Rani and T Vasudev. 2018. An efficient technique for detection and removal of lines with text stroke crossings in document images. In *ICCR*. Springer.

Mohammed Refaey. 2014. Fast detection and removal algorithms for ruled lines in full-color scanned handwritten documents. In *CSCESM*, pages 77–81. Citeseer.

Mohammed AA Refaey. 2015. Ruled lines detection and removal in grey level handwritten image documents. In *ICICS*, pages 218–221. IEEE.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Lev Yasenko, Yaroslav Klyatchenko, and Oksana Tarasenko-Klyatchenko. 2020. Image noise reduction by denoising autoencoder. In *2020 IEEE 11th International Conference on DESSERT*, pages 351–355. IEEE.

Kayo Yin. 2019. Cleaning up dirty scanned documents with deep learning. [Online; posted 30-July-2019].