DeepErase: Weakly Supervised Ink Artifact Removal in Document Text Images

Yike Qi* **W. Ronny Huang*** **Qianqian Li** **Jonathan L. DeGange**
Ernst & Young LLP
New York, NY
{yike.qi, ronny.huang, qianqian.li, jonathan.degange}@ey.com

Abstract

Still in 2019, many scanned documents come into businesses in non-digital format. Text to be extracted from real world documents is often nestled inside rich formatting, such as tabular structures or forms with fill-in-the-blank boxes or underlines whose ink often touches or even strikes through the ink of the text itself. Such ink artifacts can severely interfere with the performance of recognition algorithms or other downstream processing tasks. In this work, we propose DeepErase, a neural preprocessor to erase ink artifacts from text images. We devise a method to programmatically augment text images with real artifacts, and use them to train a segmentation network in an weakly supervised manner. In additional to high segmentation accuracy, we show that our cleansed images achieve a significant boost in downstream recognition accuracy by popular OCR software such as Tesseract 4.0. We test DeepErase on out-of-distribution datasets (NIST SDB) of scanned IRS tax return forms and achieve double-digit improvements in recognition accuracy over baseline for both printed and handwritten text.

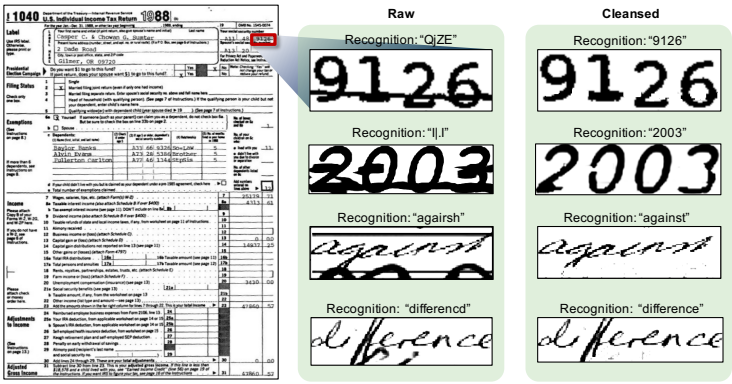


Figure 1: DeepErase cleans, or *erases*, ink artifacts from document text images, improving recognition accuracy, visual appeal, and other downstream tasks. Here we show text images cropped from scanned documents with various ink artifacts, such as underlines, boxes, smudges, and spurious strokes. DeepErase removes those artifacts, immediately improving recognition performance by Tesseract 4.0, a widely used open-source OCR tool, for printed text and by `SimpleHTR`, a popular offline handwriting classifier, for handwritten text.

* Authors contributed equally.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Document Intelligence Workshop, Vancouver, Canada.

1 Introduction

Despite the digitization of information over the past twenty years, large swaths of industry still rely on paper documents for data entry and ingestion. Optical character recognition (OCR) has thus become a widely adopted tool for automatically transcribing text images to text strings. Modern convolutional neural networks have driven many major advances in the performance of OCR systems, culminating in the large-scale adoption of OCR tools such as Tesseract 4.0, Abbyy Fine Reader, or Microsoft Computer Vision OCR.

The relevant text to be extracted from real world documents are often nestled inside of rich formatting such as tabular structures or forms with fill-in-the-blank boxes or underlines. Furthermore, documents with handwriting entries often contain handwritten strokes which do not stay within confines of the boxes or lines in which they belong and can encroach into regions occupied by other text that needs to be transcribed (henceforth such encroachment strokes will be called *spurious strokes*). When extracting text regions from such richly formatted documents, it is inevitable that such document *ink artifacts* are present in the cropped image even if the localization is perfect. Such artifacts can severely degrade the performance of recognition algorithms, as shown in Figure 1.

Despite the prevalence of these artifacts in the real world, many document text recognition datasets, including IAM Marti and Bunke [2002], NIST SDB19 Johnson [2012], and IFN/ENIT El Abed and Margner [2007] contain only images which are cleanly cropped and are more or less free from artifacts. Even the recently released FUNSD dataset of noisy scanned documents Guillaume Jaume [2019] segment their words free of underlines, boxes, and spurious strokes. Consequently, most results on text recognition have reported their performance on clean test examples Graves and Schmidhuber [2009], Bluche [2016], typically in the form of well-aligned, well-spaced text lines, which are not representative of the noisy, marked-up, richly formatted scanned documents encountered in the wild.

Little work has been done leveraging deep learning for document artifact removal. In this work, we present DeepErase, which inputs a document text image with ink artifacts and outputs the same image with artifacts erased (Figure 1). Training is weakly supervised as we use a simple artifact assembler program to produce dirty images along with their segmentation masks for training. Note that henceforth we may refer to images with artifacts as "dirty". We evaluate the performance of DeepErase by passing the cleansed images into two popular text recognition tools: Tesseract and `SimpleHTR`. On these recognition engines, DeepErase achieves a 40-60% word accuracy improvement (over the dirty images) on our validation set and a 14% improvement on the NIST SDB2 and SDB6 datasets of scanned IRS documents.

1.1 Related work

Our work is related broadly to the field of semantic segmentation Long et al. [2015], Ronneberger et al. [2015], Badrinarayanan et al. [2017], which predicts classes for different regions of the image. While semantic segmentation is typically applied to natural scenes, several works have applied it to documents for page segmentation Chen et al. [2017], structure segmentation Yang et al. [2017], or text line segmentation Renton et al. [2017]. All of these tasks discriminate large-scale structure within a document, such as tables or text lines, rather than small-scale patterns such as underlines striking through text characters.

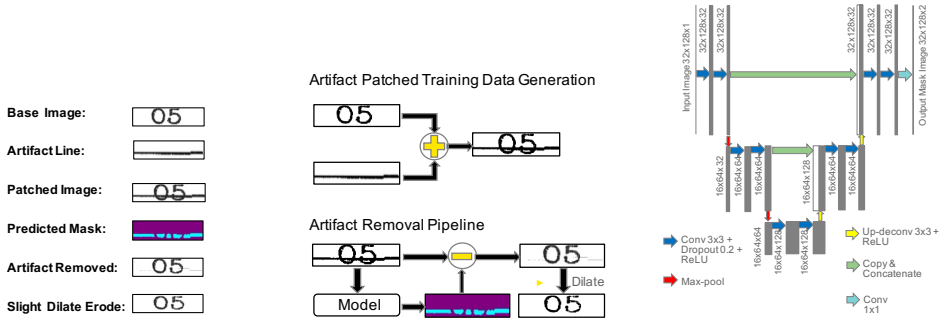
The works of Calvo-Zaragoza et al. Calvo-Zaragoza et al. [2017] and Kölsch et al. Kölsch et al. [2018] are the most similar works to ours. The task in Calvo-Zaragoza et al. [2017] is to discriminate between staff-lines and musical symbols in musical scores, while the task in Kölsch et al. [2018] is to identify handwritten annotations inside of historical documents. LIke ours, both approaches leverage fully convolutional architectures for their respective semantic segmentation tasks. There are several differences which make our task more challenging. In Calvo-Zaragoza et al. [2017], the staff-lines and musical symbols, which the task wishes to distinguish, comprise a limited set of variations. Staff-lines appear in the same position with respect to the musical notes and tend to be long continuous horizontal lines. In contrast, our artifacts include lines, smudges, and spurious strokes in a variety of orientations and positions relative to the text. The historical document text characters in Kölsch et al. [2018] are printed while the annotations are handwritten, and the annotations have a slightly different shade, both of which are telltale signs for the network to discriminate. Our images on the other hand are binarized before entering the model, forcing our segmentation to rely solely on neighborhood spatial structure. Finally, both these approaches require full supervision via manually

labeled segmentation masks, while our approach is weakly supervised—only a single artifact image assembly function needs to be written.

Our contributions are threefold:

- **Novel application**: We tackle artifact removal in printed and handwritten text images, a problem not yet approached by deep learning.
- **Weakly supervised approach**: Our approach requires only a clean, unlabeled set of printed or handwritten text images and artifacts which are widely available and a simple program to assemble them together. No manual pixel-level annotation is necessary.
- **Empirical results**: Our artifact-cleansed images achieve low test error and consequently have convincing performance upon visual inspection. Further, our artifact-cleansed images improve recognition accuracy on well-known text recognition engines such as Tesseract 4.0.

2 Method



(a) An artifact-patched image is obtained through the pixel-wise union of the base and artifact images. DeepErase predicts the artifact mask, which is used to remove artifacts.

(b) Flow diagrams showing the artifact patching and artifact removal procedures

(c) Architecture for artifact segmentation

Figure 2: Illustrations of how artifact text images are assembled for training, and how artifacts are removed during inference

2.1 Summary of approach

Like other document binarization or segmentation tasks, we use a fully convolutional network to map the raw input image to a binary segmentation mask indicating artifact or no-artifact for each pixel in the image. Once the mask is obtained, all pixels on the mask indicating the presence of an artifact are set to 255 (white) on the input image, effectively cleansing it from artifacts. For training data, we automatically assemble a corpus of dirty images paired with their segmentation masks, generated using method described below in Section 2.3, for both printed and handwritten text. The network is trained and validated on this data, and then tested in-the-wild on the NIST dataset of scanned IRS tax returns. Code for experiments is available at `https://github.com/yikeqicn/DeepErase`.

2.2 Datasets

In this work we train and test on both printed and handwritten text. Since printed text is easy to generate, we generate 280k text images in various fonts of words pulled from Wikipedia using `TextRecognitionDataGenerator` Belval [2019]. For handwritten text, we use the IAM dataset Marti and Bunke [2002] consisting of about 110k handwritten words from 657 writers. For testing, we use the NIST SDB2 Johnson [2019] and NIST SDB6 Johnson [2019] datasets consisting of about 6k pages (each) of IRS tax return forms with printed and handwritten entries, respectively, each containing the types of artifacts that we wish to tackle in this work. We pre-crop text regions

from the IRS dataset using image registration (the IRS documents all share the same template, making image registration especially effective) and manually defined crop regions for the template. In total, we have 22165 printed text images and 35202 handwritten text images from the IRS forms for testing. All images are binarized prior to being input into the model.

2.3 Programmatic assembly of text images with artifacts

In order to automatically obtain a corpus of dirty images, we create a program which imposes realistic-looking artifacts on the readily available datasets of clean images. Similar ways of programmatically generating labeled data has been done for natural language processing tasks Ratner et al. [2016]. We focus on four types of artifacts: machine-printed underlines, machine-printed fill-in-the-blank boxes, random smudges, and handwritten spurious strokes.

Algorithm 1 Generation of text images with artifacts

Input clean image $\mathbf{x} \in [0, 255]^{n \times m}$, artifact sample $\mathbf{x}_{art} \in [0, 255]^{o \times p}$, offset
Begin
  Binarize $\mathbf{x}$ and $\mathbf{x}_{art}$ with threshold of 128
  Translate $\mathbf{x}_{art}$ by offset, expanding image if needed
  and filling additional pixels with intensity 255
  Crop $\mathbf{x}_{art}$ to the same size as $\mathbf{x}$
  Superimpose $\mathbf{x}_{art}$ onto $\mathbf{x}$ to get the dirty image,
  i.e. $\mathbf{x}_{dirty} \leftarrow \texttt{min}\,(\mathbf{x}, \mathbf{x}_{art})$
  Create segmentation mask,
  i.e. $\mathbf{s} \leftarrow \mathbf{x}_{art} + (255 - \texttt{max}\,(\mathbf{x}, \mathbf{x}_{art}))$
Return dirty image $\mathbf{x}_{dirty}$, segmentation mask $\mathbf{s}$

For random smudges and spurious strokes, we take a sampling of the IAM handwriting dataset to act as the artifacts. For line and box artifacts, we extract 5000 crops of horizontal and vertical lines and blank boxes from various sources of scanned forms, including the NIST IRS dataset as well as some internally scanned forms. See Figure 2a for an example of a base image and an artifact used in the assembly process.

The datasets contain many examples of forms from the same template (e.g. the 1040 tax form). To automate extraction of lines or boxes, we first apply conventional homography-based image registration to the entire dataset, and then iteratively crop the same region from each image.

We then binarize both the clean and artifact images. This ensures that our network cannot rely on subtle differences in shading to predict artifacts.

Next we sample an offset by which to translate the artifact image with respect to the clean image. This offset is sampled from a uniform distribution with bounds set such that the artifact falls within regions of the text that are consistent with the real-world. For instance, spurious strokes usually occur at the top or bottom of the image, while underlines usually occur at the bottom. We leave the boundaries of the distribution loose enough such that there is significant randomness and the artifacts overlap with the text characters a significant portion of the time.

After translating the artifact image by the offset amount, we then superimpose it onto the clean images by taking the lower intensity pixel (0 intensity corresponds to black) of the two (artifact and clean) images for each pixel in the clean image. Examples of the resulting dirty images are shown in Figure 3. The entire artifact text image generation pipeline is presented in Algorithm 1. Figures 2a shows examples of the intermediate images or masks and Figure 2b shows the artifact assembly (used during training) and removal (used during inference) pipelines.

Finally, the segmentation mask should contain all the markings of the artifact image minus the markings of the clean image. In other words, suppose that A was the set of pixels containing the artifact marks, and B is the set of pixels containing the clean marks. Then the segmentation mask (or pixels containing an artifact) would be $S = A - A \cap B$. During inference, once a segmentation mask is predicted, one can use it as a mask to erase the artifacts out of the image, as depicted in Figure 2b.

2.4 Model architecture and training

The network, schematic in Figure 2c is a simple U-net architecture Badrinarayanan et al. [2017] which predicts a segmentation mask of artifact or no-artifact for each pixel. Convolutions are performed in blocks of two layers. At the end of each block, the feature map is downsampled via maxpooling, and the number of channels is doubled. After two blocks, the feature maps are upsampled via deconvolution (or transposed convolution) for two blocks until the feature map resolution is same as the original image. The first feature map in each upsampling block is concatenated with the last feature map from the corresponding downsampling block, as is done in U-net.

The training objective is simply to minimize the cross entropy loss between the true segmentation mask and the predicted segmentation mask on a per pixel basis, with averaging in the end. To address the class imbalance issue (there are a lot more pixels labeled not-artifact than as artifact) we use the median frequency balancing scheme from Eigen and Fergus [2015]. No regularizers are used in the training objective. The RMSProp optimizer is used to minimize the objective.

To encourage translation and size invariance, we apply data augmentation in the form of resizing, followed by horizontal and vertical shifts of the image within the fixed 32×128 canvas.

3 Evaluation

3.1 Comparative artifact detectors

We compare DeepErase to two comparative artifact detectors.

Hough: The first is the widely used Hough-transform line detector, a classical computer vision method ubiquitous over the past several decades to detect and remove lines and other simple shapes from images. We utilize the standard OpenCV 3.0 Hough Line OpenCV [2019] implementation.

Manual Supervision CNN: Second, we implement the approaches of Calvo-Zaragoza et al. [2017] and also of Kölsch et al. [2018] without ImageNet pretraining, which are nearly identical to ours except for the use of full, manual supervision. The authors of Calvo-Zaragoza et al. [2017] manually annotated 20 scans of music documents for staff line removal. To be comparable, we manually annotated 60 document text images at the pixel level for training, costing about 3 man-hours. With such few examples, it is unlikely that the trained network will be able to model all the intricacies of artifact text, as we will see in Sections 3.3 and 3.4; this further highlights the need for weakly supervised approaches in order to achieve the dataset sizes needed for high model performance. We henceforth call this approach the "Manual Supervision" approach. More information on our implementation is found in the appendix.

In our validation set results (Table 2) we evaluate the Hough, Manual Supervision, and DeepErase approaches on a split of the datasets containing only line artifacts in order to ensure a fair comparison. Since the error for Manual Supervision and DeepErase on the line-artifacts-only split was always lower than its error for the entire dataset, we report only the error on the entire dataset for Manual Supervision and DeepErase. Since the Hough approach is validated on a split of the full validation set, it has a different value for recognition accuracy on dirty images in Table 2. Meanwhile the IRS dataset is consisted entirely of line (vertical or horizontal) artifacts so the dirty recognition accuracies in Table 3 are identical.

3.2 Metrics

Other than visual inspection, we use two metrics to determine our performance on artifact removal.

Segmentation error: First, we use the segmentation error on the validation set, which is the probability that a pixel on the predicted segmentation mask does not match the ground truth. *Baseline*: to compare our results, we include the segmentation error on the original clean text images before artifact assembly, which has a ground-truth segmentation mask that is uniformly annotated with no-artifact. This baseline ensures that when the artifact detector sees an image with no artfact inside, it does not falsely claim that there are artifacts.

Recognition error: The secondary metric that we use for evaluating performance is recognition error. The simple assumption is that images cleaned from artifacts will make it easier for recognition models

to discriminate. Two recognition error metrics are reported. Character error rate (CER) is the string edit distance between the predicted string and the ground truth string, or in other words, the minimum number of per-character add, delete, or replace operations needed to match the two strings. Word error rate (WER) is the probability that the predicted word does not match the ground truth, regardless of how far off it is. *Baseline*: Like the baseline for segmentation error, we use the recognition accuracy on the "gold-standard" original clean images without any artifacts superimposed as our recognition baseline. These are the raw unmodified images from `TextRecognitionDataGeneratoror` for printed and IAM for handwritten.

For printed text recognition we use the widely used open-source Tesseract v4 software. Since there is no widely available offline handwriting recognition software, we used the model from the `SimpleHTR` repo sim. Both softwares are based on an LSTM-CTC architecture.

3.3 Validation results

We first test our model on a held-out set of examples from our dirty datasets. Since we used a train/validation split of 9:1, the held-out set consists of 28k examples for printed and about 11k for handwritten. Since our dirty dataset was crafted from a base dataset (raw images from `TextRecognitionDataGenerator` or IAM), we report the performance of the original base images (which do not have artifacts) on the recognition models as our baseline.

Using DeepErase, we observe segmentation error of less than 5% on printed and handwritten text, which means that most pixels are correctly erased (see Table 1). In contrast, the Hough transform-based line removal achieves significantly higher error, since it removes entire lines including the parts which overlap with the text. The Manual Supervision approach performs better than Hough, but does not achieve as low of error as DeepErase, due to the shortage of available Manual Supervision data as discussed in Sec. 3.1.

Good segmentation leads to greatly improved recognition performance as well as shown in Table 2. When the artifacts are erased before inputting into Tesseract or `SimpleHTR`, the recognition accuracy improves by 60.56% and 31.20%, respectively, compared to no cleaning. DeepErase-cleaned images also achieve 20-60% lower downstream recognition word error than those clean by the Hough and Manual Supervision approaches. The segmentation is not perfect though—when compared with the "gold standard" base images, cleansed images get about 15-30% higher recognition error. Figure 3 shows some example images before and after artifact erasing.

Table 1: Segmentation results on validation set

	Segmentation error	
Setting	Baseline	Cleaned
Hough on printed	1.2	17.62
Manual Supervision on printed	0.55	6.16
DeepErase on printed	0.4	**3.38**
Hough on handwritten	0.56	15.31
Manual Supervision on handwritten	0.45	7.20
DeepErase on handwritten	0.25	**4.36**

Table 2: Recognition results on validation sets

	Baseline		Dirty		Cleaned	
Setting	CER	WER	CER	WER	CER	WER
Hough on printed	13.23	20.89	129.53	95.05	132.83	93.67
Manual Supervision on printed	13.23	20.89	104.98	93.89	53.12	54.94
DeepErase on printed	13.23	20.89	104.98	93.89	29.13	**34.71**
Hough on handwritten	6.89	20.22	50.51	78.34	52.32	81.71
Manual Supervision on handwritten	6.89	20.22	46.24	77.78	37.63	66.67
DeepErase on handwritten	6.89	20.22	46.24	77.78	28.58	**47.20**

3.4 Results on real-world NIST IRS dataset

In addition to evaluating on the validation set, we wish to test DeepErase in the wild on text from scanned IRS tax return forms. In-the-wild data tends to experience distribution shift Quionero-Candela et al. [2009], leading to lower performance when tested on models trained on data from other distributions. Typically this results in an iterative process where the training data is better adapted to the distribution in-the-wild, and the system is re-tested. We present results from our first-pass here, where we had not seen the IRS data before designing our artifact generation algorithm 1.

On the IRS printed data, removing artifacts via DeepErase lowers the Tesseract recognition error by 14.67% compared to not removing them, as shown in Table 3. Similarly on the handwritten data, it lowers the `SimpleHTR` recognition error by 13.52%. In both cases, DeepErase performs better than the Hough and Manual Supervision comparables.

Figure 4 shows examples of artifact removal in both printed and handwritten IRS text. Despite the relatively high recognition error on handwritten data even after cleaning (which is primarily due to distribution shift), upon visual inspection the erased images look reasonably good and indicate that the objective of artifact removal (to yield better results on other downstream recognition engines or other tasks) is satisfied.

Table 3: Recognition results on NIST IRS datasets

	Baseline		Cleaned	
Setting	CER	WER	CER	WER
Hough on printed	97.26	78.87	194.13	94.98
Manual Supervision on printed	97.26	78.87	67.66	73.89
DeepErase on printed	97.26	78.87	60.87	**64.20**
Hough on handwritten	94.93	98.38	81.19	93.09
Manual Supervision on handwritten	94.93	98.38	70.04	91.18
DeepErase on handwritten	94.93	98.38	59.91	**84.86**

4 Conclusion

We have presented DeepErase, a neural-based approach to removing artifacts from document text images. This task is challenging because it must rely solely on spatial structure (rather than differences in shading since the images are binarized) to do semantic segmentation of a wide variety of artifacts. We present a method to programmatically assemble unlimited realistic-looking text artifact images from real data and use them to train DeepErase in weakly supervised manner. The results on the validation set are excellent, showing good segmentation along with a 40 to 60% boost in recognition accuracy for both printed and handwritten text using common recognition software. On the real-world IRS dataset, DeepErase improves recognition accuracy by about 14% on both printed and handwritten text. The cleansed images on both printed and handwritten examples look visually convincing. Next steps include better modeling the test distribution during the artifact generation process such that the trained model performs better at test time.

Acknowledgments

The authors thank Ali Khan, Neil Khatu, Rajarajan Sampath, Darrin Williams, Sameer Gupta, and Joe Kruse for project oversight and/or financial support. The views and conclusions expressed in this material are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of Ernst & Young LLP.

References

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

Sherena G Johnson. Nist special database 33. 2012.

H. El Abed and V. Margner. The ifn/enit-database - a tool to develop arabic handwriting recognition systems. In *2007 9th International Symposium on Signal Processing and Its Applications*, pages 1–4, Feb 2007. doi: 10.1109/ISSPA.2007.4555529.

Jean-Philippe Thiran Guillaume Jaume, Hazim Kemal Ekenel. Funsd: A dataset for form understanding in noisy scanned documents. In *Accepted to ICDAR-OST*, 2019.

Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

Théodore Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *Advances in Neural Information Processing Systems*, pages 838–846, 2016.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12): 2481–2495, 2017.

Kai Chen, Mathias Seuret, Jean Hennebert, and Rolf Ingold. Convolutional neural networks for page segmentation of historical document images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 965–970. IEEE, 2017.

Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5315–5324, 2017.

Guillaume Renton, Clément Chatelain, Sébastien Adam, Christopher Kermorvant, and Thierry Paquet. Handwritten text line segmentation using fully convolutional network. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 5, pages 5–9. IEEE, 2017.

Jorge Calvo-Zaragoza, Gabriel Vigliensoni, and Ichiro Fujinaga. One-step detection of background, staff lines, and symbols in medieval music manuscripts with convolutional neural networks. In *ISMIR*, pages 724–730, 2017.

Andreas Kölsch, Ashutosh Mishra, Saurabh Varshneya, Muhammad Zeshan Afzal, and Marcus Liwicki. Recognizing challenging handwritten annotations with fully convolutional networks. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 25–31. IEEE, 2018.

Belval. Belval/textrecognitiondatagenerator, Jul 2019. URL https://github.com/Belval/TextRecognitionDataGenerator.

Sherena G. Johnson. Nist special database 2, Apr 2019. URL https://www.nist.gov/srd/nist-special-database-2.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575, 2016.

David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

OpenCV. https://docs.opencv.org, Jul 2019. URL https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html.

https://github.com/githubharald/simplehtr.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

A Example image results from validation set

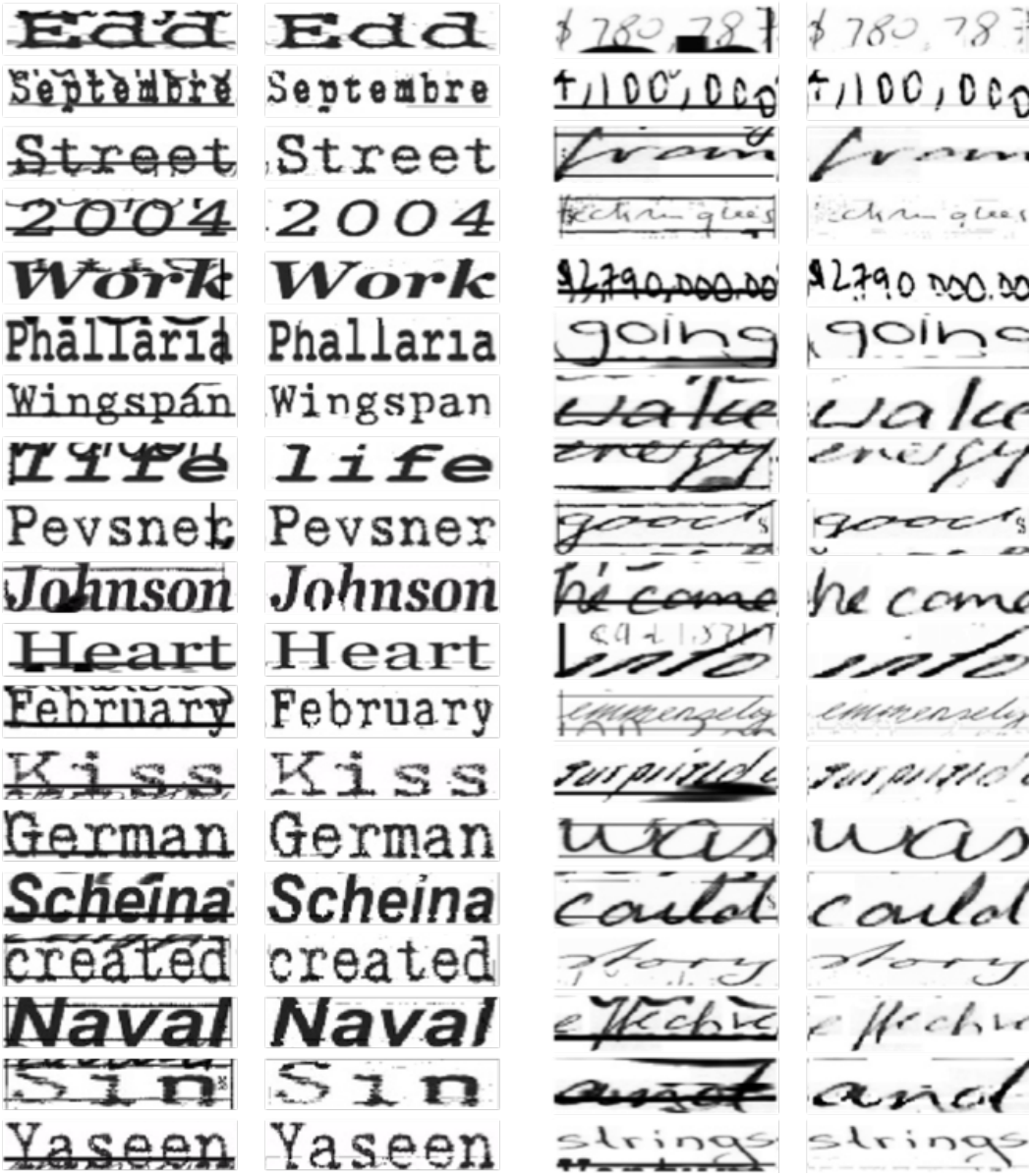


Figure 3: Examples from validation results. Columns 1 and 3 are before cleansing, 2 and 4 are after cleansing.

B Example image results from IRS dataset

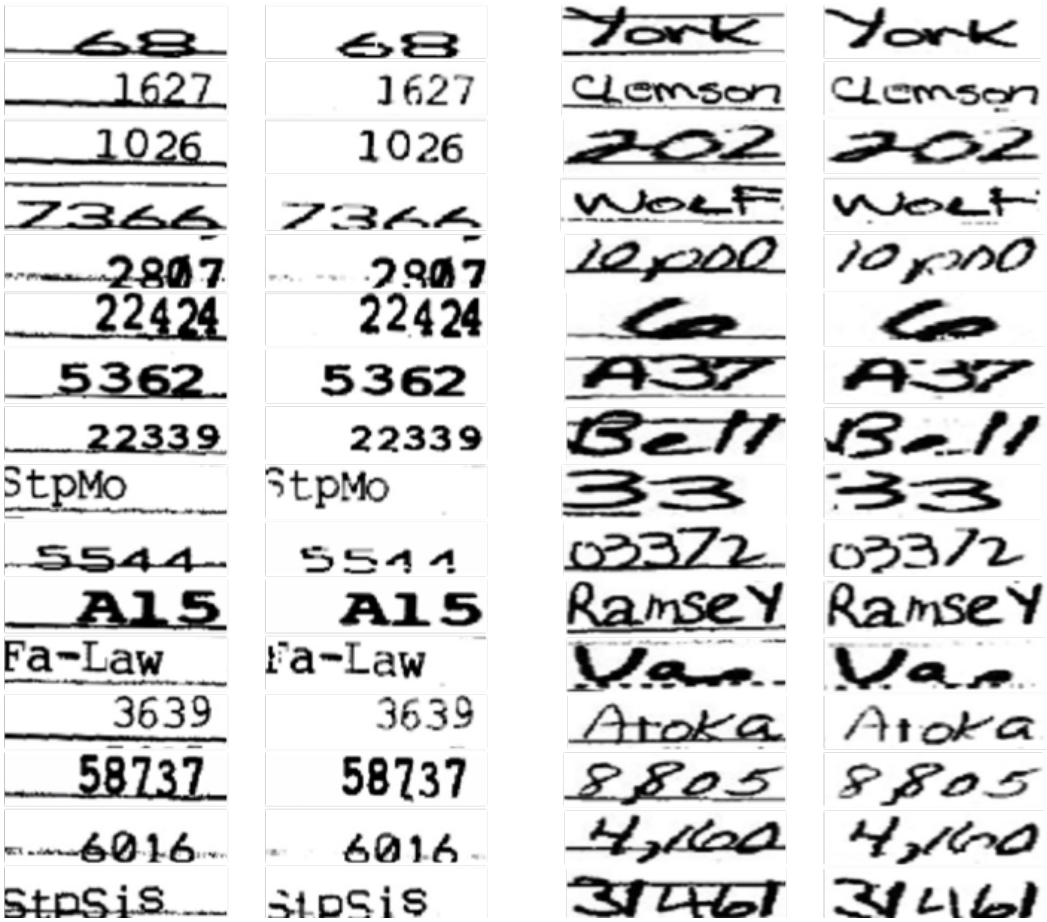


Figure 4: Examples from IRS results. Columns 1 and 3 are before cleansing, 2 and 4 are after cleansing.

C Manual Supervision CNN Model

We have provided a set of illustrations around the DeepErase model. In the appendix, we are presenting further details around the benchmark "Manual Supervision CNN" model (Section 3.1 in the main paper).

In Section 3.1 main paper, we described the training method for the benchmark Manual Supervision CNN Model. In Figure 5, we presents the process of model training as below:

- **Training Images**: The 60 synthetic images with artifacts were used as development images. See "Patched Image (x)" in the Figure 5.
- **Manual Coloring**: The artifact areas in the images were colored manually. See "Manual Label" in Figure 5.
- **Finalize Labeling**: The manual colored pixels were compared against the dark areas of original patched images. The pixels of the overlapped areas were verified as positive pixels. See "Overlapped Label(y)" in Figure 5.

The model prediction is the same as the DeepErase Model, which includes the following procedures:

- **Predict Masks**: Apply the model upon images with artifacts. The pixels of artifact areas would be predicted as positive. See "Predicted Mask" in the Figure 5.
- **Remove Artifacts**: Force the identified artifact area to be the background color. See "Artifact Removal" in Figure 5.
- **Downstream Process**: The cleaned images would be ready to use for downstream process. i.e. text recognition.

D Text Recognition Result Examples

In Section 3.3 and 3.4 main paper, we discussed the text recognition results for validation data and real-world NIST IRS datasets. In this section, we attach a few of result images as illustration. Overall, similar to DeepErase model, the Manual Supervision model was able to help remove artifacts.However, The removal was less accurate due to limited labeled training data and potentially less accurate manual labelling. The removal could be incomplete or overactive. See the result images in Figure 6 and Figure 7.

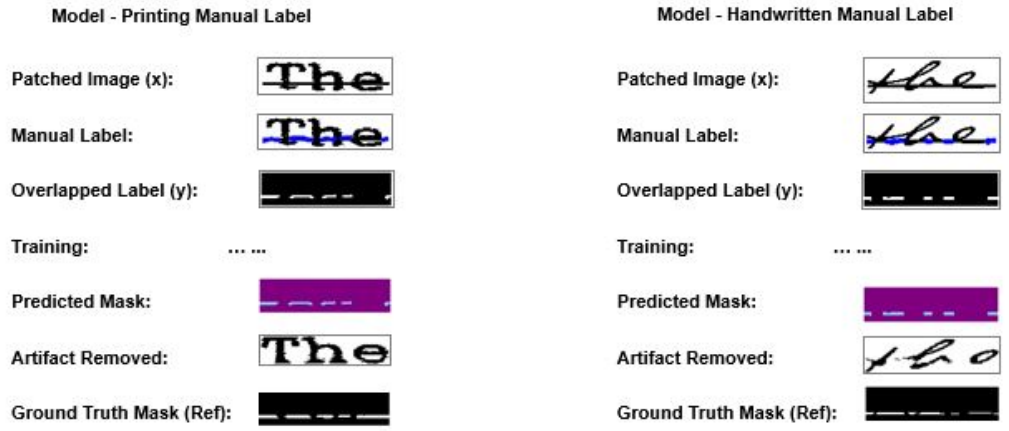


Figure 5: Manual Supervision CNN Model Training and Usage

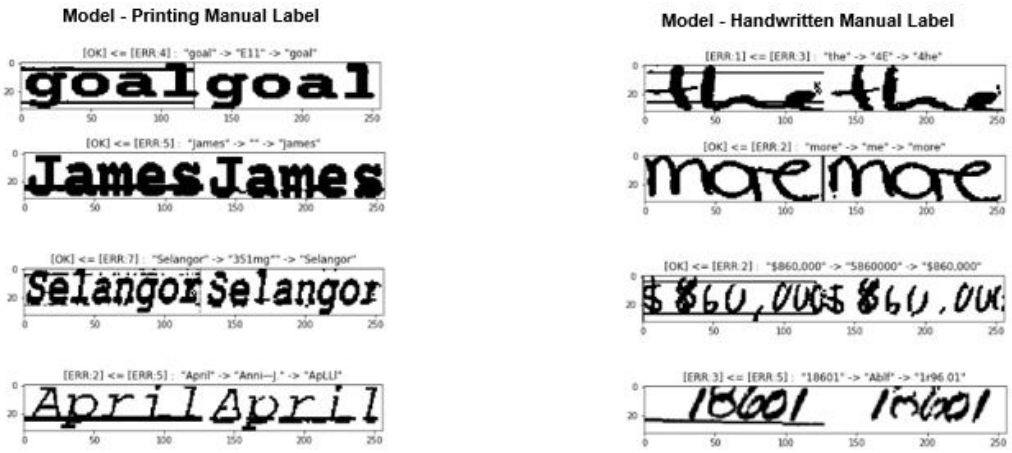


Figure 6: Manual Supervision CNN Model - Text Recognition Results - Validation Datasets

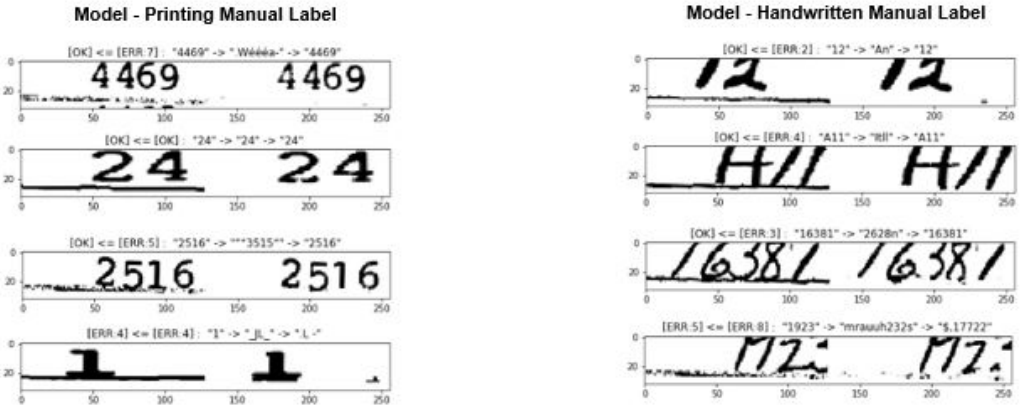


Figure 7: Manual Supervision CNN Model - Text Recognition Results - IRS NIST Datasets