DeepErase: Unsupervised Ink Artifact Removal in Document Text Images

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

Abstract

Paper-heavy industries like insurance, law, and government have long leveraged optical character recognition (OCR) to automatically transcribe hordes of scanned documents into text strings for downstream processing. 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. Further, the text region could have random ink smudges or spurious strokes. Such ink artifacts can severely interfere with the performance of recognition algorithms or other downstream processing tasks. In this work, we proprose DeepErase, a neural-based pre-processor to erase ink artifacts from text images. We devise a method to programmatically generate artificial text images with realistic-looking artifacts, and use them to train the U-net-like model in a totally unsupervised manner. In additional to high segmentation accuracy, we show that our cleansed images achieve a significant boost in recognition accuracy by popular OCR software such as Tesseract. Finally, we test DeepErase in-the-wild on scanned IRS tax return forms and achieve double-digit improvements in accuracy on printed text. All experiments are performed on both printed and handwritten text.

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

*Authors contributed equally.

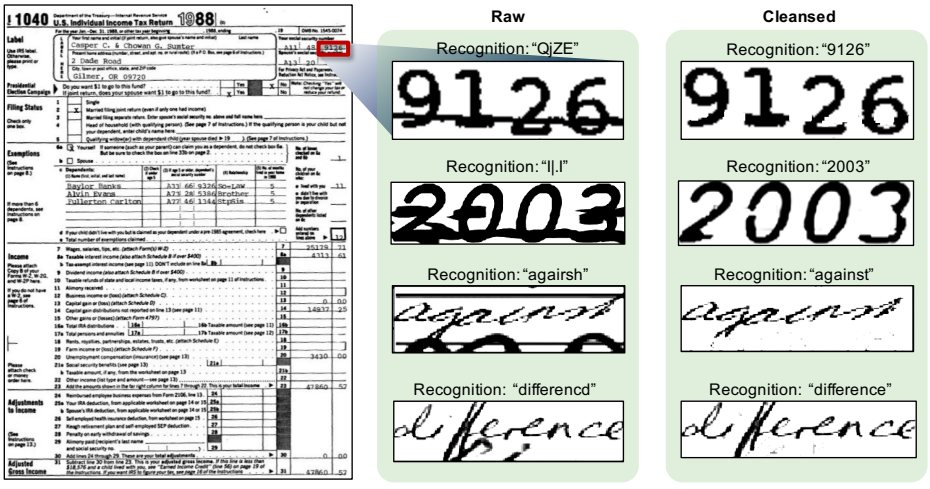


Figure 1: DeepErase cleans, or *erases*, ink artifacts from document text images, improving recognition accuracy, visual appeal, and other downstream tasks. Here we show text images cropped from scanned documents with various ink artifacts, such as underlines, boxes, smudges, and spurious strokes. DeepErase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

major advances in the performance of OCR systems, culminating in the large-scale adoption of OCR tools such as Tesseract, Abbyy, or Microsoft OCR.

OCR, or more generally document text recognition, relies on a two-step process: (1) Localization: determine regions of the image (i.e. bounding boxes) which contain text and crop out those regions. (2) Recognition: transcribe cropped text image into a text string. Localization was traditionally performed via sliding window-based techniques and nowadays is performed via region proposal networks [30]. Meanwhile, convolutional feature extractors [17] coupled with recurrent classifiers with the CTC loss has long been the workhorse of text recognition algorithms [11, 12], although more recent approaches use attention-based networks [5, 24].

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 [21], NIST SDB19 [14], and IFN/ENIT [9] 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 [13] segment their words free of underlines, boxes, and spurious strokes. Consequently, most results on text recognition have reported their performance on clean test examples [12, 4], 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.

One possible way to robustify a text recognition system is to train it on images containing the types of artifacts typically present in documents, making it robust against such perturbations, a method akin to data augmentation or adversarial training [10]. However, today most organizations are already set up with industrial-grade recognition systems wrapped in cloud and security infrastructure, rendering the prospect of overhauling the existing system with a homemade classifier (which is likely trained on much fewer data and therefore less performant) too risky an endeavor for most.

Nonetheless, many industrial-grade classifiers are still not robust to document images with ink artifacts (Figure 1. An alternative way to address this problem is to *erase* artifacts from the image before feeding it into the recognition engine. One might want artifact-cleansed images for other downstream tasks as well besides recognition, including signature extraction/verification [20] and document restoration, or simply for visual appeal; thus it is important to have an image pre-processing step that erases these artifacts.

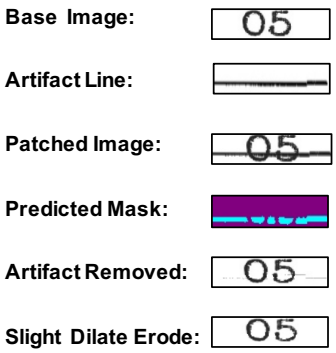
Little work has been done leveraging deep learning for document artifact removal. In this work, we present DeepErase, which inputs a document text image with ink artifacts and outputs the same image with artifacts erased (Figure 1). Training is totally unsupervised as we use a simple artifact generator 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 Tesseract, DeepErase achieves a 40-60% word accuracy improvement (over the dirty images) on our validation set and a 14% improvement on the NIST SDB2 dataset of scanned IRS documents.

1.1. Related work

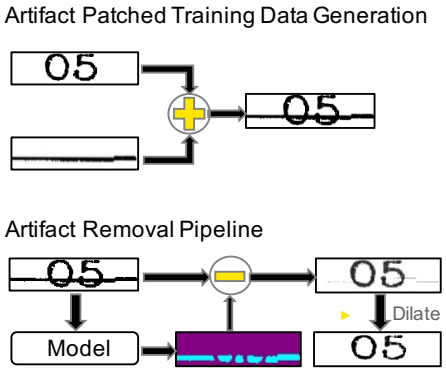
Our work is related broadly to the field of semantic segmentation [19, 28, 2], which predicts classes for different regions of the image. While semantic segmentation is typically applied to natural scenes, several works have applied it to documents for page segmentation [7], structure segmentation [31], or text line segmentation [27]. 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.

Classical methods for line artifact detection used the Hough transform to detect lines and other simple shapes in documents, such as ellipses [18, 22]. Such methods, however, do not pay attention to the spatial structure beyond specified shapes, and may erase parts of the clean text that overlapped with the artifact. Since the dawn of deep learning, similar tasks involving semantic segmentation in documents have been actively researched. Document binarization is a task in which each pixel in an RGB or grayscale image is assigned a binary value of either on or off. Binarization in low-contrast, degraded documents cannot rely solely on neighborhood-independent pixel thresholds and, like our task, must pay attention to the spatial patterns in the image. Recent approaches in binarization leverage multiscale convolutional networks to perform per-pixel binarization prediction [29].

The works of Calvo-Zaragoza et al. [6] and Kölsch et al. [16] are the most similar works to ours. The task in [6] is to discriminate between staff-lines and musical symbols



(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. A slight dilation/erosion operation further cleans residual artifact pixels.



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

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

in musical scores, while the task in [16] is to identify handwritten annotations inside of historical documents. Both approaches leverage fully convolutional architectures for their respective semantic segmentation tasks. There are several differences which make our task more challenging. In [6], 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 [16] 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 segmenter to rely solely on neighborhood spatial structure. Finally, both these approaches require manually labeled segmentation masks, while our approach is totally unsupervised.

1.2. Contributions

Our contributions are threefold:

- **Novel application**: We tackle artifact removal in printed and handwritten text images, a problem not yet approached by deep learning.
- **Unsupervised approach**: Our approach is unsupervised, requiring only a clean, unlabeled set of printed or handwritten text images which is widely available.
- **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.

2. Method

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 generate 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.

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` [3]. For handwritten text, we use the IAM dataset [21] consisting of about 110k handwritten words from 657 writers. For testing, we use the NIST SDB2 [15] and NIST SDB6 [15] 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 generation of text images with artifacts

In order to automatically generate 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 [26]. We focus on four types of artifacts: machine-printed underlines, machine-printed fill-in-the-blank boxes, random smudges, and handwritten spurious strokes.

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. 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 4. The entire artifact text image generation algorithm is presented in Algorithm 1.

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 $S = A - A \cap B$. We show our implementation in Algorithm 1.

Algorithm 1 Generation of text images with artifacts

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

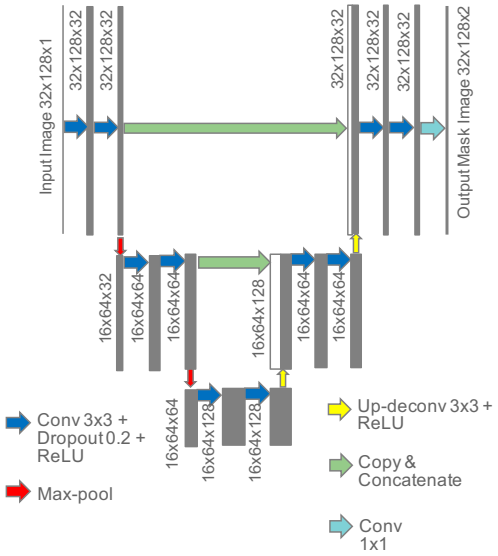


Figure 3: Our U-net architecture used for artifact segmentation.

2.4. Model architecture and training

The network, schematic in Figure 3 is a simple U-net architecture [2] 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

Table 1: Segmentation results on validation set

	Segmentation error	
	Baseline	Cleaned
Hough on printed	50	17.62
DeepErase on printed	50	**3.38**
Hough on handwritten	50	15.31
DeepErase on handwritten	50	**4.36**

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 [8]. No regularizers are used in the training objective. The RMSProp optimizer is used to minimize the objective.

To encourage translation and size invariances, 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. Baseline artifact detector

As a baseline artifact detector, we use the Hough transform line detector, a method ubiquitous over the past several decades to detect and remove underlines and other simple shapes from images. We utilize the standard OpenCV 3.0 Hough Line [23] implementation. In our validation set results (Table 2) we evaluate the Hough transform and DeepErase on a split of the datasets containing only line artifacts in order to ensure a fair comparison. Since the error for 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 DeepErase.

3.2. Evaluation metrics

Other than visual inspection, we use two metrics to determine our performance on artifact removal. 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. To compare our results, we include the baseline of a random segmentation, which has an error of 50% since the segmentation output is binary.

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.

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 [1]. 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 base images on the recognition models as a baseline from which to compare our results on the cleaned images.

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.

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. DeepErase also beats Hough line removal by similar margins. The segmenta-

Table 2: Recognition results on validation set

	Base		Dirty		Cleaned	
Setting	CER	WER	CER	WER	CER	WER
Hough on printed	13.23	20.89	129.53	95.05	132.83	93.67
DeepErase on printed	13.23	20.89	104.98	93.89	21.49	**33.33**
Hough on handwritten	6.89	20.22	50.51	78.34	52.32	81.71
DeepErase on handwritten	6.89	20.22	48.97	78.40	28.58	**47.20**

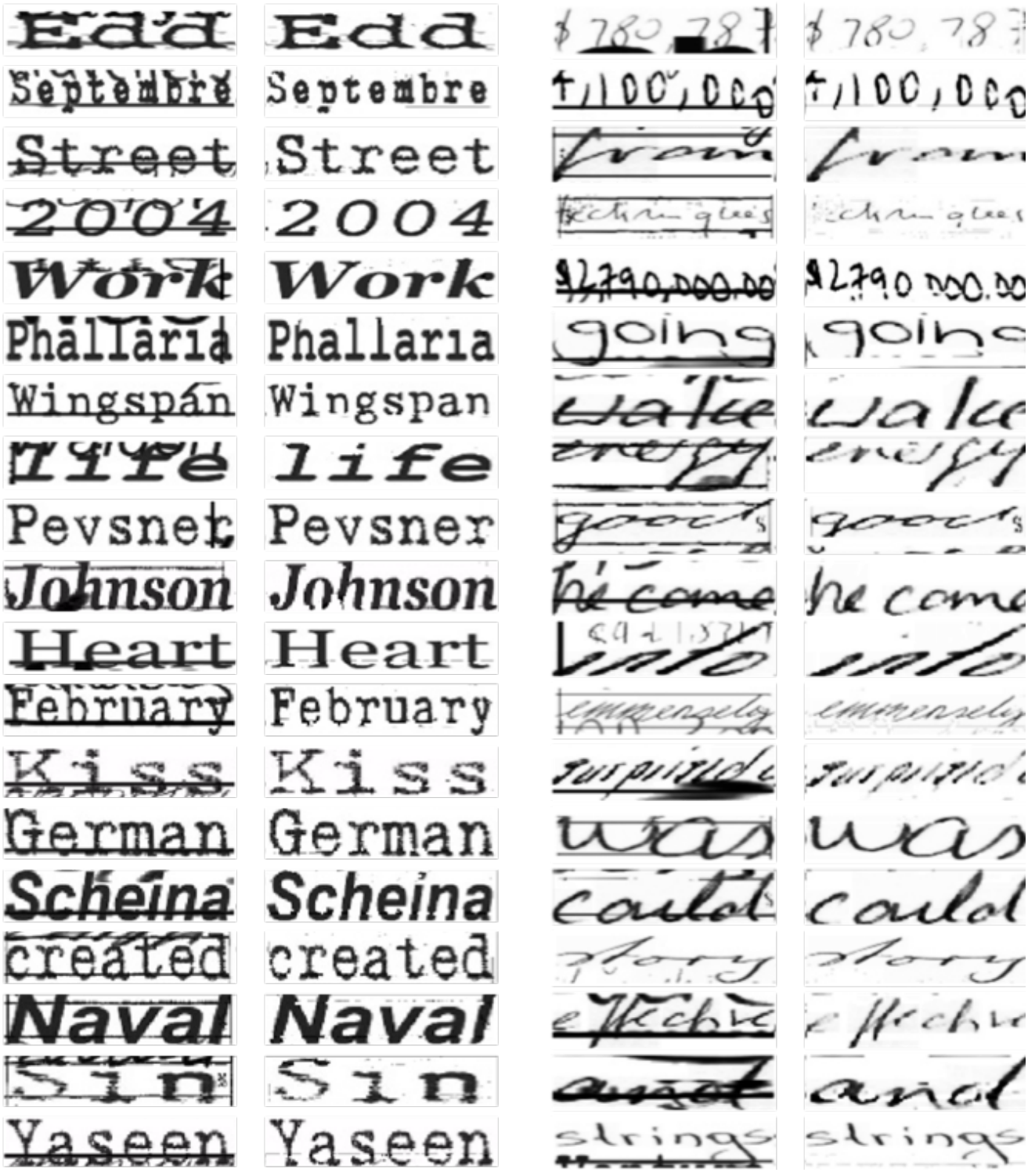


Figure 4: Examples from validation results. Columns 1 and 3 are before cleansing, 2 and 4 are after cleansing.

tion is not perfect though—when compared with the "gold standard" base images, cleansed images get about 15-30% higher recognition error. Figure 4 shows some example images before and after artifact erasing.

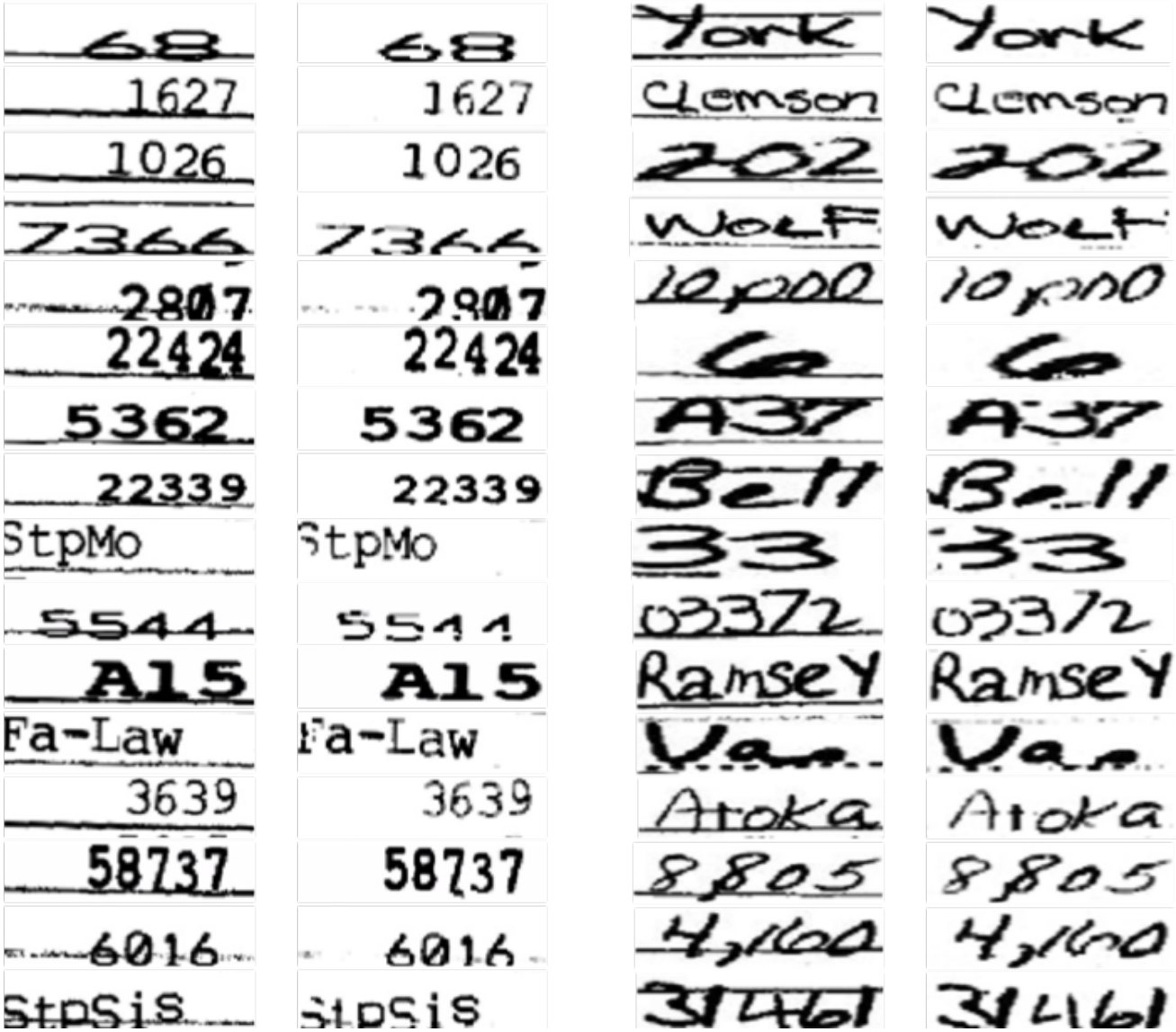


Figure 5: Examples from IRS results. Columns 1 and 3 are before cleansing, 2 and 4 are after cleansing.

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 [25], 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% as shown in Table 3. On the handwritten data, however, it actually increases word error by about 10%, although it decreases character error by about 4%. This could be due to a couple reasons. First, the research-codebase handwriting recognition model we use (`SimpleHTR`) was trained on limited data and performs unpredictably on slight image perturbations. One example is that "33" image on Figure 5 is classified as a "33" before cleansing and as "333" after cleansing. Such anomalies suggest that the result could have to do in part with an unstable recognition model which gets individual characters wrong for having very slight flaws. The fact that the character error decreases

Table 3: Recognition results on NIST IRS datasets

	Base/Dirty		Cleaned	
Setting	CER	WER	CER	WER
Hough on printed	97.26	78.87	194.13	94.98
DeepErase on printed	97.26	78.87	60.87	**64.20**
Hough on handwritten	20.31	38.67	19.04	50
DeepErase on handwritten	20.31	**38.67**	**16.88**	48.01

seems to support this. Second, the `SimpleHTR` model could have already been trained to be robust to underlines (the main source of artifact in the IRS dataset); therefore our erasing attempts provide no improvement to the recognition accuracy and occasionally remove a legitimate region of the image by accident (see "WOLF" in Figure 5 as an example of this).

In addition, Figure 5 shows examples of artifact removal in both printed and handwritten IRS text. Despite the higher recognition error garnered by DeepErase on the handwritten data, upon visual inspection the erased images look reasonably good and may yield better results on other downstream tasks, or on recognition with a different model.

4. Conclusion

In short, 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 generate unlimited realistic-looking text artifact images and use them to train DeepErase in an unsupervised 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 printed text. Despite having inconclusive recognition results on handwritten text, the cleansed images on handwritten IRS text look reasonable. Next steps would be to better model the test distribution during the artifact generation process such that the trained model performs better at test time.

References

[1] https://github.com/githubharald/simplehtr. 5

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017. 2, 4

[3] Belval. Belval/textrecognitiondatagenerator, Jul 2019. 3

[4] T. Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *Advances in Neural Information Processing Systems*, pages 838–846, 2016. 2

[5] T. Bluche, J. Louradour, and R. Messina. Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1050–1055. IEEE, 2017. 2

[6] J. Calvo-Zaragoza, G. Vigliensoni, and I. Fujinaga. One-step detection of background, staff lines, and symbols in medieval music manuscripts with convolutional neural networks. In *ISMIR*, pages 724–730, 2017. 2, 3

[7] K. Chen, M. Seuret, J. Hennebert, and R. 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. 2

[8] D. Eigen and R. 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. 5

[9] 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. 2

[10] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 2

[11] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006. 2

[12] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009. 2

[13] J.-P. T. Guillaume Jaume, Hazim Kemal Ekenel. Funsd: A dataset for form understanding in noisy scanned documents. In *Accepted to ICDAR-OST*, 2019. 2

[14] S. G. Johnson. Nist special database 33. 2012. 2

[15] S. G. Johnson. Nist special database 2, Apr 2019. 3

[16] A. Kölsch, A. Mishra, S. Varshneya, M. Z. Afzal, and M. 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. 2, 3

[17] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990. 2

[18] L. Likforman-Sulem, A. Hanimyan, and C. Faure. A hough based algorithm for extracting text lines in handwritten documents. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 2, pages 774–777. IEEE, 1995. 2

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 2

[20] C. I. Ltd. Icdar 2019 competition on signature verification based on an on-line and off-line signature dataset. 2

[21] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002. 2, 3

[22] J. Matas, C. Galambos, and J. Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000. 2

[23] OpenCV. https://docs.opencv.org, Jul 2019. 5

[24] J. Poulos and R. Valle. Attention networks for image-to-text. *arXiv preprint arXiv:1712.04046*, 2017. 2

[25] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009. 7

[26] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575, 2016. 4

[27] G. Renton, C. Chatelain, S. Adam, C. Kermorvant, and T. 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. 2

[28] O. Ronneberger, P. Fischer, and T. 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. 2

[29] C. Tensmeyer and T. Martinez. Document image binarization with fully convolutional neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 99–104, Nov 2017. 2

[30] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *European conference on computer vision*, pages 56–72. Springer, 2016. 2

[31] X. Yang, E. Yumer, P. Asente, M. Kraley, D. 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. 2