Hand-written Text Removal from Papers

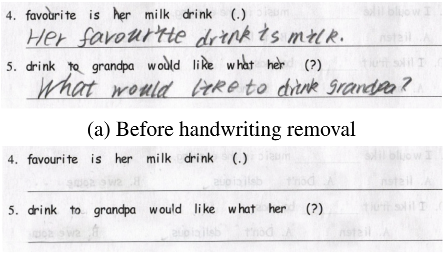
Jiangyi Liu
jiangyi.liu@wisc.edu

Venkata Abhijeeth Balabhadruni
balabhadruni@wisc.edu

Sujay Chandra Shekara Sharma
schandrashe5@wisc.edu

1. Introduction

The current maturity of computer vision algorithms enables its real-world application. Therefore, in this project, we also want to work on a problem that originated in our daily lives. In short, we focus on the problem of *removing hand-written text from papers*: Consider a sheet of paper (Figure 1a) with both printed text and hand-written text (e.g., exam papers or application forms). Given a photo of the paper, how can we remove hand-written text while still keeping the printed text? It is expected that ink from handwriting be replaced in a context-aware way (Figure 1b), so that the paper can be recovered to its state before being written by hand.



4. favourite is her milk drink (.)
Her favourtte drink ts milk.
5. drink to grandpa would like what her (?)
What would like to drink Grandpa?

(a) Before handwriting removal

4. favourite is her milk drink (.)
5. drink to grandpa would like what her (?)

(b) After handwriting removal

Figure 1. Example of hand-written text removal.

This problem has potential applications in privacy protection, education, and office work, e.g., redacting personal information from a scanned file.

2. Related Works

In the literature of Computer Vision, the *text removal* problem is closely related to the problem of removing hand-written text. Before the emergence of deep neural networks, manual marking of hand-written text is required. The algorithm then uses interpolation to overwrite hand-written areas with blanks. Such a method requires significant human intervention and effort and performs poorly when the background is complex or handwriting appears on images[2].

However, deep learning brings new possibilities to the text removal problem. Deep neural networks can understand the semantic information of images and can be used for locating texts from an image [8]. The classical approach for this task [6] uses a two-step structure: (1) apply techniques in object detection to identify areas with texts, and (2) use image inpainting algorithms to erase these areas. In some works, Generative Adversarial Networks[5] or sliding window [4] is used, but the main idea of a two-step structure remains unchanged. On the other hand, [9] is the first work that introduces an end-to-end network for this task, directly generating results after text removal.

The task of handwriting removal is similar to the text removal problem but slightly more difficult. While the objective of text removal is removing all text, handwriting removal requires first classifying handwritten and printed text and then only removing the former. There is only limited research in this area. [1] is the first work (to our knowledge) that attempts to solve the handwriting removal problem, introducing a “stroke mask” based on characteristics of hand-written text. However, its robustness is not good and it requires high-quality photos of paper.

Another issue is that only one dataset [1] is openly available for the handwriting removal problem, hindering related research.

3. Methods

We use deep learning to solve the task of hand-written text removal because the task inherently requires splitting printed and hand-written text. The bounding box level precision provided by traditional approaches is not enough.

Starting with the EnsExam network (which we reimplemented with publicly available EraseNet[3] code) , we first attempted to replicate the network and results mentioned in their paper [1], which is currently the state-of-the-art method in this area. Then, we proceeded by optimizing their network by:

- Switching to a different loss function;
- Add better data augmentation;
- Alter network structure for output image generation;
- Remove image compression in output.

3.1. New loss function

To start with, we noted that the *stroke normalization loss* mentioned in the EnsExam paper [1] is not performing well. Its formula is listed below:

$$L_{SN} = \frac{\|M_{gt_s} - M_s\|_1}{\min(\sum_{x,y} M_{gt_s}, \sum_{x,y} M_s)}$$

We argue that such a loss is not a good fit. Consider two photos, one with nearly no strokes, and another with a lot of strokes. If the number of mismatched pixels are roughly the same, the loss for photo with nearly no strokes will become much larger. Thus, such a loss prefers photos with no strokes in the train data, and this preference decreases the efficiency of learning.

We proposed another loss for stroke normalization, based on weighted loss function. Given that stroke pixels are much less than non-stroke pixels, we calculate their loss separately, and then perform re-balancing by dividing them with ratios of stroke pixels (resp. ratios of non-stroke pixels). If we have $p = \sum_{x,y} M_{gt_s}^{x,y} / (w \cdot h)$, the new loss then becomes:

$$L'_{SN} = \sum_{x,y} \frac{\|M_{gt_s}^{x,y} - M_s^{x,y}\|_1}{p \cdot M_{gt_s}^{x,y} + (1-p) \cdot (1 - M_{gt_s}^{x,y})}$$

The new loss solved the problem of preferring a certain kind of images by using a technique similar to train set rebalancing in classification tasks. In our benchmarks, it performs better than the original loss at identifying strokes.

3.2. Output Image Generation

In the EnsExam paper [1], they generated the final output I_{comp} by blending the predicted network output I_{re} with network input I_{in} and predicted mask M_b:

$$I_{comp} = I_{re} \cdot M_b + I_{in} \cdot (1 - M_b)$$

We proposed the following modifications, and looked into their performance by ablation studies:

- Use stroke-level mask M_s instead of M_b for blending;
- Apply image dilation to M_s before blending.

Image dilation is a filter that thickenes strokes. By widening the width of each stroke, we hope that the stroke mask covers each stroke fully, and consequently, replace them with pixels after stroke removal. We also hope that by using stroke mask M_s, which is more fine-grained than block mask M_b, performance can become better.

Also, we noted that JPEG format was used in the original EraseNet [3] (which is also likely to be inherited by EnsExam). We switched to using the lossless format PNG, which led to a performance boost.

Another improvement we made about output generation is clamping color values of masks (M_b, M_s) into the range of $[0, 1]$, which is not done by EnsExam. Such clamping will ensure the blending of M_b always generate I_{comp} inside the SDR color space, which matches our expectation. Without such a fix, outputs can be over-exposed, leading to lower metrics.

3.3. Data Augmentation

For better performance on different colors of papers, we performed color jittering on input data. We kept the existing data augmentation of random rotation.

4. Evaluation

Our evaluation consists of (a) our attempt to reproduce reported metrics in [1], and (b) several ablation studies:

- Comparison between L_{SN} and L'_{SN}, using visualized strokes;
- Comparison among different output image generation schemes;
- Metrics with different output image formats;

4.1. Metrics Explanation

Following [1, 3], we use the following metrics for our evaluation:

PSNR Peak signal-to-noise ratio is calculated between the ground truth image and the produced image after written text removal. Residual handwritten text is treated as noise. A higher signal-to-noise ratio means less leftover text.

MSSIM Multi-scale structural similarity is defined in [7], used to characterize similarity of images under the human visual system.

MSE Mean squred error between two images.

AGE Average of gray level difference between two images.

pEPs Percentage of error pixels between two images.

pCEPs Percentage of “clustered” error pixels, i.e., pixels whose neighboring four pixels mismatch.

4.2. Reproduction of EnsExam network

Due to the fact that EnsExam is not open source, we have to start from the code of its predecessor – EraseNet. Based on that, the following changes are made according to [1]:

- Add CBAM for deconvolution layers;
- Introduce loss function L_{SN} for strokes;
- Train stroke-level segmentation head.

Then, we trained the model with hyperparameters suggested in [3] and batch size 8. All training is performed on a GPU with model name “AMD(R) Radeon(TM) RX 7900 XTX”, and we stopped training after 100 epoches. Unfortunately, we were not able to reproduce claims made in [1]. Table 1 is a comparison between their claimed metrics and our results.

Given the results in Table 1, we propose the following guesses on possible reasons:

	PSNR	MSSIM	MSE	AGE	pEPs	pCEPs
Claimed	36.05	96.59	0.05	1.43	0.47	0.11
Ours	34.22	94.55	0.06	2.25	7.10	1.58

Table 1. Claimed metrics and our results on EnsExam. MSSIM, MSE, pEPs, and pCEPs are represented by percentage.

Differences in training platforms We are using an AMD GPU for training, which is not quite normal for machine learning. Such a device difference may lead to slightly different results.

Implementation errors Although we thoroughly checked our code, we cannot guarantee that our implementation strictly matches the desired algorithm in [1]. Also, there can be mismatch between [1] and their own implementation. On the other hand, we found that EraseNet's publicly available code [3] uses ground truth masks M_b^{gt} for blending. Theoretically speaking, this should not be allowed. We suspect that EnsExam could have benefited from such a trick if they also started from the code base of [3]. In fact, our implementation for midterm report suffered from this bug, and after fixing it, our metrics dropped.

4.3. Evaluate Stroke Normalization Loss

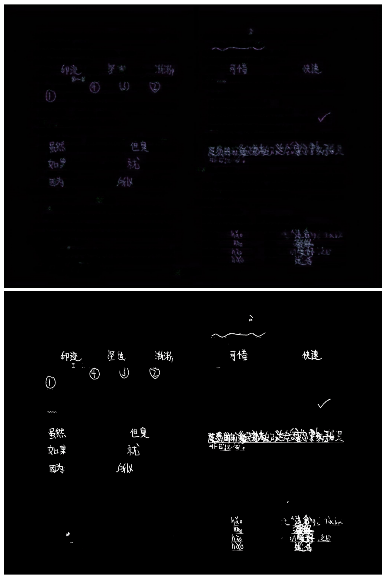


Figure 2. Strokes with L_{SN} (top) and $L_{SN'}$ (bottom).

Finding Our new loss L'_{SN} positively contribute to performance.

Figure 2 is an randomly chosen example that demostrated predicted strokes when the network is trained using different stroke normalization loss functions. By comparing side by side, we find that the stroke using old loss is not clear: some strokes are not visible at all, and we can see a grid-like noise on some longer lines. None of these problems are present when trained with the new loss L'_{SN}, which shows our new loss is better. We listed our a comparison of metrics in Table 2.

	PSNR	MSSIM	MSE	AGE	pEPs	pCEPs
Original	36.12	96.94	0.05	1.37	6.00	1.55
New loss	36.38	96.99	0.05	1.28	5.84	1.50

Table 2. Comparing original network with network using new loss L'_{SN}. Both are modified to output lossless images during evaluation.

4.4. Evaluate Output Image Generation

As mentioned in Section 3.2, our modification to output image generation is mainly about using lossless formats and modifying the mask used for blending.

Finding Using lossless format boost performance dramatically.

	PSNR	MSSIM	MSE	AGE	pEPs	pCEPs
JPEG	34.22	94.55	0.06	2.25	7.10	1.58
PNG	36.12	96.94	0.05	1.37	6.00	1.55

Table 3. Comparing network outputing JPEG images and PNG images. Here, only the original EnsExam network is used.

Table 3 shows metrics calculated with different output formats. We conclude that also JPEG compression usually does not introduce human-identifiable artifacts, it does negatively impact the metrics.

Finding The block mask M_b performs the best in output image generation.

	PSNR	MSSIM	MSE	AGE	pEPs	pCEPs
M_s	35.90	96.47	0.05	1.36	6.41	1.55
M_s^d	36.06	96.94	0.05	1.29	6.18	1.62
M_b	36.38	96.99	0.05	1.28	5.84	1.50
no mask	35.25	96.70	0.06	1.62	6.20	1.62

Table 4. Comparing usage of different masks. New loss and lossless output image format are used.

Table 4 shows the metrics when output image generation uses block mask M_b, stroke mask M_s, or stroke mask with dilation M_s^d. For the stroke mask with dilation, dilation kernel size is set to be $d = 7$. According to the table, when we use a mask, if a mask covers more area the performance becomes better. This might be because the stroke mask may failed to cover some strokes which the block mask may cover thanks to its larger covered area. However, if the mask covers the whole image (see the line "no mask") the performance drops. This is because for areas without strokes it's better to copy pixels from the input image, since the network may slightly alter these pixels.

4.5. Data Augmentation

We applied the data augmentation, and re-trained the model with the same hyperparameters and epoches. The results are listed in Table 5.

Enabled?	PSNR	MSSIM	MSE	AGE	pEPs	pCEPs
No	36.38	96.99	0.05	1.28	5.84	1.50
Yes	36.10	97.13	0.05	1.26	5.82	1.50

Table 5. Results after applying data augmentation

Finding Introducing data augmentation lead to a slightly better MSSIM, but decreases PSNR. In total, it's not quite effective.

5. Conclusion

In this project, we optimized state-of-the-art methods for handwritten text removal. Such a task is useful for education and privacy protection. We first attempted to reproduce metrics reported in [1] but failed, so we analyzed possible reasons. Then, we proposed several ways of improving metrics. Our improved version is available on GitHub. We improved the peak signal-noise ratio from 34.22 to 36.38. The full comparison is shown in Table 6.

	PSNR	MSSIM	MSE	AGE	pEPs	pCEPs
Original	34.22	94.55	0.06	2.25	7.10	1.58
Optimized	36.38	96.99	0.05	1.28	5.84	1.50

Table 6. Comparison between the original network and our optimized version.

Contribution of Team Members Everyone participated in evaluation design. Jiangyi took the role of coding and literature research. Abhijeeth and Sujay were in charge of all the writeup and slides.

References

[1] Liufeng Huang, Bangdong Chen, Chongyu Liu, Dezhi Peng, Weiying Zhou, Yaqiang Wu, Hui Li, Hao Ni, and Lianwen Jin. Ensexam: A dataset for handwritten text erasure on examination papers. In *International Conference on Document Analysis and Recognition*, pages 470–485. Springer, 2023. 1, 2, 3, 4

[2] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International journal of computer vision*, 116:1–20, 2016. 1

[3] Chongyu Liu, Yuliang Liu, Lianwen Jin, Shuaitao Zhang, Canjie Luo, and Yongpan Wang. Erasenet: End-to-end text removal in the wild. *IEEE Transactions on Image Processing*, 29:8760–8775, 2020. 1, 2, 3

[4] Toshiki Nakamura, Anna Zhu, Keiji Yanai, and Seiichi Uchida. Scene text eraser. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 832–837. IEEE, 2017. 1

[5] Siyang Qin, Jiahui Wei, and Roberto Manduchi. Automatic semantic content removal by learning to neglect. *arXiv preprint arXiv:1807.07696*, 2018. 1

[6] Osman Tursun, Simon Denman, Sabesan Sivapalan, Sridha Sridharan, Clinton Fookes, and Sandra Mau. Component-based attention for large-scale trademark retrieval. *IEEE Transactions on Information Forensics and Security*, 17: 2350–2363, 2019. 1

[7] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 2

[8] Qixiang Ye and David Doermann. Text detection and recognition in imagery: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 37(7):1480–1500, 2014. 1

[9] Shuaitao Zhang, Yuliang Liu, Lianwen Jin, Yaoxiong Huang, and Songxuan Lai. Ensnet: Ensconce text in the wild. In *Proceedings of the AAAI conference on artificial intelligence*, pages 801–808, 2019. 1