MSdocTr-Lite: A Lite Transformation for Full Page Multi-script Handwritting Recognition

Kim Long Hoang   
May 31, 2024

***Abstract*— The Transformer architecture excels in pattern recognition but requires large datasets for training and validating. In Handwritten Text Recognition (HTR), gathering extensive labeled data is challenging and costly. This paper presents a lightweight transformer for full-page multiscript handwriting recognition, addressing data scarcity by enabling training on reasonably sized datasets with- out external data. It learns page-level reading order using a curriculum learning strategy, eliminating line segmentation errors and reducing segmentation annotation needs. Additionally, it supports easy adaptation to other scripts through transfer learning with page-level labeled images. Experiments on various datasets (French, English, Spanish...) demonstrate the model’s effective- ness. Keywords: Seq2Seq model, page-level recognition, Handwritten Text Recognition, Multi-script, Transformer, Transfer Learning**

# Introduction

Handwritten Text Recognition (HTR) converts scanned handwritten documents into machine-readable text but faces challenges due to handwriting variability and segmentation issues. Traditional methods struggle with segmentation, leading to errors. Recent approaches using deep learning, like trans- former models, aim to recognize text at the page level, avoiding segmentation. This paper proposes a lightweight transformer model trained with a curriculum learning strategy, which is efficient, adaptable to various scripts, and performs well across multiple languages. The structure includes related work, proposed approach, experimental results, and conclusions.

# Theoretical Background

## Background on Handwritten Text Recognition (HTR)

In the past, Various approaches have been developed but it stil have a lot of challenging problem due to the high variability in writing style, illegible handwriting, and the degradation of document quality over time. Early methods focused on character-level segmentation, but they struggled with the inherent difficulties of cursive handwriting.  
To address this, word-based and line-based segmentation techniques were introduced, each offering improvements yet still facing significant hurdles.

## *Overview of current HTR approaches*

* Character-Level Segmentation

A close-up of a paper

Description automatically generated

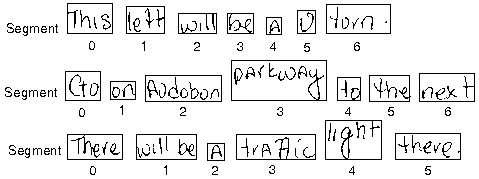
Figure 1:

What it does: Extracts individual character images from documents for recognition.

Weaknesses: Performance is generally low due to the difficulty in segmenting cursive and unconstrained handwriting accurately.

Proposed Solution:Move towards word-based or line-based segmentation to reduce the segmentation errors at the character level.

* Word-Based Segmentation



What it does:

Extracts word images from documents before feeding them into the HTR system.

Weaknesses:

Faces issues due to irregular inter-word and intra-word spaces, making segmentation inconsistent

Proposed Solution:

Adopt text line recognition to utilize more context and reduce the inconsistencies faced in word segmentation.

* Line Segmentation

What it does:

Segments documents into individual lines to surpass inconsistencies between words and provides more context for recognition.

Weaknesses:

Challenges arise from non-uniform text line skew/slant or closely situated and touching text lines, affecting recognition performance.

Proposed Solution:

Explore paragraph or page-level recognition to avoid intermediate segmentation steps and leverage deep learning models.

* Paragraph-Level Recognition with MDLSTM and Attention

What it does:

Uses Multi-Dimensional Long Short-Term Memory Recurrent Neural Networks (MDLSTM) and attention mechanisms for paragraph-level HTR.

Weaknesses:

High memory requirements, lack of GPU acceleration for training, and intractable inference time led to abandonment

Proposed Solution:

Investigate transformer-based architectures and curriculum learning strategies to improve efficiency and scalability.

* Sequence-to-Sequence (Seq2Seq) Models Based on Transformer Architecture

What it does:

Utilizes the transformer architecture for page-level handwritten document text recognition.

Weaknesses:

Requires a massive amount of annotated data for training.

Long feature sequences lead to extensive training times and high GPU resource consumption.Difficult to port to low-resource languages due to the need for extensive data for retraining.

Proposed Solution:

Develop a lite transformer model that requires fewer parameters and can be trained on standard GPUs. Implement curriculum learning to efficiently train the model with limited annotated data and facilitate transfer learning to adapt to different scripts.

## Motivation for developing a lite transformer model

Motivated by the need for efficient and effective handwritten document recognition, we propose a lite transformer model for page-level handwritten text recognition. This model uses a limited number of parameters and can be trained without external data. Employing a curriculum learning strategy, the model learns reading order and scales to large text images. This strategy is applied once, making the model adaptable to different scripts with minimal additional training. This architecture requires less memory, allowing training on standard GPUs. Key contributions include:

**Key Contributions:**

* An end-to-end lite transformer-based model for page-level handwritten text recognition, avoiding early segmentation errors and leveraging larger context.
* Curriculum learning strategy to reduce the need for extensive annotated data and enable the model to learn reading order at the page level.
* Adaptability to different scripts using a simple transfer-learning process with page-level labeled images.
* Extensive comparative experiments demonstrate effectiveness across multiple scripts and languages (e.g., English, French, Spanish, Arabic).

By addressing the weaknesses of previous approaches, the lite transformer model aims to offer a more efficient and adaptable solution for handwritten text recognition.

# Related Work

## Review of line level HTR systems

The Line-level HTR system are specialized tools designed to recognize and transcribe text at a line level, meaning they process one line of text at a time. This system are crucial for digitizing handwritten documents, enabling easier search, editing, and storage.  
The Key Components on Line-Level HTR-systems are

* Preprocessing
* Feature Extraction
* Recognition
* Postprocessing  
  The problem of mapping transcript to images has motivated some research in the past decade, either for the alignment of Optical Character Recognition (OCR) output with book content (e.g. in [5], [6]), or for mapping the transcription of historical documents to segmented words or lines (such as [7], [8]).

## Review of Page level

## A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have already been defined in the abstract. Abbreviations such as IEEE, SI, ac, and dc do not have to be defined. Abbreviations that incorporate periods should not have spaces: write “C.N.R.S.,” not “C. N. R. S.” Do not use abbreviations in the title unless they are unavoidable (for example, “IEEE” in the title of this article).

# Proposed Approach

# Experimental Results

## Experimental Setup

* Database:

To assess the quality, the advantages and weaknesses of this system, we will use databases for which we know the ground truth for both the line segmentation and the line transcription. Such databases are publicly available and extensively used in automatic text recognition problems. The Rimes database [1] consists of a training set of 1,500 images of handwritten paragraphs in French, and an evaluation set of 100 images. The IAM database [2] consists of 747 images of handwritten documents in English for training, 116 for validation, and 336 for evaluation. Examples of images are shown in Fig. 3. We carried out experiments on parts of the databases which were not used to train the recognizers.

# IV. Guidelines for Graphics Preparation and Submission

## E. Resolution

The proper resolution of your figures will depend on the type of figure it is as defined in the “Types of Figures” section. Color and grayscale figures should be at least 300dpi. Line art, including tables should be a minimum of 600dpi.

## F. Vector Art

In order to preserve the figures’ integrity across multiple computer platforms, we accept files in the following formats: .EPS/.PDF/.PS. All fonts must be embedded or text converted to outlines in order to achieve the best-quality results.

## G. Color Space

The term “color space” refers to the entire sum of colors that can be represented within the said medium. For our purposes, the three main color spaces are grayscale, RGB (red/green/blue), and CMYK (cyan/magenta/yellow/black). RGB is generally used with on-screen graphics, whereas CMYK is used for printing purposes.

All color figures should be generated in RGB or CMYK color space. Grayscale images should be submitted in grayscale color space. Line art may be provided in grayscale OR bitmap colorspace. Note that “bitmap colorspace” and “bitmap file format” are not the same thing. When bitmap color space is selected, .TIF/.TIFF/.PNG are the recommended file formats.

## H. Accepted Fonts Within Figures

When preparing your graphics, IEEE suggests that you use one of the following Open Type fonts: Times New Roman, Helvetica, Arial, Cambria, or Symbol. If you are supplying EPS, PS, or PDF files, all fonts must be embedded. Some fonts may only be native to your operating system; without the fonts embedded, parts of the graphic may be distorted or missing.

A safe option when finalizing your figures is to strip out the fonts before you save the files, creating “outline” type. This converts fonts to artwork which will appear uniformly on any screen.

## I. Using Labels Within Figures

1. **Figure Axis Labels**
   1. Figure axis labels are often a source of confusion. Use words rather than symbols. As an example, write the quantity “Magnetization” or “Magnetization *M*,” not just “*M*.” Put units in parentheses. Do not label axes only with units. For example, write “Magnetization (A/m)” or “Magnetization (Am−1),” not just “A/m.” Do not label axes with a ratio of quantities and units. For example, write “Temperature (K),” not “Temperature/K.”
   2. Multipliers can be especially confusing. Write “Magnetization (kA/m)” or “Magnetization (103 A/m).” Do not write “Magnetization (A/m) × 1000” because the reader would not know whether the top axis label means 16000 A/m or 0.016 A/m. Figure labels should be legible, approximately 8- to 10-point type.
2. **Subfigure Labels in Multipart Figures and Tables**

Multipart figures should be combined and labeled before final submission. Labels should appear centered below each subfigure in 8-point Times New Roman font in the format of (a) (b) (c).

## J. Referencing a Figure or Table Within Your Article

When referencing your figures and tables within your article, use the abbreviation “Fig.” even at the beginning of a sentence. Do not abbreviate “Table.” Tables should be numbered with Roman numerals.

## K. Submitting Your Graphics

Because IEEE will do the final formatting of your article, all figures, figure captions, and tables can be placed at the end of your article. However, if you do place your figures within the article, they should be placed at the top of the page, closest to the first mention in the text. Figures should be submitted as individual files, separate from the manuscript in one of the file formats listed above. Place figure captions below the figures; place table headings above the tables. Do not include captions as part of the figures, or put them in “text boxes” linked to the figures. Also, do not place borders around the outside of your figures.

## L. Color Processing / Printing in IEEE Transactions, Journals, and Letters

All IEEE Transactions, Journals, and Letters allow an author to publish color figures on IEEE *Xplore* at no charge, and automatically convert them to grayscale for print versions. In most journals, figures and tables may alternatively be printed in color if an author chooses to do so. Please note that this service comes at an extra expense to the author. If you intend to have print color graphics, you will have the opportunity to indicate this in the Author Gateway and will be contacted by PubOps to confirm the charges.

V. Conclusion

A conclusion section is not required. Although a conclusion may review the main points of the article, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

# Appendix

Appendixes, if needed, appear before the acknowledgment.

# References and Footnotes

## A. References

REFERENCES

[5] S. Feng and R. Manmatha, “A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books,” Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries- JCDL ’06, pp. 109–118, 2006. [6] I. Z. Yalniz and R. Manmatha, “A fast alignment scheme for automatic ocr evaluation of books,” in Document Analysis and Recognition (ICDAR), 2011 International Conference on. IEEE, 2011, pp. 754–758. [7] J. Rothfeder, R. Manmatha, and T. M. Rath, “Aligning Transcripts to Automatically Segmented Handwritten Manuscripts,” in Proc. 7th Int. Workshop on Document Analysis Systems, 2006, pp. 84–95. [8] E. Kornfield, R. Manmatha, and J. Allan, “Text alignment with hand written documents,” in First International Workshop on Document Image Analysis for Libraries, 2004. Proceedings. IEEE, 2004, pp. 195–209.