Published in Image Processing On Line on 2026–04–00.
Submitted on 2024–10–30, accepted on 2026–03–11.
ISSN 2105–1232 © 2026 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2026.587

Transcribing Lines of Handwritten Text Using TrOCR: An Encoder-Decoder Model Based on Pre-Trained Image and Text Transformers

Natalia Bottaioli[1], Daniel Parres[2], and Yung-Hsin Chen[3]

[1]Université Paris-Saclay, ENS Paris-Saclay, Centre Borelli, France
natalia.bottaioli@ens-paris-saclay.fr
[2]PRHLT Research Center, Universitat Politècnica de València, Valencia, Spain
dparres@prhlt.upv.es
[3]RobotiFAI, Zürich, Switzerland
yunghsin.c@robotif.ai

Communicated by Jean-Michel Morel *Demo edited by* Natalia Bottaioli

Abstract

This article focuses on analyzing several aspects of the handwritten text recognition (HTR) models belonging to the TrOCR family introduced by Minghao Li et al. in [TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models, AAAI Conference on Artificial Intelligence, 2023]. The TrOCR models are designed to recognize single lines of English text using a transformer-based encoder-decoder architecture. All models incorporate a pre-trained vision transformer as the encoder and a pre-trained text transformer as the decoder. The encoder is responsible for extracting key features from the image, while the decoder autoregressively transcribes the text, subword by subword, based on the extracted features. The authors report state-of-the-art performance across different text types, including handwritten, scene, and printed text. Our analysis has several objectives. The first one is to gain a better understanding of the training process and the data used for producing the handwritten models. The second one is to highlight and explore the functionality and limitations of the TrOCR model in the context of HTR, which poses unique challenges such as variations in individual writing styles. Additionally, we propose an architecture diagram that helped us better understand what the model actually does with the input text line image, which we hope will be useful for the research community using TrOCR.

Source Code

The source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the `README.md` file of the archive. The authors' original method implementation is available here: TrOCR[2].
This is an MLBriefs article. The source code has not been reviewed!

.

Keywords: optical character recognition; transformer-based HTR; handwritten text recognition

[1]https://doi.org/10.5201/ipol.2026.587
[2]https://huggingface.co/microsoft/trocr-base-handwritten

1 Introduction

Optical Character Recognition (OCR) has evolved through several distinct technological eras, each shaped by the limitations and possibilities of its time. Early systems relied on handcrafted rules and geometric heuristics, where characters were recognized by comparing strokes or silhouettes to predefined shapes [23][6][24]. These methods worked reasonably well for clean, machine-printed text but were vulnerable to the variation in writing styles or distortions [31][26]. The emergence of machine learning brought the first substantial leap: classical neural networks and Hidden Markov Models allowed models to learn statistical properties of characters and their sequences [36]. This era was followed by the dominance of convolutional neural networks (CNNs) [16], which extracted spatial features far more effectively, often paired with recurrent neural networks (RNNs) such as LSTMs for sequence modeling [2]. OCR research then widely adopted the Connectionist Temporal Classification (CTC) loss [8][7], which enabled end-to-end training without explicit character segmentation and secured the CNN–RNN–CTC pipeline as the standard for handwritten and scene text recognition [29][27].

Despite their success, CNN–RNN systems faced inherent constraints: sequential recurrence limited parallelization and long-range dependencies, and CTC-based decoding struggled with highly irregular text or stylistic variation [10]. The introduction of transformers [32] resolved this by replacing recurrence with attention mechanisms capable of modeling global context in a single step. Vision transformers (ViT) demonstrated that images could be treated as sequences of patches, while text transformers captured linguistic structure more effectively than RNNs [5]. OCR models began integrating these components – either by augmenting CNN backbones with attention layers or by adopting fully transformer-based architectures. TrOCR [17] represents the culmination of this shift: a unified encoder–decoder model that applies a vision transformer to image patches and a text transformer to generate transcriptions autoregressively. Crucially, TrOCR leverages large-scale pre-training for both components, marking a shift from task-specific OCR architectures to general-purpose foundation models adapted to the OCR domain.

Despite this strong reported performance, TrOCR remains a complex model whose training process, data lineage, and downstream behavior are not fully transparent in the original publication. The model is trained through a multi-stage pipeline involving large-scale text extraction from PDF documents, extensive synthetic data generation, and fine-tuning on curated benchmarks. Each stage produces intermediate model variants that differ not only in parameter scale, but also in their exposure to specific types of textual input. Understanding this training trajectory is essential for interpreting TrOCR's strengths and limitations, especially in handwritten text recognition, where variability in writing style, stroke continuity, background color, and character regularity pose challenges that differ markedly from printed or scene text.

This article focuses on the handwritten branch of the TrOCR model family. First, it reconstructs and clarifies the multi-stage training pipeline that leads to the publicly available handwritten models, including an examination of the data used at each stage. Second, it provides an exploratory analysis of the behavior of TrOCR on handwritten text, illustrating where the model performs reliably and where it exhibits systematic weaknesses. Particular attention is paid to issues such as sensitivity to framing, difficulties with cursive handwriting, lack of robustness to non-white backgrounds, and the model's tendency to ignore case distinctions. Through these analyses, we hope to provide practical guidance for researchers and practitioners who intend to use TrOCR for handwritten text recognition and to highlight areas where further model development or fine-tuning may be necessary.

2 TrOCR Architecture

TrOCR is the name given to the end-to-end text recognition encoder-decoder family of models introduced in [17], where a pre-trained image transformer acts as the encoder and a pre-trained text transformer acts as the decoder. Its architecture, according to the original paper, is summarized in Figure 1.

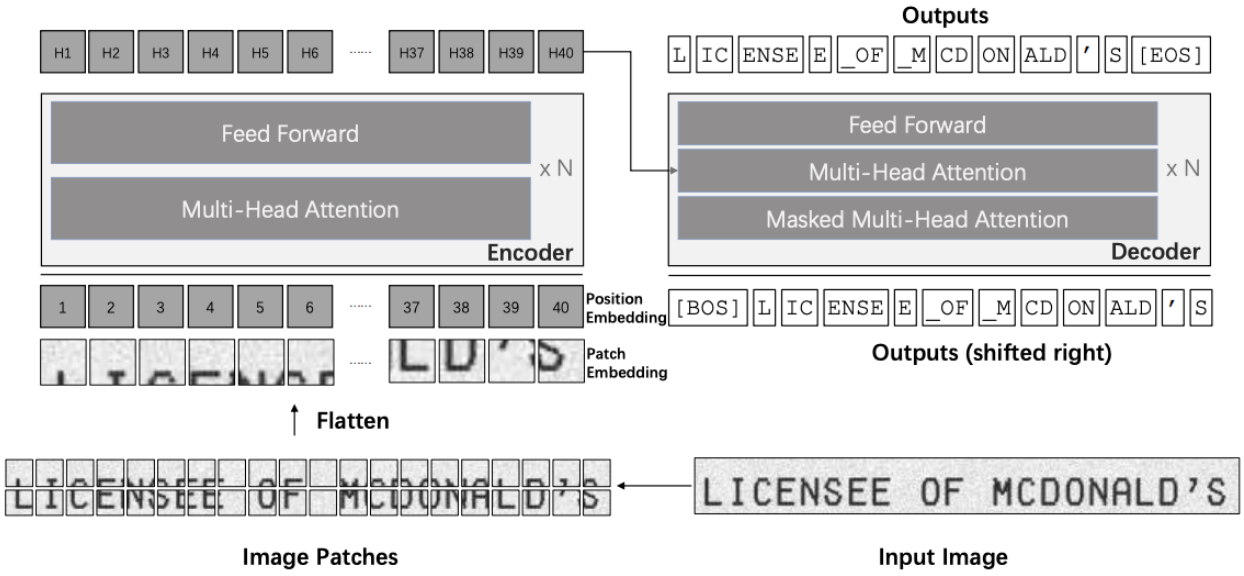


Figure 1: The architecture of the TrOCR model, as shown in the original paper [17].

In summary, the encoder computes representations of the image patches, and the decoder generates word piece sequences based on the visual features and on previous predictions.

It should be noted that, according to [17], the preprocessing of the input text line deviates slightly from the original architecture diagram shown in Figure 1. More specifically, the authors claim to have followed Dosovitskiy et al. [5] by first resizing the input image to a 384×384 square image and then splitting it into a sequence of 16×16 image patches, which are used as input to the image transformer that acts as the encoder. This can also be verified in the code. In Figure 2, we propose an alternative diagram showing the 576 (24×24) patches of size 16×16 in which the 384×384 image obtained after resizing the input image is divided.

It is also important to note that, because the demo expects single lines of text as input (where the width is typically much greater than the height), resizing and generating patches in this manner gives different emphasis to the horizontal and vertical dimensions of the input images. Specifically, each 16×16 patch contains more information about the variation of the text line in the horizontal dimension than in the vertical dimension.

2.1 Transformers

A transformer is a type of neural network architecture that is particularly effective in handling sequential data. It consists of an encoder and a decoder. The encoder extracts features from the input sequences (which can be images or text) and passes them to the decoder. The decoder then autoregressively generates the output sequence based on these features.

TrOCR follows the same transformer structure, with encoder parameters initialized using an image transformer and decoder parameters initialized using text transformers.

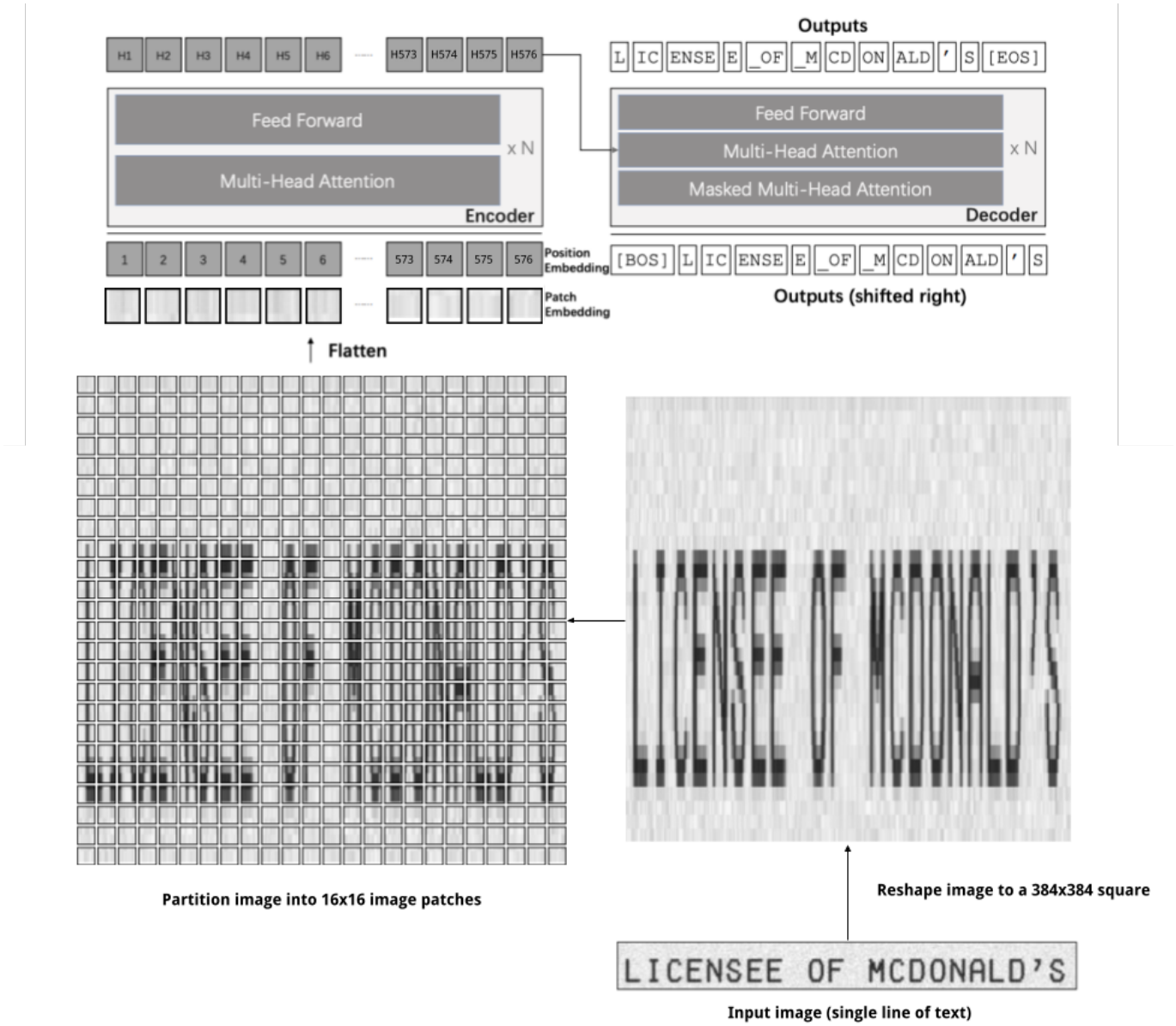


Figure 2: Proposed diagram of the architecture of the TrOCR model, as implemented in the code. In particular, the line of text is first reshaped to a 384×384 pixel image, and then partitioned into 16×16 image patches. It is worth noting that when input images are lines of text extracted from a PDF file, the reshaping step is even more aggressive on the horizontal axis than the one shown in this diagram for the 3-word expression 'LINCENSEE OF MCDONALD'S'.

2.1.1 Autoregressiveness

The model operates in two distinct modes: training and inference. For an autoregressive transformer, the input of the decoder differs between these two phases.

During the supervised training phase, the entire target sequence, including special tokens such as <bos> (beginning of sequence), <eos> (end of sequence), and <pad> (used to ensure a uniform sequence length across a batch[3]), is fed into the decoder. This full sequence is used to compute the loss against the true labels.

[3]Text transformer models are often trained on batches of data. Each batch typically consists of multiple input sequences. However, these sequences of text are often of different lengths (i.e., text outputs of different lengths). To process them in a single batch, all sequences must have the same length. Padding is used to make each sequence in a batch have the same length by adding the special padding token (<pad>) to the shorter sequences.

However, during the inference phase, the transformers generate outputs autoregressively. This means that tokens are produced one at a time, with each new token depending on the previously generated ones. At the first time step, since no output has yet been generated, the input of the decoder consists only of the <bos> token. The decoder processes this token to produce the first output token, which is then appended to the decoder input for subsequent steps. This process continues iteratively until the <eos> token is generated or a predefined maximum length is reached. An illustration of this process is shown in Figure 3.

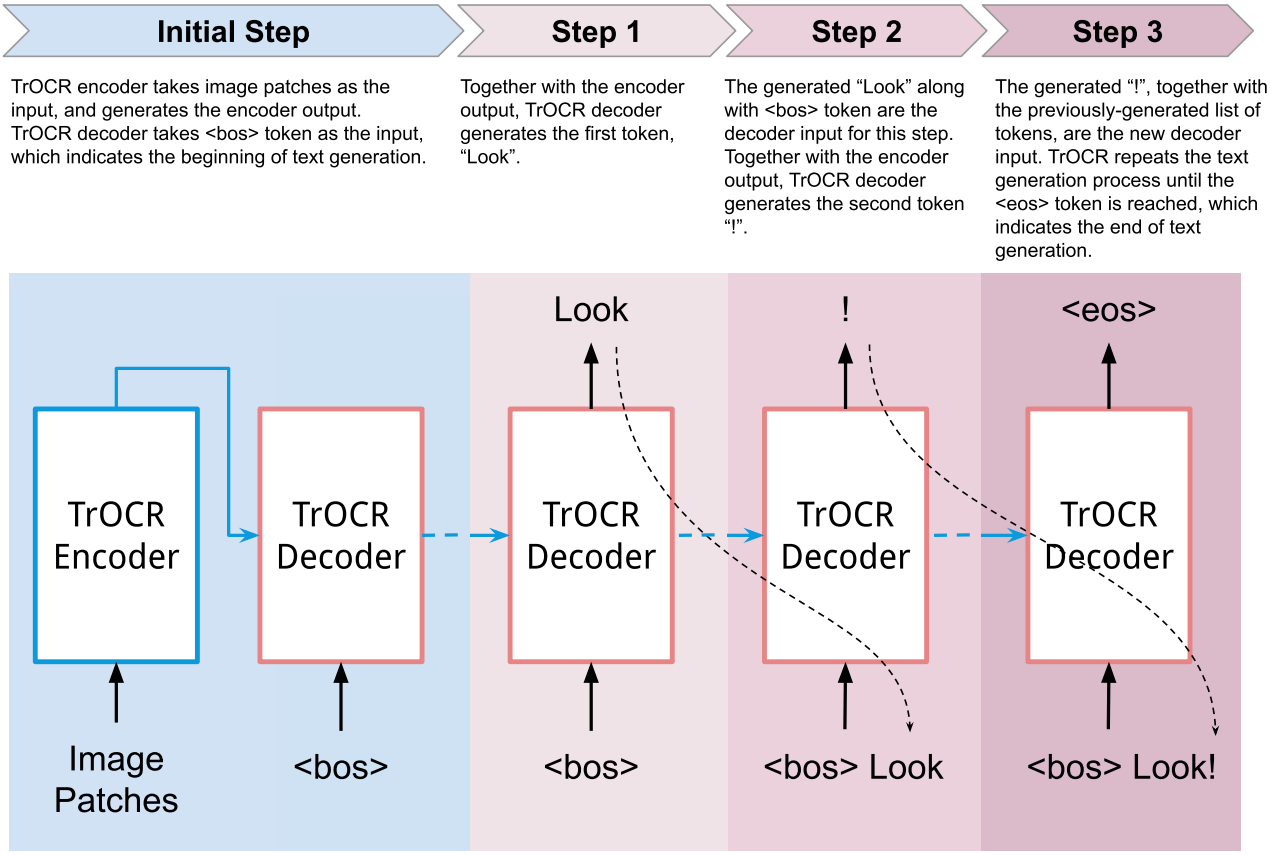


Figure 3: Transformer decoder generating the textual output autoregressively during inference, after an image containing the text "Look!" is provided to the model.

2.1.2 Transformer Encoder

The encoder consists of several key components, including positional encoding, multi-head attention, and feed-forward layers.

Positional Encoding. Since transformers process entire sequences in parallel, rather than sequentially like Recurrent Neural Networks (RNNs), positional information can be lost, making it difficult for the model to recognize the order of tokens in the input. To address this issue, positional encoding is added to the input embedding, enabling the model to identify the position of each token within the sequence.

Multi-Head Attention. The multi-head attention mechanism in the encoder combines multiple self-attention "heads" that run in parallel, allowing the model to focus on different aspects of the input simultaneously. The self-attention mechanism is one of the most distinctive features of transformers.

It calculates a weight matrix for each token in the input sequence by computing the dot products between tokens. This matrix enables the transformer to determine the importance of different tokens all at once, identifying which parts of the input are most relevant for predicting the next token, rather than processing data sequentially like RNNs or LSTMs.

Feed-Forward Layers. After applying multi-head attention, the transformer employs a feed-forward neural network to capture non-linear transformations in the data.

2.1.3 Transformer Decoder

The decoder takes the input sequence, adds positional encoding, and processes it through a masked multi-head attention mechanism. The output of this masked multi-head attention, together with the encoder's output, is then passed through a multi-head attention layer followed by a feed-forward neural network. This sequence of operations produces the final decoder output. Two key attention mechanisms in the decoder, which differentiate it from the encoder, are the masked multi-head attention and the cross attention. These mechanisms enable the decoder to generate output sequences efficiently by using both the previously generated tokens and the encoder's output.

Masked Multi-Head Attention. The masked multi-head attention is similar to the multi-head attention in the encoder, performing self-attention on the decoder input. However, it ensures that the model does not "cheat" by looking ahead at future tokens that have not yet been generated, using a masking mechanism. This is particularly important during training, where the entire sequence is provided as input to the decoder. The model achieves this by setting the attention weights for future positions to zero, effectively masking them.

Cross Attention. Unlike the multi-head attention in the encoder, which uses self-attention, the decoder employs cross-attention. In this mechanism, the decoder pays attention to the output of the encoder. Instead of calculating the weight matrix from the dot products of tokens within the same sequence, cross-attention calculates it by taking the dot products between the decoder input and the encoder outputs. This process is crucial because it enables the decoder to generate contextually relevant output based on the input sequence.

2.1.4 Advantages and Limitations

Transformers offer several advantages, including parallel processing, the ability to capture long-range dependencies, and SOTA performance.

- **Parallel Processing**: The transformer architecture uses the self-attention mechanism, which enables it to capture relationships between all elements in a sequence simultaneously, rather than processing them one by one. This parallel processing capability contrasts with RNNs, which rely on sequential processing. As a result, transformers can process sequences much more efficiently, leading to significantly faster computation.
- **Long-Range Dependencies**: In transformers, self-attention enables each element in a sequence (such as a word in a sentence) to pay attention to all other elements in that same sequence. This mechanism is particularly powerful because it goes beyond relying on immediate neighbors (like adjacent words) to determine meaning. It captures long-range dependencies and connections between distant elements by computing attention scores, which guide the model on how much focus to allocate to different elements.

- **SOTA Performance**: Transformers have achieved SOTA performance across a wide range of tasks, including translation (e.g., BERT [3], RoBERTa ([18]), question answering, and text generation, as seen in models like GPT.

However, transformers also have weaknesses. First, they require a fixed input length, which means that they cannot natively handle variable input sizes. As a result, for models like TrOCR, the input must be resized and divided into patches to ensure a fixed input length. Second, transformers require large amounts of data to generalize effectively. To address this, TrOCR leverages transfer learning, initializing the model with weights from well-trained models to mitigate the need for extensive data. Finally, transformers are sensitive to spatial distortions: the default positional encodings used in transformers struggle to capture relative spatial relationships, particularly when these relationships are irregular or distorted. This weakness will be further explored in this study.

3 Training Pipeline

As mentioned in the previous section, the TrOCR architecture is composed of one pre-trained vision transformer (the encoder) and one pre-trained text transformer (the decoder). The TrOCR authors further pre-train models on synthetically generated data, and fine-tune them on human-labeled databases. In this section, we explore the initialization, pre-training and fine-tuning pipeline. In order to better visualize the data used in the training process, examples of training data are included in each subsection describing a training step. The overview of the training workflow, described in more detail in Figure 4, is as follows:

1. **Model initialization**: both the encoder (vision transformer) and the decoder (text transformer) models are initialized using weights available online.

2. **Stage 1 Training (First stage pre-training)**: the model is trained for the task of transcribing single lines of text using images of lines of text extracted from PDF files available online (together with their transcriptions).

3. **Stage 2 Training (Second stage pre-training)**: the model is trained using lines of synthetically generated handwritten text.

4. **Stage 3 Training (Fine-tuning)**: the model is trained with handwritten lines of text.

3.1 Model Lineage

As shown in the left column of Figure 4, three different sizes of TrOCR models are produced in Step 1 (Model initialization), by combining different Computer Vision (CV) and Natural Language Processing (NLP) pre-trained models. $\text{TrOCR}_{\text{SMALL}}$ contains 62 million parameters, $\text{TrOCR}_{\text{BASE}}$ includes 334 million, and $\text{TrOCR}_{\text{LARGE}}$ comprises 558 million parameters.

In all cases, the TrOCR models [17] undergo three distinct training stages. In Stage 1 (referred to as "first stage pre-training" in [17]), each model is trained on 684 million lines of text in English extracted from PDF files available on the Internet. This initial phase provides models with a broad understanding of the task of transcribing single lines of text. The three models generated at this stage are called `trocr-small-stage1`, `trocr-base-stage1`, and `trocr-large-stage1`, as shown both in the middle column of Figure 4 and in the line corresponding to Stage 1 in Table 1 (the word "Any" under "Text type" indicates that, up to that step, models have not been trained to transcribe a specific type of digital-born text). It is expected that these models perform better on machine-printed text, as this constitutes the entirety of the Stage 1 training data. Models are intended to be

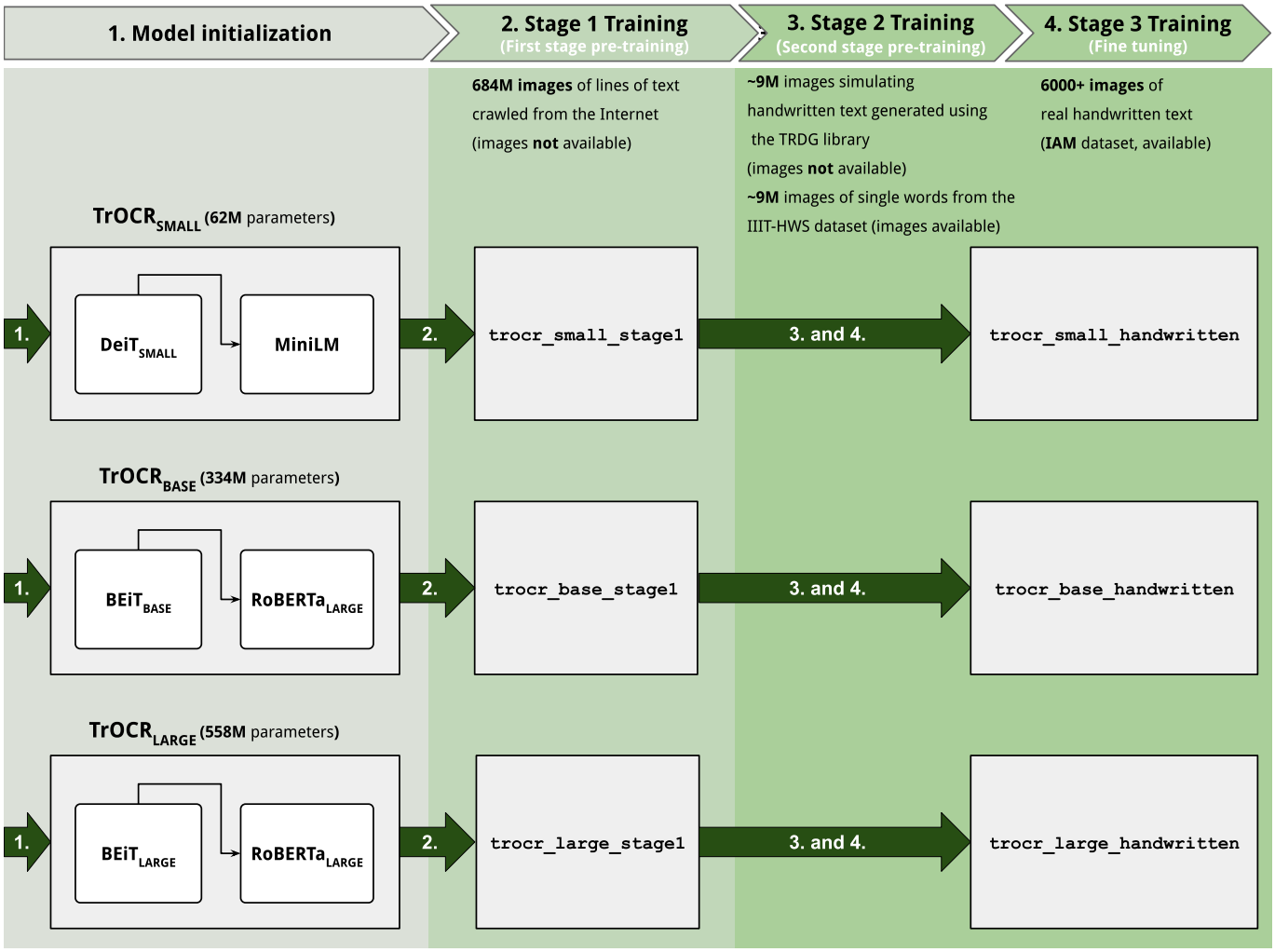


Figure 4: Training pipeline of TrOCR models for handwritten text. Note that no models are shown under Stage 2 Training since they were not published by the authors of [17].

further trained into models capable of processing one specific desired text type (i.e., handwritten, printed, scene text), as shown in Table 1.

Following this, each of the models goes through another training stage that we call Stage 2 (referred to as "second stage pre-training" in [17]), which involves training them to recognize text of a specific type among handwritten, printed, and scene text. For this purpose, close to 18 million images of lines of text are synthetically generated to train the handwritten models, 3.3 million images are produced to train the printed text models, and 16 million images are generated to train the scene text models. In other words, new TrOCR models are generated for each text category, tailored to their specific type of text. The nine models produced after this stage of training were not made available by the authors of TrOCR [17] and therefore are listed as N/A in Table 1 and not included in Figure 4.

Finally, in Stage 3 (referred to as "fine-tuning" in the original paper), each specialized model undergoes further training, but with real data: the handwritten text models are fine-tuned using the IAM dataset [20], the printed-text models are trained with the SROIE dataset [11], and the scene text models are fine-tuned on eight distinct word-based datasets: IIIT5K-3000 [22], SVT-647 [33], IC13-857 [14], IC13-1015 [14], IC15-1811 [13], IC15-2077 [13], SVTP-645 [25], and CT80-288 [28].

This would result in nine new models (i.e., three fine-tuned models for each specific text recognition category). However, only eight of these models were made publicly available (the `trocr-small-str` version for scene text was neither released nor evaluated in [17]).

Table 1 presents all available TrOCR models along with their corresponding identifiers for loading pre-trained weights through the Hugging Face library [35].

		Models		
Stage	**Text type**	**Small**	**Base**	**Large**
1	Any	`trocr-small-stage1`	`trocr-base-stage1`	`trocr-large-stage1`
2	Handwritten	N/A	N/A	N/A
	Printed	N/A	N/A	N/A
	Scene Text	N/A	N/A	N/A
3	Handwritten	`trocr-small-handwritten`	`trocr-base-handwritten`	`trocr-large-handwritten`
	Printed	`trocr-small-printed`	`trocr-base-printed`	`trocr-large-printed`
	Scene Text	N/A	`trocr-base-str`	`trocr-large-str`

Table 1: The identifiers for the TrOCR models corresponding to each text recognition category and the training stages are listed following the names used in the Hugging Face library [35]. Notably, the weights for the `trocr-small-str` model designed for scene text have not been published, which is listed as N/A.

This work focuses on models in the 'Stage 3 - Handwritten' line in Table 1, and its corresponding demo is implemented for the model `trocr-base-handwritten`[4].

3.2 Model Initialization

Both the encoder and the decoder models are initialized using the publicly available models pre-trained on large-scale labeled and unlabeled datasets.

TrOCR leverages image transformers, such as BEiT [1] and DeiT [30], along with text-transformers such as BERT [3] and RoBERTa [18] to initialize the decoder. However, since both models have an encoder-based structure, they lack the cross-attention mechanism found in transformer decoders. As a result, the parameters for cross-attention are randomly initialized, as these are not present in the original transformer models.

An ablation study on the SROIE [11] dataset is described in [17]. In this study, different combinations of pre-trained vision and text transformer models are evaluated. For the encoder, they use $\text{DeiT}_{\text{BASE}}$, $\text{BEiT}_{\text{BASE}}$ and ResNet-50 [9]. For the decoder, they use $\text{RoBERTa}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{LARGE}}$ models. The conclusion is that initializing the encoder with $\text{BEiT}_{\text{BASE}}$ and the decoder with $\text{RoBERTa}_{\text{LARGE}}$ leads to higher precision, recall and F1 results. The demo published together with this work is initialized using $\text{BEiT}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{LARGE}}$.

3.3 Stage 1 Training

As mentioned in Section 1, the training performed in Stage 1 (referred to as "first-stage pre-training" in the original paper) is common to all models. We understand that the authors generated images of lines of text from PDF files, and extracted the ground truth text by reading each line from the same PDF files. However, no access to the dataset used in this stage is provided in [17] and, to our knowledge, it is not publicly accessible.

According to [17], these images were obtained by sampling two million document pages from PDF files of born-digital documents publicly available on the Internet, and generating 684M text line images together with their corresponding text transcription.

[4]The eleven models shown in this Table 1 are available here: https://huggingface.co/models?other=trocr&sort=trending&search=microsoft

3.4 Stage 2 Training

Stage 2 Training (referred to as "second-stage pre-training" in the original paper) uses data augmentation exclusively. On the one hand, the authors use an open-source text recognition data generator called TRDG[5], which takes lines of text (randomly crawled from pages of Wikipedia) as input and generates text line images using more than 5,000 different fonts that simulate handwritten styles with varying degrees of realism. On the other hand, the authors include the IIIT-HWS dataset [15], which also consists of text lines generated by rendering 100 images for each of roughly 90k words taken from a dictionary (in different font types –chosen among 750 synthetic fonts in total– that, ideally, simulate handwritten fonts), and varying parameters such as the kernel level (inter-character spacing), stroke width (from a defined distribution) and applying Gaussian filtering to smooth the rendered image. In order to simulate other characteristics of handwritten text, they also apply small rotations and shear transformations, and apply translations by varying the padding in order to simulate incorrect segmentation of words. Figure 5 (taken from [15]) shows some good results of these artificially generated words (i.e., results that resemble text written by a human). However, a careful examination indicates that most examples are print-script rather than cursive handwriting. The letters are clearly generated individually; some of them are displaced so that they touch each other, but the transition is generally not smooth, as it should be for cursive handwriting. Furthermore, the repeated instances of one same letter present in a word are identical (e.g., the two "t" in "state" in Figure 5).

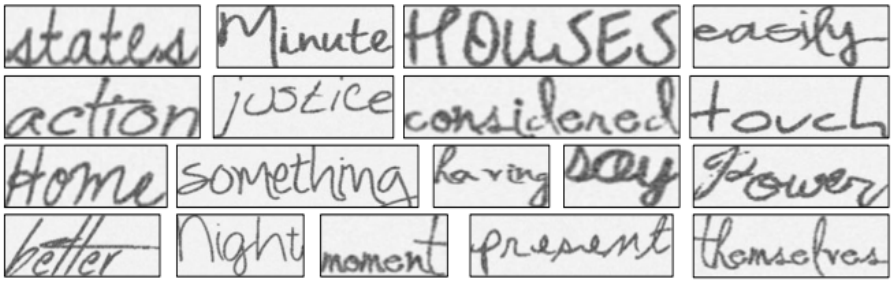


Figure 5: Sample synthetic handwritten word images, as shown in [15].

In total, these two sub-datasets together are said to contain 17.9M lines of text. To our knowledge, the training data generated with the TRDG library is not publicly available (only links to fonts [6,7] are provided).

By downloading the IIIT-HWS dataset, one can verify that:

1. The dataset consists of roughly 9M images (which means that the number of text lines generated using the TRDG library is also around 9M);

2. Each image in the dataset contains one single word;

3. All images are 48×125 pixels in size, in grayscale;

4. The roughly 9M images are organized in folders, each of which contains 100 different images of the same word generated using 100 different fonts;

[5] https://github.com/Belval/TextRecognitionDataGenerator
[6] https://fonts.google.com/?category=Handwriting
[7] https://www.1001fonts.com/handwritten-fonts.html

5. Some of these open-source fonts resemble 'handwritten fonts', like the sample images displayed in [15], included in Figure 5.

6. Some other fonts resemble standard computer fonts, and not 'handwritten fonts', as can be seen in Figure 6.

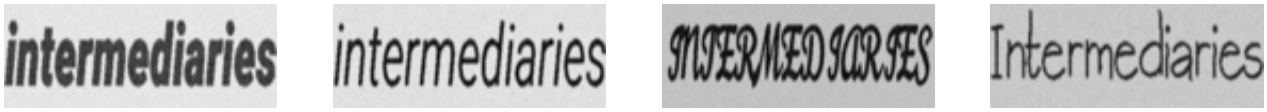


Figure 6: Sample images in the IIIT-HWS dataset created with fonts that do not look like a 'handwritten font'.

3.5 Stage 3 Training (fine-tuning)

Finally, the IAM Handwriting Database [20] was used to fine-tune the model for the downstream handwritten text recognition task, using the cross-entropy loss. This benchmark dataset, composed of handwritten text in English, is the most widely used for handwritten text recognition.

Fine-tuning was done using the Aachen's partition of the aforementioned dataset. First published in ICDAR 1999, this database contains forms of unconstrained handwritten text, scanned at a resolution of 300 dpi and saved as PNG images with 256 grayscale levels. One sample image of such forms can be seen in Figure 7. The Aachen partition used for fine-tuning consists of 6161 lines of text extracted from 747 forms in the training set, 966 lines from 115 forms in the validation set and 2915 lines from 336 forms in the test set. The total number of distinct writers is 657.

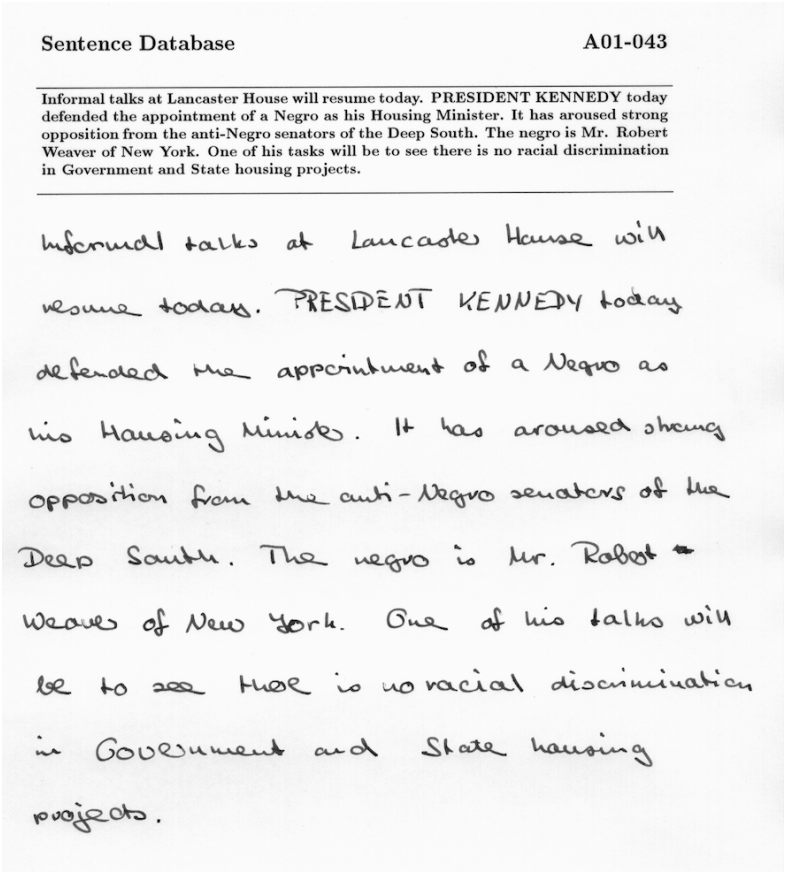


Figure 7: Sample of a complete form from the IAM Handwriting Database.

3.6 Evaluation Metric

The metric used for evaluating the performance of text recognition of handwritten text is the Character Error Rate (CER), which measures the minimum number of character edits relative to the number of ground-truth characters and can be computed as

$$CER = \frac{S + D + I}{N}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the number of characters in the ground truth (including spaces).

3.7 Reported Results on the IAM Dataset

In Table 2 we include the results of TrOCR on the IAM handwritten dataset reported in [17], compared to other state-of-the-art models. The only model reporting a lower CER is one of the three models reported in [4] but, unlike TrOCR, it uses an external language model.

Model	Architecture	External LM	CER
(Bluche and Messina 2017 [2])	GCRNN/CTC	Yes	3.20
(Michael et al. 2019 [21])	LSTM/LSTM w/Attn	No	4.87
(Wang et al. 2020a [34])	FCN/GRU	No	6.40
(Kang et al. 2020 [12])	Transformer w/CNN	No	4.67
(Diaz et al. 2021 [4])	S-Attn/CTC	No	3.53
(Diaz et al. 2021 [4])	S-Attn/CTC	Yes	2.75
(Diaz et al. 2021 [4])	Transformer w/CNN	No	2.96
$\text{TrOCR}_{\text{SMALL}}$	Transformer	No	4.22
$\textbf{TrOCR}_{\textbf{BASE}}$	Transformer	No	**3.42**
$\text{TrOCR}_{\text{LARGE}}$	Transformer	No	2.89

Table 2: Evaluation results (CER) on the IAM Handwriting dataset (as reported on [17]).

4 Experiments

The experiments discussed in this section intend to serve as an initial exploratory walk-through of how one specific TrOCR model trained for handwritten text recognition works, with the aim of saving time for users approaching TrOCR for the first time.

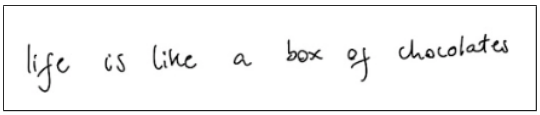
As mentioned in Section 1, the demo that accompanies this paper receives an image containing a single line of text and returns (ideally) the text's transcription. The quality of this transcription has its limitations, as can be seen in the following sections.

4.1 Handwritten Text

Among all the possible forms of handwriting text, there is usually a distinction between print and cursive handwriting. The latter usually involves continuous strokes, which makes delimiting each single character more difficult. In [17], no specific mention is made of whether there is a difference in the quality of the transcription for cursive or print handwritten text. In the following subsections, we explore how the model behaves when exposed to different forms of handwritten text (i.e., print, upper case print, cursive).

4.1.1 Print handwritten text

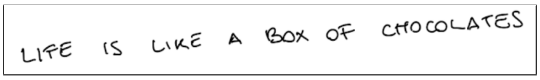
Figures 8 and 9 show the behavior of the TrOCR model when transcribing print handwritten text. In both cases, the result is precise. However, note that no upper-case letters are used in the transcription of the phrase in Figure 9 despite it being fully written in upper-case.



TrOCR output: `life is like a box of chocolates.`

Ground truth: `life is like a box of chocolates`

Figure 8: Example of lower case print handwritten text. Note that the algorithm added a full stop at the end of the transcription (which is not present in the image).



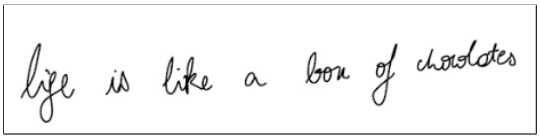
TrOCR output: `life is like a box of chocolates.`

Ground truth: `LIFE IS LIKE A BOX OF CHOCOLATES`

Figure 9: Example of mostly upper case print handwritten text. Note that the transcription is done using lower case only.

4.1.2 Cursive handwritten text

As mentioned at the beginning of this section, transcribing cursive handwritten text is one of the most challenging of all OCR tasks. In this case, the same phrase introduced in the previous subsection is evaluated again, but now in cursive handwritten text, as Figure 10 shows.



TrOCR output: `lige is like a boon of chondates.`

Ground truth: `life is like a box of chocolates`

Figure 10: Example of cursive (a.k.a. continuous) handwritten text.

One could argue that this specific continuous handwriting style is not clear or readable enough, but that is usually the case in handwritten text, especially in cursive handwriting.

4.1.3 Text written in languages other than English

As mentioned in Section 3.5, the model used in this demo was fine-tuned on IAM, which contains handwritten text in English. However, some texts in other languages can be transcribed with high precision, as Figure 11 shows. This is most likely due to the fact that Spanish and English share not only the same script but also some word roots (e.g. the word 'declarant' does exist in English), and also because the model does not have a language post-processing step.

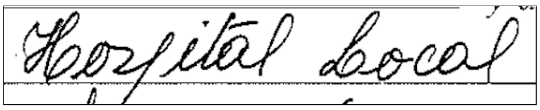
TrOCR output: `la declarante`

Ground truth: `la declarante`

Figure 11: Example of cursive handwritten text written in Spanish extracted from a Uruguayan birth certificate.

4.1.4 Background color

Another aspect worth examining is how well text is transcribed when it is not written on a white background. To explore that, we propose an example of text written on a white background in which the text is correctly transcribed, shown in Figure 12, and then compare the result of transcribing the same image after inverting the colors (note that, in both cases, images are binary).



TrOCR output: `Hospital Local`

Ground truth: `Hospital Local`

Figure 12: Example of cursive handwritten text extracted from a birth certificate scan.

TrOCR output: `Hospital health`

Ground truth: `Hospital Local`

Figure 13: Same image from Figure 12 after inverting colors.

This is most likely because training was done on images containing black text on white paper (at least, this is the case in all datasets used for training that were publicly available and described in Section 3).

4.2 Scene Text

Scene text recognition is one of the main tasks in OCR, and it is the area in which the first step of traditional OCR pipelines, text detection, is more challenging. Since the method described in this article is capable of transcribing only single lines of text, and a single line of text extracted from a scene resembles a line of either handwritten or printed text, one could argue that analyzing this OCR task may not seem necessary. However, we will explore what happens with a single image shown in Figure 14 when cropped using Python[8], or using other segmentation tools, such as the method introduced in [19] and available online in the IPOL demo titled 'Unified scene-text detection and layout analysis'[9].

[8] All images in this paper were cropped by removing pixels directly in Python, without any resizing. This preserves the original resolution of the retained region and ensures that only one variable (framing) is altered.

[9] https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000421

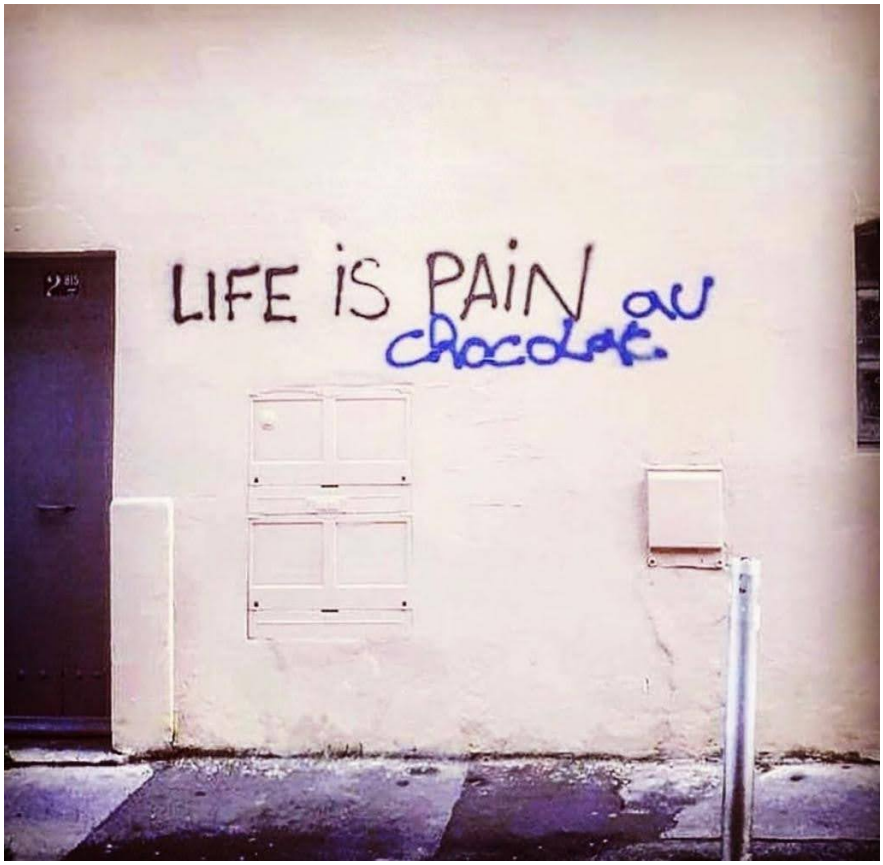


Figure 14: Scene text image with both upper-case and lower-case print handwritten text. TrOCR outputs "5 References" which is clearly not an attempt to transcribe the text present in the image but has more to do with the fact that the model was trained to transcribe single lines of text as input.

One could, for instance, want to see how well the model recognizes the phrase 'LIFE iS PAiN' by cropping the image, as shown in Figure 15. The model outputs the correct transcription (although in lower-case, as observed in a previous example).

TrOCR output: `life is pain.`

Ground truth: `LIFE iS PAiN`

Figure 15: Cropped from Figure 14. Size: 515×112 pixels.

Figure 16 shows how a different crop can lead to incorrect text recognition. The crop shown in Figure 17 is almost exactly the same as the one shown in Figure 16. In fact, the only difference is that the second one contains five fewer rows of pixels than the first one. This shows how even a minor difference in the crop, barely perceptible by the human eye, can lead to incorrect text recognition.

Figure 18, which was cropped to remove a few more pixel rows at the bottom of the image shown in Figure 17 while still leaving the text readable to the human eye, provides yet another (quite controversial!) transcription of the same text.

An instructive comparison between Figures 16, 17 and 18, visualized through their difference map shown in Figure 19, underscores how subtle variations in the cropping process can alter TrOCR's behavior. The three cropped figures differ only in height when the overlaid words are aligned.

TrOCR output: `life is paid.`

Ground truth: `LIFE iS PAiN`

Figure 16: Cropped from Figure 14. Size: 515×125 pixels.

TrOCR output: `life is rain.`

Ground truth: `LIFE iS PAiN`

Figure 17: Cropped from Figure 16. Size: 515×120 pixels.

TrOCR output: `wife is pain.`

Ground truth: `LIFE iS PAiN`

Figure 18: Cropped from Figure 17. Size: 515×105 pixels.

Figure 19: Difference map of Figures 16, 17, and 18 (compared when aligned at the top). Yellow indicates pixels where all three images match. Blue indicates pixels where only Figures 16 and 17 match (they were removed from Figure 18). Red indicates pixels present only in Figure 16. Colors are intended to help visualize how the content is progressively cropped across the sequence of images, and demonstrate that even small visual differences between the images lead to different TrOCR outputs.

Although this additional height does not modify the textual content, it does lead to a noticeable change in TrOCR's output.

Delegating the cropping step to an algorithm that detects text lines may seem a more objective and possibly trustworthy way of cropping images. However, Figure 20 shows that problems can still arise. One could argue that Figure 20 also contains the word 'au' in the crop (note that this is because the cited demo detects entire lines and paragraphs, and provides them as outputs). However, it should be noted that cropping the 'au' part from Figure 20, as Figure 21 shows, does not shorten the transcription provided by TrOCR for the image shown in Figure 20: it completely changes the transcription, even if the only difference between the two images is the presence or absence of the rightmost part of the image containing the 'au' text.

TrOCR output: `life is_rainlaw`

Ground truth: `LIFE iS PAiN au`

Figure 20: Cropped from Figure 14, using the algorithm presented in [19].

TrOCR output: `life is pain.`

Ground truth: `LIFE iS PAiN`

Figure 21: Image obtained by cropping the "au" part from Figure 20.

5 Conclusions

The TrOCR model appears capable of correctly transcribing English text written by hand in a printed style on light-colored paper. However, it is notably less effective when transcribing handwritten cursive text, or when transcribing text written in languages other than English or on dark backgrounds.

Fine-tuning a model on a dataset that is more suited to the specific task is a logical approach, especially when attempting to transcribe handwritten text in datasets that differ from those used during the initial training.

Lastly, the model shows to be quite sensitive to the way in which text is cropped from documents or images from scenes, resulting in a very unstable behavior.

Acknowledgments

The research underlying the results presented in this publication was partly supported by the Agencia Nacional de Investigación e Innovación (ANII) and the France 2030 CollabNext project.

Sincere thanks are extended to Rafael Grompone, Javier Preciozzi, and Vincent Christelin for their invaluable insight and support throughout the process of writing this article, to Jérémy Anger for his generous support while creating the online demo accompanying this article, and to Roy He and Seginus Mowlavi for their valuable comments which helped improve the final version.

Image Credits

Images without a specific authorship credit were created by the authors of this paper.

 Image extracted from the IAM Handwriting Database [20].

 Anonymous photo of Paris street art, sourced from online archives *circa* November 2018.

References

[1] H. Bao, L. Dong, S. Piao, and F. Wei, *BEiT: BERT Pre-Training of Image Transformers*, in International Conference on Learning Representations, 2022. https://openreview.net/forum?id=p-BhZSz59o4.

[2] T. Bluche and R. Messina, *Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition*, in IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, 2017, pp. 646–651, https://doi.org/10.1109/ICDAR.2017.111.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding*, in North American Chapter of the Association for Computational Linguistics, 2019, https://doi.org/10.18653/v1/N19-1423.

[4] D. H. Diaz, S. Qin, R. Ingle, Y. Fujii, and A. Bissacco, *Rethinking Text Line Recognition Models*. arXiv preprint 2104.07787, 2021, https://doi.org/10.48550/arXiv.2104.07787.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv preprint 2010.11929, 2021, https://doi.org/10.48550/arXiv.2010.11929.

[6] H. Fujisawa, *Forty Years of Research in Character and Document Recognition - An Industrial Perspective*, Pattern Recognition, 41 (2008), pp. 2435–2446, https://doi.org/10.1016/j.patcog.2008.03.015.

[7] A. Graves, *Connectionist Temporal Classification*, in Supervised Sequence Labelling with Recurrent Neural Networks, 2012, pp. 61–93, https://doi.org/10.1007/978-3-642-24797-2_7.

[8] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, *A Novel Connectionist System for Unconstrained Handwriting Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 31 (2008), pp. 855–868, https://doi.org/10.1109/tpami.2008.137.

[9] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.

[10] W. Hu, X. Cai, J. Hou, S. Yi, and Z. Lin, *GTC: Guided Training of CTC Towards Efficient and Accurate Scene Text Recognition*, in AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 11005–11012, https://doi.org/10.1609/aaai.v34i07.6735.

[11] Z. Huang, K. Chen, J. He, X. Bai, D. Karatzas, S. Lu, and C. V. Jawahar, *ICDAR 2019 Competition on Scanned Receipt OCR and Information Extraction*, in IEEE International Conference on Document Analysis and Recognition (ICDAR), 2019, pp. 1516–1520, https://doi.org/10.1109/icdar.2019.00244.

[12] L. Kang, P. Riba, M. Rusiñol, A. Fornés, and M. Villegas, *Pay Attention to What You Read: Non-Recurrent Handwritten Text-Line Recognition*, Pattern Recognition, 129 (2022), p. 108766, https://doi.org/10.1016/j.patcog.2022.108766.

[13] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny, *ICDAR 2015 Competition on Robust Reading*, in International Conference on Document Analysis and Recognition (ICDAR), 2015, pp. 1156–1160, https://doi.org/10.1109/ICDAR.2015.7333942.

[14] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazàn, and L. P. de las Heras, *ICDAR 2013 Robust Reading Competition*, in International Conference on Document Analysis and Recognition (ICDAR), 2013, pp. 1484–1493, https://doi.org/10.1109/ICDAR.2013.221.

[15] P. Krishnan and C. V. Jawahar, *Generating Synthetic Data for Text Recognition*. arXiv preprint 1608.04224, 2016, https://doi.org/10.48550/arXiv.1608.04224.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-Based Learning Applied to Document Recognition*, Proceedings of the IEEE, 86 (2002), pp. 2278–2324, https://doi.org/10.1109/5.726791.

[17] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, *TrOCR: Transformer-Based Optical Character Recognition with Pre-Trained Models*, in AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 13094–13102, https://doi.org/10.1609/aaai.v37i11.26538.

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv preprint 1907.11692, 2019, https://doi.org/10.48550/arXiv.1907.11692.

[19] S. Long, S. Qin, D. Panteleev, A. Bissacco, Y. Fujii, and M. Raptis, *Towards End-To-End Unified Scene Text Detection and Layout Analysis*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, https://doi.org/10.1109/CVPR52688.2022.00112.

[20] U. Marti and H. Bunke, *The IAM-Database: An English Sentence Database for Off-Line Handwriting Recognition*, International Journal on Document Analysis and Recognition (IJDAR), 5 (2002), pp. 39–46, https://doi.org/10.1007/s100320200071.

[21] J. Michael, R. Labahn, T. Grüning, and J. Zöllner, *Evaluating Sequence-To-Sequence Models for Handwritten Text Recognition*, in IEEE International Conference on Document Analysis and Recognition, 2019, https://doi.org/10.1109/ICDAR.2019.00208.

[22] A. Mishra, K. Alahari, and C. V. Jawahar, *Scene Text Recognition Using Higher Order Language Priors*, in British Machine Vision Conference (BMVC), 2012, https://doi.org/10.5244/C.26.127.

[23] S. Mori, C. Y. Suen, and K. Yamamoto, *Historical Review of OCR Research and Development*, Proceedings of the IEEE, 80 (1992), pp. 1029–1058, https://doi.org/10.1109/5.156468.

[24] A. Pandey, S. Sawant, and E. M. Schwartz, *Handwritten Character Recognition Using Template Matching*, in Florida Conference on Recent Advances in Robotics (FCRAR), 2010, pp. 20–21.

[25] T. Q. Phan, P. Shivakumara, S. Tian, and C. L. Tan, *Recognizing Text with Perspective Distortion in Natural Scenes*, in IEEE International Conference on Computer Vision (ICCV), 2013, pp. 569–576, https://doi.org/10.1109/ICCV.2013.76.

[26] R. Plamondon and S. N. Srihari, *On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2002), pp. 63–84, https://doi.org/10.1109/34.824821.

[27] J. Puigcerver, *Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?*, in IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, 2017, pp. 67–72, https://doi.org/10.1109/ICDAR.2017.20.

[28] A. Risnumawan, P. Shivakumara, C. S. Chan, and C. L. Tan, *A Robust Arbitrary Text Detection System for Natural Scene Images*, Expert Systems with Applications, 41 (2014), pp. 8027–8048, https://doi.org/10.1016/j.eswa.2014.07.008.

[29] B. Shi, X. Bai, and C. Yao, *An End-To-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (2016), pp. 2298–2304, https://doi.org/10.1109/TPAMI.2016.2646371.

[30] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, *Training Data-Efficient Image Transformers & Distillation Through Attention*, in PMLR International Conference on Machine Learning, 2021, pp. 10347–10357. http://proceedings.mlr.press/v139/touvron21a/touvron21a.pdf.

[31] Ø. D. Trier, A. K. Jain, and T. Taxt, *Feature Extraction Methods for Character Recognition - A Survey*, Pattern Recognition, 29 (1996), pp. 641–662, https://doi.org/10.1016/0031-3203(95)00118-2.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention Is All You Need*, Advances in Neural Information Processing Systems, 30 (2017). https://dl.acm.org/doi/10.5555/3295222.3295349.

[33] K. Wang, B. Babenko, and S. Belongie, *End-To-End Scene Text Recognition*, in International Conference on Computer Vision (ICCV), 2011, pp. 1457–1464, https://doi.org/10.1109/ICCV.2011.6126402.

[34] T. Wang, Y. Zhu, L. Jin, C. Luo, X. Chen, Y. Wu, Q. Wang, and M. Cai, *Decoupled Attention Network for Text Recognition*, AAAI Conference on Artificial Intelligence, 34 (2020), pp. 12216–12224, https://doi.org/10.1609/aaai.v34i07.6903.

[35] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, *Transformers: State-Of-The-Art Natural Language Processing*, in Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, 2020, pp. 38–45, https://doi.org/10.18653/v1/2020.emnlp-demos.6.

[36] M. Zimmermann and H. Bunke, *Hidden Markov Model Length Optimization for Handwriting Recognition Systems*, in International Workshop on Frontiers in Handwriting Recognition, 2002, pp. 369–374, https://doi.org/10.1109/IWFHR.2002.1030938.