TrOCR: Transformer-Based Optical Character Recognition with Pre-trained Models

Minghao Li\textsuperscript{1}, Tengchao Lv\textsuperscript{2}, Jingye Chen\textsuperscript{2}, Lei Cui\textsuperscript{2}, Yijuan Lu\textsuperscript{2}, Dinei Florencio\textsuperscript{2}, Cha Zhang\textsuperscript{2}, Zhoujun Li\textsuperscript{1}, Furu Wei\textsuperscript{2}

\textsuperscript{1}Beihang University
\textsuperscript{2}Microsoft Corporation
\{liminghao1630, liz\}@buaa.edu.cn
\{tengchaolv, v-jingyechen, lecu, yijlu, dinei, chazhang, fuwei\}@microsoft.com

Abstract

Text recognition is a long-standing research problem for document digitalization. Existing approaches are usually built based on CNN for image understanding and RNN for character-level text generation. In addition, another language model is usually needed to improve the overall accuracy as a post-processing step. In this paper, we propose an end-to-end text recognition approach with pre-trained image Transformer and text Transformer models, namely TrOCR, which leverages the Transformer architecture for both image understanding and wordpiece-level text generation. The TrOCR model is simple but effective, and can be pre-trained with large-scale synthetic data and fine-tuned with human-labeled datasets. Experiments show that the TrOCR model outperforms the current state-of-the-art models on the printed, handwritten and scene text recognition tasks. The TrOCR models and code are publicly available at https://aka.ms/trocr.

Introduction

Optical Character Recognition (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene photo or from subtitle text superimposed on an image. Typically, an OCR system includes two main modules: a text detection module and a text recognition module. Text detection aims to localize all text blocks within the text image, either at word-level or textline-level. The text detection task is usually considered as an object detection problem where conventional object detection models such as YOLO\textsuperscript{5} and DBNet (Liao et al. 2019) can be applied. Meanwhile, text recognition aims to understand the text image content and transcribe the visual signals into natural language tokens. The text recognition task is usually framed as an encoder-decoder problem where existing methods leveraged CNN-based encoder for image understanding and RNN-based decoder for text generation. In this paper, we focus on the text recognition task for document images and leave text detection as the future work.

Recent progress in text recognition (Diaz et al. 2021) has witnessed the significant improvements by taking advantage of the Transformer (Vaswani et al. 2017) architectures. However, existing methods are still based on CNNs as the backbone, where the self-attention is built on top of CNN backbones as encoders to understand the text image. For decoders, Connectionist Temporal Classification (CTC) (Graves et al. 2006) is usually used compounded with an external language model on the character-level to improve the overall accuracy. Despite the great success achieved by the hybrid encoder/decoder method, there is still a lot of room to improve with pre-trained CV and NLP models: 1) the network parameters in existing methods are trained from scratch with synthetic/human-labeled datasets, leaving large-scale pre-trained models unexplored. 2) as image Transformers become more and more popular (Dosovitskiy et al. 2021; Touvron et al. 2020), especially the recent self-supervised image pre-training (Bao, Dong, and Wei 2021), it is straightforward to investigate whether pre-trained image Transformers can replace CNN backbones, meanwhile exploiting the pre-trained image Transformers to work together with the pre-trained text Transformers in a single framework on the text recognition task.

To this end, we propose TrOCR, an end-to-end Transformer-based OCR model for text recognition with pre-trained CV and NLP models, which is shown in Figure 1. Distinct from the existing text recognition models, TrOCR is a simple but effective model which does not use the CNN as the backbone. Instead, following (Dosovitskiy et al. 2021), it first resizes the input text image into 384 × 384 and then the image is split into a sequence of 16 × 16 patches which are used as the input to image Transformers. Standard Transformer architecture with the self-attention mechanism is leveraged on both encoder and decoder parts, where wordpiece units are generated as the recognized text from the input image. To effectively train the TrOCR model, the encoder can be initialized with pre-trained ViT-style models (Dosovitskiy et al. 2021; Touvron et al. 2020; Bao, Dong, and Wei 2021) while the decoder can be initialized with pre-trained BERT-style models (Devlin et al. 2019; Liu et al. 2019; Dong et al. 2019; Wang et al. 2020b), respectively. Therefore, the advantage of TrOCR is three-fold. First, TrOCR uses the pre-trained image Transformer and
The encoder is designed to obtain the representation of the vanilla Transformer encoder-decoder structure in TrOCR. and a text Transformer for language modeling. We adopt the using an image Transformer for extracting the visual features TrOCR is built up with the Transformer architecture, includ- ing multilingual pre-trained models in the decoder- lingual text recognition with minimum efforts, where just Furthermore, we can easily extend the TrOCR for multi- age datasets without any complex pre/post-processing steps.

The contributions of this paper are summarized as fol- lows:

1. We propose TrOCR, an end-to-end Transformer-based OCR model for text recognition with pre-trained CV and NLP models. To the best of our knowledge, this is the first work that jointly leverages pre-trained image and text Transformers for the text recognition task in OCR.
2. TrOCR achieves state-of-the-art results with a stan- dard Transformer-based encoder-decoder model, which is convolution free and does not rely on any complex pre/post-processing steps.
3. The TrOCR models and code are publicly available at https://aka.ms/trocr.

**TrOCR**

**Model Architecture**

TrOCR is built up with the Transformer architecture, including an image Transformer for extracting the visual features and a text Transformer for language modeling. We adopt the vanilla Transformer encoder-decoder structure in TrOCR. The encoder is designed to obtain the representation of the image patches and the decoder is to generate the wordpiece sequence with the guidance of the visual features and previous predictions.

**Encoder** The encoder receives an input image $x_{\text{img}} \in \mathbb{R}^{3 \times H_0 \times W_0}$, and resizes it to a fixed size $(H, W)$. Since the Transformer encoder cannot process the raw images unless they are a sequence of input tokens, the encoder decomposes the input image into a batch of $N = HW/P^2$ foursquare patches with a fixed size of $(P, P)$, while the width $W$ and the height $H$ of the resized image are guaranteed to be divisible by the patch size $P$. Subsequently, the patches are flatted into vectors and linearly projected to $D$-dimensional vectors, aka the patch embeddings. $D$ is the hidden size of the Transformer through all of its layers.

Similar to ViT (Dosovitskiy et al. 2021) and DeiT (Touvron et al. 2020), we keep the special token "[CLS]" that is usually used for image classification tasks. The "[CLS]" token brings together the information from all the patch embeddings and represents the whole image. Meanwhile, we also keep the distillation token in the input sequence when using the DeiT pre-trained models for encoder initialization, which allows the model to learn from the teacher model. The patch embeddings and special tokens are given learnable 1D position embeddings according to their absolute positions.

Unlike the features extracted by the CNN-like network, the Transformer models have no image-specific inductive bi- ases and process the image as a sequence of patches, which makes the model easier to pay different attention to either the whole image or the independent patches.

**Decoder** We use the original Transformer decoder for TrOCR. The standard Transformer decoder also has a stack of identical layers, which have similar structures to the layers in the encoder, except that the decoder inserts the "encoder-decoder attention" between the multi-head self- attention and feed-forward network to distribute different at-
tion on the output of the encoder. In the encoder-decoder attention output module, the keys and values come from the encoder output, while the queries come from the decoder input. In addition, the decoder leverages the attention masking in the self-attention to prevent itself from getting more information during training than prediction. Based on the fact that the output of the decoder will right shift one place from the input of the decoder, the attention mask needs to ensure the output for the position $i$ can only pay attention to the previous output, the input on the positions less than $i$:

$$
    h_i = \text{Proj}(\text{Emb}(\text{Token}_i))
$$

$$
    \sigma(h_{ij}) = \frac{e^{h_{ij}}}{\sum_{k=1}^{V} e^{h_{ik}}} \text{ for } j = 1, 2, \ldots, V
$$

The hidden states from the decoder are projected by a linear layer from the model dimension to the dimension of the vocabulary size $V$, while the probabilities over the vocabulary are calculated on that by the softmax function. We use beam search to get the final output.

**Model Initialization**

Both the encoder and the decoder are initialized by the public models pre-trained on large-scale labeled and unlabeled datasets.

**Encoder Initialization** The DeiT (Touvron et al. 2020) and BEiT (Bao, Dong, and Wei 2021) models are used for the encoder initialization in the TrOCR models. DeiT trains the image Transformer with ImageNet (Deng et al. 2009) as the sole training set. The authors try different hyperparameters and data augmentation to make the model data-efficient. Moreover, they distill the knowledge of a strong image classifier to a distilled token in the initial embedding, which leads to a competitive result compared to the CNN-based models.

Referring to the Masked Language Model pre-training task, BEiT proposes the Masked Image Modeling task to pre-train the image Transformer. Each image will be converted to two views: image patches and visual tokens. They tokenize the original image into visual tokens by the latent codes of discrete VAE (Ramesh et al. 2021), randomly mask some image patches, and make the model recover the original visual tokens. The structure of BEiT is the same as the image Transformer and lacks the distilled token when compared with DeiT.

**Decoder Initialization** We use the RoBERTa (Liu et al. 2019) models and the MiniLM (Wang et al. 2020b) models to initialize the decoder. Generally, RoBERTa is a replication study of (Devlin et al. 2019) that carefully measures the impact of many key hyperparameters and training data size. Based on BERT, they remove the next sentence prediction objective and dynamically change the masking pattern of the Masked Language Model.

The MiniLM are compressed models of the large pre-trained Transformer models while retaining 99% performance. Instead of using the soft target probabilities of masked language modeling predictions or intermediate representations of the teacher models to guide the training of the student models in the previous work. The MiniLM models are trained by distilling the self-attention module of the last Transformer layer of the teacher models and introducing a teacher assistant to assist with the distillation.

When loading the above models to the decoders, the structures do not precisely match since both of them are only the encoder of the Transformer architecture. For example, the encoder-decoder attention layers are absent in these models. To address this, we initialize the decoders with the RoBERTa and MiniLM models by manually setting the corresponding parameter mapping, and the absent parameters are randomly initialized.

**Task Pipeline**

In this work, the task pipeline of the text recognition task is that given the textline images, the model extracts the visual features and predicts the wordpiece tokens relying on the image and the context generated before. The sequence of ground truth tokens is followed by an “[EOS]” token, which indicates the end of a sentence. During training, we shift the sequence backward by one place and add the “[BOS]” token to the beginning indicating the start of generation. The shifted ground truth sequence is fed into the decoder, and the output of that is supervised by the original ground truth sequence with the cross-entropy loss. For inference, the decoder starts from the “[BOS]” token to predict the output iteratively while continuously taking the newly generated output as the next input.

**Pre-training**

We use the text recognition task for the pre-training phase, since this task can make the models learn the knowledge of both the visual feature extraction and the language model. The pre-training process is divided into two stages that differ by the used dataset. In the first stage, we synthesize a large-scale dataset consisting of hundreds of millions of printed textline images and pre-train the TrOCR models on that. In the second stage, we build two relatively small datasets corresponding to printed and handwritten downstream tasks, containing millions of textline images each. We use the existing and widely adopted synthetic scene text datasets for the scene text recognition task. Subsequently, we pre-train separate models on these task-specific datasets in the second stage, all initialized by the first-stage model.

**Fine-tuning**

Except for the experiments regarding scene text recognition, the pre-trained TrOCR models are fine-tuned on the downstream text recognition tasks. The outputs of the TrOCR models are based on Byte Pair Encoding (BPE) (Sennrich, Haddow, and Birch 2015) and SentencePiece (Kudo and Richardson 2018) and do not rely on any task-related vocabularies.

**Data Augmentation**

We leverage data augmentation to enhance the variety of the pre-training and fine-tuning data. Six kinds of image trans-
forms plus keeping the original are taken for printed and handwritten datasets, which are random rotation (-10 to 10 degrees), Gaussian blurring, image dilation, image erosion, downsampling, and underlining. We randomly decide which image transformation to take with equal possibilities for each sample. For scene text datasets, RandAugment (Cubuk et al. 2020) is applied following (Atienza et al. 2016). In addition, we collect around 53K receipt images in the real world and recognize the text on them by commercial OCR engines. According to the results, we obtain by cropping the whole receipt images according to the ground truth bounding boxes.

Recognizing scene text images is more challenging than printed text images, as many images in the wild suffer from blur, occlusion, or low-resolution problems. Here we leverage some widely-used benchmarks, including IIIT5K-3000 (Mishra, Alahari, and Jawahar 2012), SVT-647 (Wang, Babenko, and Belongie 2011), IC13-857, IC13-1015 (Karatzas et al. 2013), IC15-1811, IC15-2077 (Karatzas et al. 2015), SVTP-645 (Phan et al. 2013), and CT80-288 (Rismanawan et al. 2014) to evaluate the capacity of TrOCR.

### Experiments

#### Data

**Pre-training Dataset** To build a large-scale high-quality dataset, we sample two million document pages from the publicly available PDF files on the Internet. Since the PDF files are digital-born, we can get pretty printed textline images by converting them into page images and extracting the textlines with their cropped images. In total, the first-stage pre-training dataset contains 684M textlines.

We use 5,427 handwritten fonts\(^1\) to synthesize handwritten textline images by the TRDG\(^2\), an open-source text recognition data generator. The text used for generation is crawled from random pages of Wikipedia. The handwritten dataset for the second-stage pre-training consists of 17.9M textlines, including IIIT-HWS dataset (Krishnan and Jawahar 2016). In addition, we collect around 53K receipt images in the real world and recognize the text on them by commercial OCR engines. According to the results, we crop the textlines by their coordinates and rectify them into normalized images. We also use TRDG to synthesize 1M printed textline images with two receipt fonts and the built-in printed fonts. In total, the printed dataset consists of 3.3M textlines. The second-stage pre-training data for the scene text recognition are MJSynth (MJ) (Jaderberg et al. 2014) and SynthText (ST) (Gupta, Vedaldi, and Zisserman 2016), totaling about 16M text images.

**Benchmarks** The SROIE (Scanned Receipts OCR and Information Extraction) dataset (Task 2) focuses on text recognition in receipt images. There are 626 receipt images and 361 receipt images in the training and test sets of SROIE. Since the text detection task is not included in this work, we use cropped images of the textlines for evaluation, which are obtained by cropping the whole receipt images according to the ground truth bounding boxes.

The IAM Handwriting Database is composed of handwritten English text, which is the most popular dataset for handwritten text recognition. We use the Aachen’s partition of the dataset\(^3\): 6,161 lines from 747 forms in the train set, 966 lines from 115 forms in the validation set and 2,915 lines from 336 forms in the test set.

Recognizing scene text images is more challenging than printed text images, as many images in the wild suffer from blur, occlusion, or low-resolution problems. Here we leverage some widely-used benchmarks, including IIIT5K-3000 (Mishra, Alahari, and Jawahar 2012), SVT-647 (Wang, Babenko, and Belongie 2011), IC13-857, IC13-1015 (Karatzas et al. 2013), IC15-1811, IC15-2077 (Karatzas et al. 2015), SVTP-645 (Phan et al. 2013), and CT80-288 (Rismanawan et al. 2014) to evaluate the capacity of TrOCR.

### Settings

The TrOCR models are built upon the Fairseq (Ott et al. 2019) which is a popular sequence modeling toolkit. For the model initialization, the DeiT models are implemented and initialized by the code and the pre-trained models from the UniLM’s official repository\(^4\). The RoBERTa models come from the corresponding page in the Fairseq GitHub repository. We use 32 V100 GPUs with the memory of 32GBs for pre-training and 8 V100 GPUs for fine-tuning. For all the models, the batch size is set to 2,048 and the learning rate is 5e-5. We use the

---

2. https://github.com/Belval/TextRecognitionDataGenerator
3. https://github.com/jpuigcerver/Laia/tree/master/egs/iam
4. https://github.com/microsoft/unilm
BPE and sentencepiece tokenizer from Fairseq to tokenize the textlines to wordpieces.

We employ the $384 \times 384$ resolution and $16 \times 16$ patch size for DeiT and BEiT encoders. The DeiT$_{SMALL}$ has 12 layers with 384 hidden sizes and 6 heads. Both the DeiT$_{BASE}$ and the BEiT$_{BASE}$ have 12 layers with 768 hidden sizes and 12 heads while the BEiT$_{LARGE}$ has 24 layers with 1024 hidden sizes and 16 heads. We use 6 layers, 256 hidden sizes and 8 attention heads for the small decoders, 512 hidden sizes for the base decoders and 12 layers, 1,024 hidden sizes and 16 heads for the large decoders. For this task, we only use the last half of all layers from the corresponding RoBERTa model, which are the last 6 layers for the RoBERTa$_{BASE}$ and the last 12 layers for the RoBERTa$_{LARGE}$. The beam size is set to 10 for TrOCR models.

We take the CRNN model (Shi, Bai, and Yao 2016) as the baseline model. The CRNN model is composed of convolutional layers for image feature extraction, recurrent layers for sequence modeling and the final frame label prediction, and a transcription layer to translate the frame predictions to the final label sequence. To address the character alignment issue, they use the CTC loss to train the CRNN model. For a long time, the CRNN model is the dominant paradigm for text recognition. We use the PyTorch implementation\footnote{https://github.com/meijieru/crnn.pytorch} and initialized the parameters by the provided pre-trained model.

**Evaluation Metrics**

The SROIE dataset is evaluated using the word-level precision, recall and f1 score. If repeated words appear in the ground truth, they are also supposed to appear in the prediction. The precision, recall and f1 score are described as:

\[
\text{Precision} = \frac{\text{Correct matches}}{\text{The number of the detected words}}
\]

\[
\text{Recall} = \frac{\text{Correct matches}}{\text{The number of the ground truth words}}
\]

\[
F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

The IAM dataset is evaluated by the case-sensitive Character Error Rate (CER). The scene text datasets are evaluated by the Word Accuracy. For fair comparison, we filter the final output string to suit the popular 36-character charset (lowercase alphanumeric) in this task.

**Results**

**Architecture Comparison** We compare different combinations of the encoder and decoder to find the best settings. For encoders, we compare DeiT, BEiT and the ResNet-50 network. Both the DeiT and BEiT are the base models in their original papers. For decoders, we compare the base decoders initialized by RoBERTa$_{BASE}$ and the large decoders initialized by RoBERTa$_{LARGE}$. For further comparison, we also evaluate the CRNN baseline model and the Tesseract OCR in this section, while the latter is an open-source OCR Engine using the LSTM network.

Table 1 shows the results of combined models. From the results, we observe that the BEiT encoders show the best performance among the three types of encoders while the best decoders are the RoBERTa$_{LARGE}$ decoders. Apparently, the pre-trained models on the vision task improve the performance of text recognition models, and the pure Transformer models are better than the CRNN models and the Tesseract on this task. According to the results, we mainly use three settings on the subsequent experiments: TrOCR$_{SMALL}$ (total parameters=62M) consists of the encoder of DeiT$_{SMALL}$ and the decoder of MiniLM, TrOCR$_{BASE}$ (total parameters=334M) consists of the encoder of BEiT$_{BASE}$ and the decoder of RoBERTa$_{LARGE}$, TrOCR$_{LARGE}$ (total parameters=558M) consists of the encoder of BEiT$_{LARGE}$ and the decoder of RoBERTa$_{LARGE}$. In Table 2, we have also done some ablation experiments to verify the effect of pre-trained model initialization, data augmentation, and two stages of pre-training. All of them have great improvements to the TrOCR models.

**SROIE Task 2** Table 3 shows the results of the TrOCR models and the current SOTA methods on the leaderboard of the SROIE dataset. To capture the visual information, all of these baselines leverage CNN-based networks as the feature extractors while the TrOCR models use the image Transformer to embed the information from the image patches. For language modeling, MSO Lab (Sang and Cuong 2019) and CLOVA OCR (Sang and Cuong 2019) use LSTM layers and H&H Lab (Shi, Bai, and Yao 2016) use GRU layers while the TrOCR models use the Transformer decoder with a pure attention mechanism. According to the results, the TrOCR models outperform the existing SOTA models with pure Transformer structures. It is also confirmed that Transformer-based text recognition models get competitive performance compared to CNN-based networks in visual feature extraction and RNN-based networks in language modeling on this task without any complex pre/post-process steps. There are some methods on the SROIE leaderboard that use additional human-labeled data and utilize ensemble methods that TrOCR does not use, which are not comparable and the results are not included in this paper.

**IAM Handwriting Database** Table 4 shows the results of the TrOCR models and the existing methods on the IAM Handwriting Database. According to the results, the methods with CTC decoders show good performance on this task and the external LM will result in a significant reduction in CER. By comparing the methods (Bluche and Messina 2017) with the TrOCR models, the TrOCR$_{LARGE}$ achieves a better result, which indicates that the Transformer decoder is more competitive than the CTC decoder in text recognition and has enough ability for language modeling instead of relying on an external LM. Most of the methods use sequence models in their encoders after the CNN-based backbone except the FCN encoders in (Wang et al. 2020a), which leads to a significant improvement on CER. Instead of relying on the features from the CNN-based backbone, the TrOCR models using the information from the image patches get similar and even better results, illustrating that the Transformer structures are competent to extract visual features well after
Table 5 shows the inference speed of the TrOCR models. Table 6 compares the TrOCR models with other methods. The Table 4 shows the CER on the IAM Handwriting dataset.

<table>
<thead>
<tr>
<th>Model Architecture</th>
<th>Training Data</th>
<th>External LM</th>
<th>CER</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Bluche and Messina 2017)</td>
<td>GCRNN / CTC</td>
<td>Synthetic + IAM</td>
<td>Yes</td>
</tr>
<tr>
<td>(Michael et al. 2019)</td>
<td>LSTM/LSTM w/Attn</td>
<td>IAM</td>
<td>No</td>
</tr>
<tr>
<td>(Wang et al. 2020a)</td>
<td>FCN / GRU</td>
<td>IAM</td>
<td>No</td>
</tr>
<tr>
<td>(Kang et al. 2020)</td>
<td>Transformer w/ CNN</td>
<td>Synthetic + IAM</td>
<td>No</td>
</tr>
<tr>
<td>(Diaz et al. 2021)</td>
<td>S-Attn / CTC</td>
<td>Internal + IAM</td>
<td>No</td>
</tr>
<tr>
<td>(Diaz et al. 2021)</td>
<td>S-Attn / CTC</td>
<td>Internal + IAM</td>
<td>Yes</td>
</tr>
<tr>
<td>(Diaz et al. 2021)</td>
<td>Transformer w/ CNN</td>
<td>Internal + IAM</td>
<td>No</td>
</tr>
<tr>
<td>TrOCR_{SMALL}</td>
<td>Transformer</td>
<td>Synthetic + IAM</td>
<td>No</td>
</tr>
<tr>
<td>TrOCR_{BASE}</td>
<td>Transformer</td>
<td>Synthetic + IAM</td>
<td>No</td>
</tr>
<tr>
<td>TrOCR_{LARGE}</td>
<td>Transformer</td>
<td>Synthetic + IAM</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 4: Evaluation results (CER) on the IAM Handwriting dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Total Sentences</th>
<th>Total Tokens</th>
<th>Time</th>
<th>Speed #Sentences</th>
<th>Speed #Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrOCR_{SMALL}</td>
<td>62M</td>
<td>2,915</td>
<td>31,081</td>
<td>348.4s</td>
<td>8.37 sentences/s</td>
<td>89.22 tokens/s</td>
</tr>
<tr>
<td>TrOCR_{BASE}</td>
<td>334M</td>
<td>2,915</td>
<td>31,959</td>
<td>633.7s</td>
<td>4.60 sentences/s</td>
<td>50.43 tokens/s</td>
</tr>
<tr>
<td>TrOCR_{LARGE}</td>
<td>558M</td>
<td>2,915</td>
<td>31,966</td>
<td>666.8s</td>
<td>4.37 sentences/s</td>
<td>47.94 tokens/s</td>
</tr>
</tbody>
</table>

Table 5: Inference time on the IAM Handwriting dataset.

pre-training. From the experiment results, the TrOCR models exceed all the methods which only use synthetic/IAM as the sole training set with pure Transformer structures and achieve a new state-of-the-art CER of 2.89. Without leveraging any extra human-labeled data, TrOCR even gets comparable results with the methods in (Diaz et al. 2021) using the additional internal human-labeled dataset.

Scene Text Datasets In Table 6, we compare the TrOCR_{BASE} and TrOCR_{LARGE} models of fine-tuning with synthetic data only and fine-tuning with synthetic data and benchmark datasets (the training sets of IC13, IC15, IIIT5K, SVT) to the popular and recent SOTA methods. Compared to all, the TrOCR models establish five new SOTA results of eight experiments while getting comparable results on the rest. Our model underperforms on the IIIT5K dataset, and we find some scene text sample images contain symbols, but the ground truth does not. It is inconsistent with the behavior in our pre-training data (retaining symbols in ground truth), causing the model to tend still to process symbols. There are two kinds of mistakes: outputting symbols but truncating the output in advance to ensure that the number of wordpieces is consistent with the ground truth, or identifying symbols as similar characters.

Inference Speed Table 5 shows the inference speed of different settings TrOCR models on the IAM Handwriting Database. We can conclude that there is no significant margin in inference speed between the base models and the large models. In contrast, the small model shows comparable results for printed and handwriting text recognition even though the number of parameters is an order of magnitude smaller and the inference speed is as twice as fast. The low number of parameters and high inference speed means fewer computational resources and user waiting time, making it more suitable for deployment in industrial applications.

Related Work

Scene Text Recognition

For text recognition, the most popular approaches are usually based on the CTC-based models. (Shi, Bai, and Yao 2016) proposed the standard CRNN, an end-to-end architecture combined by CNN and RNN. The convolutional layers are used to extract the visual features and convert them to sequence by concatenating the columns, while the recurrent layers predict the per-frame labels. They use a CTC decoding strategy to remove the repeated symbols and all the blanks from the labels to achieve the final prediction. (Gao et al. 2019) extracted the feature by the densely connected network incorporating the residual attention block and capture the contextual information and sequential dependency by the CNN network. They compute the probability distribution on the output of the CNN network instead of using an RNN network to model them.

The Sequence-to-Sequence models (Zhang et al. 2020b; Wang et al. 2019; Sheng, Chen, and Xu 2019; Bleeker and de Rijke 2019; Lee et al. 2020; Atienza 2021; Chen et al. 2021) are gradually attracting more attention, especially after the advent of the Transformer architecture (Vaswani et al. 2017). SaHAN (Zhang et al. 2020b), standing for the scale-aware hierarchical attention network, are proposed to address the character scale-variation issue. The authors use the FPN network and the CRNN models as the encoder as well as a hierarchical attention decoder to retain the multi-scale features. (Wang et al. 2019) extracted a sequence of visual features from the input images by the CNN with attention module and BiLSTM. The decoder is composed of the proposed Gated Cascade Attention Module (GCAM). For the Transformer models, (Sheng, Chen, and Xu 2019) first applied the Transformer to Scene Text Recognition. A CNN-based modality-transform block is employed to transform 2D input images to 1D sequences. (Bleeker and de Rijke 2019) added a direction embedding to the input of the decoder for the bidirectional text decoding with a single decoder, while (Lee et al. 2020) utilized the two-dimensional
Table 6: Word accuracy on the six benchmark datasets (36-char), where “Syn” indicates the model using synthetic data only and “Syn+Benchmark” indicates the model using synthetic data and benchmark datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>IIT5k</th>
<th>SVT</th>
<th>IC13</th>
<th>IC15</th>
<th>SVTP</th>
<th>CUTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlugNet (Mou et al. 2020)</td>
<td>94.4</td>
<td>92.3</td>
<td>–</td>
<td>95.0</td>
<td>–</td>
<td>82.2</td>
</tr>
<tr>
<td>SRN (Yu et al. 2020)</td>
<td>94.8</td>
<td>91.5</td>
<td>95.5</td>
<td>–</td>
<td>82.7</td>
<td>–</td>
</tr>
<tr>
<td>RobustScanner (Yue et al. 2020)</td>
<td>95.4</td>
<td>93.5</td>
<td>94.1</td>
<td>–</td>
<td>79.2</td>
<td>82.9</td>
</tr>
<tr>
<td>TextScanner (Wan et al. 2020)</td>
<td>95.7</td>
<td>92.7</td>
<td>–</td>
<td>94.9</td>
<td>–</td>
<td>83.5</td>
</tr>
<tr>
<td>AutoSTR (Zhang et al. 2020a)</td>
<td>94.7</td>
<td>90.9</td>
<td>–</td>
<td>94.2</td>
<td>81.8</td>
<td>–</td>
</tr>
<tr>
<td>RCEED (Cui et al. 2021)</td>
<td>94.9</td>
<td>91.8</td>
<td>–</td>
<td>–</td>
<td>82.2</td>
<td>–</td>
</tr>
<tr>
<td>PREN2D (Yan et al. 2021)</td>
<td>95.6</td>
<td>94.0</td>
<td>96.4</td>
<td>–</td>
<td>83.0</td>
<td>–</td>
</tr>
<tr>
<td>VisionLAN (Wang et al. 2021)</td>
<td>95.8</td>
<td>91.7</td>
<td>95.7</td>
<td>–</td>
<td>83.7</td>
<td>–</td>
</tr>
<tr>
<td>Bhunia (Bhunia et al. 2021b)</td>
<td>95.2</td>
<td>92.2</td>
<td>–</td>
<td>95.5</td>
<td>–</td>
<td>84.0</td>
</tr>
<tr>
<td>CVAE-Feed.</td>
<td>95.2</td>
<td>–</td>
<td>–</td>
<td>95.7</td>
<td>–</td>
<td>84.6</td>
</tr>
<tr>
<td>STN-CSR (Cai, Sun, and Xiong 2021)</td>
<td>94.2</td>
<td>92.3</td>
<td>96.3</td>
<td>94.1</td>
<td>86.1</td>
<td>82.0</td>
</tr>
<tr>
<td>ViTSTR-B (Atienza 2021)</td>
<td>88.4</td>
<td>87.7</td>
<td>93.2</td>
<td>92.4</td>
<td>78.5</td>
<td>75.2</td>
</tr>
<tr>
<td>CRNN (Shi, Bai, and Yao 2016)</td>
<td>84.3</td>
<td>78.9</td>
<td>–</td>
<td>88.8</td>
<td>–</td>
<td>61.5</td>
</tr>
<tr>
<td>TRBA (Baek, Matsui, and Aizawa 2021)</td>
<td>92.1</td>
<td>88.9</td>
<td>–</td>
<td>93.1</td>
<td>–</td>
<td>74.7</td>
</tr>
<tr>
<td>ABIINet (Fang et al. 2021)</td>
<td>96.2</td>
<td>93.5</td>
<td>97.4</td>
<td>–</td>
<td>86.0</td>
<td>–</td>
</tr>
<tr>
<td>Diaz (Diaz et al. 2021)</td>
<td>96.8</td>
<td>94.6</td>
<td>96.0</td>
<td>–</td>
<td>80.4</td>
<td>–</td>
</tr>
<tr>
<td>PARSeq (Bautista and Atienza 2022)</td>
<td>97.0</td>
<td>93.6</td>
<td>97.0</td>
<td>96.2</td>
<td>86.5</td>
<td>82.9</td>
</tr>
<tr>
<td>MaskOCR (ViT-B) (Lu et al. 2022)</td>
<td>95.8</td>
<td>94.7</td>
<td>98.1</td>
<td>–</td>
<td>87.3</td>
<td>–</td>
</tr>
<tr>
<td>MaskOCR (ViT-L) (Lu et al. 2022)</td>
<td>96.5</td>
<td>94.1</td>
<td>97.8</td>
<td>–</td>
<td>88.7</td>
<td>–</td>
</tr>
<tr>
<td>SVTR-L (Du et al. 2022)</td>
<td>96.3</td>
<td>91.7</td>
<td>97.2</td>
<td>–</td>
<td>86.6</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 6: Word accuracy on the six benchmark datasets (36-char), where “Syn” indicates the model using synthetic data only and “Syn+Benchmark” indicates the model using synthetic data and benchmark datasets.

Handwritten Text Recognition

(Memon et al. 2020) gave a systematic literature review about the modern methods for handwriting recognition. Various attention mechanisms and positional encodings are compared in the (Michael et al. 2019) to address the alignment between the input and output sequence. The combination of RNN encoders (mostly LSTM) and CTC decoders (Bluche and Messina 2017; Graves and Schmidhuber 2008; Pham et al. 2014) took a large part in the related works for a long time. Besides, (Graves and Schmidhuber 2008; Voigtlaender, Doetsch, and Ney 2016; Puigcerver 2017) have also tried multidimensional LSTM encoders. Similar to the scene text recognition, the seq2seq methods and the scheme for attention decoding have been verified in (Michael et al. 2019; Kang et al. 2020; Chowdhury and Vig 2018; Bluche 2016).

Conclusion

In this paper, we present TrOCR, an end-to-end Transformer-based OCR model for text recognition with pre-trained models. Distinct from existing approaches, TrOCR does not rely on the conventional CNN models for image understanding. Instead, it leverages an image Transformer model as the visual encoder and a text Transformer model as the textual decoder. Moreover, we use the wordpiece as the basic unit for the recognized output instead of the character-based methods, which saves the computational cost introduced by the additional language modeling. Experiment results show that TrOCR achieves state-of-the-art results on printed, handwritten and scene text recognition with just a simple encoder-decoder model, without any post-processing steps.
Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62276017, U1636211, 61672081).

References

Baek, J.; Matsu, Y.; and Aizawa, K. 2021. What if We Only Use Real Datasets for Scene Text Recognition? Toward Scene Text Recognition With Fewer Labels. In CVPR.
Bhunia, A. K.; Chowdhury, P. N.; Sain, A.; and Song, Y.-Z. 2021a. Towards the Unseen: Iterative Text Recognition by Distilling From Errors. In ICCV.
Bluche, T. 2016. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. NIPS.
Krishnan, P.; and Jawahar, C. V. 2016. Generating Synthetic Reading Scene Text Recognition. In NIPS.


Puigcerver, J. 2017. Are multidimensional recurrent layers really necessary for handwritten text recognition? In ICDAR.


Wan, Z.; He, M.; Chen, H.; Bai, X.; and Yao, C. 2020. Textscanner: Reading characters in order for robust scene text recognition. In AAAI.


Wang, S.; Wang, Y.; Qin, X.; Zhao, Q.; and Tang, Z. 2019. Scene text recognition via gated cascade attention. In ICME.


