EffOCR: An Extensible, Open-Source Package for Efficiently Digitizing World Knowledge

Tom Bryan[1], **Jacob Carlson**[1], **Abhishek Arora**[1], **Melissa Dell**[1,2*]
[1] Harvard University; Cambridge, MA, USA.
[2] National Bureau of Economic Research; Cambridge, MA, USA.
All authors contributed equally. *Corresponding author: melissadell@fas.harvard.edu.

Abstract

Billions of public domain documents remain trapped in hard copy or lack an accurate digitization. Modern natural language processing methods cannot be used to index, retrieve, and summarize their texts; conduct computational textual analyses; or extract information for statistical analyses, and these texts cannot be incorporated into language model training. Given the diversity and sheer quantity of public domain texts, liberating them at scale requires optical character recognition (OCR) that is accurate, extremely cheap to deploy, and sample-efficient to customize to novel collections, languages, and character sets. Existing OCR engines, largely designed for small-scale commercial applications in high resource languages, often fall short of these requirements. EffOCR (**Eff**icient**OCR**), a novel open-source OCR package, meets both the computational and sample efficiency requirements for liberating texts at scale by abandoning the sequence-to-sequence architecture typically used for OCR, which takes representations from a learned vision model as inputs to a learned language model. Instead, EffOCR models OCR as a character or word-level image retrieval problem. EffOCR is cheap and sample efficient to train, as the model only needs to learn characters' visual appearance and not how they are used in sequence to form language. Models in the EffOCR model zoo can be deployed off-the-shelf with only a few lines of code and include lightweight models designed for mobile phones that are extremely cheap to deploy. Importantly, EffOCR also allows for easy, sample efficient customization with a simple model training interface and minimal labeling requirements due to its sample efficiency. We illustrate the utility of EffOCR by cheaply and accurately digitizing 20 million historical U.S. newspaper scans, evaluating zero-shot performance on randomly selected documents from the U.S. National Archives, and accurately digitizing a Japanese document collection for which all other OCR solutions failed.

1 Introduction

Vast document collections remain trapped in hard copy or lack accurately digitized texts. For example, the U.S. National Archives holds approximately 13.28 billion pages of textual records, most of which are in the public domain.[1] These documents are preserved because they are central to the workings of the U.S. government, have long-term research value, or provide valuable information for the public, but working with most of them is costly and time-consuming. The U.S. National Archives are not unique: many other countries have national archives with public domain collections numbering in the billions of pages, not to mention state and local archives and libraries. Without accurate machine-readable data, modern natural language processing (NLP) tools cannot be used to index, retrieve, and summarize materials; conduct computational textual analyses; or extract information for statistical investigations. Public domain texts, if accurately digitized, could also provide massive scale information for training large language models, with no risks of copyright infringement.

Using optical character recognition (OCR) to digitize public domain collections on a large scale entails several challenges.

Cost: First, the OCR solution must be cheap to deploy, given document collections whose size numbers in the millions or even billions of pages. Commercial engines - as well as large open-source OCR models - fall well short of this requirement. Using them to digitize large-scale collections would require astronomical budgets.

Accuracy: Second, digitized texts need to

[1]For documents published in the United States, the public domain includes any content published by a U.S. government officer/employee in the course of official duties, all content published more than 95 years ago, and some content published before 1989 that either wasn't published with a notice or did not renew copyright. This is common, for instance, in the case of publications like local newspapers (Ockerbloom, 2019). See the supplementary materials for details.

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 579–596

be sufficiently accurate for end users' objectives, which are highly diverse. Accuracy can be particularly central for quantitative applications, for which small errors can create major statistical outliers. Models for lower resource languages, if they exist, tend to perform much worse than models for high resource settings like English.

Sample efficient, easy training: Documents are highly heterogeneous in terms of their fonts or handwritings, languages, scripts, backgrounds, and artifacts from scanning and aging. There are a diversity of documents for which no existing OCR solution works zero-shot, particularly in low resource languages. Yet stakeholders who would like to digitize these documents rarely have familiarity with deep learning frameworks. Bringing high quality OCR to low resource settings requires a simple API for training and a sample efficient architecture, with an accessible compute and annotation burden.

A diversity of pre-trained and tuneable models: Users have diverse accuracy needs, scaling requirements, and budgets. A comprehensive OCR solution would make it easy to compare the accuracy and deployment costs of models of varying sizes so that users can choose the one that best suits their needs for a particular application.

To meet these objectives, we developed `EffOCR`, an open-source OCR package designed for researchers, libraries, and archives seeking a computationally and sample efficient OCR solution for digitizing diverse document collections. EffOCR has two key ingredients: 1) a novel OCR architecture and 2) a carefully designed interface to facilitate off-the-shelf OCR usage, customization via model training if necessary, and easy sharing of OCR models.

The novel `EffOCR` model architecture is treated in detail in Carlson et al. (2023), where we compare accuracy, sample efficiency, and deployment costs to a range of popular OCR engines. In short, OCR predominantly models text recognition as a sequence-to-sequence (seq2seq) problem, in which learned representations from a vision model are taken as inputs to a learned language model. Learning how vision embeddings are used in sequence to form language requires large amounts of data. For example, the predominant transformer sequence-to-sequence OCR package (Li et al., 2021) was trained on 684 million text lines using 32 32GB V100 GPU cards. State-of-the-art seq2seq OCR is sample-inefficient to tune and infeasible for users to extend to low resource languages, which may not even have a transformer large language model (LLM) that can be used to initialize the model, as language modeling advances are concentrated in less than two dozen languages (Joshi et al., 2020). The typical stakeholder working with low resource documents has a minimal budget for training and limited experience with deep learning frameworks, underscoring the need for a much more sample efficient framework with an easy-to-use API.

Additionally, seq2seq OCR requires autoregressive decoding, which makes inference slower than it would be, all else equal, with parallel decoding.

`EffOCR` abandons the seq2seq OCR model that predominates in the literature, instead modeling OCR as a word or character level *image* retrieval problem. `EffOCR` first localizes words using highly accurate, scalable object detection methods (Ultalytics, 2023; Chen et al., 2019; Wu et al., 2019). Recognition is then modeled as a contrastively trained image retrieval problem, where image embeddings of the same character or word have similar representations, regardless of their style. `EffOCR` is trained primarily on digital fonts, combined with a modest number of character and word crops from real-world documents. At inference time, characters/words are recognized by computing their nearest neighbor in an offline dictionary of exemplar embeddings created with a digital font. Carlson et al. (2023) show, using English, Japanese, and Polytonic Greek benchmarks, that the `EffOCR` architecture is accurate, highly sample efficient, cheap to train, and extremely fast to deploy when using backbones designed for mobile phones.

To meet the challenges of digitizing large-scale and low-resource document collections, the `EffOCR` package contains the following components:

1. An off-the-shelf toolkit for applying OCR models with just a few lines of code
2. A repository of pre-trained OCR models that underlies off-the-shelf usage
3. ONNX runtime support for fast deployment
4. Comprehensive tools for efficient model tuning
5. Supports models from popular backends (Chen et al., 2019; Ultalytics, 2023) for initializing localization and any *timm*-supported model for initializing recognition

6. Easy sharing of models, to promote reusability, reproducibility, and extensibility

EffOCR has been extensively tested. For example, we have used it to cheaply digitize 20 million pages of historical public domain U.S. newspaper scans that are extremely heterogeneous, posting the massive-scale output to Hugging Face.[2] Creating this dataset within our modest budget while meeting accuracy requirements would have been impossible without EffOCR. We have also examined performance in settings where no existing OCR solutions provide usable output, and tested zero-shot performance on a random selection of U.S. National Archive documents, with a model that did not see any similar content during training. Tutorials are available at https://effocr.github.io/.

EffOCR has a GNU General Public License. It is being actively maintained and crowd-sourcing of annotations to expand the pre-trained model zoo to other languages and settings, including handwriting, is underway.

The rest of this paper is organized as follows. Section 2 briefly compares EffOCR to existing, popular OCR solutions. Section 3 describes the key features of the OCR package, and Section 4 examines several use cases: using EffOCR to digitize 20 million historical newspaper scans, using EffOCR zero-shot on randomly selected collections from the U.S. National Archives, and using EffOCR to digitize a historical Japanese publication for which all existing OCR solutions fail. Finally, Section 5 discusses the limitations of the EffOCR package.

2 Comparisons to Other OCR Engines

There is a vast literature on OCR. Of primary interest here are widely used OCR softwares, which are the most plausible alternatives to EffOCR.

EffOCR- as the name suggests - is tailored towards applications requiring computational or sample efficiency. Carlson et al. (2023) conduct detailed experiments comparing the EffOCR architecture to other widely used solutions, considering accuracy, sample efficiency, and computational efficiency. We refer the interested reader to that paper for details, summarizing the two key themes that emerge here.

Customization is highly relevant: As the preponderance of researchers still using data entry firms suggests, sometimes no existing OCR solution provides acceptable accuracy. For typewritten Japanese documents from the mid-20th century, that are of considerable relevance to studying Japan's remarkable 20th century growth performance, Carlson et al. (2023) show that the best performing engine (Baidu, the leading commercial OCR for Asian languages) gets over half of characters wrong. The widespread failure of OCR to provide acceptable results is also evidenced by a large post-OCR error correction literature (*e.g.*, Lyu et al. (2021); Nguyen et al. (2021); van Strien. et al. (2020)).

EffOCR is significantly more sample efficient than leading open-source OCR engines: EasyOCR (JaidedAI, 2021), TrOCR (Li et al., 2021), and PaddleOCR (Du et al., 2022), as shown in the supplementary materials.[3] Learning to recognize the visual features of individual characters is a highly parsimonious problem, making EffOCR cheap to tune or train from scratch. Because EffOCR does not need to understand language, it is straightforward to extend to new languages and scripts, including those that lack a transformer large language model to initialize state-of-the-art seq2seq. The convolutional models in the EffOCR model zoo can be trained on a Google Colab account, whereas training TrOCR on 684 million text lines required 32 32GB V100 cards.

A central aim of EffOCR is to democratize access to OCR to low resource languages and settings that are difficult to study because existing solutions are not suitable to these use cases. While we do not have the resources to train OCR models for all these settings, our simple APIs for training models and uploading them to the EffOCR model hub can encourage the crowdsourcing of this effort.

The most accurate OCR engines in high resource settings (*e.g.,* English) are costly to deploy at scale: TrOCR (Base) is a highly accurate state-of-the-art English OCR. With 334 million parameters, it is nearly 50 times slower to deploy than our pre-trained lightweight EffOCR English word recognition model, while offering only relatively modest gains on the evaluation tasks in Carlson

[2] https://huggingface.co/datasets/dell-research-harvard/AmericanStories

[3] EasyOCR uses a seq2seq convolutional recurrent neural network (CRNN) framework (Shi et al., 2016), TrOCR uses a seq2seq encoder-decoder transformer (Li et al., 2021), and PaddleOCR's uses Single Vision Text Recognition (SVTR), which like EffOCR abandons seq2seq, dividing text images into small (non-character) patches, using mixing blocks to perceive inter- and intra-character patterns, and recognizing text by linear prediction (Du et al., 2022).

et al. (2023).[4] For English, Google Cloud Vision (GCV) - a proprietary commercial product - dominated all open-source solutions (including EffOCR), but would have been orders of magnitude more costly to deploy. In our experience, it is frequently outside academic budgets for larger projects.

Lightweight EffOCR models are also faster than Tesseract and PaddleOCR - with the comparison to EasyOCR depending on the hardware used for deployment. This is despite having around 8x more parameters than Tesseract and around 4x more than EasyOCR (parameter count is similar to PaddleOCR). This is achieved through parallel rather than sequential decoding and ONNX integration. EffOCR is also significantly more accurate on tasks like digitizing the 20 million U.S. historical newspaper scans.

Users for whom neither computational nor sample efficiency is of concern - because they are working in a well-resourced context and don't face cost constraints for the scale of their problem - are not our target audience and may well find an existing OCR engine like Google Cloud Vision better meets their needs. In practice, academic or large-scale archival digitization of document collections often involves low-resource languages or settings, tight budget constraints, or both.

3 The EffOCR Library

3.1 Off-the-shelf Usage

At the core of EffOCR is an off-the-shelf toolkit. EffOCR is a modular framework, that first localizes lines, characters, and (for some models) words using object detection, and then recognizes characters and words by embedding their crops and retrieving their nearest neighbor from an offline index of exemplar embeddings created from a digital font.

Localization: EffOCR supports two widely used backends for localization inference: MMDetection (Chen et al., 2019), which includes state-of-the-art object detection models, and Yolo (Ultalytics, 2023), which includes fast, efficient object detection models. Users can deploy line, word, and character models from the pre-trained model zoo, that use Yolo v8 (Ultalytics, 2023) (optimized for efficiency), Yolo v5 (Jocher, 2020) (fewer dependencies) or Cascade R-CNN (Cai and Vasconcelos, 2018) (optimized for accuracy). Pre-trained localization models are available for alphabetic English/Latin, Polytonic Greek, and CJK characters (which vary significantly in their aspect ratios and groupings).

Recognition: EffOCR recognizes word and character crops using contrastively trained image retrieval models. The EffOCR model zoo currently contains 30 pre-trained models, covering English, Polytonic Greek, and horizontally and vertically-written Japanese. We chose these languages to examine the utility of EffOCR in a high resource setting, in a setting where existing solutions fail, and in an intermediate case.

The EffOCR pre-trained models use a variety of backbones: two lightweight convolutional backbones that are very efficient to deploy (Howard et al., 2019; Maaz et al., 2022), a state-of-the-art CNN encoder (Liu et al., 2022), and three vision transformers (Ali et al., 2021; Li et al., 2022; Liu et al., 2021). For English, there is a word level model that defaults to character recognition when the word is below a default (tuneable) cosine similarity threshold, as well as a character-only model.

The documentation provides more guidance on model selection. A description of the training dataset is provided alongside with the trained models such that users can quickly identify the most suitable models for their tasks.

EffOCR can be used off-the-shelf with just a few lines of code:

```
import effocr
engine = effocr.EffOCR(
    line_detector = "./line_model",
    localizer = "./localizer",
    char_recognizer = "./char_recognizer",
    word_recognizer = "./word_recognizer"
)
results = engine.infer('image.jpg')
```

ONNX (ONNX, 2021) integration is an important component, as it allows for efficient CPU deployment and interoperability between deep learning frameworks. All EffOCR stages can optionally employ ONNX-format models and ONNX-runtime inference, and models can be converted to ONNX format within the package. ONNX-runtime increases CPU throughput by up to four times (Jocher, 2020) for YOLO models used in EffOCR, which allows for cost-effective cloud deployment for processing large document sets. ONNX compatibility allows additional model speedups through graph optimizations, quantization, and pruning.

[4] TrOCR has a small model (62M parameters), but Carlson et al. (2023) find it is outperformed by the 334M parameter base model by a wide margin on historical documents.

3.2 Customized Model Training

Many low-resource settings are poorly served by existing OCR engines, and a central aim of EffOCR is to democratize OCR for these settings by providing a simple interface for custom model training that can be used by researchers and others who have limited experience with deep learning frameworks. Custom training can be initialized using a Yolo object detection model for localization and any timm image encoder model for recognition. In the near future, support for training localization models with MMDetection will be added. This futureproofs `EffOCR`, as new models are developed.

`EffOCR` supports logging of a training run on Weights and Biases (Biewald, 2020). It takes industry standard coco json labels as inputs, and hence is compatible with the outputs of a range of both open-source and proprietary labeling softwares. It also exports output in the same format, so that users can easily correct model predictions if desired to speed up labeling.

Model training with `EffOCR` is highly efficient, *e.g.,* the convolutional backbones can be trained on Google Colab. We trained all models on either a single Nvidia RTX 3090 or A6000 card.

3.3 Visualization, Storage and Export

`EffOCR` comes with a tool to visualize the OCR, side-by-side with the original image, as well as to visualize the line, word, and character predictions. These greatly facilitate quality checking the output and troubleshooting potential problems.

`EffOCR` offers users different options for data export. The default outputs of `EffOCR` include line coordinates, word coordinates, character coordinates, and the text associated with each of these annotations. The text for the full image is also assembled in the correct order. Users may choose to export only the assembled text, only text annotations associated with a given level of bounding box (line, word, or character), or all of the above.

3.4 User Contributions

By making OCR sample efficient and easy to train, `EffOCR` aims to promote the reusability and reproducibility of OCR pipelines. This is particularly important for low resource settings and languages, where there is little commercial incentive for product development and few alternatives to crowd-sourcing models. `EffOCR` users can upload their self-trained models to the `EffOCR` Hugging Face hub. Whenever a model is saved, a model card is automatically generated that follows best practices outlined in Hugging Face's Model Card Guidebook.[5] Moreover, the automatically generated card contains instructions on how to use the model in the context of `EffOCR` and model-specific architecture and training details in the interest of reproducibility.

3.5 Integration with Layout Parser

OCR engines typically detect lines, versus detecting and classifying different layout objects in a document. Many documents have complex layouts - *e.g.,* newspapers have headlines, articles, captions, ads, and headers arranged in complex multicolumn layouts, and tables likewise have different types of information arranged in oftentimes complex layouts. These structures necessitate applying object detection models for document layout analysis, which have been trained to detect the coordinates of each layout object and classify its type (*e.g.,* headline, articles, etc).

To facilitate combining `EffOCR` with deep learning-based document layout models, wrappers will be integrated into a popular open-source layout detection package, Layout Parser (Shen et al., 2020), that will allow Layout Parser users to call any `EffOCR` model. Layout Parser also has wrappers to call GCV and Tesseract, which will allow users to easily compare `EffOCR` output to these other packages to decide what best meets their accuracy and cost objectives. Layout Parser and `EffOCR` were designed by the same lab, facilitating long-run coordination between the packages.

4 Applications

Scalability: We have tested the utility of `EffOCR` with various real-world applications. In the first application, we cheaply and accurately digitized 20 million newspaper page scans from Library of Congress's Chronicling America collection (Library of Congress, 2022). The resulting dataset, American Stories, is available for download on Hugging Face.[6]. Figure 1 illustrates why this is a challenging task: newspapers are extremely heterogeneous in their fonts and image quality. Dell et al. (2023) provide a detailed analysis of the quality of the resulting text dataset.

[5] https://huggingface.co/docs/hub/model-card-guidebook

[6] `https://huggingface.co/datasets/dell-research-harvard/AmericanStories`

Figure 1: This figure shows a diversity of examples processed with EffOCR, with predicted transcriptions on the right.

We first trained character EffOCR using synthetic data plus a labeled set of 291 newspaper lines (Carlson et al., 2023), created in a couple of hours. We then bootsrapped word level annotations by creating them with the character level EffOCR model, filtering out lines with a high non-word rate.

With EffOCR, combined with layout analysis using Layout Parser, we could digitize the dataset with a $60K USD cloud compute budget (plus pipeline development costs). GCV makes significant layout errors when fed full newspaper scans and achieves best performance when fed individual lines. At current prices, digitizing the collection at the line level, since GCV charges per image, would have cost over $23 million USD. TrOCR Base, the most accurate open-source OCR, would have exceeded our budget by a factor of nearly 50.

Zero-Shot Performance: Second, we show that our English lightweight word-level model has strong zero-shot performance on randomly selected document collections from the U.S. National Archives. This model saw only newspapers in training, to test true zero shot performance. We selected a single textline from each of 300 random documents from separate National Archive record groups. EffOCR achieved a 11.2% CER on the diverse collection, compared with a 11.8% CER from Tesseract (Best), a 12.1% CER from EasyOCR, and a 51% CER from TrOCR (Small), which appeared to struggle with blurry and partially obscured text. We suspect the results could be significantly improved by including a random sample of documents from the National Archives in training, to broaden the set of real world documents that the model is exposed to.

All open-source models performed significantly worse than GCV (1.2% CER), but as discussed earlier cost concerns presently preclude its use at scale. Despite being engineered for low-resource, few-shot contexts, EffOCR remains competitive in high-resource, zero-shot situations.

Low Resource Settings: Finally, we use EffOCR to digitize historical Japanese firm level records for vertically written Japanese documents (Teikoku Koshinjo, 1957), where the best available solution (from Baidu OCR) mispredicts over half of characters. We use the evaluation set in Carlson et al. (2023), which consists of randomly selected segments that were double labeled.

Using a training set of 898 labeled table cells, we achieve a CER of 0.7%, 80 times more accurate than the best existing solution. As a result, we are able to study a variety of questions about Japan's remarkable growth performance that would have been impossible to examine without EffOCR.

To further examine the limits of sample efficiency, we calculate the character classification error when the (character) model only sees one (or up to 5) labeled character(s) for each of the characters that appear in the training set, which comprise 77% of the characters in the test set. This results in character classification errors of 13.4% and 2.0% respectively. While the model does clearly benefit from seeing multiple crops of characters that appear frequently, this illustrates viable few shot performance.

5 Limitations

If large portions of a document are illegible, vision-only OCR will not be suitable and language understanding may be helpful for inferring content. For high resource languages such as English when cost is not a concern, users may get the best mileage

from a leading commercial product such as GCV.

Currently, the `EffOCR` model zoo has pre-trained models supporting typewritten English, Japanese, and Polytonic Greek. Over the coming months, we will be crowd-sourcing annotations (including handwriting) from package users and colleagues. We will use them, along with digital fonts, to pre-train additional models. In addition, users are encouraged to contribute their models.

`EffOCR` does not currently support handwriting. We started with typewritten documents because there are billions of public domain typeface documents that are of considerable interest to researchers and the general public. We are planning to expand the model zoo to include handwriting and users have already offered to contribute annotations. Synthetic handwriting generators, *e.g.* Bhunia et al. (2021), can provide extensive data for pre-training for scripts that they support, analogous to the use of digital fonts for typeface documents. We will make synthetic handwriting datasets available so that package users can also use them for training their own custom models.

References

Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. 2021. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34.

Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Mubarak Shah. 2021. Handwriting transformers. *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1086–1094.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Zhaowei Cai and Nuno Vasconcelos. 2018. Cascade r-cnn: Delving into high quality object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162.

Jacob Carlson, Tom Bryan, and Melissa Dell. 2023. Efficient ocr for building a diverse digital history. *arXiv preprint arXiv:2304.02737*.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. 2019. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

Melissa Dell, Jacob Carlson, Tom Bryan, Emily Silcock, Abhishek Arora, Zejiang Shen, Luca D'Amico-Wong, Quan Le, Pablo Querubin, and Leander Heldring. 2023. American stories: A large-scale structured text dataset of historical u.s. newspapers.

Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, and Yu-Gang Jiang. 2022. Svtr: Scene text recognition with a single visual model. *arXiv preprint arXiv:2205.00159*.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324.

JaidedAI. 2021. Easyocr. https://github.com/JaidedAI/EasyOCR.

Glenn Jocher. 2020. YOLOv5 by Ultralytics.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the nlp world. *arXiv preprint arXiv:2004.09095*.

Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2021. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*.

Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. 2022. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*.

Library of Congress. 2022. Chronicling America: Historic American Newspapers.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986.

Lijun Lyu, Maria Koutraki, Martin Krickl, and Besnik Fetahu. 2021. Neural ocr post-hoc correction of historical corpora. *Transactions of the Association for Computational Linguistics*, 9:479–483.

Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. 2022. Edgenext: efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *European Conference on Computer Vision*, pages 3–20. Springer.

Thi Tuyet Hai Nguyen, Adam Jatowt, Mickael Coustaty, and Antoine Doucet. 2021. Survey of post-ocr processing approaches. *ACM Comput. Surv.*, 54(6).

John Mark Ockerbloom. 2019. Newspaper copyrights, notices, and renewals.

ONNX. 2021. Onnx runtime. https://www.onnxruntime.ai. Version: x.y.z.

Zejiang Shen, Kaixuan Zhang, and Melissa Dell. 2020. A large dataset of historical japanese documents with complex layouts. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 548–549.

Baoguang Shi, Xiang Bai, and Cong Yao. 2016. 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(11):2298–2304.

Teikoku Koshinjo. 1957. *Teikoku Ginko Kaisha Yoroku*. Teikoku Koshinjo.

Ultalytics. 2023. Yolo v8 github repository. https://github.com/ultralytics/ultralytics.

Daniel van Strien., Kaspar Beelen., Mariona Coll Ardanuy., Kasra Hosseini., Barbara McGillivray., and Giovanni Colavizza. 2020. Assessing the impact of ocr quality on downstream nlp tasks. In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 1: ARTIDIGH,*, pages 484–496. INSTICC, SciTePress.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. https://github.com/facebookresearch/detectron2.

Supplementary Materials

S-1 Model Architecture and Model Zoo

Figure S-1 shows the `EffOCR` model architecture, and Table S-1 summarizes the models in the `EffOCR` model zoo. Readers seeking technical details for the `EffOCR` models contained in the pre-trained model zoo are referred to the detailed supplementary materials in Carlson et al. (2023).

S-2 Sample Efficiency

To examine how efficiently EffOCR learns in comparison to leading open source architectures, we train different OCR models from scratch using varying amounts of annotated data. EffOCR-C (Base) is compared to SVTR (implemented via PaddleOCR) (Du et al., 2022), CRNN (implemented via EasyOCR) (Shi et al., 2016), and TrOCR (Li et al., 2021b). All architectures are pre-trained from scratch on 8,000 synthetic text lines, starting from pre-trained checkpoints not customized for OCR when supported by the framework. They are then fine-tuned on the study's benchmark datasets, with varying train-test-validation splits: 70%-15%-15%, 50%-25%-25%, 20%-40%-40%, 5%-47.5%-47.5%, and 0%-50%-50% (i.e., zero-shot). These exercises are performed for the English newspaper character level models and horizontal Japanese, as vertical Japanese is not supported by the comparison architectures.

Figure S-2 plots the percentage of the benchmark dataset used in training on the x-axis and the CER on the y-axis. On just 99 labeled table cells for Japanese and 21 labeled rows for LoCCA (the 5% train split), EffOCR's CER is only 5% (Japanese) and 7% (English), showing viable few shot performance. The other architectures remain unusable. EffOCR performs nearly as well using 20% or training data as using 70%, where it continues to outperform all other alternatives. This illustrates that its parsimonious architecture learns efficiently.

S-3 Training Config Details

The EffOCR package exposes a wide variety of training options and hyperparameters to users. A few key elements are described here, readers looking for more details are directed to the package documentation.

Recognizer Training Options:

- `timm_model_name` Model name from timm (Wightman, 2019) package used as a base encoder for the recognizer.
- `render_dict` Folder to store crop renders and gold training data locally.
- `font_dir_path` Local path to draw tff (font) files from, which are used to create character/word renders.
- `hns_txt_path` Local file path to draw hard negative samples from. Hard negative text files are created by default at the end of recognizer training. Most recognizer training applications use two stages, an initial run and a hard negative sampling run.
- `latin_suggested_args` Uses default arguments for alphabetic writing systems such as Latin, Greek, and Cyrillic.

In addition to these options, a wide variety of standard model training parameters are exposed, including learning rate, optimizer options, weight decay, batch size, device selection, and number of training epochs.

Localizer Training Options:

- `vertical` Whether model should expect characters aligned horizontally (as in English and many Latin scripts) or vertically (as in many character-based scripts).
- `no_words` Detect only characters, not words. Recommended for languages without word groupings.
- `iou_thresh` Training and validation IOU threshold for character/word detection.

- `conf_thresh` Training and validation confidence threshold for character/word detection.

As with the recognizer, other standard training parameters are exposed. In particular, adjusting the image input shape may be valuable for particularly long or short lines.

Hyperparameters and training procedure used to generate models listed in the Model Zoo (Table S-1) are listed in Carlson et al. (2023).

S-4 Visualization

Figure S-3 shows the `EffOCR` visualization interface.

S-5 American Stories

Figure S-4 plots the number of articles in the `American Stories` dataset, created with `EffOCR`, across time.

S-6 The Public Domain

Table S-2 provides detailed information about the requirements for information published in the United States to be in the public domain, in order to give readers a better sense of these collections.

S-7 Inference Speed

EffOCR implements two features designed to increase computational efficiency. First, both localization and recognition inference is run in a multithreaded fashion, ensuring that compute resources are fully utilized. Second, EffOCR provides support for ONNX runtime and ONNX-format models, which provide up to a 3x speedup on a CPU compared to native PyTorch runtime (ONNX, 2021). GPUs are typically cost prohibitive for digitization at scale.

Table S-3 provides a comparison between EffOCR and other commonly used OCR frameworks' python implementations. It is important to note that these numbers - across softwares - can vary significantly dependending on the hardware resources available. All comparisons are made on four 2200 MHz CPU cores, selected to represent a plausible and relatively affordable research compute setup. EffOCR performance is competitive with other widely used frameworks, with EffOCR (Small) having the fastest performance. Tesseract (Ooms, 2023) testing used the `pytesseract` package with default settings. EasyOCR (JaidedAI, 2021) testing used the `easyocr` package with default English settings. PaddleOCR (PaddlePaddle, 2022) testing used the `paddleocr` package with `use_angle_cls` option and default English settings. TrOCR (Li et al., 2021a) testing used the `transformers` package implementation, with `trocr-base-printed` and `trocr-small-printed` models for Base and Small tests, respectively. EffOCR testing used default settings with pretrained ONNX English newspaper models from the model zoo.

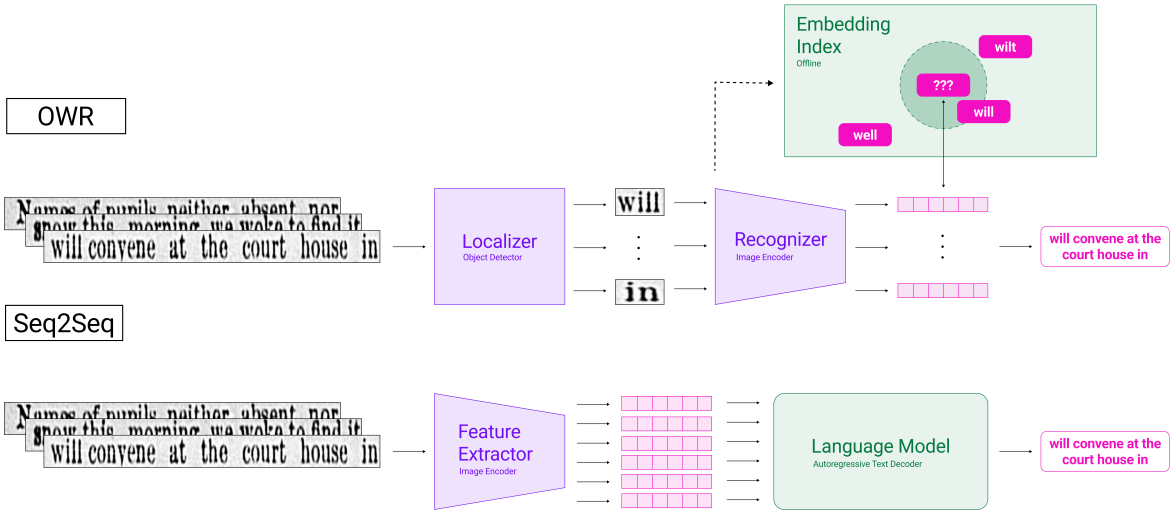

Figure S-1: **EffOCR and Seq2Seq Model Architectures.** This figure represents the EffOCR architecture, as compared to a typical sequence-to-sequence OCR architecture.

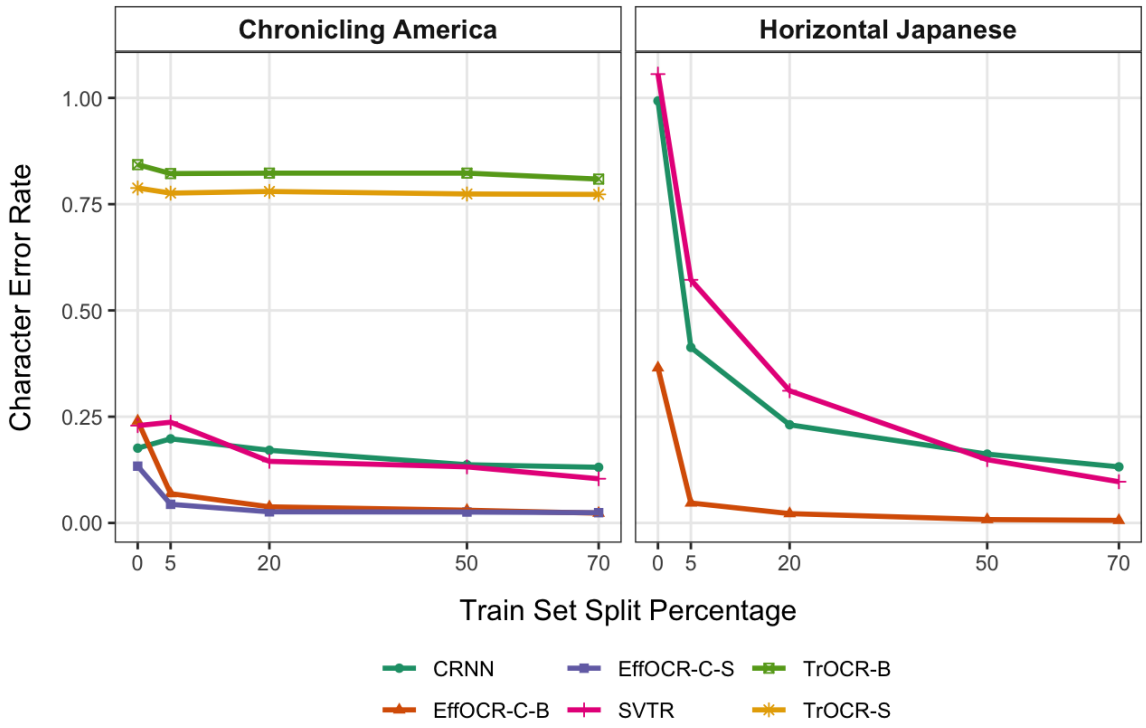

Figure S-2: **Sample Efficiency.** This figure plots the percentage of the benchmark dataset used in training against the character error rate, for different OCR model architectures: CRNN (EasyOCR), SVTR (PaddleOCR), TrOCR (Transformer OCR), and `EffOCR` small and base convolutional models.

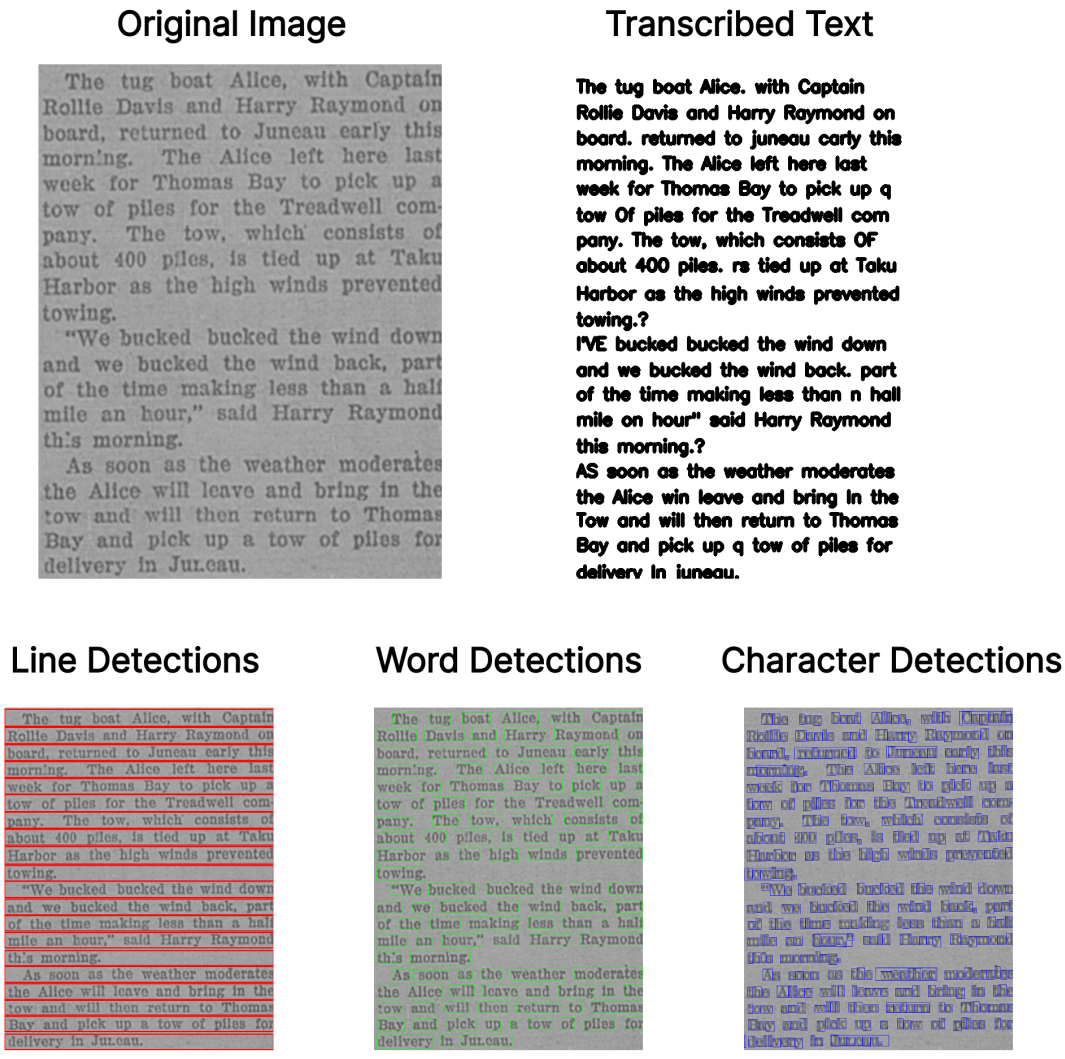

Figure S-3: **Visualization.** This figure shows the `EffOCR` visualization interface.

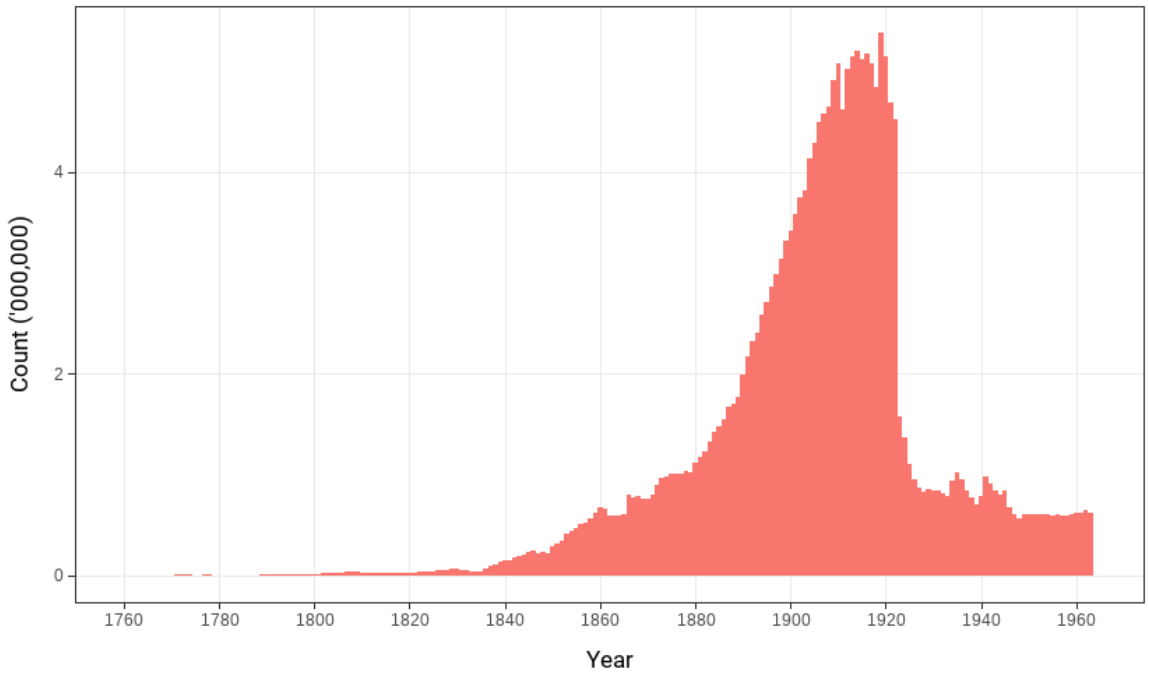

Figure S-4: **American Stories.** This figure plots the number of articles in the `American Stories` dataset, created with `EffOCR`, across time.

Training Set	Line Detection	Localizer		Word Recognition		Character Recognition				
	YOLO	YOLO	MaskRCNN	MobileNetV3	EdgeNeXt	MobileNetV3	EdgeNeXt	ViT	ConvNeXt	XCiT
English Newspapers	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
English Mixed Archival	✓	-	✓	✓	-	✓	-	-	-	-
Japanese Vertical	✓	✓	✓	N/A	N/A	✓	✓	✓	✓	✓
Japanese Horizontal	✓	✓	✓	N/A	N/A	✓	✓	✓	✓	✓
Polytonic Greek	✓	✓	-	N/A	N/A	✓	-	-	✓	-

Table S-1: Models Currently Available in the EffOCR Model Zoo. Note Japanese models do not use word-level recognition.

Date of Publication	Conditions	Copyright Term
Public Domain		
Anytime	**Works prepared by an officer/employee of the U.S. Government as part of their official duties**	**None**
Before 1928	**None**	**None. Copyright expired.**
1928 through 1977	**Published without a copyright notice**	**None. Failure to comply with required formalities**
1978 to 1 March 1989	**Published without notice and without subsequent registration within 5 years**	**None. Failure to comply with required formalities**
1928 through 1963	**Published with notice but copyright was not renewed**	**None. Copyright expired**
Copyrighted		
1978 to 1 March 1989	Published without notice, but with subsequent registration within 5 years	70 (95) years after the death of author (corporate author)
1928 through 1963	Published with notice and the copyright was renewed	95 years after publication
1964 through 1977	Published with notice	95 years after publication
1978 to 1 March 1989	Created after 1977 and published with notice	70 (95) years after the death of author (corporate author) or 120 years after creation, if earlier
1978 to 1 March 1989	Created before 1978 and first published with notice in the specified period	The greater of the term specified in the previous entry or 31 December 2047
From 1 March 1989 through 2002	Created after 1977	70 (95) years after the death of author (corporate author) or 120 years after creation, if earlier
From 1 March 1989 through 2002	Created before 1978 and first published in this period	The greater of the term specified in the previous entry or 31 December 2047
After 2002	None	70 (95) years after the death of author (corporate author) or 120 years after creation, if earlier

Table S-2: This table summarizes U.S. copyright law, based on a similar table produced by the Cornell libraries. For concision, we focus on works initially published in the United States. A variety of other cases are also covered at https://guides.library.cornell.edu/copyright.

Model	Textline/s	Article/s
EffOCR Base	0.46	0.02
EffOCR Small	21.07	1.08
Tesseract	4.47	0.21
EasyOCR	19.80	1.03
PaddleOCR	13.56	0.61
TrOCR (Base)	0.43	0.02
TrOCR (Small)	0.97	0.05

Table S-3: Comparison of EffOCR speeds with other popular OCR frameworks in CPU environment. Tests included both Textline (single lines of text) and Article (5-40 lines of text) examples.

References

Jacob Carlson, Tom Bryan, and Melissa Dell. 2023. Efficient ocr for building a diverse digital history. *arXiv preprint arXiv:2304.02737*.

Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, and Yu-Gang Jiang. 2022. Svtr: Scene text recognition with a single visual model. *arXiv preprint arXiv:2205.00159*.

JaidedAI. 2021. Easyocr. https://github.com/JaidedAI/EasyOCR.

Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2021a. Trocr github repository. https://github.com/microsoft/unilm/tree/master/trocr.

Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2021b. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*.

ONNX. 2021. Onnx runtime. https://www.onnxruntime.ai. Version: x.y.z.

J Ooms. 2023. Tesseract: Open source ocr engine.

PaddlePaddle. 2022. PaddleOCR.

Baoguang Shi, Xiang Bai, and Cong Yao. 2016. 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(11):2298–2304.

Ross Wightman. 2019. Pytorch image models. https://github.com/rwightman/pytorch-image-models.