Materials and Methods

Encoders

Different encoders can be used interchangeably for EffOCR's character localization module (hereafter, "localizer") and character recognizing module (hereafter "recognizer"). We use the following:

- **EffOCR-C (Base)**: ConvNeXt (Tiny) (Liu et al., 2022) for both the localizer and recognizer. Both models are initialized from the officially released checkpoint with specifications:
  {`size: "tiny"`}
- **EffOCR-T (Base)**: XCiT (Small) (Ali et al., 2021) for both the localizer and recognizer. Both models are initialized from the officially released checkpoint with specifications:
  {`size: "small", depth: 12, patch_size: 8, resoultion: 224`}
- **EffOCR-C (Small) and EffOCR-Word (Small)**: YOLOv5 (Small) (Jocher, 2020) for the localizer and MobileNetV3 (Small) for the recognizer. YOLOv5 is initialized from the officially released `YOLOv5s` checkpoint, and MobileNetV3 is initially from the PyTorch Image Models ("timm") (Wightman, 2019) produced checkpoint with specifications:
  {`size: "small", channel_multiplier: 0.50`}

For ablations, we also examine:

- Swin (Tiny) (Liu et al., 2021) for both the localizer and recognizer. Both models are initialized from the officially released checkpoint with specifications:
  {`size: "tiny", patch_size: 4, window: 7, resolution: 224`}
- ViTDet (Base) (Li et al., 2022) for the localizer and a vanilla vision transformer, ViT (Base), for the recognizer. Both models are initialized from the officially released checkpoint with specifications:
  {`size: "base", patch_size: 16, resolution: 224`}

These architectures were selected for the following reasons:

- **EffOCR-C (Base)**: ConvNeXt is a new state-of-the-art CNN backbone, in contrast to the other three vision transformer encoders.
- **EffOCR-T (Base)**: XCiT was chosen because of its comparative advantage in modeling fine-grained features via the ability to accommodate smaller patch sizes through a linear complexity attention mechanism, which may be especially suitable for character images with small spatial extents (as measured in pixels).
- **EffOCR-C (Small) and EffOCR-Word (Small)**: MobileNetV3 (Small) and YOLOv5 (Small) were collectively chosen to produce a speed optimized EffOCR, as both architectures are popular, easily customizable, and speed-optimized by design.
- The Swin transformer was selected because of its state-of-the-art performance on object detection tasks.
- The original ViT embeddings perform well for image retrieval, and have become a new baseline for image retrieval (El-Nouby et al., 2021).

The inference speed advantages offered by a smaller transformer encoder, such as MobileViT, are much more modest than that offered by MobileNetV3, and hence an EffOCR-T (small) model is not developed, although it would be straightforward to do so should users desire it. In tests, a MobileViTv2 (small) Recognizer model was approximately 6.5 times slower than a comparable MobileNetv3 Recognizer.

As the deep learning literature advances and new models are developed, EffOCR's modular framework and simple training recipes make it straightforward to swap in new encoders, granting the model a degree of future-proofness.

These models are all trained on a single A6000 GPU card, with hyperparameters selected using the 15% validation split, save for the models with XCiT (Small) or ViT (Base) encoders, which were trained on two A6000 GPU cards.

Character Localization

All models use an MMDetection (Chen et al., 2019) backend for localization, except for the ViTDet ablation, which uses Detectron2 (Wu et al., 2019) and YOLOv5 (Small) (Jocher, 2020) for EffOCR-C (Small), which uses its own custom training scripts. Only one EffOCR configuration, EffOCR-C (Small), has a localizer that uses a one-stage object detection framework: YOLOv5 (Small) (Jocher, 2020). All others use a two-stage object detector, specifically a Cascade R-CNN (Cai & Vasconcelos, 2019). One stage object detection is faster, and hence makes sense for the small model, where a central objective is fast inference speed.

The localizers built with ConvNeXt (EffOCR-C Base), XCiT (EffOCR-T Base), and Swin (ablation) are trained on 8,000 textlines of synthetic data for 40 epochs at a constant learning rate of $1e-4$ and fine-tuned on benchmark data for 100 epochs at a $2.5e-5$ constant learning rate, all with anchor generator scales $[2, 8, 32]$. ViTDet is trained on 8,000 textlines of synthetic data for 40 epochs with a constant learning rate of $1e-4$, and then fine-tuned for 100 epochs on benchmark data with a $1e-5$ constant learning rate. The YOLO localizer is trained on 8,000 textlines of synthetic data for 30 epochs at a constant learning rate of $1e-2$ and fine-tuned on benchmark data for 30 additional epochs, still at a constant $1e-2$ learning rate.

The synthetic data used for pre-training the localizers and comparison models was created using a custom synthetic data generator.

This generator was used to create six synthetic dataset variants, each consisting of 10,000 synthetic lines with an 80%-10%-10% train-test-validation split. The six dataset variants are: horizontal English with character sequences generated at random, horizontal Japanese with character sequences generated at random, vertical Japanese with character sequences generated at random, horizontal English with text sequences generated from Wikipedia, horizontal Japanese with text sequences generated from (Japanese) Wikipedia, and vertical Japanese with text sequences generated from (Japanese) Wikipedia. Localizers for detecting Greek text were pretrained on synthetic English data due to broad similarities between lines. Text sequence based synthetic datasets were used to pre-train seq2seq models that rely on language context, e.g., TrOCR and CRNN; character sequence based synthetic datasets were used to pre-train non-seq2seq models, e.g., EffOCR and SVTR.

Character Recognition

The EffOCR recognizer is trained using the Supervised Contrastive ("SupCon") loss function (Khosla et al., 2020), a generalization of the InfoNCE loss (Oord et al., 2018) that allows for multiple positive and negative pairs for a given anchor. In particular, we work with the "outside" SupCon loss formulation

$$\mathcal{L}_{\text{out}}^{\text{sup}} = \sum_{i \in I} \mathcal{L}_{\text{out},i}^{\text{sup}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau)}{\sum_{a \in A(i)} \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_a / \tau)}$$

as implemented in PyTorch Metric Learning (Musgrave et al., 2020), where τ is a temperature parameter, i indexes a sample in a "multiviewed" batch (in this case multiple fonts/augmentations of characters with the same identity), $P(i)$ is the set of indices of all positives in the multiviewed batch that are distinct from i, $A(i)$ is the set of all indices excluding i, and z is an embedding of a sample in the batch (Khosla et al., 2020).

To create training batches for the recognizer, EffOCR uses a custom m per class sampling algorithm *without replacement* adapted from the PyTorch Metric Learning repository (Musgrave et al., 2020). This metric learning batch sampling algorithm also implements batching and training with hard negatives, where the negative samples in a batch are selected to be semantically close to one another, and thus contrasts made between anchors and hard negatives may be especially informative for the model to

update on. Indeed, one of the main advantages of contrastive training is that it allows the learning process to exploit hard negative mining.

More specifically, the custom batch sampling algorithm samples m character variants for each class (character) - drawn from both target documents and augmented digital fonts. We choose $m = 4$ and the batch size is 128, meaning 4 styles/representations of each of 32 different characters appear in each batch. The model learns to map character crops of the same identity to similar dense vectors in a semantically rich, high-dimensional vector space, and vice versa. There is no natural definition of an epoch in the context of batch-based sampling for contrastive learning with data augmentation in the way EffOCR formulates this procedure. For EffOCR recognizer training, an epoch is defined as some number P passes through all unique characters N in the character set under consideration, i.e., $N = 13,738$ for Japanese, $N = 91$ for English, and $N = 186$ for Polytonic Greek. Empirically, a good setting for Japanese is $P = 1$, so the total number of classes in an epoch is 13,738, for English $P = 10$, so the total number of classes in an epoch is 910, and for Greek $P = 4$, so the total number of classes in an epoch is 744. Sampling for each class occurs without replacement, for better coverage of character variants. Because of this, the number of passes P matters, as it determines the number of character variants used for contrastive training in each epoch.

Every character crop that appears in the training set is embedded using a model first trained without hard negative mining/sampling, and for each we find its 8 nearest neighbors. The EffOCR recognizer is then trained again from scratch, with batches being sampled with an m per class sampler (without replacement) that is further modified to randomly intersperse hard negative sets (8 nearest neighbor characters, $m = 4$ variants of each) throughout batches.

EffOCR is trained on digital font renders from readily available fonts (13 for Japanese, 14 for English, and 8 for Greek), along with a modest number of labeled crops from the target datasets.[1] The digital fonts are augmented by randomly applying affine transformations (translation and scaling); background coloring, color jittering, color inversion, and grayscaling; and Gaussian blurring. The model trains on digital fonts and labeled crops *together*, since the objective is to learn general purpose embeddings that would map target crops nearby to digital renders. All recognizer models except MobileNetV3 use an AdamW optimizer with weight decay of $5e-4$, a SupCon loss with temperature of 0.1, a learning rate of $2e-5$, and a batch size of 128. MobileNetv3 uses the same parameters except a learning rate of $2e-3$. The Japanese datasets are trained for 60 epochs and the English and Greek datasets for 30.

After recognizer training is completed, the recognizer is used as an encoder to create an offline index of exemplar character embeddings to be searched at inference time for the purposes of character recognition. Specifically, the exemplar character embedding index is created by embedding image renders for all the unicode characters supported by the Google Noto Serif font series, i.e., Noto Serif CJK JP Regular for models trained for Japanese OCR and Noto Serif Regular for models trained for English and Greek OCR. The Google Noto series is chosen as an exemplar font due to both its extremely wide coverage of glyphs and the simplicity of its style, though, by virtue of EffOCR's training, other fonts could be used as well. At inference time, FAISS (Johnson et al., 2019) is used to perform an *inner product* similarity search that compares character embeddings in the sample being inferenced to exemplar character embeddings in this offline index; identities are assigned to inferenced characters using the identity of that character's nearest neighbor in the offline exemplar index, i.e., k-NN classification with $k = 1$.

For case sensitive applications, EffOCR character recognition for English text can also be lightly post-processed to help better differentiate uppercase and lowercase letters from one another: one can force a character to be uppercased or lowercased through simple rules based statistics about the dimensions of bounding boxes (in the sample undergoing inference). This procedure is irrelevant for results reported in this text, however, for which CER is measured uncased.

[1]Fonts for Japanese included: Dela Gothic One Regular; Hachi Maru Pop Regular; Hina Mincho Regular; Komorebi Gothic; Kosugi Regular; New Tegomin Regular; Noto Serif CJK JP Regular; Reggae One Regular; Shippori Mincho B1 Regular; Stick Regular; taisyokatujippoi7T5; Tanugo Regular; and Yomogi Regular. Fonts for English included: Anton Regular; Cutive Mono Regular; EB Garamond Regular; Fredoka Regular; IM Fell DW Pica Regular; NewYorker-jLv; Noto Serif Regular; Oldnewspapertypes-449D; Orbitron Regular; Special Elite Regular; Ultra Regular; VT323 Regular; ZaiConsulPolishTypewriter-MVAxw; and ZaiCourierPolski1941-Yza4q. Fonts for Polytonic Greek included EB Garamond Regular; Noto Serif Regular; SBL Greek; Gentium Book Plus Regular; Gentium Plus Regular; Gentium Plus Italic; Orbiton Regular; Ultra Regular; and

Greek text is also case-sensitive and CER from Greek data is also presented uncased, although upper- and lowercase Greek characters bear less resemblance than in English. Two additional rules were also applied when evaluating Greek text: apostrophes (') and accents (‘) were considered equivalent, and the *stigma* ligature was considered equivalent to the *terminal sigma* (ς) character.

Checkpoints/weights for all recognizers are supported by implementations from timm (Wightman, 2019).

Word Recognition

We train word recognition as a nearest neighbor image retrieval problem. The training dataset for the model consists of digital renders of words created using 43 fonts, silver quality data from the target dataset created by applying the EffOCR-C (Small) model to a random sample of days, and a small number of randomly selected hand labeled word crops. We limited the number of crops with model-generated labels to 20 - so each word can have 0-20 silver-quality crops depending upon its frequency of occurrence in our random sample. This limit is binding for common words, *e.g.,* "the".

The recognizer is trained using the Supervised Contrastive ("SupCon") loss function (Khosla et al., 2020), as above. To create training batches for the recognizer, we use a custom m per class sampling algorithm without replacement, adapted from the PyTorch Metric Learning repository (Musgrave et al., 2020). The m word variants for each class (word) are drawn from both target documents and augmented digital fonts. We select $m = 4$ and the batch size is 1024, meaning 4 styles of each of 256 different words appear in each batch. For training without hard negatives, we define an epoch as letting the model see each word (case-sensitive) exactly $m = 4$ times. Sampling for each class occurs without replacement until all variants are exhausted.

In order to converge faster with limited compute, we also implement offline-hard negative mining, batching similar negatives and their corresponding positive anchors together - thus making the contrasts between the positive and negative pairs within a batch especially informative. To create hard negative sets, we render each word using a reference font (Noto-Serif Regular) and embed it to create a reference index. We find $k = 8$ nearest neighbors for each word using this index and the model trained without hard negatives, which yields sets of 8 words that have a similar appearance when rendered with the reference font. We use only the reference font to create these sets because using crops corresponding to all 43 fonts for each word is computationally costly and creates more hard negative sets than we can use in training. We also use each word crop from the target dataset (both silver quality annotations generated with model predictions and gold quality human-annotated predictions) to create hard negative sets. Hence, the total number of hard-negative sets equals the number of words in our dictionary (generated with the reference font) plus the number of word crops from the newspaper data in the training set.

Each hard negative set contains 8 words, with $m = 4$ views per word, which means we can fit 32 randomly sampled hard negative sets within each batch. An epoch is defined as seeing each hard negative set once. Since the number of synthetic views of an image is much larger than the number of target newspaper crops, whenever newspaper crops are available we force the m views of a word to contain an equal number of synthetic and target crops.

We use a MobileNetV3 (Small) encoder pre-trained on ImageNet1k sourced from the timm (Wightman, 2019) library, more specifically, the model *mobilenetv3_small_050*. We use 0.1 as the temperature for SupCon loss and AdamW as the optimizer with Pytorch defaults for all parameters other than weight decay (5e-4) and learning rate. We used Cosine Annealing with Warm Restarts as the learning rate scheduler with a maximum learning rate of $2e-3$, a minimum learning rate of 0, time to first restart (T_0) as the number of batches in an epoch, and restart factor, T_{mult} of 2 using the implementation provided in Pytorch.

While fonts and newspaper crops for each word act as an augmentation on the skeleton of the word, we also add more image-level transformations to improve generalization. These include Affine transformation (only slight translation and scaling allowed), Random Color Jitter, Random Autocontrast, Random Gaussian Blurring, Random Grayscale, Random Solarize, Random Sharpness, Random Invert, Random Equalize, Random Posterize and Randomly erasing a small number of pixels of the image. Additionally, we pad the word to make the image square while preserving the aspect ratio of the word render. We

do not use common augmentations like Random Cropping or Center Cropping, to avoid destroying too much information.

The model trained without hard negatives was trained for 50 epochs and with hard negatives, it was trained for 40 epochs. For selecting the best checkpoint, we use 1-CER (OCR Character Error Rate) as the validation metric on the validation set. We chose the model that performed best in terms of CER when detecting only words on the validation set. This means that if a word is outside of our dictionary, it is forcefully matched to the nearest neighbor in the dictionary. The best model achieved a CER of 4.9% with word-only recognition.

At inference time, words are recognized by retrieving their nearest neighbor from the offline embedding index created with the reference font, using a Facebook Artificial Intelligence Similarity Search backend (Johnson et al., 2019). The code to train the model and generate training data, as well as the model checkpoints, are made publicly available.

Comparisons

To examine sample efficiency, we train alternative architectures from scratch, on the same number of synthetic text lines used to train EffOCR. Specifically, the comparison architectures are, as applicable, initialized with "default" pre-trained checkpoints that have not yet been exposed to an OCR task, e.g., masked language model pre-trained weights for text transformers or ImageNet pre-trained weights for CNNs and vision transformers. These comparison architectures are then trained on 8,000 synthetic text lines per the applicable synthetic dataset variant (see: Methods - Synthetic Data) as a form of standardized OCR-task-specific pre-training. They are then fine-tuned on the same benchmark datasets used to assess EffOCR, but 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).

The hyperparameters used for initializing and training comparison models are as follows:

- The EasyOCR implemented **CRNN** (Shi et al., 2016) comparison is trained from a random initialization (as is the default in EasyOCR) for 100,000 iterations on the horizontal English text sequence and horizontal Japanese text sequence synthetic datasets, respectively. The learning rate is fixed at 1.0 with an Adadelta optimizer and the batch size is 128, per the EasyOCR configuration defaults. The architecture uses VGG for feature extraction, a BiLSTM for seq2seq/language modeling, and a CTC loss, as also is the EasyOCR default. A new prediction head is used to match the character set associated with EffOCR for Japanese. The resulting model is then fine-tuned for 30,000 iterations with a batch size of 64, and all other hyperparameters the same, on the benchmark datasets of varying splits.
- The **SVTR** (Du et al., 2022) comparison is first trained from a random initialization for 500 epochs with an Adam optimizer with cosine-scheduled learning rate of 0.001 and batch size of 32 on horizontal English character sequence and horizontal Japanese character sequence synthetic datasets, respectively. All these hyperparameters are PaddleOCR defaults, which are also used for fine-tuning on the benchmark dataset splits.
- The **TrOCR** (Li et al., 2021a) comparison models are initialized from the appropriate vision transformer and language transformer pre-trained encoder and decoder checkpoints: for TrOCR (Base) this is the officially released BEiT (Base) checkpoint and the officially released RoBERTa (Large) checkpoint used by the TrOCR authors for model initialization; for TrOCR (Small) these are similarly the officially released checkpoints for DeiT (Small) and MiniLM used by the TrOCR authors for their model initialization. These checkpoints are exported directly from the TrOCR GitHub repository (Li et al., 2021b) using a modified script originally authored by Hugging Face (Wolf et al., 2020), such that training is possible in native PyTorch with Huggingface model implementations. TrOCR (Base) is trained on the horizontal English synthetic text sequence dataset for 60 epochs at a fixed learning rate of $5e-7$ with a batch size of 16; TrOCR (Small) is trained for 40 epochs, with all other hyperparameters the same. (The learning rate was selected based on experiments with the validation set.) The resulting models are then fine-tuned with the same hyperparameters on the various benchmark dataset splits.

To evaluate how existing solutions perform when fine-tuned on the EffOCR benchmark datasets, existing pre-trained checkpoints from the EasyOCR CRNN, PaddleOCR SVTR, and TrOCR (Base) and TrOCR (Small) models are fine-tuned on the baseline 70%-15%-15% split of the benchmark datasets. Specifically, the 15% validation set is used for hyperparameter tuning and the 15% test set is used to construct the results reported in the study.

For all comparison models, training hyperparamters are the same as used during the sample efficiency assessments with standardized synthetic pre-training, save that prediction heads for relevant models are left as they are by default. Model initialization differs, accordingly: TrOCR (Base) and TrOCR (Small) use `microsoft/trocr-base-stage1` and
`microsoft/trocr-small-stage1` checkpoints, respectively; EasyOCR CRNN uses the most recently released `japanese_g2.pth` and `english_g2.pth` checkpoints; and PaddleOCR SVTR uses the most recently released `japan_PP-OCRv3_rec_train` and
`en_PP-OCRv3_rec_train` best accuracy checkpoints.

Inference Speed Comparisons

For digitizing large-scale collections, fast inference on a CPU is necessary, due to the high costs of GPU compute. All comparisons are made on four 2200 MHz CPU cores, selected to represent a plausible and relatively affordable research compute setup. To standardize measurements of speed, each model generated predictions on the same 15% test set. All EffOCR models are implemented with ONNX Runtime for cross-compatibility and speed.

Inference speed is inherently dependent on implementation and it is plausible that the other open-source architectures may be updated in the future to achieve faster inference speeds. A strong correlation between model size and inference speed is apparent and intuitive, highlighting the utility of the EffOCR-C (Small) model for digitizing knowledge - like the Chronicling America collection - at scale.

A random sample of 10 LoCCA scans shows an average of 1944 column x lines per scan (historical newspapers used small fonts and contained few images), which implies the cost at current prices to digitize the LoCCA collection at the line level using GCV would be over 23 million US dollars.

Using FS4 VM instances in Microsoft Azure to process all content in the LoCCA collection for one randomly selected day per decade, on average it took 17.21 seconds to process 1,000 lines with EffOCR-C (small). At current prices, this translates to a cost of $0.000908 per one thousand lines, as compared to GCV's current prices of $1.50 (first 5 million units) and $0.60 (above 5 million units) per thousand lines to process Chronicling America at the line level.

Benchmark Dataset Creation

Figure S-1 illustrates the documents used to create this study's benchmarks. The OCR systems evaluated in this study take lines (cells in tables or individual lines from columns in prose) as inputs. These segments were created using a Mask R-CNN (He et al., 2017) model custom-trained with Layout Parser (Shen et al., 2021), an open-source package that provides a unified, deep learning powered toolkit for recognizing document layouts. Mask R-CNN was applied to the three Japanese publications considered and to ten different newspapers randomly selected from Chronicling America. Segments were selected at random for inclusion in this study's benchmark datasets. Table S-1 provides dataset statistics.

To create the character region and text annotations, three highly skilled annotators - undergraduate and graduate students - annotated each segment. All discrepancies were then hand checked and resolved by the study authors. Each of the datasets has a 70%-15%-15% train-validate-test split used for baseline evaluations. The validation set was used for model development, whereas the test set was used only once, to create the results reported in this study.

Supplementary Results

Ablations

To elucidate which components of EffOCR are essential for its performance, several ablations are examined in Table S-2: using a simple feedforward neural network classifier head for recognition instead of performing k-nearest neighbors classification[2], training with and without hard negatives, disabling training on synthetic data for the recognizer and localizer, and the use of alternative vision encoders. All ablations use a fixed set of hyperparameters that are associated with a specific localizer-recognizer configuration; these hyperparameters are outlined in the sections on Character Localization and Character Recognition.

Modeling character-level classification as an image retrieval problem weakly dominates the classification performance when using a standard multilayer perceptron with softmax procedure for classification. OCR as retrieval is chosen as the baseline not only due to its performance, but because it also allows for adding new characters at inference time (just embed a new exemplar character and add it to the offline index) - common in historical and archaeological settings - and because efficient similarity search technologies like FAISS (Johnson et al., 2019) provide fast inference.

Removing hard negatives increases the character error rate substantially, particularly for Japanese, which has many characters with highly similar visual appearances, e.g., some multi-stroke kanji are nearly identical to one another and differ only in the slants of some strokes. Using hard negatives in constrastive training effectively incentivizes the model to distinguish between these very visually similar characters.

Training on only labels from the target documents leads to a large deterioration in performance for Japanese. This is as expected, given that only a fraction of *kanji* characters appear in the small training datasets. The deterioration in performance is modest for English, where there are far fewer characters. The opposite is true for character localization. Localization for English is a harder problem than for Japanese because character silhouettes and aspect ratios are more variable.

Two additional vision transformer encoders are explored: Swin (Tiny) (Liu et al., 2021) for both the localizer and recognizer and ViTDet (Base) (Li et al., 2022) for the localizer and a vanilla vision transformer, ViT (Base), for the recognizer. The performance is similar to the base EffOCR-C and EffOCR-T models.

Out-of-Distribution Performance

To evaluate how EffOCR does out-of-distribution, we compare EffOCR-Word (Small) to other solutions on a highly diverse, out-of-distribution dataset sampled randomly from 64 randomly selected (out of 638) document record groups in the U.S. National Archives. This is a challenging dataset, with examples shown in Figure S-2.

EffOCR performs similarly to other open-source OCR engines (CER = 12.6), having only been exposed to highly out-of-distribution newspapers during training (Table S-3. GCV performs significantly better than any open-source solution, but with 14 billion documents in the National Archives, the cost of digitizing any appreciable share of them would be astronomical.

Using EffOCR to liberate data at scale

EffOCR can convert the publications examined in this study (Jinji Koshinjo, 1954; Teikoku Koshinjo, 1957; Jinji Koshinjo, 1939) into a knowledge graph showing relationships through shareholding patterns, family ownership, financing, boards, occupational histories, family connections, spatial locations, and supply chains.

[2]Implicitly, retrieving the nearest neighbor character from an index of offline exemplar character embeddings, as the EffOCR recognizer does by default, is k-NN classification with $k = 1$.

Figure S-3 provides an illustrative example of one component of this graph, showing supply chain networks in 1956 that were constructed by using EffOCR to digitize the customers and suppliers of Japan's 7,000 largest firms. Fine-grained control through EffOCR allowed detecting an atypical character separating firms in the customer-supplier lists - required for accurate digitization - that other OCR solutions did not systematically recognize, as well as accurate digitization of firm names. Each node in the graph is a firm, whose size is proportional to its degree centrality in the supply chain network. Shading denotes the big-three firms in pre-war Japan (Mitsui, Mitsubishi, and Sumitomo), as well as other firms - comprising Japan's largest conglomerates - targeted by the Holding Company Liquidation Commission in the late 1940s. The graph underscores that the largest pre-war firms remained the most central in Japanese supply chain networks in the 1950s, despite various policies in the late 1940s designed to curb their influence (of Staff, 1945; Commission), 1973; Hadley, 2015; Cohen, 1987).

Supplementary Tables

	Horiz. Jap. Tables	Vert. Japanese Tables	Vert. Jap. Prose	Chronicling America
Train Lines	1309	898	459	291
Val Lines	280	192	98	62
Test Lines	281	193	100	64
Total	1870	1283	657	417
Train Chars	3089	3296	5832	7438
Val Chars	673	677	1063	1708
Test Chars	682	701	1111	1727
Total	4444	4674	8006	10873

Table S-1: This table reports the number of annotated lines and characters in the training, validation, and test sets of this study's four benchmarks.

		Feed Forward	Hard Neg.	No Synthetic Data		Encoder	
	EffOCR-C (Base)	Neural Net	Off	Recognizer	Localizer	Swin (Tiny)	ViT (Base)
Horizontal Japanese	**0.006**	0.006	0.041	0.594	0.009	0.009	0.010
Vertical Japanese (tables)	**0.007**	0.010	0.087	0.700	0.016	0.016	0.010
Vertical Japanese (prose)	0.030	0.038	0.076	0.788	0.032	0.036	**0.027**
Chronicling America	**0.023**	0.037	0.045	0.027	0.068	0.025	0.037

Table S-2: This table provides the character error rate. *Feed Forward Neural Net* models the recognizer as a classification problem with a feed forward neural network, *Hard Neg. Off* does not include hard negatives in recognizer training, *No Synthetic Data* turns off synthetic data training in the recognizer and localizer, respectively, and *Swin (Tiny)* and *ViT (Base)* are alternative vision encoders.

OCR Model	Character Error Rate
EffOCR-Word (Small)	0.126
Tesseract OCR (Best)	0.118
EasyOCR CRNN	0.129
PaddleOCR SVTR	0.160
Google Cloud Vision OCR	0.018
TrOCR (Base)	0.103
TrOCR (Small)	0.537

Table S-3: **Zero Shot Performance on National Archives Dataset.** This table reports off-the-shelf performance of different OCR architectures on a diverse dataset of US National Archives documents.

Supplementary Figures

Figure S-1: **Dataset Description.** Representative samples of the publications examined in this study.

Figure S-2: **Diversity in the National Archives Dataset.** This figure shows examples sampled from the National Archives Zero-Shot evaluation dataset, along with EffOCR predicted transcriptions.

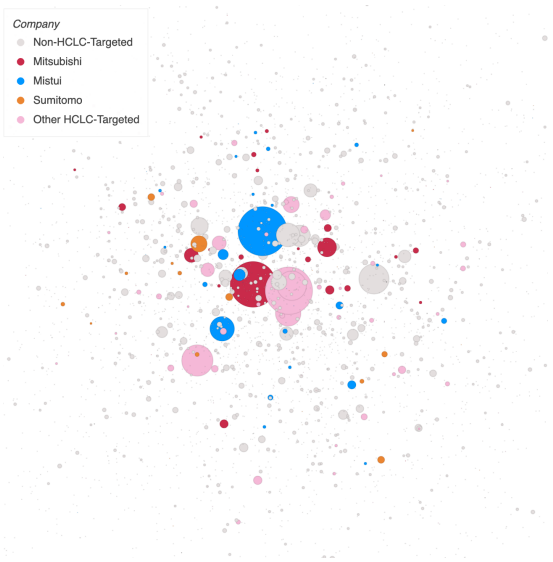

Figure S-3: **Supply Chain Networks (Japan, 1956).** Each node in the graph is a firm, whose size is proportional to its degree centrality in the supply chain network. Shading denotes three of the largest firms in pre-war Japan - Mitsui, Mitsubishi, and Sumitomo - as well as other firms - comprising Japan's largest conglomerates - targeted by the Holding Company Liquidation Commission (HCLC) in the late 1940s.

References

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

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1483–1498, 2019. ISSN 1939-3539. doi: 10.1109/tpami.2019.2956516. URL `http://dx.doi.org/10.1109/tpami.2019.2956516`.

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

Theodore Cohen. *Remaking Japan: The American occupation as new deal*. Free Press, New York, 1987.

Mochikabu Kaisha Seiri Iinkai(Holding Company Liquidation Commission). *Nihon zaibatsu to sono kaitai (The Dissolution of Japan's Zaibatsu*, volume 2. Hara Shobo, Toyko, 1973.

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

Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021.

Eleanor M Hadley. *Antitrust in Japan*. Princeton University Press, Princeton, NJ, 2015.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Jinji Koshinjo. *Jinji koshinroku*. Jinji Koshinjo, 1939.

Jinji Koshinjo. *Nihon shokuinrokj*. Jinji Koshinjo, 1954.

Glenn Jocher. YOLOv5 by Ultralytics, 5 2020. URL `https://github.com/ultralytics/yolov5`.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

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

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

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

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

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

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. Pytorch metric learning, 2020.

Joint Chiefs of Staff. Basic initial post surrender directive to supreme commander for the allied powers for the occupation and control of japan, jcs 1380/15, 1945. URL `https://www.ndl.go.jp/constitution/e/shiryo/01/036/036tx.html`.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Zejiang Shen, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, and Weining Li. Layoutparser: A unified toolkit for deep learning based document image analysis. *International Conference on Document Analysis and Recognition*, 12821, 2021.

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

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

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

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