On Web-based Visual Corpus Construction for Visual Document Understanding

Donghyun Kim[1], Teakgyu Hong[2⋆],
Moonbin Yim[1], Yoonsik Kim[1], and Geewook Kim[1⋆⋆]

[1] NAVER CLOVA
[2] Upstage AI

Abstract. In recent years, research on visual document understanding (VDU) has grown significantly, with a particular emphasis on the development of self-supervised learning methods. However, one of the significant challenges faced in this field is the limited availability of publicly accessible visual corpora or extensive collections of images with detailed text annotations, particularly for non-Latin or resource-scarce languages. To address this challenge, we propose Web-based Visual Corpus Builder (Webvicob), a dataset generator engine capable of constructing large-scale, multilingual visual corpora from raw Wikipedia HTML dumps. Our experiments demonstrate that the data generated by Webvicob can be used to train robust VDU models that perform well on various downstream tasks, such as DocVQA and post-OCR parsing. Furthermore, when using a dataset of 1 million images generated by Webvicob, we observed an improvement of over 13% on the DocVQA Task 3 compared to a dataset of 11 million images from the IIT-CDIP. The implementation of our engine is publicly available on `https://github.com/clovaai/webvicob`.

Keywords: Visual Document Understanding · Optical Character Recognition · Document Image Processing.

1 Introduction

Language modeling has been a long-standing fundamental task in natural language processing (NLP). The trained language models (LMs) are utilized in a range of downstream NLP applications, such as information extraction (IE) [17] and question answering (QA) [9]. To build a powerful LM, recent methods utilize large-scale text corpus at the pretraining phase. The text corpus is generally constructed with a specialized engine or software. For example, WikiExtractor [1] extracts texts from Wikipedia HTML dumps and builds a clean text corpus.

Visual Document Understanding (VDU) [42,15,22] has been developed to conduct a wide range of real-world tasks on document images. For example,

⋆ This work is done at NAVER CLOVA.
⋆⋆ Correspondence to `gwkim.rsrch@gmail.com`

Document Parsing [42,22] aims to extract some key texts from a document image [17,18]. Most recent VDU backbones can be regarded as an extension of LM.

Inspired by recent advances in LMs [40,9], recent VDU methods share a similar approach that (1) it first collects large-scale real document images, (2) conducts OCR on the images to extract texts, and (3) trains a BERT-like LM backbone on the extracted texts [42,41,15,16].

Although conventional VDU methods have shown promising results, several practical challenges exist, especially in the training dataset preparation phase. Most recent works [42,15] rely on a large-scale real image dataset. For example, IIT-CDIP dataset [27] consists of 11M industrial document images and is utilized in a range of VDU works. However, in most low-resourced languages, there is no public dataset like IIT-CDIP. In addition, off-the-shelf OCR engines (e.g., CLOVA OCR API,[3] MS Read API,[4] Amazon Textract[5]) are required to extract texts in the pre-processing. This often requires enormous costs.

Moreover, using OCR can have other negative consequences; the constructed data is strongly tied to the OCR engine, and OCR errors are propagated throughout the process. This problem becomes severe, especially in some non-Latin languages such as Korean and Japanese, where OCR is known to be complicated.

In this work, we propose an engine for building a web-based visual corpus. As shown in Figure 2 and Figure 3, the proposed Web-based Visual Corpus Builder (Webvicob) renders a web page into an image file and generates rich text annotations. With expressive Document Object Model (DOM) APIs, Webvicob produces accurate bounding boxes for all characters.

Moreover, Webvicob covers a wide range of word contexts. Wikipedia is a huge dataset consisting of over 270 languages and containing 60 million HTML documents. The results of PCA analysis on 3,626 samples are available in Section 5.3.

Compared to the traditional model trained on IIT-CDIP, the Webvicob-trained model shows a competitive result on DocVQA Task 1 and a higher score[6] on Task 3, showing the effectiveness of the de-biased Webvicob-based visual corpora.

The proposed method is simple yet effective. We show that the Webvicob-generated corpus is critical in building a powerful VDU backbone. Through extensive experiments and analyses, we show the effectiveness of Webvicob. The contributions of this work can be summarized as follows:

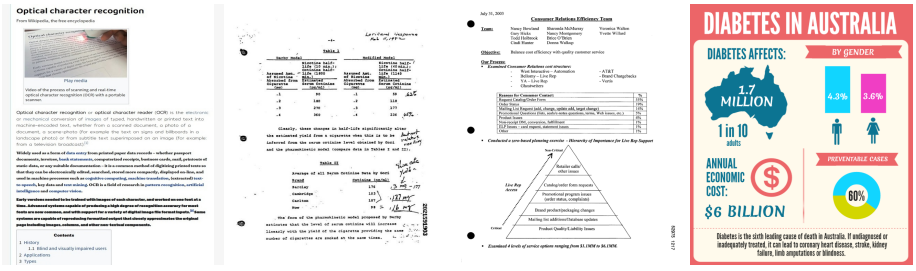
1. We propose Webvicob, which can be used for pretraining the visual document understanding models. Webvicob provides rich annotations, including character, word, line, and paragraph information.
2. Webvicob provides support for a wide range of fonts, allowing visual documents with identical content to appear differently. Additionally, we have

[3] https://clova.ai/OCR

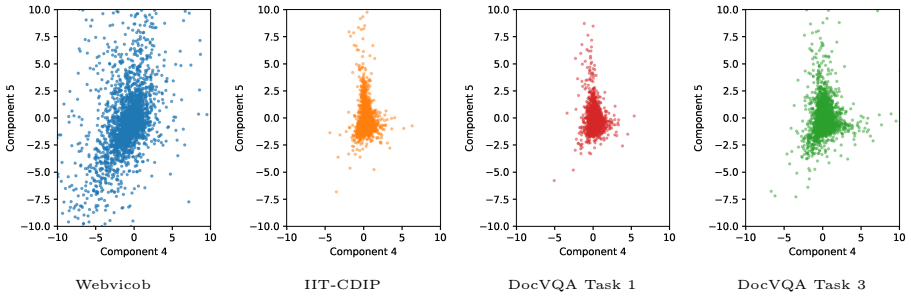
[4] https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision

[5] https://learn.microsoft.com/ko-kr/azure/cognitive-services/computer-vision/overview-ocr

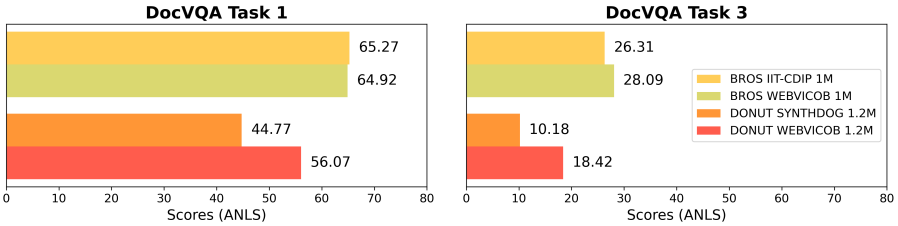
[6] https://rrc.cvc.uab.es/?ch=17&com=evaluation&task=3



(a) Document image samples.



(b) PCA visualization.



(c) Document VQA benchmarks.

Fig. 1: **Image Samples and Visualization of Main Results.** (a) Samples of Webvicob, IIT-CDIP [27], DocVQA Task 1 [39], and Task 3 [32] are shown, respectively. (b) Visualization of principal component analysis (PCA) with bag of words (BoW) representations. The visualization was carried out using the 4^{th} and 5^{th} principal components. The analysis method and further visualization results can be found in section 5.3. (c) Results on two benchmarks with pretrained $\text{BROS}_{\text{BASE}}$ [15] and $\text{Donut}_{\text{Proto}}$ [22] are shown. As can be seen in (a) and (b), DocVQA Task 1 has a similar distribution to IIT-CDIP, while Task 3 does not. This also affects the final score.

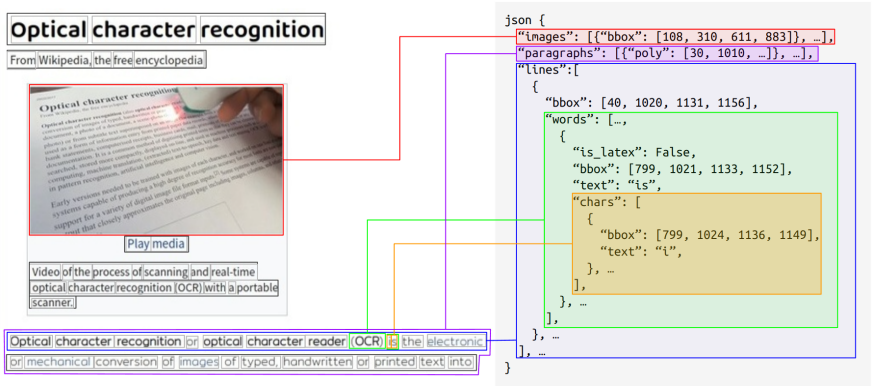


Fig. 2: **A sample of Webvicob dataset.** The dataset contains large-scale web document images with hierarchical text annotations (i.e., character, word, line, and paragraph-level annotations). If box contains LaTeX [34] (i.e., math formula), we set "is_latex" value as True.

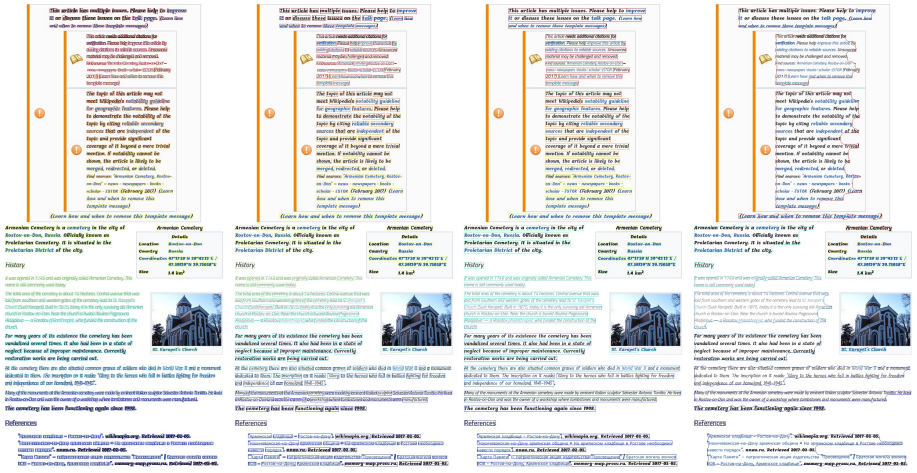
taken into account the characteristics of each font to construct precise character-level bounding box annotations.
3. We conduct extensive experiments to verify the effectiveness of the proposed engine and dataset.
4. The source code of our engine is publicly available to promote research on VDUs in low-resource languages.

2 Background and Related Work

In this section, we introduce a traditional VDU pipeline and datasets. Most of the current methods share a similar approach of (1) collecting large-scale real images, (2) conducting OCR on the images, and (3) training a BERT-like backbone on the extracted texts [42,15].

2.1 VDU Backbones

Inspired by BERT [9] and the recent advancements in language modeling, a range of BERT-like Transformer-based VDU backbones have been proposed [42,15,41,16,8]. For handling layout information of document images, spatial coordinates of OCR text boxes are fed to the VDU backbone [42,15,43]. Using visual encoders like ResNet [14], visual features of an input image are also being incorporated into the recent VDU backbones [41,16]. More recently, with the advances in Vision Transformer (ViT) [10], training a Transformer encoder-decoder VDU backbone without OCR has also been attempted [22,7,26]. Our engine can be used together in various VDU backbones. In this paper, we verified the performance of the Webvicob engine by pretraining BROS [15], LayoutXLM [43], and Donut [22].



(a) Character (b) Word (c) Line (d) Paragraph

Fig. 3: **Visualization of Webvicob annotations.** We use a colormap to show the order of box annotation.

2.2 Visual Corpus Construction for VDU

Most existing OCR annotated datasets have small sizes, leading to difficulties in training VDU backbones. To construct a rich corpus, in the traditional pipeline, large-scale real-world document images (e.g., IIT-CDIP) and an OCR engine (e.g., CLOVA OCR API [3]) are used.

The quality of the OCR engine significantly affects the downstream processes [22,7]. Hence, there have been difficulties in training and testing the VDU backbone. For example, since BROS [15] and LayoutLM [43] use different in-house OCRs, it has been difficult to make a fair comparison.

LayoutXLM [43] collects large-scale digital-born PDF data from the world wide web and extracts text annotations from the PDF via an open-source PDF renderer.[7] Although this showed another promising direction, it is not easy to follow the pipeline in practice. A practitioner has to collect the PDF data as there is no publicly available dataset ([43] did not open-source the dataset).

Moreover, since PDF files cannot easily be editable, augmentation is limited. Unlike the existing approach, Webvicob can efficiently modify and augment data (i.e., layout, background image) with javascript as Webvicob renders HTML directly. Moreover, Webvicob can easily be incorporated with the HTML dumps (e.g., Wikipedia dumps[8]), which are easily accessible and have been widely used in building powerful NLP backbones [9].

[7] `https://github.com/pymupdf/PyMuPDF`

[8] `https://dumps.wikimedia.org`

3 Web-based Visual Corpus Builder

Webvicob uses HTML dumps and modifies Document Object Model (DOM) to generate data for pretraining VDU backbones with rich corpora. As seen in Figure 2, Webvicob provides box annotations of characters, words, LaTeX [34], images, lines, and paragraphs, and also for images and LaTeXs.

In this section, we explain the generation procedures (Algorithm 1) in detail.

Algorithm 1 Get annotations from HTML

Input: `html`
Output: `image, annotations`

procedure `get_annotations_from_html(html)`
1: `html` ← `add_spans(html)` {3.1}
   // From `<p>ab</p>` to `<p><span>a</span><span>b</span></p>`
2: `driver` ← `get_selenium_driver(html)`
3: `remove_unusable_elements(driver)` {3.1}
4: `change_paragraph_fonts(driver)` {3.2}
5: `image` ← `capture(driver)`
6: `boxes` ← `get_glyph_box(driver)` {3.2}
7: `annotations` ← `get_annots(boxes)` {3.1}
   return `image, annotations`

3.1 Annotation

Adding Spans We can access each Element[9] using the DOM and find out the bounding box (bbox) of the Element through the `getBoundingClientRect()`[10] function. Webvicob modifies the HTML so that all characters can be bounded with a tag to get the bbox of each character (i.e., from `<p>abc</p>` to `<p><span>a</span><span>b</span><span>c</span></p>`). With this procedure, the `getBoundingClientRect()` function can be applied to all spans, allowing us to obtain bounding box annotations for all characters.

Remove Unusable Elements The essential step before creating data is to remove unusable Elements. For example, there are various ways to hide a specific Element in a web document. Even if the Element is invisible, the function `getBoundingClientRect()` returns results since Element occupies space. Specifically, the following Elements are removed:

- Pseudo Elements.[11]

[9] https://developer.mozilla.org/en-US/docs/Web/API/Element
[10] https://developer.mozilla.org/en-US/docs/Web/API/Element/getBoundingClientRect
[11] https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements

- Child Elements whose Element size is larger than the parent node.
- Elements that invisible style applied.
  (display: none / visibility: hidden / visibility: collapse / opacity: 0)
- Elements located outside the rendering screen.
- Element with placeholder attribute.

Construct Annotations We construct word annotations and line annotations by calculating the spaces between the character boxes and the line boxes. We define LaTex Elements with "mwe-math-fallback-image-inline" className, and define image Elements with {image, canvas, SVG, video} tags. As MarkupLM [21] did, paragraph annotations are extracted using a well-ordered tree structure. Elements with the same depth are grouped into one paragraph.

3.2 Rendering with Various Fonts

Random Paragraph Fonts Visual diversity in pretraining datasets is generally associated with improved performance of VDU backbones [19,12,45,22].

For visual diversity, Webvicob renders HTML with various fonts for each paragraph (See Figure 4a and 4b). We randomly select fonts from 3,567 Google-Fonts[12] in our experiments and analyses.

(a) (b) (c)

Fig. 4: **Visualization of rendered images with various fonts.** (a) Original font with `getBoundingClientRect()` results. (b) Random font with `getBoundingClientRect()` results. (c) Random font with actual glyph boxes.

Precise Bounding Boxes The actual bounding box of the glyph and the result of `getBoundingClientRect()` are different. As shown in Figure 4a and 4b, the result of `getBoundingClientRect()` has a large margin. We extract a ratio of the actual glyph box to the bounding box via rendering vector images in a font file with a Pygame FreeType handler.[13] Using the ratio, the final tight glyph box can be obtained (Figure 4c).

[12] `https://fonts.google.com`
[13] `https://www.pygame.org/docs/ref/freetype.html`

4 Experiment and Analysis

4.1 Setup

Donut experiments are carried out using 8 NVIDIA V100 GPUs for a fair comparison with $\text{Donut}_{\text{Proto}}$, while other experiments are conducted using 4 NVIDIA A100 80G GPUs. We use mixed precision training technique [33].

Employed Models To show the effectiveness of Webvicob, We use three models with different properties.

BROS [15] was proposed with an effective pretraining method (i.e., area masking) and a relative positional encoding. To validate the effectiveness of Webvicob-generated data, we pretrain $\text{BROS}_{\text{BASE}}$ and measure performance on DocVQA Task 1 [39] and Task 3 [32].

LayoutXLM [43] is a widely-used multilingual VDU backbone. We re-implement and pretrain $\text{LayoutXLM}_{\text{BASE}}$ for eight languages to validate Webvicob in a multilingual scenario. FUNSD [20] and XFUND [43] datasets are used as benchmark datasets.

Donut [22] introduced a novel approach that utilizes images alone without relying on OCR results as input. We pretrain $\text{Donut}_{\text{Proto}}$ using data generated by Webvicob to demonstrate the versatility of the data for different architectures. This has been verified through experiments on DocVQA Task 1, DocVQA Task 3, FUNSD, and XFUND (Japanese) datasets.

Donut for Entity Recognition Since Donut only takes an image input and cannot utilize the popular method of token classification using OCR information, we trained the model to decode sequences in a specific format (as depicted in Figure 5) in order to extract all entity fields within the document.

To assess the model's ability to identify the entity set, regardless of the order, we used Hungarian Matching [25] to adjust the sequence so that the Tree Edit Distance (TED) is minimized. The final evaluation score was based on TED-based accuracy [47,18,48,22].

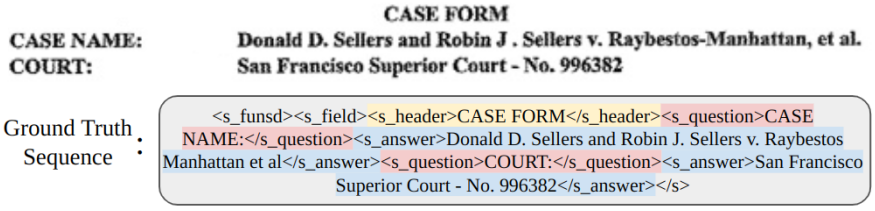


Fig. 5: **A sample of ground truth for Donut Entity Recognition.**

[14] https://en.wikipedia.org/wiki/List_of_Wikipedias

Table 1: **Statistics of Webvicob and conventional datasets.** ISO 639-1 language code is used for Language Support. Webvicob† is used in our multilingual experiments (Section 4.2). Webvicob‡ is a virtual dataset that can be constructed with Webvicob. It contains more than 60M samples with rich annotations and 270+ language supports. The total number of Wikipedia articles (60M) can be found on the Wikipedia statistic webpage.[14]

Dataset	#Image			Annotation Level				Language Supports
	Train	Val	Test	Char	Word	Line	Para	
MSRA-TD500 [44]	300	0	200			✓		EN
IC15 [13]	1,000	0	500		✓			EN
CTW-1500 [46]	1,000	0	500			✓		EN
IC17 MLT [36]	7,200	1,800	9,000		✓			EN ZH JA KO BN AR
IC19 MLT [35]	10,000	0	10,000		✓			EN ZH JA KO BN AR HI
IC19 LSVT [38]	30,000	0	20,000			✓		EN ZH
IC19 ArT [4]	5,603	0	4,563		✓			EN ZH
Total-Text [5]	1,255	0	300		✓			EN ZH
TextOCR [37]	21,778	3,124	3,232		✓			EN
IntelOCR [24]	191,059	16,731	0		✓			EN
HierText [28]	8,281	1,724	1,634		✓	✓	✓	EN
SynthText [12]	858,750	0	0	✓	✓			EN
IIT-CDIP [27]	11,434,146	0	0					EN
OCR-IDL [2]	26,600,964	0	0		✓	✓		EN
Webvicob†	18,584,173	4000	4000	✓	✓	✓	✓	EN ZH JA ES FR PT IT DE
Webvicob‡	60,475,636			✓	✓	✓	✓	270+ languages

Pretraining In pretraining $\text{BROS}_{\text{BASE}}$, we used 6.4 million English samples generated by Webvicob and trained the model for 5 epochs. The training procedure involved randomly selecting 512 consecutive tokens from each document, similar to LayoutLMv2 [41]. The remaining training hyperparameters were kept the same as $\text{BROS}_{\text{BASE}}$.

As can be seen in Table 1 and Table 2, we pretrained $\text{LayoutXLM}_{\text{BASE}}$ using 18.6 million multilingual data generated by Webvicob, which included eight languages: English, Chinese, Japanese, Spanish, French, Portuguese, Italian, and German. The model was pretrained for 5 epochs with a batch size of 64, following the procedure described in LayoutXLM. In line with the multilingual sampling strategy [6,3,43], each batch was sampled based on the language's frequency of occurrence in the data, with the probability $p_l \propto (\frac{N_l}{N})^\alpha$, where N is the total number of data and N_l is the number of data for language l. The parameter α was set to 0.7 to account for imbalanced data distribution.

The open-sourced $\text{Donut}_{\text{Proto}}$ was pretrained with 1.2 million samples generated by SynthDoG, including 400,000 samples each for Korean, Japanese, and English. The pretraining was performed for a duration of 5 days using 8 NVIDIA V100 GPUs, totaling 40 GPU days, with a batch size of 8. A similar approach was taken, where 400,000 samples each for Korean, Japanese, and English were generated by Webvicob and $\text{Donut}_{\text{Proto}}$ was trained for 3 days on 8 NVIDIA

Table 2: **Number of samples per language** in our multilingual data. ISO 639-1 language code is denoted as "Code".

Language (Code)	#Train	#Val	#Test
English (EN)	6,403,095	500	500
Chinese (ZH)	1,248,720	500	500
Japanese (JA)	1,293,628	500	500
Spanish (ES)	1,712,046	500	500
French (FR)	2,409,552	500	500
Portuguese (PT)	1,087,824	500	500
Italian (IT)	1,747,412	500	500
German (DE)	2,681,896	500	500
Total	18,584,173	4,000	4,000

V100 GPUs (24 GPU days) with a batch size of 16. The images are cropped and then resized to a size of 2,048 in width and 1,536 in height for training purposes.

Finetuning We finetune BROS with DocVQA datasets for 16K iterations. 64 batch size, Adam [23] optimizer, 5e-5 learning rate, and cosine annealing learning rate scheduler [29] are adapted.

Since the exact finetuning schedule has not been disclosed, we set up a "rough" finetuning schedule and compared results. For XFUND Semantic Entity Recognition (SER), we finetune LayoutXLM with 10K iterations, 64 batch-size, AdamW [30] optimizer, 1e-4 learning rate, linear decay learning rate, and warmup [11] learning rate for 10% of total iterations. We use unilm library[15] for LayoutXLM finetuning experiments.

$\text{Donut}_{\text{Proto}}$ was finetuned for 300 epochs for the DocVQA tasks with an image size of 2,560 width and 2,048 height. For FUNSD and XFUND tasks, we finetuned the model for 2,000 epochs using images with a width of 2,048 and a height of 1,536. For all downstream tasks, 8 batch size, Adam [23] optimizer, 1.5e-5 learning rate, and cosine annealing learning rate scheduler are adapted.

4.2 Experimental Results

BROS The corpus of DocVQA Task 1 and IIT-CDIP are very similar, while Webvicob-generated data has more various corpus (will be shown in Section 5.3).

The results also reflect this pattern. The results of Table 3 are categorized based on the quantity of data used in pretraining. Because of the significant amount of data specific to a particular domain within IIT-CDIP, it is possible to build a domain-specialized model. On the other hand, the extensive diversity of domains represented in Webvicob-generated data resulted in the development of a model that excels in the DocVQA Task 3.

[15] https://github.com/microsoft/unilm

Table 3: **Quantitative comparison in DocVQA tasks** with $\text{BROS}_{\text{BASE}}$ according to various settings of pretraining corpus. For an apples-to-apples comparison, we control the amount of pretraining data in experiments. The score is measured with Average Normalized Levenshtein Similarity (ANLS).

Model	IIT-CDIP	Webvicob	Task 1	Task 3
BROS	-	-	62.98	25.22
BROS	500K	-	65.01	25.12
BROS	-	500K	64.62	26.20
BROS	250K	250K	65.56	26.56
BROS	1M	-	65.27	26.31
BROS	-	1M	64.92	28.09
BROS	500K	500K	65.67	26.51
BROS	11M	-	68.07	24.76
BROS	-	6.4M	65.63	27.85

Table 4: **Multitask finetuning Accuracy (F1) for Semantic Entity Recognition (SER) with FUNSD and XFUND datasets.** We perform multitask finetuning $\text{LayoutXLM}_{\text{BASE}}$ under same setting with eight languages. Despite using only approximately 60% of the training data and iterations, the ultimate mean performance being competitive.

Model	FUNSD	ZH	JA	ES	FR	IT	DE	PT	Avg.
LayoutXLM (PDF 22M + IIT 8M)	0.785	0.897	0.787	0.744	0.791	0.811	0.832	0.824	0.785
LayoutXLM (Webvicob 18.6M)	0.727	0.893	0.794	0.686	0.749	0.765	0.760	0.764	0.767

The use of IIT-CDIP data for pretraining resulted in favorable performance on DocVQA Task 1. Conversely, when Webvicob data was used, DocVQA Task 3 performed much better. Surprisingly, the performance of Task 3 using Webvicob 1M has better performance than BROS using IIT-CDIP 11M. When a combination of both IIT-CDIP and Webvicob-generated data were used in equal portions for pretraining, there was a moderate enhancement in performance for both tasks.

LayoutXLM We report the f1 scores of Semantic Entity Recognition (SER) for FUNSD and XFUND datasets. Table 4 shows f1 scores of multitask finetuning with eight languages. Under the same finetuning setup, Webvicob data shows comparable performance. Please note that the total number of pretraining iterations of LayoutXLM (Webvicob) is $\sim$1.45M, much smaller than the iteration of LayoutXLM (PDF 22M + IIT 8M), which is $\sim$2.34M.

Table 5: **Finetuning results of Donut$_{\text{Proto}}$.** The XFUND (JA) dataset is denoted as "JA", while DocVQA Task 1 and DocVQA Task 3 are denoted as "Task 1" and "Task 3" respectively. The evaluation of the Entity Recognition task in FUNSD and XFUND was conducted using TED-based accuracy. ANLS scores are used for the DocVQA tasks. Despite utilizing only 60% of GPU days, the Donut$_{\text{Proto}}$ model trained on Webvicob-generated data exhibited better performance than the Donut$_{\text{Proto}}$ model trained on SynthDoG-generated data.

Model	FUNSD	JA	Task 1	Task 3
Donut (Not Pretrained)	0.1390	0.1942	0.1391	0.0877
Donut (SynthDoG 1.2M)	0.6059	0.7817	0.4477	0.1018
Donut (Webvicob 1.2M)	0.7397	0.8455	0.5607	0.1842
Donut (SynthDoG 0.6M + Webvicob 0.6M)	0.7576	0.8488	0.5622	0.2039
Donut (SynthDoG 1.2M + Webvicob 1.2M)	0.7738	0.8611	0.5636	0.2165

Donut We report finetuning results of Donut$_{\text{Proto}}$ on Table 5. Donut$_{\text{Proto}}$ trained using Webvicob data has achieved superior performance on all four tasks, despite 60% GPU training days compared to the Donut$_{\text{Proto}}$ trained with SynthDoG-generated data. While SynthDoG also made use of the Wikipedia corpus, the outstanding performance of Webvicob-generated data is noteworthy. We speculate that there are two reasons for this. First, the presence of text and images that are contextually relevant in Webvicob helps to accurately reflect the relationship between images and text. Second, the use of various types of real documents, such as tables, in the learning process can implicitly improve the learning of semantic information, such as key-value pairing.

5 Discussion

5.1 Scale-up using CommonCrawl

Our initial goal was to handle tons of HTML data, such as CommonCrawl dataset[16] which consists of petabytes of data.

However, the massive variety in the HTML format made it difficult to design an integrated solution. Currently, Webvicob is only specialized in a Wikipedia format. Expanding the scope could be future work.

5.2 Webvicob with Augraphy

We expect Webvicob to easily be integrated with any augmentation project, such as Augraphy [31], to boost the performance further. As can be seen in Figure 6, combining Webvicob and Augraphy can generate document images with rich visual effects.

[16] https://commoncrawl.github.io/cc-crawl-statistics/plots/crawlsize

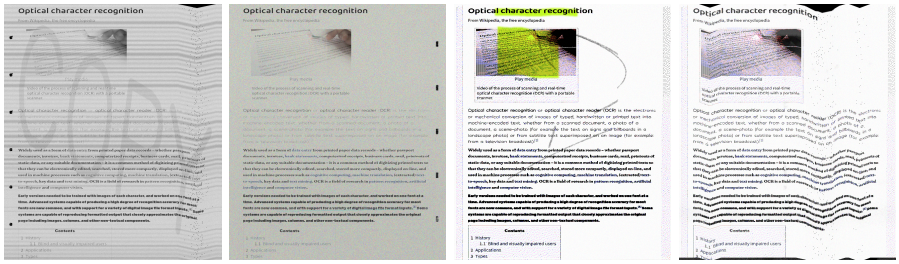


Fig. 6: **Webvicob with Augraphy.** Augraphy has the capability to produce a pencil sketch or folding appearance.

5.3 PCA analysis

We sampled 3,626 data points from each of the four datasets (Webvicob, IIT-CDIP, DocVQA Task 1, and DocVQA Task 3), creating a total of 14,504 vectors for PCA analysis. We constructed a vocabulary of 100,000 words using Wikipedia 1M data and represented the data using Bag of Words (BOW) representation, excluding stop words[17]. The figures [7, 8] in the i^{th} row display the analysis between the i^{th} and $(i+1)^{th}$ principal components. For example, the first row shows the plot of the 0^{th} and 1^{st} principal components.

We visualized the data using 10 principal components and found that the BOW vectors of IIT-CDIP and DocVQA Task 1 are very similar, while Webvicob contains a diverse range of BOW vectors.

Further visualization with Google vocabulary[18] can be found in figures [9, 10].

6 Conclusion

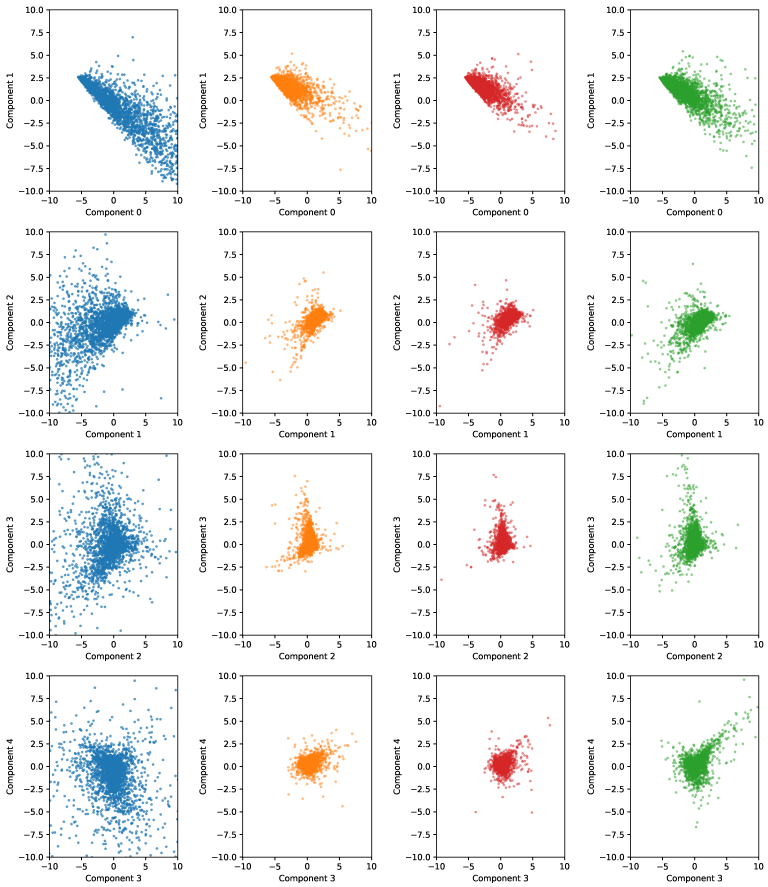
In this work, we propose an engine, Webvicob, that builds visual corpora from web resources. The constructed visual corpora can be utilized in building VDU backbones. In our experiments and analyses, we observe that the Webvicob-generated data helps the VDU backbone perform robustly across a variety of document formats and domains.

7 Acknowledgements

The authors thank NAVER CLOVA Text Vision Team and Information Extraction Team.

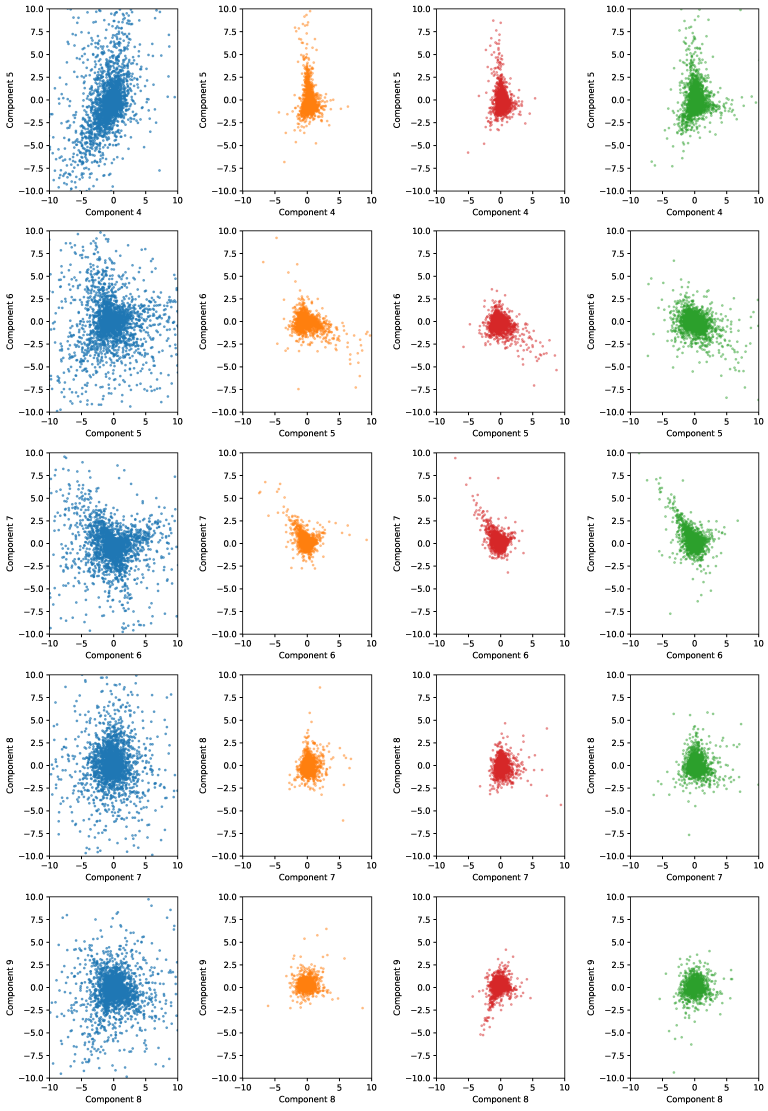
[17] `https://www.nltk.org/index.html`

[18] `https://github.com/first20hours/google-10000-english/blob/master/google-10000-english-no-swears.txt`



(a) Webvicob (b) IIT-CDIP (c) DocVQA Task 1 (d) DocVQA Task 3

Fig. 7: **Visualization of PCA Results.** Components 0, 1, 2, 3, 4



(a) Webvicob (b) IIT-CDIP (c) DocVQA Task 1 (d) DocVQA Task 3

Fig. 8: **Visualization of PCA Results.** Components 4, 5, 6, 7, 8, 9

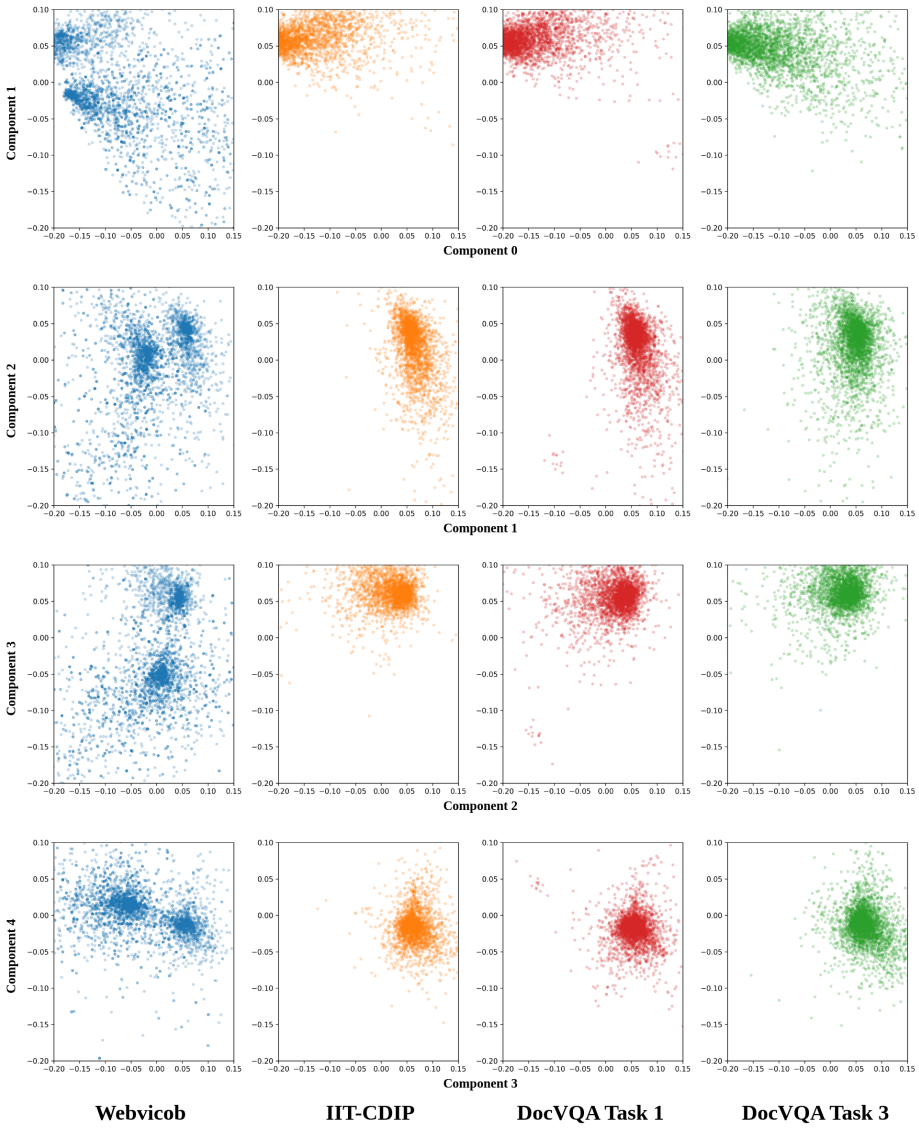


Fig. 9: **Visualization of PCA Results with Google vocabulary.** Components 0, 1, 2, 3, 4

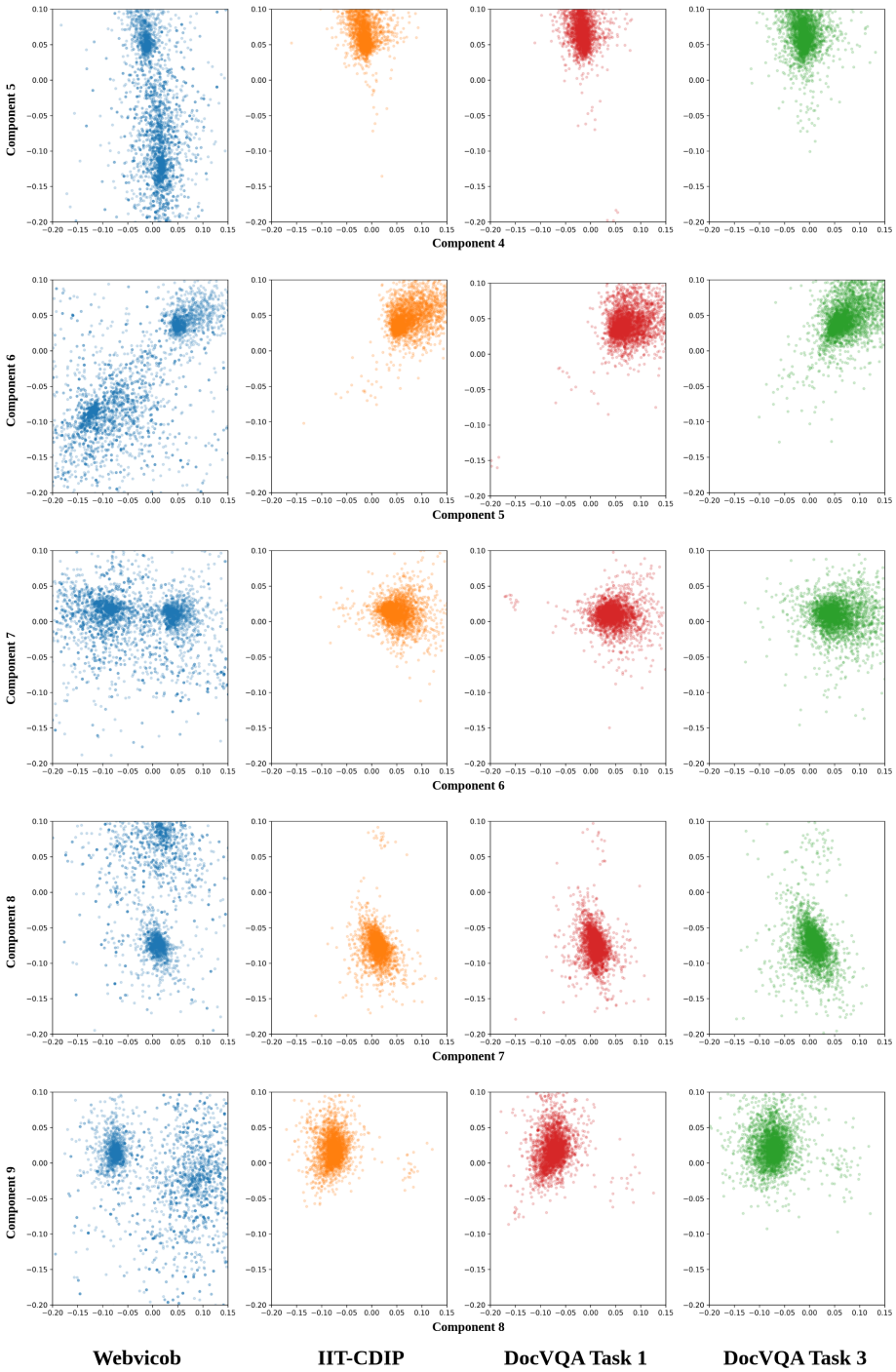


Fig. 10: **Visualization of PCA Results with Google vocabulary.** Components 4, 5, 6, 7, 8, 9

References

1. Attardi, G.: WikiExtractor. https://github.com/attardi/wikiextractor (2015)
2. Biten, A.F., Tito, R., Gomez, L., Valveny, E., Karatzas, D.: Ocr-idl: Ocr annotations for industry document library dataset. In: European Conference on Computer Vision Workshop (ECCV Workshop) (2022)
3. Chi, Z., Dong, L., Wei, F., Yang, N., Singhal, S., Wang, W., Song, X., Mao, X.L., Huang, H., Zhou, M.: InfoXLM: An information-theoretic framework for cross-lingual language model pre-training. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) (2021)
4. Chng, C.K., Ding, E., Liu, J., Karatzas, D., Chan, C.S., Jin, L., Liu, Y., Sun, Y., Ng, C.C., Luo, C., Fang, C., Zhang, S., Han, J.: ICDAR2019 robust reading challenge on arbitrary-shaped text - rrc-art. In: International Conference on Document Analysis and Recognition (ICDAR) (2019)
5. Ch'ng, C.K., Chan, C.S., Liu, C.: Total-Text: Towards orientation robustness in scene text detection. International Journal on Document Analysis and Recognition (IJDAR) (2020)
6. Conneau, A., Lample, G.: Crosslingual language model pretraining. In: In Advances in Neural Information Processing Systems (NeurIPS) (2019)
7. Davis, B., Morse, B., Price, B., Tensmeyer, C., Wigington, C., Morariu, V.: End-to-end document recognition and understanding with dessurt. In: European Conference on Computer Vision (ECCV) (2022)
8. Deng, X., Shiralkar, P., Lockard, C., Huang, B., Sun, H.: Dom-lm: Learning generalizable representations for html documents (2022), https://arxiv.org/abs/2201.10608
9. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) (2019)
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (ICLR) (2021)
11. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch SGD: Training imagenet in 1 hour (2017), https://arxiv.org/abs/1706.02677
12. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
13. Harley, A.W., Ufkes, A., Derpanis, K.G.: Evaluation of deep convolutional nets for document image classification and retrieval. In: International Conference on Document Analysis and Recognition (ICDAR) (2015)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
15. Hong, T., Kim, D., Ji, M., Hwang, W., Nam, D., Park, S.: BROS: A pre-trained language model for understanding texts in document. In: AAAI Conference on Artificial Intelligence (AAAI) (2022)
16. Huang, Y., Lv, T., Cui, L., Lu, Y., Wei, F.: LayoutLMv3: Pre-training for document ai with unified text and image masking. In: ACM International Conference on Multimedia (ACM MM) (2022)

17. Hwang, W., Kim, S., Yim, J., Seo, M., Park, S., Park, S., Lee, J., Lee, B., Lee, H.: Post-ocr parsing: building simple and robust parser via bio tagging. In: Workshop on Document Intelligence at NeurIPS (NeurIPS Workshop) (2019)
18. Hwang, W., Lee, H., Yim, J., Kim, G., Seo, M.: Cost-effective end-to-end information extraction for semi-structured document images. In: Empirical Methods in Natural Language Processing (EMNLP) (2021)
19. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. In: International Conference on Neural Information Processing Systems Workshop (NIPS Workshop) (2014)
20. Jaume, G., Ekenel, H.K., Thiran, J.P.: Funsd: A dataset for form understanding in noisy scanned documents. In: ICDAR Workshop on Open Services and Tools for Document Analysis (ICDAR-OST) (2019)
21. Junlong, L., Xu, Y., Cui, L., Wei, F.: Markuplm: Pre-training of text and markup language for visually rich document understanding. In: Annual Meeting of the Association for Computational Linguistics (ACL) (2022)
22. Kim, G., Hong, T., Yim, M., Park, J., Yim, J., Hwang, W., Yun, S., Han, D., Park, S.: Donut: Document understanding transformer without ocr. In: European Conference on Computer Vision (ECCV) (2022)
23. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
24. Krylov, I., Nosov, S., Sovrasov, V.: Open images v5 text annotation and yet another mask text spotter. In: Asian Conference on Machine Learning (ACML) (2021)
25. Kuhn, H.W.: The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly (NRLQ) (1955)
26. Lee, K., Joshi, M., Turc, I., Hu, H., Liu, F., Eisenschlos, J., Khandelwal, U., Shaw, P., Chang, M.W., Toutanova, K.: Pix2struct: Screenshot parsing as pretraining for visual language understanding (2022), https://arxiv.org/abs/2210.03347
27. Lewis, D., Agam, G., Argamon, S., Frieder, O., Grossman, D., Heard, J.: Building a test collection for complex document information processing. In: International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR) (2006)
28. Long, S., Qin, S., Panteleev, D., Bissacco, A., Fujii, Y., Raptis, M.: Towards end-to-end unified scene text detection and layout analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
29. Loshchilov, I., Hutter, F.: SGDR: Stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (ICLR) (2017)
30. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (ICLR) (2019)
31. Maini, S., Groleau, A., Chee, K.W., Larson, S., Boarman, J.: Augraphy: A data augmentation library for document images (2022), https://arxiv.org/abs/2208.14558
32. Mathew, M., Bagal, V., Tito, R., Karatzas, D., Valveny, E., Jawahar, C.: InfographicVQA. In: IEEE Winter Conference on Applications of Computer Vision (WACV) (2022)
33. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaev, O., Venkatesh, G., Wu, H.: Mixed precision training. In: International Conference on Learning Representations (ICLR) (2018)
34. Mittelbach, F., Schöpf, R.: With latex into the nineties. TUGboat (1989)
35. Nayef, N., Liu, C.L., Ogier, J.M., Patel, Y., Busta, M., Chowdhury, P., Karatzas, D., Khlif, W., Matas, J., Pal, U., Burie, J.C.: ICDAR2019 robust reading chal-

lenge on multi-lingual scene text detection and recognition — rrc-mlt-2019. In: International Conference on Document Analysis and Recognition (ICDAR) (2019)
36. Nayef, N., Yin, F., Bizid, I., Choi, H., Feng, Y., Karatzas, D., Luo, Z., Pal, U., Rigaud, C., Chazalon, J., Khlif, W., Luqman, M., Burie, J.C., Liu, C.L., Ogier, J.M.: ICDAR2017 robust reading challenge on multi-lingual scene text detection and script identification - rrc-mlt. In: International Conference on Document Analysis and Recognition (ICDAR) (2017)
37. Singh, A., Pang, G., Toh, M., Huang, J., Galuba, W., Hassner, T.: TextOCR: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
38. Sun, Y., Karatzas, D., Chan, C.S., Jin, L., Chng, C.K., Liu, Y., Luo, C., Ng, C.C., Han, J., Ding, E., Liu, J.: ICDAR 2019 competition on large-scale street view text with partial labeling - rrc-lsvt. In: International Conference on Document Analysis and Recognition (ICDAR) (2019)
39. Tito, R., Mathew, M., Jawahar, C.V., Valveny, E., Karatzas, D.: ICDAR 2021 competition on document visual question answering. In: International Conference on Document Analysis and Recognition (ICDAR) (2021)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: International Conference on Neural Information Processing Systems (NIPS) (2017)
41. Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florencio, D., Zhang, C., Che, W., et al.: LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In: Annual Meeting of the Association for Computational Linguistics (ACL) (2021)
42. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M.: LayoutLM: Pre-training of text and layout for document image understanding. In: Knowledge Discovery and Data Mining (KDD) (2019)
43. Xu, Y., Lv, T., Cui, L., Wang, G., Lu, Y., Florencio, D., Zhang, C., Wei, F.: LayoutXLM: Multimodal pre-training for multilingual visually-rich document understanding (2021), https://arxiv.org/abs/2104.08836
44. Yao, C., Bai, X., Liu, W., Ma, Y., Tu, Z.: Detecting texts of arbitrary orientations in natural images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
45. Yim, M., Kim, Y., Cho, H.C., Park, S.: SynthTIGER: Synthetic text image generator towards better text recognition models. In: International Conference on Document Analysis and Recognition (ICDAR) (2021)
46. Yuliang, L., Lianwen, J., Zhang, S., Sheng, Z.: Detecting curve text in the wild: New dataset and new solution (2017), https://arxiv.org/abs/1712.02170
47. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing (SICOMP) (1989)
48. Zhong, X., ShafieiBavani, E., Jimeno Yepes, A.: Image-based table recognition: Data, model, and evaluation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) European Conference on Computer Vision (ECCV) (2020)