Supplementary Material for OCR-free Document Understanding Transformer

Geewook Kim[1*], Teakgyu Hong[4†], Moonbin Yim[2†], JeongYeon Nam[1], Jinyoung Park[5†], Jinyeong Yim[6†], Wonseok Hwang[7†], Sangdoo Yun[3], Dongyoon Han[3], and Seunghyun Park[1]

[1]NAVER CLOVA [2]NAVER Search [3]NAVER AI Lab
[4]Upstage [5]Tmax [6]Google [7]LBox

Table of Contents

A Appendix

A.1 Details of OCR Engines (MS, CLOVA, Easy, Paddle)

Current state-of-the-art visual document understanding (VDU) backbones, such as BROS [6], LayoutLM [17] and LayoutLMv2 [16], are dependent on off-the-shelf OCR engines. These backbones take the output of OCR as their (one of) input features. For the OCR-dependent methods, in our experiments, we use state-of-the-art OCR engines that are publicly available, including 2 OCR API products (i.e., MS OCR[1] and CLOVA OCR[2]) and 2 open-source OCR models (i.e., Easy OCR[3] and Paddle OCR[4]). In the main paper, Paddle OCR is used for the Chinese train ticket dataset [5] and CLOVA OCR is used for the rest datasets in the document information extraction (IE) tasks. MS OCR is used to measure the running time of the LayoutLM family in document classification and visual question answering (VQA) tasks, following the previous work of Xu et al. [16]. Each OCR engine is explained in the following.

* Corresponding author: gwkim.rsrch@gmail.com

† This work was done while the authors were at NAVER CLOVA.

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

[2] https://clova.ai/ocr/en.

[3] https://github.com/JaidedAI/EasyOCR.

[4] https://github.com/PaddlePaddle/PaddleOCR.

MS OCR MS OCR[1] is the latest OCR API product from Microsoft and used in several recent VDU methods, e.g., LayoutLMv2 [16]. This engine supports 164 languages for printed text and 9 languages for handwritten text (until 2022/03).

CLOVA OCR CLOVA OCR[2] is an API product from NAVER CLOVA and is specialized in document IE tasks. This engine supports English, Japanese and Korean (until 2022/03). In the ablation experiments on the CORD dataset [11] (Figure 9 in the main paper), the CLOVA OCR achieved the best accuracy.

Easy OCR Easy OCR[3] is a ready-to-use OCR engine that is publicly available at GitHub. This engine supports more than 80 languages (until 2022/03). Unlike the aforementioned two OCR products (i.e., MS OCR and CLOVA OCR), this engine is publicly opened and downloadable.[3] The entire model architecture is based on the modern deep-learning-based OCR modules [2,1] with some modifications to make the model lighter and faster. The total number of model parameters is 27M which is small compared to the state-of-the-art models [2,1].

Paddle OCR Paddle OCR[4] is an open-source OCR engine available at GitHub. We used a lightweight (i.e., mobile) version of the model which is specially designed for a fast and light OCR of English and Chinese texts. The model is served on a CPU environment and the size of the model is extremely small, which is approximately 10M.

A.2 Details of Synthetic Document Generator (SynthDoG)

In this section, we explain the components of the proposed Synthetic Document Generator (SynthDoG) in detail. The entire pipeline basically follows Yim et al. [19]. Our source code is available at `https://github.com/clovaai/donut`. More samples are shown in Figure A.

Background Background images are sampled from ImageNet [3]. Gaussian blur is randomly applied to the background image to represent out-of-focus effects.

Document Paper textures are sampled from the photos that we collected. The texture is applied to an white background. In order to make the texture realistic, random elastic distortion and Gaussian noise are applied. To represent various view angles in photographs, a random perspective transformation is applied to the image.

Text Layout and Pattern To mimic the layouts in real-world documents, a heuristic rule-based pattern generator is applied to the document image region to generate text regions. The main idea is to set multiple squared regions to represent text paragraphs. Each squared text region is then interpreted as multiple lines of text. The size of texts and text region margins are chosen randomly.

Fig. A. Examples of SynthDoG. English, Chinese, Japanese and Korean samples are shown (from top to bottom). Although the idea is simple, these synthetic samples play an important role in the pre-training of Donut. Please, see Figure 7 in the main paper for details

Text Content and Style We prepare the multi-lingual text corpora from Wikipedia.[5] We use Noto fonts[6] since it supports various languages. SynthDoG samples texts and fonts from these resources and the sampled texts are rendered in the regions that are generated by the layout pattern generator. The text colors are randomly assigned.

Post-processing Finally, some post-processing techniques are applied to the output image. In this process, the color, brightness, and contrast of the image are adjusted. In addition, shadow effect, motion blur, Gaussian blur, and JPEG compression are applied to the image.

[5] https://dumps.wikimedia.org.
[6] https://fonts.google.com/noto.

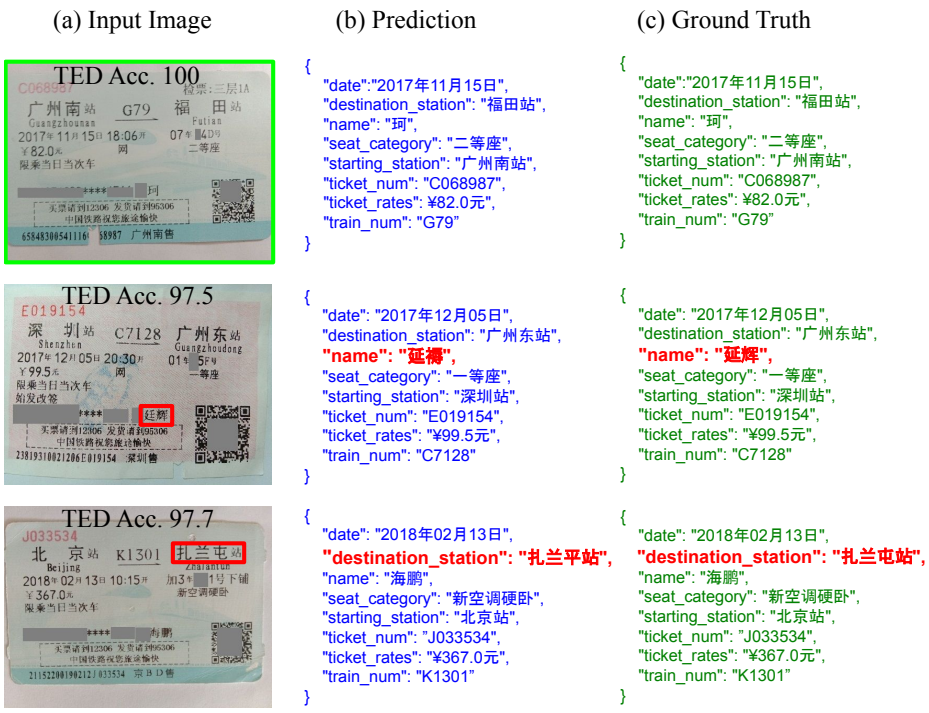

Fig. B. Examples of *Ticket* [5] with Donut predictions. There is no hierarchy in the structure of information (i.e., depth = 1) and the location of each key information is almost fixed. Failed predictions are marked and bolded (red)

A.3 Details of Document Information Extraction

Information Extraction (IE) on documents is an arduous task since it requires (a) reading texts, (b) understanding the meaning of the texts, and (c) predicting the relations and structures among the extracted information. Some previous works have only focused on extracting several pre-defined key information [5]. In that case, only (a) and (b) are required for IE models. We go beyond the previous works by considering (c) also. Although the task is complex, its interface (i.e., the format of input and output) is simple. In this section, for explanation purposes, we show some sample images (which are the raw input of the IE pipeline) with the output of Donut.

In the main paper, we test four datasets including two public benchmarks (i.e., *CORD* [11] and *Ticket* [5]) and two private industrial datasets (i.e., *Business Card* and *Receipt*). Figure B shows examples of *Ticket* with the outputs of Donut. Figure C shows examples of *CORD* with the outputs of Donut. Due to strict industrial policies on the private industrial datasets, we instead show some real-like high-quality samples of *Business Card* and *Receipt* in Figure D.

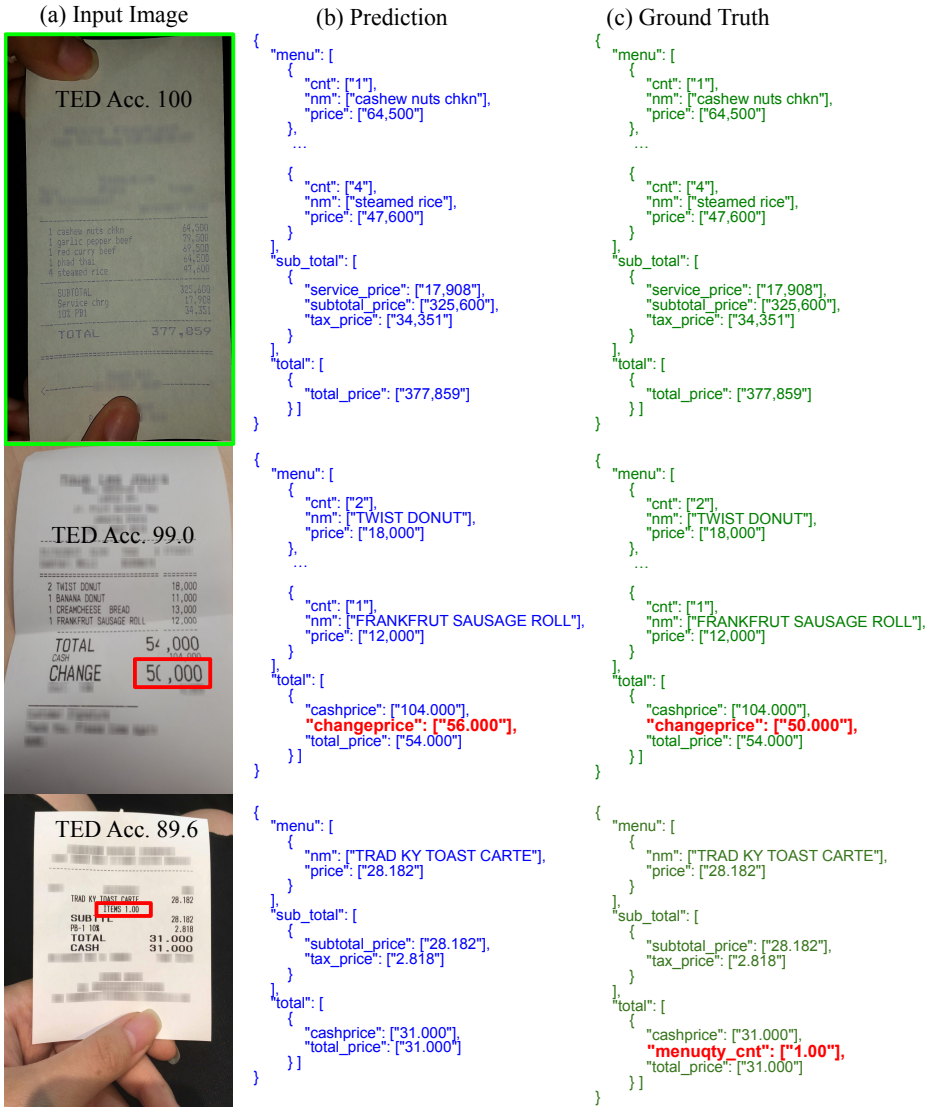

Fig. C. Examples of *CORD* [11] with Donut predictions. There is a hierarchy in the structure of information (i.e., depth = 2). **Donut** not only reads some important key information from the image, but also predicts the relationship among the extracted information (e.g., the name, price, and quantity of each menu item are grouped)

A.4 Details of Model Training Scheme and Output Format

In the model architecture and training objective, we basically followed the original Transformer [12], which uses a Transformer encoder-decoder architecture

Fig. D. Examples of *Business Card* (top) and *Receipt* (bottom). Due to strict industrial policies on the private industrial datasets from our active products, real-like high-quality samples are shown instead

and a teacher-forcing training scheme. The teacher-forcing scheme is a model training strategy that uses the ground truth as input instead of model output from a previous time step. Figure E shows a details of the model training scheme and decoder output format.

A.5 Implementation and Training Hyperparameters

The codebase and settings are available at GitHub[7]. We implement the entire model pipeline with Huggingface's `transformers`[8] [15] and an open-source library `TIMM` (PyTorch image models)[9] [14].

For all model training, we use a half-precision (fp16) training. We train Donut using Adam optimizer [10] by decreasing the learning rate as the training progresses. The initial learning rate of pre-training is set to 1e-4 and that of fine-tuning is selected from 1e-5 to 1e-4. We pre-train the model for 200K steps with 64 NVIDIA A100 GPUs and a mini-batch size of 196, which takes about 2-3 GPU days. We also apply a gradient clipping technique where a maximum gradient norm is selected from 0.05 to 1.0. The input resolution of Donut is set to 2560×1920 at the pre-training phase. In downstream tasks, the input resolutions are controlled. In some downstream document IE experiments, such

[7] https://github.com/clovaai/donut.
[8] https://github.com/huggingface/transformers.
[9] https://github.com/rwightman/pytorch-image-models.

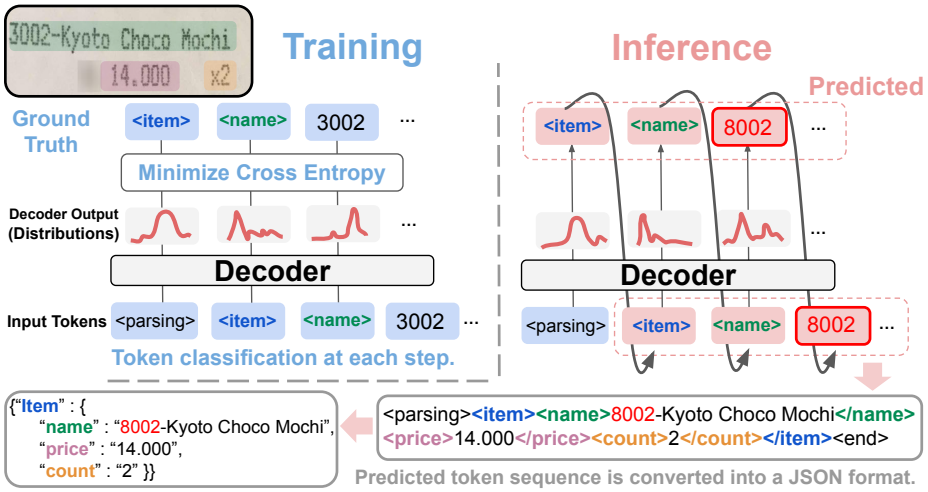

Fig. E. Donut training scheme with teacher forcing and decoder output format examples. The model is trained to minimize cross-entropy loss of the token classifications simultaneously. At inference, the predicted token from the last step is fed to the next

as, *CORD* [11], *Ticket* [5] and *Business Card*, smaller size of input resolution, e.g., 1280×960, is tested. With the 1280×960 setting, the model training cost of Donut was small. For example, the model fine-tuning on *CORD* or *Ticket* took approximately 0.5 hours with one A100 GPU. However, when we set the 2560×1920 setting for larger datasets, e.g., *RVL-CDIP* or *DocVQA*, the cost increased rapidly. With 64 A100 GPUs, *DocVQA* requires one GPU day and *RVL-CDIP* requires two GPU days approximately. This is not surprising in that increasing the input size for a precise result incurs higher computational costs in general. Using an efficient attention mechanism [13] may avoid the problem in architectural design, but we use the original Transformer [12] as we aim to present a simpler architecture in this work. Our preliminary experiments in smaller resources are available in Appendix A.6.

For the implementation of document IE baselines, we use the `transformers` library for BERT [4], BROS [6], LayoutLMv2 [16,18] and WYVERN [8]. For the SPADE [9] baseline, the official implementation[10] is used. The models are trained using NVIDIA P40, V100, or A100 GPUs. The major hyperparameters, such as initial learning rate and number of epochs, are adjusted by monitoring the scores on the validation set. The architectural details of the OCR-dependent VDU backbone baselines (e.g., LayoutLM and LayoutLMv2) are available in Appendix A.7.

[10] https://github.com/clovaai/spade.

A.6 Preliminary Experiments in Smaller Resources

In our preliminary experiments, we pre-trained Donut with smaller resources (denoted as $\text{Donut}_{\text{Proto}}$), i.e., smaller data (SynthDoG 1.2M) and fewer GPUs (8 V100 GPUs for 5 days). The input size was 2048×1536. In this setting, $\text{Donut}_{\text{Proto}}$ also achieved comparable results on *RVL-CDIP* and *CORD*. The accuracy on *RVL-CDIP* was 94.5 and *CORD* was 91.6. After the preliminaries, we have scaled the model training with more data.

A.7 Details of OCR-dependent Baseline Models

In this section, we provide a gentle introduction to the general-purpose VDU backbones, such as LayoutLM [17] and LayoutLMv2 [16]. To be specific, we explain how the conventional backbones perform downstream VDU tasks; document classification, IE, and VQA. Common to all tasks, the output of the OCR engine is used as input features of the backbone. That is, the extracted texts are sorted and converted to a sequence of text tokens. The sequence is passed to the Transformer encoder to get contextualized output vectors. The vectors are used to get the desired output. The difference in each task depends on a slight modification on the input sequence or on the utilization of the output vectors.

Document Classification At the start of the input sequence, a special token `[CLS]` is appended. The sequence is passed to the backbone to get the output vectors. With a linear mapping and softmax operation, the output vector of the special token `[CLS]` is used to get a *class-label* prediction.

Document IE With a linear mapping and softmax operation, the output vector sequence is converted to a *BIO-tag* sequence [7].

IE on 1-depth structured documents When there is no hierarchical structure in the document (See Figure B), the tag set is defined as {"B_k", "I_k", "O" | $k \in$ pre-defined keys}. "B_k" and "I_k" are tags that represent the beginning (B) and the inside (I) token of the key k respectively. The "O" tag indicates that the token belongs to no key information.

IE on n-depth structured documents When there are hierarchies in the structure (See Figure C), the BIO-tags are defined for each hierarchy level. In this section, we explain a case where the depth of structure is $n = 2$. The tag set is defined as {"$\text{B}_g.\text{B}_k$", "$\text{B}_g.\text{I}_k$", "$\text{I}_g.\text{B}_k$", "$\text{I}_g.\text{I}_k$", "O" | $g \in$ pre-defined parent keys, $k \in$ pre-defined child keys}. For instance, the Figure C shows an example where a parent key is "menu" and related child keys are {"cnt", "nm", "price"}. "B_g" represents that one group (i.e., a parent key such as "menu") starts, and "I_g" represents that the group is continuing. Separately from the BI tags of the parent key (i.e., "B_g" and "I_g"), the BI tags of each child key (i.e., "B_k" and "I_k") work the same as in the case of $n = 1$. This BIO-tagging method is also known as *Group BIO-tagging* and the details are also available in Hwang et al. [7].

Document VQA With a linear mapping and softmax operation, the output vector sequence is converted to a *span-tag* sequence. For the input token sequence, the model finds the beginning and the end of the answer span. Details can also be found in the Section 4.2 of Devlin et al. [4].

References

1. Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S.J., Lee, H.: What is wrong with scene text recognition model comparisons? dataset and model analysis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019) 2
2. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character region awareness for text detection. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9357–9366 (2019). https://doi.org/10.1109/CVPR.2019.00959 2
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 2
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423, https://aclanthology.org/N19-1423 7, 9
5. Guo, H., Qin, X., Liu, J., Han, J., Liu, J., Ding, E.: Eaten: Entity-aware attention for single shot visual text extraction. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 254–259 (2019). https://doi.org/10.1109/ICDAR.2019.00049 1, 4, 7
6. Hong, T., Kim, D., Ji, M., Hwang, W., Nam, D., Park, S.: Bros: A pre-trained language model focusing on text and layout for better key information extraction from documents. Proceedings of the AAAI Conference on Artificial Intelligence **36**(10), 10767–10775 (Jun 2022). https://doi.org/10.1609/aaai.v36i10.21322, https://ojs.aaai.org/index.php/AAAI/article/view/21322 1, 7
7. 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 2019 (2019) 8
8. Hwang, W., Lee, H., Yim, J., Kim, G., Seo, M.: Cost-effective end-to-end information extraction for semi-structured document images. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 3375–3383. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic (Nov 2021). https://doi.org/10.18653/v1/2021.emnlp-main.271, https://aclanthology.org/2021.emnlp-main.271 7
9. Hwang, W., Yim, J., Park, S., Yang, S., Seo, M.: Spatial dependency parsing for semi-structured document information extraction. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. pp. 330–343. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.findings-acl.28, https://aclanthology.org/2021.findings-acl.28 7
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980 6
11. Park, S., Shin, S., Lee, B., Lee, J., Surh, J., Seo, M., Lee, H.: Cord: A consolidated receipt dataset for post-ocr parsing. In: Workshop on Document Intelligence at NeurIPS 2019 (2019) 2, 4, 5, 7

12. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf 5, 7
13. Wang, S., Li, B., Khabsa, M., Fang, H., Ma, H.: Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768 (2020) 7
14. Wightman, R.: Pytorch image models. https://github.com/rwightman/pytorch-image-models (2019). https://doi.org/10.5281/zenodo.4414861 6
15. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.: 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. Association for Computational Linguistics, Online (Oct 2020). https://doi.org/10.18653/v1/2020.emnlp-demos.6, https://aclanthology.org/2020.emnlp-demos.6 6
16. Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florencio, D., Zhang, C., Che, W., Zhang, M., Zhou, L.: LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 2579–2591. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.acl-long.201, https://aclanthology.org/2021.acl-long.201 1, 2, 7, 8
17. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M.: Layoutlm: Pre-training of text and layout for document image understanding. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. p. 1192–1200. KDD '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3394486.3403172, https://doi.org/10.1145/3394486.3403172 1, 8
18. 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. arXiv preprint arXiv:2104.08836 (2021) 7
19. Yim, M., Kim, Y., Cho, H.C., Park, S.: Synthtiger: Synthetic text image generator towards better text recognition models. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) Document Analysis and Recognition – ICDAR 2021. pp. 109–124. Springer International Publishing, Cham (2021) 2