DocVisor: A Multi-purpose Web-Based Interactive Visualizer for Document Image Analytics

Khadiravana Belagavi, Pranav Tadimeti, and Ravi Kiran Sarvadevabhatla(✉)

Centre for Visual Information Technology, International Institute of Information Technology, Hyderabad 500032, India
ravi.kiran@iiit.ac.in
https://github.com/ihdia/docvisor

Abstract. The performance for many document-based problems (OCR, Document Layout Segmentation, etc.) is typically studied in terms of a single aggregate performance measure (Intersection-Over-Union, Character Error Rate, etc.). While useful, the aggregation is a trade-off between instance-level analysis of predictions which may shed better light on a particular approach's biases and performance characteristics. To enable a systematic understanding of instance-level predictions, we introduce DocVisor - a web-based multi-purpose visualization tool for analyzing the data and predictions related to various document image understanding problems. DocVisor provides support for visualizing data sorted using custom-specified performance metrics and display styles. It also supports the visualization of intermediate outputs (e.g., attention maps, coarse predictions) of the processing pipelines. This paper describes the appealing features of DocVisor and showcases its multi-purpose nature and general utility. We illustrate DocVisor's functionality for four popular document understanding tasks – document region layout segmentation, tabular data detection, weakly-supervised document region segmentation and optical character recognition. DocVisor is available as a documented public repository for use by the community.

1 Introduction

Tasks in document image analysis are evaluated using task-specific performance measures. For example, measures such as average Intersection-over-Union (IoU) or the Jaccard Index, mAP@x (mean Average Precision for an IoU threshold of x), and mean HD (Hausdorff distance) are used to evaluate semantic segmentation approaches [6,10]. Similarly, measures such as Character Error Rate (CER) and Word Error Rate (WER) are used to evaluate Optical Character Recognition (OCR) systems [5]. These measures are undoubtedly informative as aggregate performance measures. However, they are prone to statistical bias arising from the imbalanced distribution of performance measures - e.g., mean

E. H. Barney Smith and U. Pal (Eds.): ICDAR 2021 Workshops, LNCS 12917, pp. 206–219, 2021.
https://doi.org/10.1007/978-3-030-86159-9_14

averaging [11]. Therefore, it is helpful to have a mechanism that enables predictions to be visualized on a per-instance (image or region) basis after sorting them by the performance measure of interest. Doing so can be valuable in understanding factors contributing to performance or lack thereof. Sometimes, it may also be essential to visualize outputs of the intermediate stages in the systems' pipeline. For example, in an end-to-end OCR system, the prediction from a skew correction module is fed to a line segmenter module, which then produces an input to the recognizer module [15]. Similarly, in attention-based OCR systems, visualizing attention maps intermediately may help better understand the final prediction [25].

Moreover, multiple tasks may be of interest to different project groups working on the same document corpus. One group could be working on document layout segmentation, while another could be working on OCR. Instead of maintaining separate visualization mechanisms, it may be more efficient to develop and maintain a single interface that can be used concurrently by all project groups.

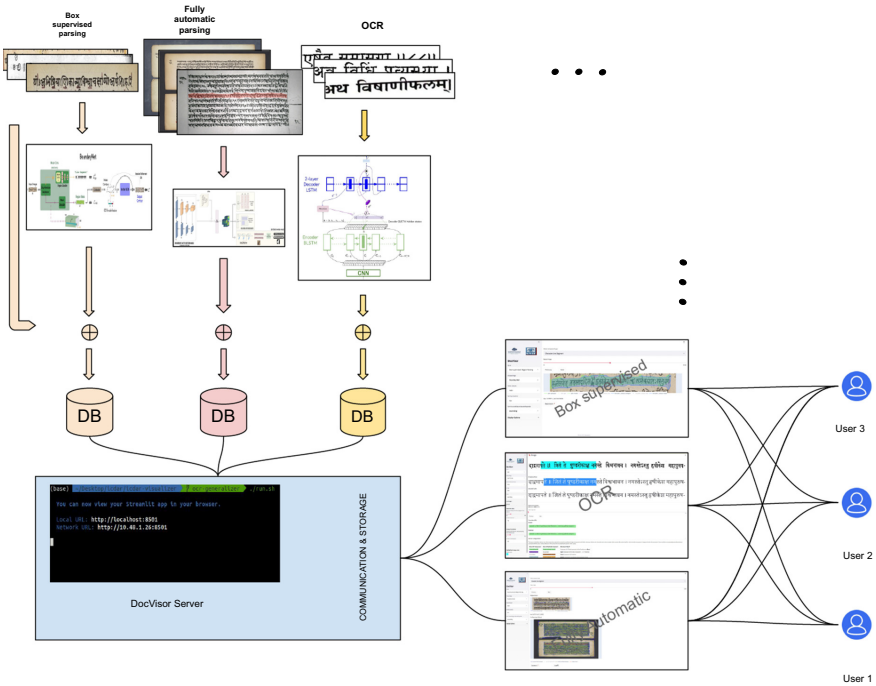


Fig. 1. Architecture of the DocVisor tool.

Motivated by the above requirements, we introduce DocVisor - a configurable web-based multi-purpose visualization tool for analyzing the data and predictions related to various document image understanding problems. Currently,

DocVisor supports four popular document understanding tasks and their variants – document region layout segmentation, tabular data detection, weakly-supervised document region segmentation, and optical character recognition. The tool is flexible and provides support for visualizing data sorted using custom-specified performance metrics and display styles.

Table 1. Salient attributes of popular visualization tools.

Methods	Released in	Metric-sorted	Allow comparison with g.t	Multi-model comparison	Task(s)	Web-based
dhSegment [17]	2018	×	✓	×	Layout	✓
PAGEViewer[a]	2014	×	Partially	×	Layout	×
OpenEvaluationTool [1]	2017	×	✓	×	Layout	✓
HInDoLA [23]	2019	×	×	×	Layout	✓
DocVisor	2021	✓	✓	✓	Multiple	✓

[a] https://www.primaresearch.org/tools/PAGEViewer.

The architecture of the DocVisor tool is depicted in Fig. 1. Model outputs for the three document-analysis tasks (Fully Automatic Region Parsing, Box Supervised Region Parsing and OCR) are pre-processed and given as an input to the DocVisor tool, which then renders an interactive multi-layout web-page. The tool and associated documentation can be accessed from https://github.com/ihdia/docvisor.

2 Related Work

Tools for document analysis and Optical Character Recognition (OCR) form an essential part of improving the related models. They are broadly classified into annotation tools, visualization tools, post-correction tools and evaluation tools. Prominent tools for layout parsing, image-based text search and historical manuscript transcription include HInDoLA[1], Aletheia[2], Monk[3] and Transkribus[4] respectively [4,13,20,23]. Kiessling et al. [14] propose eScriptorium for historical document analysis, which allows to segment and transcribe the manuscripts manually or automatically.

AnyOCR model is designed to be unsupervised by combining old segmentation-based methods with recent segmentation-free methods to reduce the annotation efforts [12]. It also includes error correction services like anyLayoutEdit and anyOCREdit [2]. OpenOCRCorrect[5] is designed to detect and correct OCR errors by exploiting multiple auxiliary sources [19]. It includes

[1] https://github.com/ihdia.
[2] www.primaresearch.org/tools/Aletheia.
[3] http://monkweb.nl/.
[4] https://readcoop.eu/transkribus/.
[5] https://tinyurl.com/5dd7dh6a.

features like error detection, auto-corrections, and task specific color coding to reduce the cognitive load. OCR4all[6] is designed with semi-automatic workflow for historical documents OCR.

Some of these tools naturally allow analysis of annotations and related metadata as a secondary feature. One work close to ours is open evaluation tool[7] by Alberti et al. [1], which allows for evaluation of document layout models at pixel level. Another is dhSegment [17] which enables predictions and ground-truth to be serially visualized using the Tensorboard functionality of the popular machine learning library, Tensorflow.

Unlike DocVisor, these tools omit certain important features such as interactive overlays, visualization at multiple levels of granularity, metric-sorted visualization of all test set predictions, simultaneous comparison of multiple model outputs etc. The few tools which provide a reasonable degree of functionality for visualization are summarized in Table 1.

We describe the functionalities of DocVisor in subsequent sections.

3 Fully Automatic Region Parsing

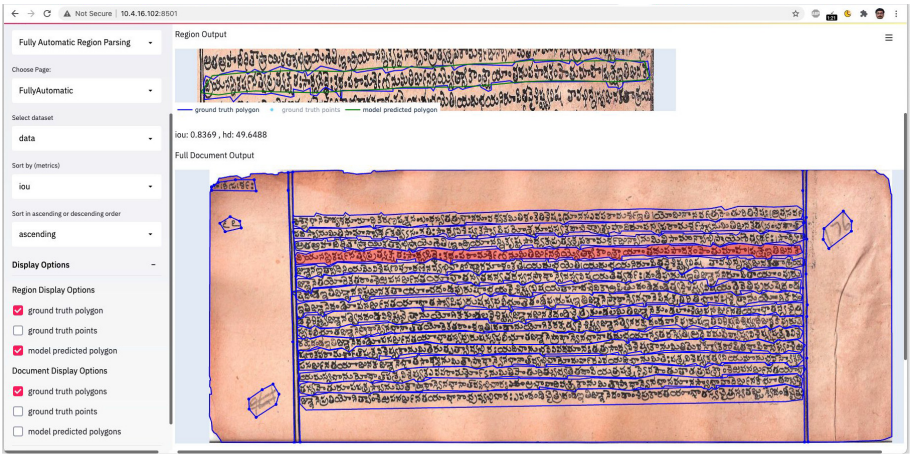


Fig. 2. Fully Automatic Region Parsing interface in DocVisor. Refer to Sect. 3.

Segmenting a document or tabular image into regions based on their semantic category (e.g. text lines, pictures, boundary lines, tabular rows) is an important problem [7,16,18,26]. This is especially crucial for isolating popularly sought regions such as text lines within handwritten and historical manuscripts. We consider three problem settings - historical manuscript region layout segmentation, printed document region layout detection and tabular data layout detection.

[6] https://github.com/OCR4all/OCR4all.
[7] https://tinyurl.com/69thzj3c.

3.1 Historical Manuscript Region Layout Segmentation

For our setting, we consider a state-of-the-art deep network for handwritten document layout segmentation [21]. The document images are sourced from Indiscapes, a large-scale historical manuscript dataset [18]. The dataset is annotated using the HInDoLA annotation system [23]. As part of the annotation process, each region of interest is captured as a user-drawn polygon along with a user-supplied region category label ('text line','picture','library stamp' etc.). These polygons are rendered into binary image masks and used along with the associated document images to train the deep network. The outputs from the deep network are also binary image masks. Selecting `Go to→Fully Automatic Region Parsing` and `Page→Fully Automatic` at top-left corner of DocVisor enables visualization of the document images, associated ground-truth and deep network predictions (Fig. 2). To analyze the performance at a per-region level, the region type is first selected at the top. The metric by which the selected regions are to be sorted (IoU, HD) are selected from the `Sort by (metrics)` dropdown in the left navigation bar. An important utility of the setup becomes evident when regions with IoU value = 0 are examined. These regions correspond to false negatives (i.e. they failed to be detected by the network). In the `Display Options→Region Display Options` on the bottom-left, the visualization elements of interest are checked. This section allows ground-truth or prediction to be displayed (on the top-right) either in terms of annotator provided polygon vertices or the associated polygon boundary.

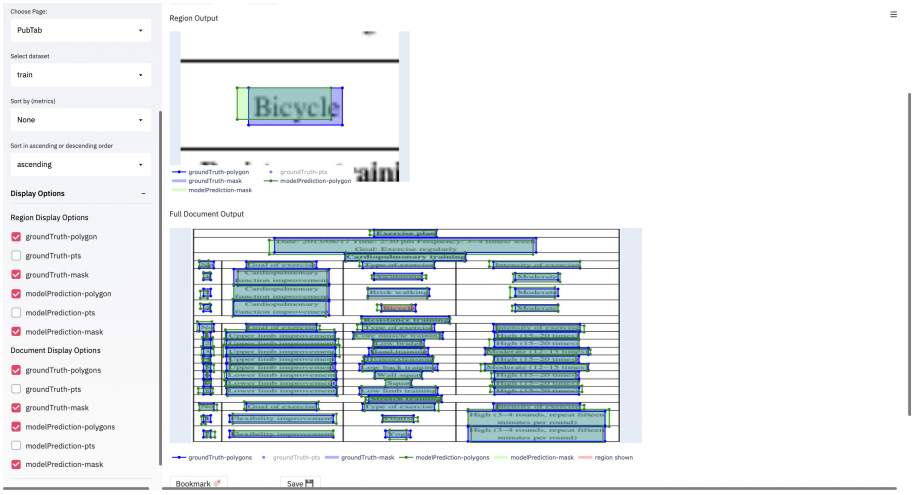


Fig. 3. PubTabNet [26] data visualized using DocVisor.

In the main panel on the right, the region image (sorted by the metric) is shown at the top. To enable viewing in the original document context, the

region is highlighted as a shaded region in the full image. As with the region display elements, the navigation bar on left can also be used to view visualization elements related to the full document image (`Display Options`→`Document Display Options`).

Finally, two additional options are also provided for selective navigation. The `Bookmark` button at the bottom enables the user to shortlist select regions for later viewing during the session. The `Save` button enables select regions to be saved to the user's device for offline analysis.

3.2 Table Data Layout Segmentation

Image-based table recognition is an important problem in document analysis. In our case, we have obtained the table-based images from PubTabNet [26]. Given the prominence of this dataset, we have provided all the necessary scripts to parse the data in the format akin to PubTabNet, and convert it to a format recognized by DocVisor.

This particular instance of Fully Automatic Region Parsing can be viewed by selecting `Go To`→`Fully Automatic Region Parsing` as well as selecting `Choose Page`→`PubTab` in the sidebar to the left. Since this page is an instance of the Fully Automatic Region Parsing layout, the functionalities available are identical to the `FullyAutomatic` instance. Refer to Fig. 3 for a depiction of an example visualization.

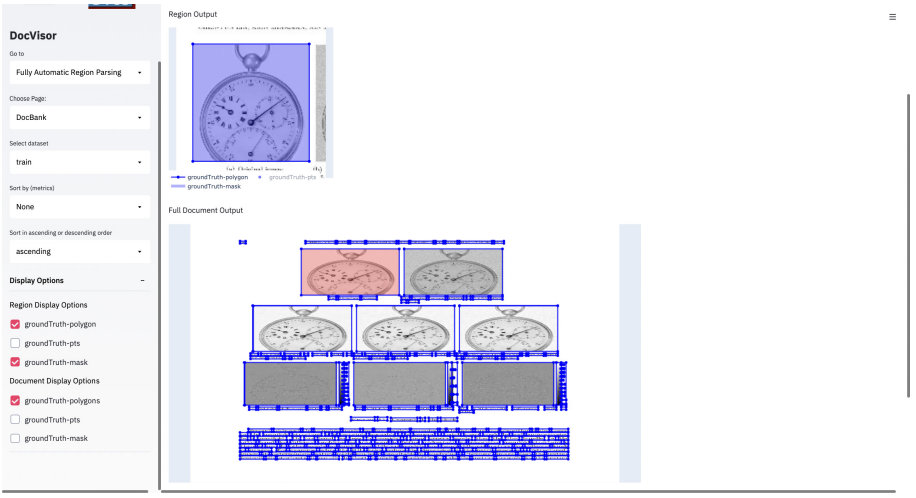


Fig. 4. DocBank [16] data visualized using DocVisor.

3.3 DocBank Dataset

DocBank [16] is a popular, large-scale dataset which consists of around 500,000 annotations, which have been obtained from a weak supervision approach. This data is also visualized as an instance of `Fully Automatic Region Parsing` layout. To select the DocBank page, select `Go To`→`Fully Automatic Region Parsing` and `Choose Page`→`DocBank`. Additionally, a script is provided in the DocVisor repository for parsing and converting data from the DocBank format to the DocVisor format. Much like PubTabNet, since DocBank is an instance of Fully Automatic Region Parsing layout, the features are identical to FullyAutomatic. Figure 4 shows an example visualization of DocBank data.

4 Box-Supervised Region Parsing

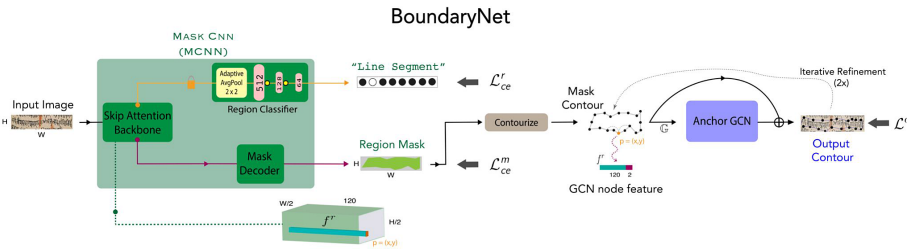


Fig. 5. The architectural diagram of the box-supervised deep network BoundaryNet [24].

A popular alternative to fully supervised region parsing (of the type described in previous section) is weakly supervised region parsing. In this setup, a weak supervisory input is provided (e.g. region class or the bounding box enclosing the region). We consider a state-of-the-art bounding box-supervised deep network called BoundaryNet [24] which outputs a sequence of predicted points denoting the vertices of the polygon enclosing the region (Fig. 5). The network also predicts the class label associated with the region. BoundaryNet is trained on an earlier version of the dataset [18] mentioned in the context of previous section.

As Fig. 5 shows, BoundaryNet comprises of two component modules. The first module, Mask-CNN, produces an initial estimate of the region boundary mask. This estimate is processed by a contourization procedure to obtain the corresponding polygon. This polygon estimate is refined by the Anchor-GCN module to produce the final output as a sequence of predicted points. Note that the output is not a mask unlike the full image setup in previous section.

In our setup, the set of intermediate and final outputs from BoundaryNet can be viewed in DocVisor by making `Box-supervised Region Parsing` selection at top-left corner. The various menu and visualization choices resemble the ones present for Fully Supervised Region Parsing as described in previous section (Fig. 6).

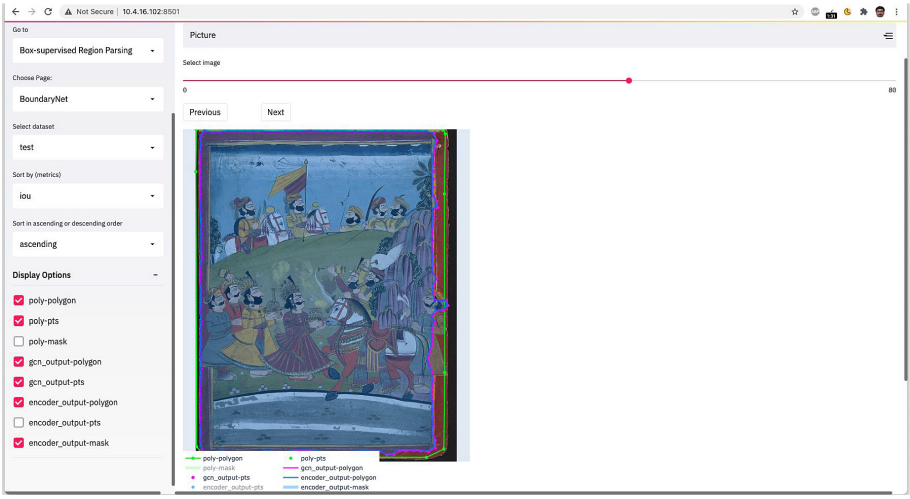


Fig. 6. Box-Supervised Region Parsing interface in DocVisor. A region image of type 'Picture' is shown in the example. Refer to Sect. 4.

5 Optical Character Recognition (OCR)

The OCR task involves recognizing text from images and is an intensely studied problem in the domain of document image understanding [3]. Although OCR systems work on various levels of granularities, we focus on the line-image variant. In this variant, the input is a single line of a document page, and the output is corresponding text.

We consider the line-level OCR developed by Agam et al. [5]. This OCR has been developed for historical documents in Sanskrit, a low-resource Indic language. We use DocVisor for analyzing two OCR models - one trained on printed texts and another on handwritten manuscripts. The accuracy of an OCR is typically characterized by Word Error Rate (WER) and Character Error Rate (CER); the smaller these quantities, the better the performance. The final numbers reported in OCR literature are typically obtained by averaging over WER and CER across line images from the test set. However, we use WER and CER scores at line-level as the metric for sorting the image set (train, validation, or test) in DocVisor.

The DocVisor interface enables selection between WER and CER metrics for sorting. The slider bar enables random access across the sorted image sequence. For each line image, the predicted text, corresponding ground-truth text, WER, and CER are shown (see Fig. 7). To enable aligned comparison with the image, we provide an option for changing the font size in the left-side navigation bar (under `Settings`→`Render`). For quick comprehension of differences between ground-truth and predicted text, we numerically index and color-code matching and mismatched segments (see `Visualize diffs` in Fig. 7).

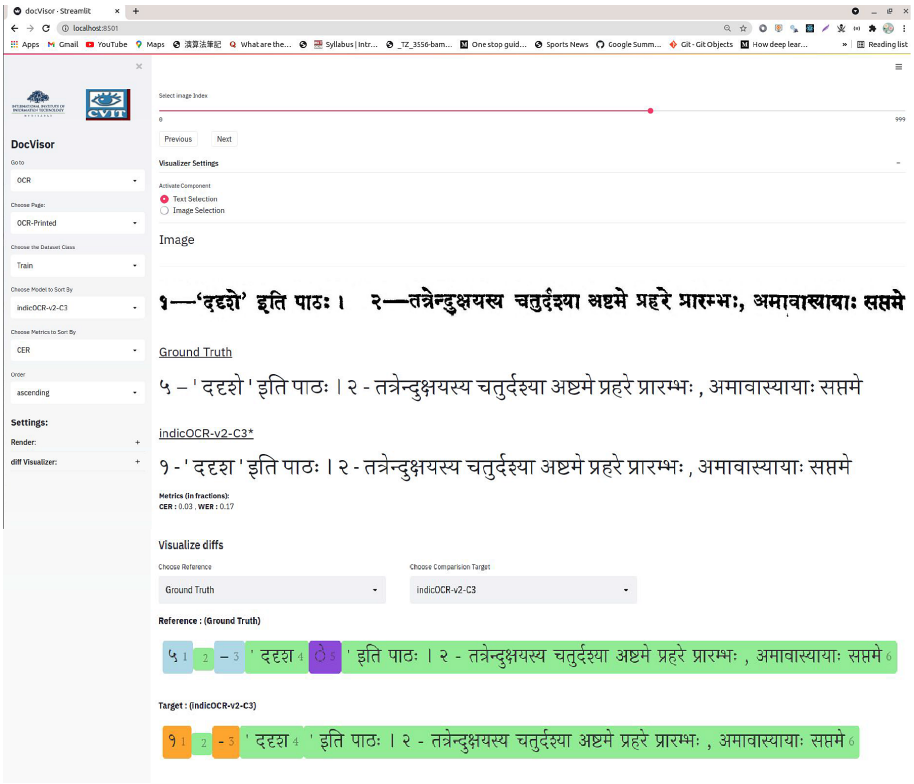


Fig. 7. OCR interface in DocVisor depicting line image, predicted text and ground-truth text. The metrics (CER, WER) and the color-coded mismatches and matches can also be seen. Refer to Sect. 5.

An interesting aspect of the OCR mentioned above lies in its 'attention' component [5]. In the process of learning to map a line image to its corresponding text, an intermediate stage produces an alignment matrix via the attention component mentioned before. The alignment matrix can be used to map a line image segment to an estimate of the corresponding matching sub-string in the predicted text and vice-versa. In the DocVisor UI, we enable the mapping via a selection process (Fig. 8). For instance, selecting `Visualizer Settings`→`Image Selection` enables selecting a segment of the line image. Upon completing the selection, the corresponding portion of the predicted text is highlighted. A similar mechanism for highlighting the image segment corresponding to a selection within the predicted text is achieved by switching to `Visualizer Settings`→`Text Selection`. From an explainability point of view, this feature is useful for understanding the image to text mapping decision process of the deep network, particularly for incorrectly predicted text portions. DocVisor also allows the user to select the color used for highlighting the image through a color picker for added flexibility.

Fig. 8. OCR interface in DocVisor depicting the image-text alignment by leveraging the attention mechanism from the OCR architecture of Agam et al. [5]. Refer to Sect. 5. Notice that there are multiple (2) attention OCR models along with non-OCR models that can be loaded in the OCR layout that allows the user to compare line level results of multiple models simultaneously.

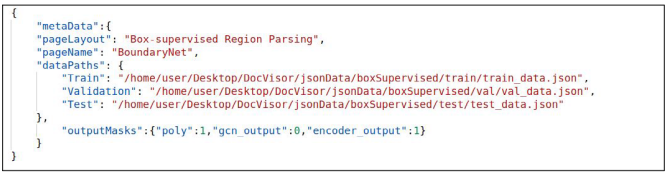
While analysing the results of a model on a particular dataset, sometimes it is important to compare the predictions of two different models or even two identical models trained with different hyper-parameter settings (e.g., learning rates) or different user-defined configurations. The DocVisor tool interface allows for visualization and comparison of multiple OCR outputs for the same image or dataset. The user can load multiple attention models along with outputs of other non-attention models (models that do not produce an alignment matrix) to compare results. The user can select the model by which the dataset is to be sorted using the metrics of interest. The user can choose to select and visualize the difference between output texts with the Ground-Truth or between outputs of any two models.

In our setting, we use the DocVisor tool to compare the results of C1 and C3 configurations of the IndicOCR model as defined by Agam et al. [5] along with results of popular state of the art non-attention models such as Google-OCR [8], Tessaract [22], and IndSenz [9]. Figure 8 shows an example of this setting.

```
{
    "metaData":{
        "pageLayout": "Fully Automatic Region Parsing",
        "pageName": "FullyAutomatic",
        "dataPaths": {
            "Train": "/home/user/Desktop/DocVisor/jsonData/fullyAutomatic/train/train_data.json",
            "Validation": "/home/user/Desktop/DocVisor/jsonData/fullyAutomatic/val/val_data.json",
            "Test": "/home/user/Desktop/DocVisor/jsonData/fullyAutomatic/test/test_data.json"
        },
        "outputMasks": {"groundTruth":1,"modelPrediction":1}

    }
}
```


(a)



```
{
    "metaData":{
    "pageLayout": "Box-supervised Region Parsing",
    "pageName": "BoundaryNet",
    "dataPaths": {
        "Train": "/home/user/Desktop/DocVisor/jsonData/boxSupervised/train/train_data.json",
        "Validation": "/home/user/Desktop/DocVisor/jsonData/boxSupervised/val/val_data.json",
        "Test": "/home/user/Desktop/DocVisor/jsonData/boxSupervised/test/test_data.json"
    },
        "outputMasks":{"poly":1,"gcn_output":0,"encoder_output":1}
    }
}
```


(b)

```
{
    "metaData": {
        "pageLayout": "OCR",
        "pageName": "OCR-Printed",
        "dataPaths": {
            "Train": "/home/user/Desktop/DocVisor/jsonData/ocr/train/train_data.json",
            "Validation": "/home/user/Desktop/DocVisor/jsonData/ocr/val/val_data.json",
            "Test": "/home/user/Desktop/DocVisor/jsonData/ocr/test/test_data.json"
        }
    }
}
```


(c)

Fig. 9. Example metadata files to launch the DocVisor tool. (a) Metadata file for Fully Automatic Layout. (b) Metadata file for Box Supervised Region Parsing Layout and (c) Metadata file for OCR layout. Refer to Sect. 6.

6 Configuration

To launch DocVisor, the user needs to provide the meta-information required to set up the tool through metadata files. Essentially, the metadata files act as configuration files in which the user not only provides the relevant information regarding the data to be visualized, but also certain settings the user can tweak for their own use cases. Figure 9 shows the structure of example metadata files for the three different layouts of DocVisor.

The user can load DocVisor with one or more of DocVisor's layouts. Additionally, a single layout can have multiple instances in the app: for example, a user may be interested in viewing the results of a model on two different datasets. In such cases, the user provides metadata files which would each contain the meta-information for the corresponding layout instance.

Figure 10 depicts standard data-file formats for each of the three layouts provided in the DocVisor tool. Each layout can be loaded provided the data files have the mandatory fields. Another advantage of DocVisor lies in its ability to import standard, pre-existing configuration formats for tasks which omits the need to write custom configuration files.

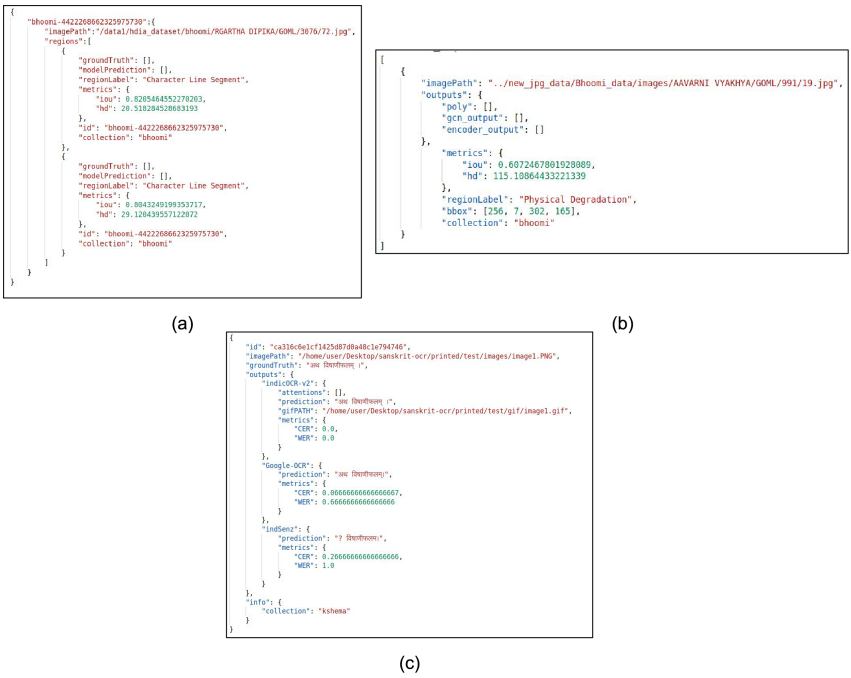


Fig. 10. Example data files for (a) Fully Automatic Layout (b) Box Supervised Region Parsing (c) OCR Layout. Refer to Sect. 6.

For more details of the data file format, refer to the tool's documentation at https://github.com/ihdia/docvisor.

7 Future Work

The existing work on DocVisor can be extended to include formats for some popular image segmentation tasks such as hOCR, PAGE format, LatexMath etc., making it much easier for users to integrate their data into DocVisor.

Additionally, support can be included for approaches with a large amount of data, where it may not be feasible to include all data into a single file.

8 Conclusion

We have presented an overview of DocVisor - our web-based multi-purpose visualization tool for analyzing the data and predictions related to various document image understanding problems. DocVisor currently supports four popular document understanding tasks and their variants – document region layout segmentation, tabular data detection, weakly-supervised document region segmentation, and optical character recognition. Given the lack of general-purpose data

and prediction analysis tools in the document image community, we expect our tool with its interactive and flexible display options to be a valuable contribution, readily available for reuse and modification given its open-source, public availability. Given the tool's capability for displaying point, polygon, and mask overlays, we also expect the tool to be helpful to researchers working on the broader computer vision problems of layout prediction.

References

1. Alberti, M., Bouillon, M., Ingold, R., Liwicki, M.: Open evaluation tool for layout analysis of document images. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 4, pp. 43–47. IEEE (2017)
2. Bukhari, S.S., Kadi, A., Jouneh, M.A., Mir, F.M., Dengel, A.: anyOCR: an open-source OCR system for historical archives. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 305–310. IEEE (2017)
3. Cheriet, M., Kharma, N., Liu, C.L., Suen, C.: Character Recognition Systems: A Guide for Students and Practitioners. John Wiley & Sons, Hoboken (2007)
4. Clausner, C., Pletschacher, S., Antonacopoulos, A.: Aletheia - An advanced document layout and text ground-truthing system for production environments. In: 2011 International Conference on Document Analysis and Recognition, pp. 48–52. IEEE (2011)
5. Dwivedi, A., Saluja, R., Kiran Sarvadevabhatla, R.: An OCR for classical indic documents containing arbitrarily long words. In: The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2020
6. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The PASCAL visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
7. Gatos, B., et al.: Ground-truth production in the tranScriptorium project. In: 2014 11th IAPR International Workshop on Document Analysis Systems, pp. 237–241. IEEE (2014)
8. Google: Convert pdf and photo files to text (2020). https://support.google.com/drive/answer/176692?hl=en. Accessed 26 March 2020
9. Hellwig, O.: Indsenz OCR (2020). http://www.indsenz.com/. Accessed on 26 March 2020
10. Huttenlocher, D.P., Klanderman, G.A., Rucklidge, W.J.: Comparing images using the hausdorff distance. IEEE Trans. Pattern Anal. Mach. Intell. **15**(9), 850–863 (1993)
11. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning, vol. 112. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-7138-7
12. Jenckel, M., Bukhari, S.S., Dengel, A.: anyOCR: a sequence learning based OCR system for unlabeled historical documents. In: 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 4035–4040. IEEE (2016)
13. Kahle, P., Colutto, S., Hackl, G., Mühlberger, G.: Transkribus-a service platform for transcription, recognition and retrieval of historical documents. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 4, pp. 19–24. IEEE (2017)

14. Kiessling, B., Tissot, R., Stokes, P., Ezra, D.S.B.: escriptorium: an open source platform for historical document analysis. In: 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW), vol. 2, p. 19. IEEE (2019)
15. Kumar, M.P., Kiran, S.R., Nayani, A., Jawahar, C., Narayanan, P.: Tools for developing OCRS for Indian scripts. In: 2003 Conference on Computer Vision and Pattern Recognition Workshop, vol. 3, p. 33. IEEE (2003)
16. Li, M., Xu, Y., Cui, L., Huang, S., Wei, F., Li, Z., Zhou, M.: DocBank: a benchmark dataset for document layout analysis (2020)
17. Oliveira, S.A., Seguin, B., Kaplan, F.: dhsegment: a generic deep-learning approach for document segmentation. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 7–12. IEEE (2018)
18. Prusty, A., Aitha, S., Trivedi, A., Sarvadevabhatla, R.K.: Indiscapes: instance segmentation networks for layout parsing of historical indic manuscripts. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 999–1006. IEEE (2019)
19. Saluja, R., Adiga, D., Ramakrishnan, G., Chaudhuri, P., Carman, M.: A framework for document specific error detection and corrections in Indic OCR. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 4, pp. 25–30. IEEE (2017)
20. Schomaker, L.: Design considerations for a large-scale image-based text search engine in historical manuscript collections. IT-Inf. Technol. **58**(2), 80–88 (2016)
21. Sharan, S.P., Aitha, S., Amandeep, K., Trivedi, A., Augustine, A., Sarvadevabhatla, R.K.: Palmira: a deep deformable network for instance segmentation of dense and uneven layouts in handwritten manuscripts. In: International Conference on Document Analysis Recognition, ICDAR 2021 (2021)
22. Smith, R.: Tesseract-OCR (2020). https://github.com/tesseract-ocr/. Accessed 26 Mar 2020
23. Trivedi, A., Sarvadevabhatla, R.K.: HInDoLA: A Unified Cloud-based Platform for Annotation, Visualization and Machine Learning-based Layout Analysis of Historical Manuscripts. In: 2nd International Workshop on Open Services and Tools for Document Analysis, OST@ICDAR 2019, Sydney, Australia, September 22–25, 2019. pp. 31–35. IEEE (2019). https://doi.org/10.1109/ICDARW.2019.10035, https://doi.org/10.1109/ICDARW.2019.10035
24. Trivedi, A., Sarvadevabhatla, R.K.: BoundaryNet: an attentive deep network with fast marching distance maps for semi-automatic layout annotation. In: International Conference on Document Analysis Recognition, ICDAR 2021 (2021)
25. Wojna, Z., et al.: Attention-based extraction of structured information from street view imagery. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 844–850. IEEE (2017)
26. Zhong, X., ShafieiBavani, E., Yepes, A.J.: Image-based table recognition: data, model, and evaluation. arXiv preprint arXiv:1911.10683 (2019)