A Large Dataset of Historical Japanese Documents with Complex Layouts

Zejiang Shen Kaixuan Zhang Melissa Dell
Harvard University
{zejiang_shen, kaixuanzhang, melissadell}@fas.harvard.edu

Figure 1: **Examples of HJDataset document images and annotations**. (a) to (d) show images of the four page categories, and (e) provides a simplified illustration of layout annotations for `main` pages. The seven types of hierarchically constructed layout elements are highlighted in different colors.

Abstract

Deep learning-based approaches for automatic document layout analysis and content extraction have the potential to unlock rich information trapped in historical documents on a large scale. One major hurdle is the lack of large datasets for training robust models. In particular, little training data exist for Asian languages. To this end, we present HJDataset, a Large Dataset of ***H****istorical* ***J****apanese Documents with Complex Layouts. It contains over 250,000 layout element annotations of seven types. In addition to bounding boxes and masks of the content regions, it also includes the hierarchical structures and reading orders for layout elements. The dataset is constructed using a combination of human and machine efforts. A semi-rule based method is developed to extract the layout elements, and the results are checked by human inspectors. The resulting large-scale dataset is used to provide baseline performance analyses for text region detection using state-of-the-art deep learning models. And we demonstrate the usefulness of the dataset on real-world document digitization tasks. The dataset is available at* `https://dell-research-harvard.github.io/HJDataset/`.

1. Introduction

Complex layouts significantly complicate the automated digitization of historical documents, which contain a variety of rich information of interest to researchers and the public more generally. In particular, many documents of relevance to social science researchers and business analysts contain complex, heterogeneous tabular and column structures, which off-the-shelf tools cannot recognize. Moreover, unique layout patterns appear in different languages. For example, complex layouts with vertical text orientation are common in Asian languages. Complex layouts disrupt Optical Character Recognition (OCR) and result in text from different columns, rows, or text regions being incorrectly garbled together, making automated digitization results unusable.

Various algorithms [2, 3] have been proposed to analyze the layouts geometrically. They utilize visual properties like text spacing and gaps to correct skewness and segment content regions with fine-tuned parameters. Recently, there has been an increased interest in adopting deep learning (DL) methods to build end-to-end layout understanding models. For example, Oliveira *et al*. [16] and Xu *et al*. [23] build models upon fully convolutional networks [13] to detect page frames and text lines with high accuracy.

Central to the success of DL models are many labeled samples for training and evaluating the neural networks. There have been long term efforts to develop layout analysis datasets [1], and recently a very large-scale dataset has been developed for modern documents [24]. However, for historical documents, the existing datasets are small. For example, there are only 150 instances in the DIVA-HisDB dataset [20] and 528 in the European Newspapers Project Dataset [5]. Because deep neural nets tend to overfit small datasets, models trained on them are less robust and performance evaluation is less reliable. Because older documents are subject to wearing, stains, and other noise that do not appear in modern documents, they require dedicated large datasets for training.

Additionally, most open-sourced historical document layout datasets are in western languages [1, 5, 4]. Models trained on them are not exposed to layout patterns that appear commonly and exclusively in Asian languages. Asian language datasets will be required to build more generalized layout analysis models.

To attack these problems, we present the HJDataset: a Large Dataset of **H**istorical **J**apanese Documents with Complex Layouts. Currently, this dataset contains 2,271 document image scans with various document information, from the Japanese Who's Who biographical directory published in 1953, which contains biographies for around 50,000 prominent Japanese citizens [10]. For each document image, HJDataset contains its content category (`main`, `index`, `advertisement`, or `other`). For the `main` and `index` pages, we create 25k layout region annotations of seven types at different levels (from page frames to individual text blocks). Besides the bounding box coordinates, we also include the dependency structures and reading orders for all the layout elements. The data are stored in the COCO format [12], which is commonly used in computer vision research. The resulting dataset provides a ground truth for different document image analysis tasks, from page classification to layout element detection. Extensive experiments have been conducted, and state-of-the-art models are trained and evaluated on this dataset.

Manual creation of such a dataset would be highly laborious, prohibitively costly, and potentially quite noisy. Therefore, similar to PubLayNet [24], HJDataset is generated in a near-automatic fashion. With the help of a carefully designed semi-rule-based method, the layout elements are accurately extracted. To ensure label quality, possible errors are identified based on annotation statistics, and human inspectors correct some minor errors accordingly.

The contribution of this work is twofold. First, we build the HJDataset, the first large layout analysis dataset of historical Japanese documents to the best of our knowledge. A semi-ruled-based method is designed for generating this dataset. Second, we show that models pre-trained on our dataset can improve performance on other tasks with small amounts of labeled data. The dataset and pre-trained models will be released online to support the development of Japanese and more general layout analysis algorithms.

2. Related Work

Layout Analysis Dataset A variety of layout analysis datasets have been created in recent years [21]. For modern documents, Antonacopoulos *et al*.'s work [1] is the first frequently-used dataset, with 305 images of magazines and technical articles available for download. The recent PubLayNet [24] dataset contains 360k samples from modern research publications. For historical documents, the work in [5] provides layout annotations for 600 historical European newspaper images. The datasets in [20] and [6] are commonly used for medieval manuscripts and have 160 and 2036 samples respectively. Historical layout datasets tend to be small and are largely unavailable for Asian languages. Large-scale digital libraries, such as the millions of scans placed online by Japan's National Diet Library, provide the raw inputs for creating large datasets for historical document layout analysis, but developing these datasets requires methods that do not rely on costly human labeling.

Deep Learning for Layout Analysis As deep learning has revolutionized computer vision research, DL-based document image analysis methods are also being developed to tackle challenging tasks. [7] evaluates convolutional neural networks for document image classification tasks, and [16] adapts the fully convolutional network (FCN) [13] to detect layout element objects inside the page. For more complicated tabular data, Schreiber *et al*. [19] adapt Faster R-CNN [17] and FCN to identify their structures and parse the contents. Behind their success, large datasets are required to train and evaluate the models.

Table 1: Page types and numbers included in HJDataset

Page Type	Number of images	Category ID [a]
`main`	2048	8
`advertisement`	87	9
`index`	82	10
`other`	54	11

[a] As COCO format does not contain an image-level category field, we add a new key for each *image* record called *category id*.

3. Page Type Labeling

Contents are organized very differently on pages of different purposes, and hence the first step of the layout analysis pipeline identifies the page type. We manually labeled the page types according to their purposes.

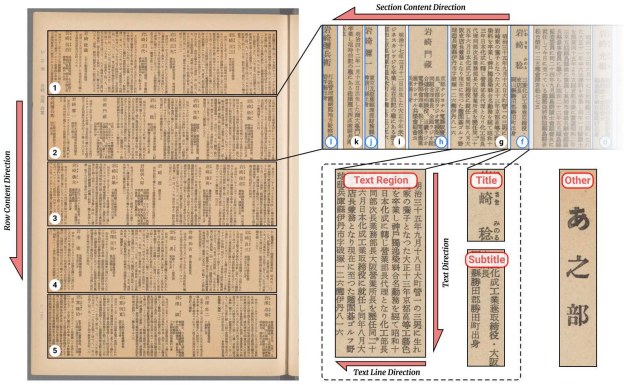

Figure 2: **The hierarchical content structure in `main` pages.** Each page contains five rows that are vertically stacked, and the text regions are horizontally arranged within each row. Texts are vertically written inside the text region, *e.g.* (g) in the figure. The title region, *e.g.* (f), can be further split into title and subtitle blocks. An other category is reserved for chapter headers and other irrelevant text regions.

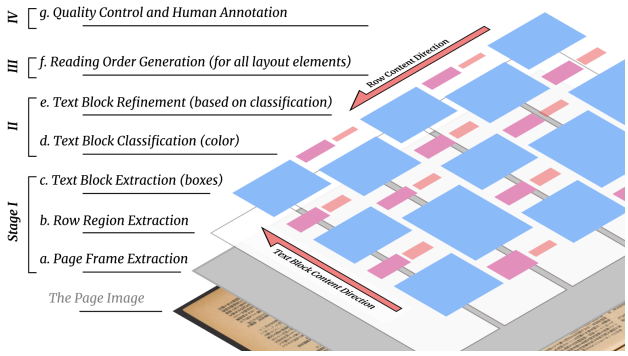

Figure 3: **The four stages in layout element annotation.** Our method detects the coordinates of the page frames, row regions, and text blocks. A text block classifier is then used to predict the block categories (indicated by the different colors in the figure), and the detections are refined accordingly. Reading orders and hierarchical dependency are generated for all layout elements. Finally human annotators check the results and correct the errors.

As shown in Figure 1, four labels, *i.e.* `main`, `index`, `advertisement`, and `other`, have been created for the 2k images. `main` pages present the detailed biographical information of around 50,000 Japanese business, political, and cultural leaders with complex structure, forming our primary focus. Table 1 provides a detailed description of the classes and the number of samples contained in HJDataset.

4. Document Layout Annotation

As shown in Figure 2, the contents in the `main` pages are organized in a hierarchical manner. Five rows are vertically stacked in a page, while text region and title region are horizontally arranged inside each row. The title region can be further broken down into title and subtitle blocks, and other irrelevant texts are labeled as the *other* type. The text region blocks contain only vertical text lines and read from right to left. Our objective is to segment the pages into units of simple layouts, namely, text region, title, and subtitle blocks. Similar rules apply to the `index` pages.

Based on the hierarchical structures, we design a multi-stage pipeline for robustly extracting the layout elements, illustrated in Figure 3. For the input page scan, the *Text Block Detector* first extracts the bounding boxes of page frame, row, text region, and title region sequentially, as explained in Section 4.1. A CNN is trained to predict the contextual labels for the extracted regions, and the block segmentation is refined accordingly (detailed in Section 4.2). After that, we construct the *reading orders* based on Japanese reading rules, as described in Section 4.3. Finally, Section 4.4 discusses measures to identify and correct possible errors to ensure high quality of the generated annotations. The dataset statistics are provided in Section 4.5.

4.1. Text Block Detector

The *Text Block Detector* extracts the content boxes in the input scan in a hierarchical fashion. After binarizing the color scans, the recognition is conducted on different resolutions to identify blocks of different scales. The algorithm downsamples the image with a $1/8$ ratio when detecting the page frame and row boxes, while using the full resolutions for extracting regions in each row. To account for possible rotations and irregularities, we characterize the page frame boxes with quadrilaterals. The row, text region, and title region are represented with rectangles as the distortions are largely eliminated within the page frames.

As illustrated in Figure 3.a, we first estimate the page frame box using contour detection. This method groups pixels with similar visual properties like color or intensity and can be used for extracting different regions [14]. In our case, the largest intensity contour in the input delineates the page boundary, and we estimate the four vertex coordinates $\{(x_i, y_i)\}_{i=1}^{4}$ of the circumscribed quadrilateral for this contour as the page box. We convert the page image inside the quadrilateral to a rectangle based on a warp affine transformation.

Connected Component Labeling (CCL) [18] and Run Length Smoothing Algorithm (RLSA) [15] are used for splitting the five rows of contents vertically inside the page frame. As we apply the RSLA algorithm horizontally, each row is connected, and CCL can be applied to differentiate the rows. This approach is robust when the page is the end

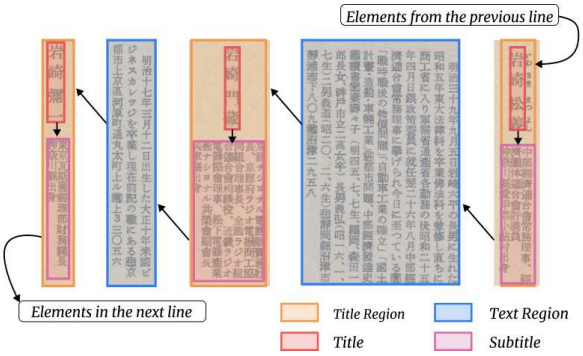

Figure 4: **Examples of the layout annotations and their reading order.**

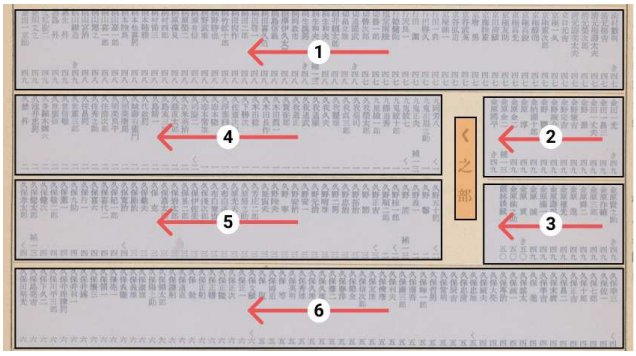

Figure 5: **Irregular reading orders in the index pages.** The section header in row 2 and 3 disrupts the reading order.

of a chapter, where there could be fewer than five rows, and the last row is not "full". Similarly, for text and title regions in a row, we apply RLSA vertically and split the connected components. Since the prediction is performed row-wise, it is impossible to connect text blocks in different rows, and the segmentation result is more robust. Rectangular bounding box coordinates (x_1, y_1, w, h) are generated for each row, text and title region, where (x_1, y_1) is the coordinate for the top-left corner, and w and h are width and height for the rectangle, respectively.

Text Block Detector finds the layout regions with high accuracy (details in Section 4.4). However, text and title regions are sometimes mis-segmented due to various noise. Hence, *Text Region Classifier* is developed to identify layout categories and correct segmentation errors.

4.2. Text Region Classification and Refinement

A three-class CNN classifier is trained to identify the text, title, and wrongly-segmented regions. After obtaining the region bounding box from *Text Region Detector*, we crop the page image based on the coordinates and predict its category. If it is classified as mis-segmented, a CCL-based method is applied to split it into text and title region. Title regions are further broken down into more refined title and subtitle segments, as illustrated in Figure 4.

We use the NASNet Mobile [25] architecture to build our CNN. It is a neural network generated via Neural Architecture Search (NAS) and achieves excellent performance over many benchmarks. Our classifier is trained on 1,200 hand-labeled samples and tested on 100 samples. As mis-segmentation rarely appears (only 3 in 1000 samples), we re-balance the dataset distribution by manually creating 250 mis-segmented images. The input images are rescaled to the same size of 200 height and 522 width. We train the model from scratch, without loading pre-trained weights. Using a stochastic gradient descent optimizer, the loss converges in 40 epochs with a final test accuracy of 0.99.

4.3. Reading Order Generation

This publication contains non-trivial reading orders which must also be deduced. The texts inside the basic elements (text region, title, and subtitle) are written vertically, and read from right to left. Additionally, for text blocks in a row, they also follow a right-to-left order. Black arrows in Figure 4 shows the topological orders of different elements in a row. However, some different structures also exist. As indicated in Figure 5, section titles (shown in orange) disrupt the regular right-to-left order of texts (see rows 2 and 3). As texts are usually densely arranged in each row, by searching the large gaps between blocks, we identify the discontinuity and correct the special reading order accordingly. We incorporate this irregularity in the dataset to include the real-world noise and support the development of more general layout understanding models.

4.4. Quality Control and Human Annotations

Historical scans are challenging to analyze due to various noise. Despite the carefully engineered method described above, detection errors inevitably exist and need to be handled carefully. However, considering the sheer number of layout elements in this dataset, manual checking of all the predictions would be highly laborious and potentially noisy.

To identify the small number of incorrect predictions without searching the whole dataset, we examine statistics about blocks and pages. As the `main` pages are densely printed, we find the number of layout elements remains consistent across pages, and blocks in a row are usually evenly spaced. Hence, by filtering layout elements that are significantly different in these statistics, we obtain a limited number of misdetection candidates. As the specificity (true negative rate) of the subsample is much higher, we can correct the problems more efficiently.

Misdetected Page Frames When a page is not appropriately scanned, or it is physically broken, the page frame de-

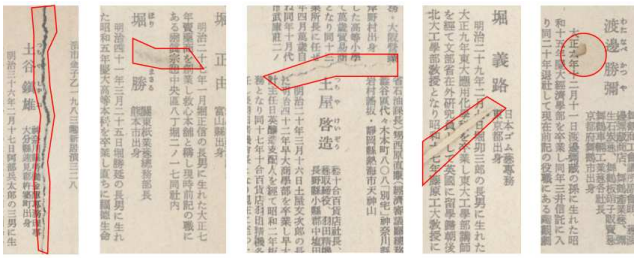

Figure 6: **Various noises in the page scans.**

tection will become inaccurate, and it will disrupt the subsequent extraction of row and text regions. A large increase or decrease in the number of layout elements in a page often implies a misdetection of the page frame. Therefore, we select pages with more than 118 (95th percentile) or less than 88 (5th percentile) layout elements and check them manually. This selects 182 pages, and 18 (9.9%) errors are identified. Their page frame coordinates are re-labeled manually. After correction, we re-run the pipeline over the pages in order to detect other layout regions more accurately.

Missed Text Lines The last text lines in a text region are sometimes missed if they contain only a few characters. This results in unusually large gaps between text blocks. This error can be easily identified by filtering the widths of the block gaps. We select 1,011 blocks with gaps larger than 54 pixels (99th percentile), and correct 487 of them.

Additional Correction Figure 6 shows issues like cracks, stains, and holes that appear frequently and can disrupt the prediction pipeline. It is difficult to pre-screen all the mis-segmentation and incorrect predictions due to these irregularities. Hence, during the manual checking process, human annotators are asked to identify such errors and correct them. A total of 111 layout elements have been found and corrected so far.

In all, we fix more than 616 errors in total (since fixing page frames leads to more improvements), and 80% are identified by the statistical approach. We estimate that before correction, there are around 1,560 blocks detected inaccurately.[1] After correcting the errors, the resultant dataset achieves 99.6% accuracy, and the remaining 0.4% errors can be neglected as random noise.

4.5. Dataset Statistics and Partition

A total of 259,616 layout elements of seven categories have been extracted, as detailed in Table 2. Figure 7 shows examples of the annotations. Layout elements like text region and other do not appear in the `index` pages, as we characterize the texts in index pages as title.

We partition our dataset into training, validation, and testing subsets: 70% for training and 15% each for validation and testing. The breakdown is stratified based on the page type to ensure the equal exposure of different page types in the three subsets. Because the characteristics of the pages vary, categories appear in different frequencies, and the dataset is unbalanced with respect to the object types.

[1] We randomly choose 20 pages, and count the error rate. This process is repeated 3 times, and the average inaccuracy is 0.6%, which is equivalent to 1,560 out of 260k blocks.

Table 2: Layout element categories and numbers

Category	Training	Validation	Test	Total
Page Frame	1490	320	320	2130
Row	7742	1657	1660	11059
Title Region	33637	7184	7271	48092
Text Region	38034	8129	8207	54370
Title	66515	14931	14366	95812
Subtitle	33576	7173	7256	48005
Other	103	16	29	148
Total	181097	39410	39109	259616

5. Experiments

In this section, we first report results from training state-of-the-art object detection models on the HJDataset. Performance is evaluated and provided as a benchmark. Second, based on the pre-trained model, we study how HJDataset can assist other layout analysis tasks.

5.1. Deep Learning Benchmark

Without considering the dependency between contents, layout analysis can be treated as *detecting layout objects* inside each page. As object detection has been extensively studied in current deep learning research, well-established models like Faster R-CNN [17], RetinaNet [11], and Mask R-CNN [8] have achieved excellent performance in various benchmarks [12]. Hence, we adopt these models and train them on our dataset. The implementation is based on Detectron2 [22], and the neural networks are trained on a single NVIDIA RTX 2080Ti GPU.

The three models are trained on all layout elements of `main` pages from the training set. For fair comparison, they are all being trained for 60k iterations, with a base 0.00025 learning rate, and a decay rate of 0.1 for each 30k iterations. The batch size is 2, and the backbone CNN structure is `R-50-FPN-3x` (details in [22]), loaded with pre-trained weights from the COCO dataset. The training configuration will also be open-sourced for reproduciblility.

Table 3 shows the per-category bounding box prediction mean Average Precision (mAP) for intersection, at intersec-

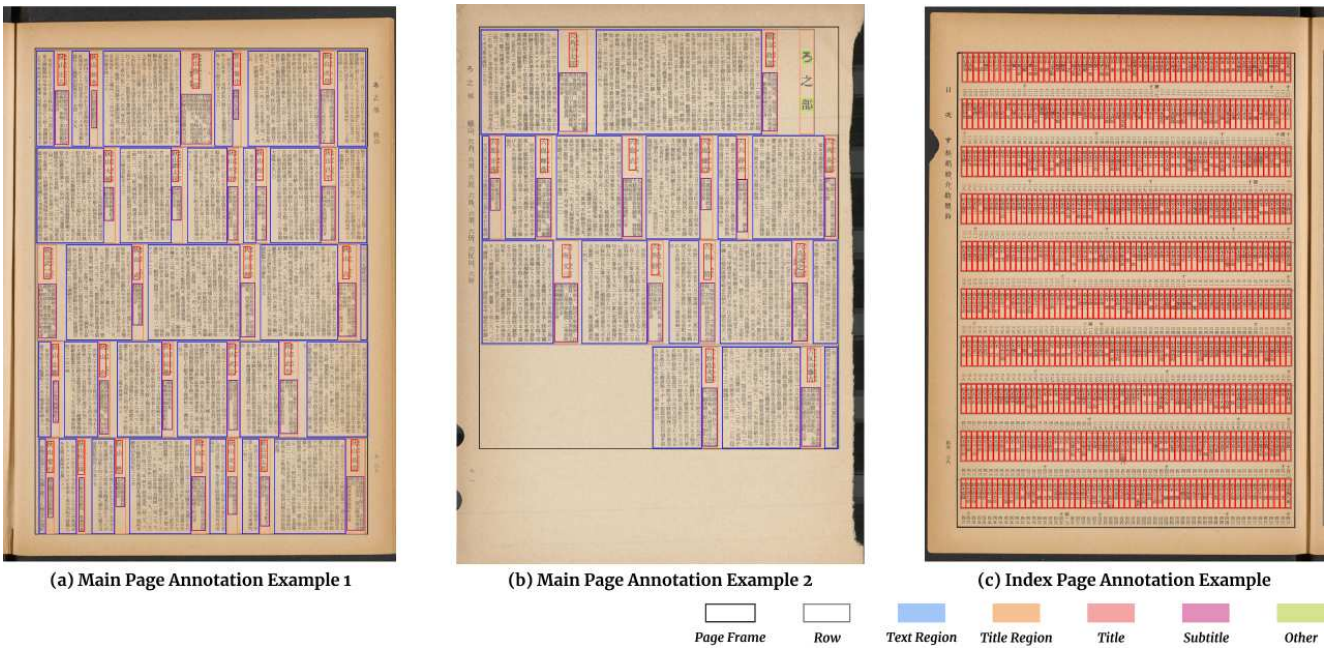

Figure 7: **Annotation Examples in HJDataset.** (a) and (b) show two examples for the labeling of `main` pages. The boxes are colored differently to reflect the layout element categories. Illustrated in (c), the items in each `index` page row are categorized as title blocks, and the annotations are denser.

tion over union (IOU) level [0.50:0.95][2], on the test data. In general, the high mAP values indicate accurate detection of the layout elements. The Faster R-CNN and Mask R-CNN achieve comparable results, better than RetinaNet. Noticeably, the detections for small blocks like title are less precise, and the accuracy drops sharply for the title category. In Figure 8, (a) and (b) illustrate the accurate prediction results of the Faster R-CNN model.

5.2. Pre-training for other datasets

We also examine how our dataset can help with a real-world document digitization application. When digitizing new publications, researchers usually do not generate large scale ground truth data to train their layout analysis models. If they are able to adapt our dataset, or models trained on our dataset, to develop models on their data, they can build their pipelines more efficiently and develop more accurate models. To this end, we conduct two experiments. First we examine how layout analysis models trained on the `main` pages can be used for understanding `index` pages. Moreover, we study how the pre-trained models perform on other historical Japanese documents.

Table 4 compares the performance of five Faster R-CNN models that are trained differently on `index` pages. If the model loads pre-trained weights from HJDataset, it includes information learned from `main` pages. Models trained over all the training data can be viewed as the benchmarks, while training with few samples (five in this case) are considered to mimic real-world scenarios. Given different training data, models pre-trained on HJDataset perform significantly better than those initialized with COCO weights. Intuitively, models trained on more data perform better than those with fewer samples. We also directly use the model trained on `main` to predict `index` pages without fine-tuning. The low zero-shot prediction accuracy indicates the dissimilarity between `index` and `main` pages. The large increase in mAP from 0.344 to 0.471 after the model is

Table 3: Detection mAP @ IOU [0.50:0.95] of different models for each category on the test set. All values are given as percentages.

Category	Faster R-CNN	Mask R-CNN[a]	RetinaNet
Page Frame	99.046	99.097	99.038
Row	98.831	98.482	95.067
Title Region	87.571	89.483	69.593
Text Region	94.463	86.798	89.531
Title	65.908	71.517	72.566
Subtitle	84.093	84.174	85.865
Other	44.023	39.849	14.371
mAP	81.991	81.343	75.223

[a] For training Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

[2] This is a core metric developed for the COCO competition [12] for evaluating the object detection quality.

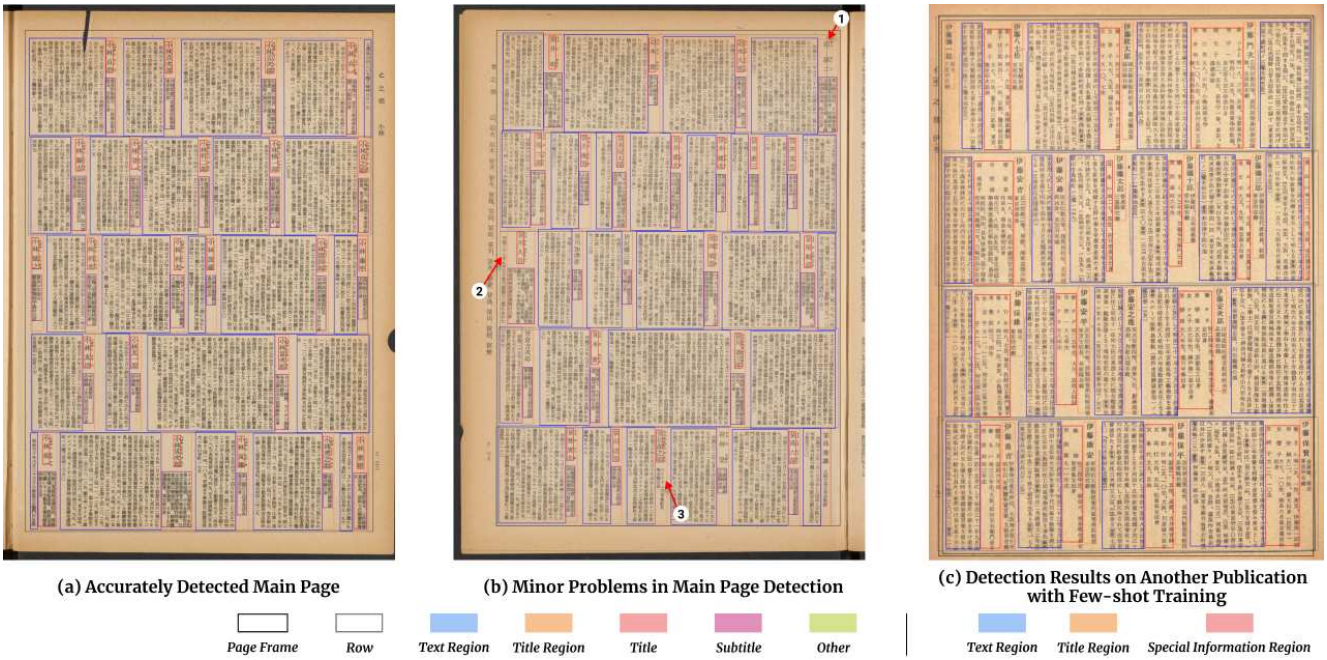

Figure 8: **The prediction results of Faster R-CNN on Main pages in HJDataset and another publication.** (a) shows that the Faster R-CNN model is robust to noise like cracks and can detect most of the layout elements accurately. (b) highlights some minor errors in the Faster R-CNN predictions like inaccurate row blocks, *e.g.* (1), and missed text and title regions, *e.g.* (2) and (3). (c) shows the results of few-shot trained Faster R-CNN on another publication. They are generally correct. We label the new publication differently to increase the difficulty for training, and the red boxes in the image denote a special information region.

Table 4: Comparison of the test set AP of Faster R-CNN models trained differently on `index` pages. All values are given as percentages.

Initialization	Training Data	mAP	AP_{50}	AP_{75}
COCO	All[a]	34.408	53.342	37.533
COCO	Few-shot	9.988	18.572	9.669
HJDataset	All	47.125	67.502	54.410
HJDataset	Few-shot	10.275	21.353	10.423
HJDataset	Zero-shot	9.411	44.299	0.068

[a] *All* indicates the model is trained on all 57 training `index` samples, *few-shot* refers to model trained on 5 random samples, and *zero-shot* means the model directly use the weights without training.

trained on five samples shows that the model can be quickly adapted to similar tasks. As the AP_{50} and AP_{75} (AP calculated with IOU=0.50 and 0.75) are higher than mAP, we conclude that the models can learn to detect the *general* position of layout objects.

To evaluate our models on other historical Japanese documents, we manually annotate 12 pages from another publication with different layouts, the Japanese Who's Who biographical directory published in 1939 [9], and we train the models on 4 samples. Performance is assessed on the remaining 8 samples, as reported in Table 5. Similar to the previous experiment, pre-training on HJDataset has a large positive influence on the detection accuracy given few training samples. And shown in Figure 8 (c), the layout elements are detected accurately. In summary, these two experiments demonstrate the usefulness of our dataset for other layout analysis tasks.

Table 5: Comparison of the test set AP of Faster R-CNN models trained differently on another publication. All values are given as percentages.

Initialization	Training Data	mAP	AP_{50}	AP_{75}
COCO	Few-shot	69.925	95.119	78.667
HJDataset	Few-shot	81.638	98.364	88.203
HJDataset	Zero-shot	38.959	50.971	42.269

6. Conclusion

In this paper, we introduce the HJDataset, a large layout analysis dataset for historical Japanese documents. With a combination of semi-rule-based segmentation and statistical error identification and correction, 260k layout annotations of seven categories are extracted from 2.2k page scans. Page type labels, block dependency, and reading orders are also included. Stored in COCO format, HJDataset allows

state-of-the-art object detection models to be easily trained and evaluated. Moreover, we show that deep learning models trained on HJDataset can be adapted to other datasets, facilitating real-world document digitization tasks.

Acknowledgement. This project is supported in part by NSF Grant #1823616.

References

[1] Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. A realistic dataset for performance evaluation of document layout analysis. In *2009 10th International Conference on Document Analysis and Recognition*, pages 296–300. IEEE, 2009.

[2] Thomas M Breuel. High performance document layout analysis. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 209–218, 2003.

[3] Roldano Cattoni, Tarcisio Coianiz, Stefano Messelodi, and Carla Maria Modena. Geometric layout analysis techniques for document image understanding: a review. *ITC-irst Technical Report*, 9703(09), 1998.

[4] Christian Clausner, Apostolos Antonacopoulos, and Stefan Pletschacher. ICDAR2019 competition on recognition of documents with complex layouts-rdcl2019. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1521–1526. IEEE, 2019.

[5] Christian Clausner, Christos Papadopoulos, Stefan Pletschacher, and Apostolos Antonacopoulos. The ENP image and ground truth dataset of historical newspapers. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 931–935. IEEE, 2015.

[6] Tobias Grüning, Roger Labahn, Markus Diem, Florian Kleber, and Stefan Fiel. Read-bad: A new dataset and evaluation scheme for baseline detection in archival documents. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 351–356. IEEE, 2018.

[7] Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995. IEEE, 2015.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[9] Jinji Kōshinjo. Jinji kōshinroku, volume 11, 1939.

[10] Jinji Kōshinjo. Jinji kōshinroku, volume 17, 1953.

[11] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[14] Michael Randolph Maire. *Contour detection and image segmentation*. Citeseer, 2009.

[15] Nikos Nikolaou, Michael Makridis, Basilis Gatos, Nikolaos Stamatopoulos, and Nikos Papamarkos. Segmentation of historical machine-printed documents using adaptive run length smoothing and skeleton segmentation paths. *Image and Vision Computing*, 28(4):590–604, 2010.

[16] Sofia Ares Oliveira, Benoit Seguin, and Frederic Kaplan. dhSegment: A generic deep-learning approach for document segmentation. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 7–12. IEEE, 2018.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[18] Hanan Samet and Markku Tamminen. Efficient component labeling of images of arbitrary dimension represented by linear bintrees. *IEEE transactions on pattern analysis and machine intelligence*, 10(4):579–586, 1988.

[19] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. DeepDeSRT: Deep learning for detection and structure recognition of tables in document images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1162–1167. IEEE, 2017.

[20] Foteini Simistira, Mathias Seuret, Nicole Eichenberger, Angelika Garz, Marcus Liwicki, and Rolf Ingold. Diva-hisdb: A precisely annotated large dataset of challenging medieval manuscripts. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 471–476. IEEE, 2016.

[21] Ernest Valveny. *Datasets and Annotations for Document Analysis and Recognition*, pages 983–1009. Springer London, London, 2014.

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

[23] Yue Xu, Fei Yin, Zhaoxiang Zhang, and Cheng-Lin Liu. Multi-task layout analysis for historical handwritten documents using fully convolutional networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1057–1063. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[24] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. Publaynet: largest dataset ever for document layout analysis. *arXiv preprint arXiv:1908.07836*, 2019.

[25] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.