Improving the Robustness of Word Detection in Printed Indic Documents

Abhayram A Nair, Tallapragada Shanmukha Sreevatsa, and Ravi Kiran Sarvadevabhatla

International Institute of Information Technology, Hyderabad
abhayram.a@students.iiit.ac.in
190340085@klh.edu.in
ravi.kiran@iiit.ac.in

Abstract. Robust printed text detection is a crucial preliminary component for many document image understanding pipelines. Although many state-of-the-art text detection models exist, their evaluated performance on printed Indic documents is affected by various issues, e.g. visual diversity and density of Indic scripts, scanning artifacts and existence of malformed ground truths in the dataset. To tackle these challenges, we propose an automatic approach which first tags the documents in terms of their text detection difficulty. We then use an imbalanced sampling procedure during training to account for difficulty class imbalance. Subsequently, we fine-tune an existing state-of-the-art model on a large-scale multilingual Indic document dataset containing 53,529 documents annotated with word text layout. We also introduce an approach to filter out documents with mislabelled ground truths from the dataset. The resulting robust text detector significantly improves document-level text detection. Specifically, a 46.5% improvement over the existing state-of-the-art model's F1 score (0.65) is observed. The improvement is especially prominent for 'hard' difficulty level documents – a 117% improvement is seen over the F1 score (0.42).

Keywords: Text Detection · Printed Indic Documents · OCR.

1 Introduction and Related Works

Text layout understanding has been a very important research area for many years[18,28,30]. With the advancement of technology at the digital forefront, printed documents have become more useful and easy to access. A document typically contains textual and visual information. The task of recognizing characters/textual elements from an image is called Optical Character Recognition, generally referred to as OCR[13,12]. There are many open-source[19,8,33,29,15,9] and commercially available OCR systems[34,6,10] that perform well on multiple languages and multiples scripts.

A text detection system can produce bounding boxes for words, lines, or paragraphs. These bounding boxes are called regions of interest (ROI). For the

Fig. 1: Examples of bounding boxes where parts of words are omitted/falsely predicted

OCR pipeline, text detection at the word level is optimal. The recognition task is greatly affected if the bounding box fails to include all parts of a word. For example, in the case of many Indic scripts, if the *matras* (diacritics) are not included in the bounding box, it may lead to faulty recognition (Refer to Fig. 1).

Detecting text from printed Indic documents has many challenges[14]. Two major challenges are non-standard font designs and the bad quality of the paper used for printing[27]. Ink bleeds are one of the major issues which arise due to the quality of the paper. Some of the most frequently observed issues of the state-of-the-art text detection model on Indic scripts are:

1. White space and Ink-bleed predictions (Refer to [11])
2. Multiple words are detected as a single word and vice-versa (Refer to [11])
3. Diacritics and conjunct consonants above or below alphabets cause separated bounding boxes during prediction (Refer to [11])
4. Words towards the right-end of the page not detected (Refer to [11])
5. Non-textual objects falsely detected as words (Refer to [11])

A system that works well considering all the above-mentioned challenges and issues is required for detecting text. There have been various language-dependent and language-independent[24,22,2,32] approaches for text detection and recognition tasks[36]. Work has been done to detect text from non-Indic scripts like English[1], Chinese[37], Japanese[3,5], Korean[16], Farsi[26], and Kanji[35] as well as from Indic scripts[4,20,27,31]. Although many state-of-the-art text detection systems exist, the performance of those systems when evaluated on printed Indic documents is not satisfactory. The factors that affect the performance of state-of-the-art text detection systems are the visual diversity and density of Indic scripts, scanning artifacts and misaligned/bad ground truths in the dataset.

We introduce a semi-automatic approach to remove the documents with mislabelled ground truths. To tackle other problems, we propose an automatic approach to tagging the documents based on the difficulty of text detection and performing imbalanced learn using oversampling[25]. This paper presents an adaptation of the state-of-the-art text detection model[21] from the docTR OCR system[7] and has been finetuned on our large-scale multilingual Indic document dataset.

Refer to https://github.com/Abhayram-A-Nair/Robust-word-detector-for-Indic-Documents for the model and demo.

2 Dataset

The dataset that we use is called the Consortium dataset. It is a dataset containing images and ground truths from 13 languages - Hindi, Malayalam, Kannada, Gurumukhi, Oriya, Assamese, Urdu, Telugu, Tamil, Gujarathi, Marathi, Bangla, and Manipuri. It contains 55783 documents and the images are taken from various different printed books written in that language. The ground truths for the images in this dataset contain both line and word-level bounding boxes.

Table 1: Consortium Dataset

Language	Number of books	Number of documents
Hindi	33	5042
Malayalam	31	5458
Kannada	33	3845
Gurumukhi	32	5025
Oriya	17	5024
Assamese	19	3467
Urdu	9	1566
Telugu	27	4896
Tamil	23	4778
Gujarathi	25	5243
Marathi	22	5028
Bangla	13	2771
Manipuri	25	3640

3 Methodology

We start by using the data to perform an evaluation of the current state-of-the-art text detection model. Observing the results of the state-of-the-art text detection model gives us an understanding of the scenarios where the model is not performing well. One such scenario is when there are errors in the ground truth bounding boxes, for example when they are misaligned or shifted. Identification and removal of such kinds of documents improve the dataset which results in a better model after fine-tuning. The dataset is then sorted into three categories based on the difficulty of word detection. Then using class balancing, we fine-tune the state-of-the-art model on our dataset.

3.1 Removal of Documents with Malformed Ground Truths

The main categories of malformed ground truths observed in the dataset are shifted, scaled, and extended ground truths. These are an issue for the model while learning and hence need to be filtered out.

To do this, we introduce a filter that we call the predicted box overlap filter. We define an overlap metric that has to be satisfied for a certain percentage of the ground truth bounding boxes in a document. The overlap value(t) for a ground truth bounding box is given by

$$t = min(1 - \frac{p \cap g}{g}) \tag{1}$$

Where g is the area of the ground truth bounding box and p is the area of any prediction bounding box in that document. The overlap metric is then said to be satisfied for that ground truth bounding box if t$\leq \theta$, where θ is a threshold value which we have chosen to be 0.1.

The overlap metric is high for a ground truth bounding box if it overlaps significantly with at least one prediction bounding box. The state-of-the-art model is generally capable of correctly detecting most of the words in a document. Thus, if most of the ground truths do not have a corresponding prediction, then the ground truths are shifted with respect to the words. Similarly, if most of the ground truths are extended beyond their corresponding words, they will have a large area that does not overlap with their corresponding prediction. Therefore, for a document with shifted/extended ground truths, we have noticed that most of the ground truths do not have a significant overlap with any of the prediction bounding boxes. Hence, to keep good documents and filter out documents with bad ground truths, at least 70% of the total ground truths in a document should have at most 10% of its area not intersecting with the prediction bounding box.

3.2 Assigning Tags to Document Images Based on Difficulty

Keeping in mind the OCR pipeline, it is much more important that the model is able to correctly identify a higher proportion of bounding boxes than being able to make bounding box predictions that are always correct. Hence, an emphasis is placed on reducing the number of False Positives (FP) and correctly identifying the valid text bounding boxes. Since recall corresponds to the proportion of actual positives (ground truth bounding boxes) that are correctly identified by our model, it is chosen for tagging the documents based on difficulty.

For a model-predicted bounding box to be labeled as a true positive, we consider the maximum Intersection Over Union (IOU) of that bounding box with all the ground truth bounding boxes present in the document. A threshold value of IOU is taken, over which the predicted bounding box is considered a true positive (TP) and below which the predicted bounding box is considered a false positive (FP). A ground truth bounding box is considered a false negative (FN) if its maximum IOU is less than the threshold value i.e. when the ground truth does not have an associated true positive.

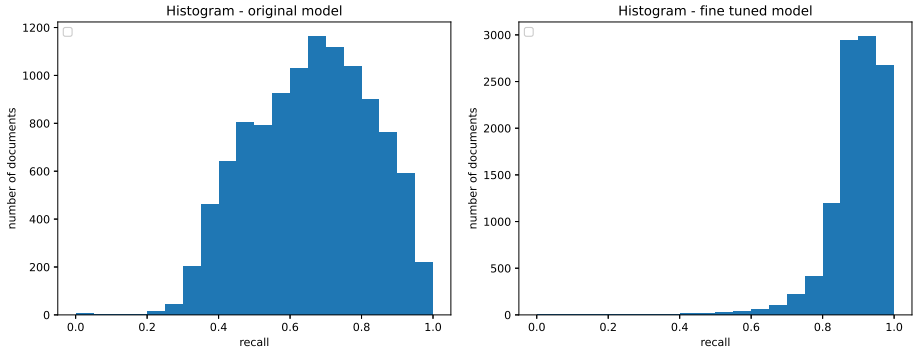
Using the above definitions, we come up with a modified recall metric (R).

$$R = \frac{\sum_{i=1}^{N} IOU_i}{\sum_{i=1}^{N} IOU_i + FN} \tag{2}$$

Where N is the number of true positives and IOU_i is the IOU of the i'th true positive.

This modified recall metric, similar to the standard recall metric expresses the proportion of actual positives that are identified correctly. However, instead of considering each true positive as binary, each true positive is weighted according to its corresponding IOU value. This places an emphasis on the bounding boxes having a higher IOU value, which would mean that the predicted bounding box more closely resembles the ground truth for higher values of R.

All the documents, irrespective of language, are sorted based on their modified recall value. The sorted values are then plotted and suitable cutoffs are chosen based on this plot for classifying these documents into "Easy", "Medium" and "Hard". The three categories refer to the difficulty faced by the model in carrying out predictions on that particular document. Therefore, from figure: 2a "Easy" documents are those documents that have R>0.696, "Medium" documents are those documents that have 0.489<R≤0.696 and "Hard" documents are those documents that have R≤0.489, where 0.489 and 0.696 are the chosen cutoffs. There is an uneven distribution of documents as a higher percentage of documents are falling into the EASY category. To prevent model learning from EASY docs, tagging the documents based on difficulty ensures equal sampling of data from all categories while training. Refer to this GitHub page [11] for examples of Easy, Medium, and Hard categories of documents.



(a) Histogram Plot - state-of-the-art model (b) Histogram Plot - Fine-tuned model

Fig. 2: (a) and (b) shows the histograms of recall values of documents given by the state-of-the-art model and the fine-tuned model.

3.3 Class Balancing the Consortium Dataset

We notice that the state-of-the-art model is able to perform well for many of the documents in the dataset. Because of this, there are a large number of "Easy" documents, a lesser number of "Medium" documents, and an even lesser number of "Hard" documents. If the dataset is used for fine-tuning the model without modification, it will lead to the model learning shortcuts for performing even better on the "Easy" documents. At the same time, while fine-tuning, the updation of weights will not be sufficiently impacted by the "Hard" documents because they are present in lesser numbers. To counteract this, we duplicate the existing "Hard" samples such that the model sees more of the "Hard" documents during each epoch. This is also done for "Medium" documents.

For each minibatch, the probability that a sample is from the "Easy" class is taken as p_e, the probability that a sample is from the "Medium" class is taken as p_m and the probability that a sample is from the "Hard" class is taken as p_h. For our original unmodified dataset, if we are randomly sampling each minibatch, $p_h < p_m < p_e$ because the number of "Easy" documents is greater than the number of "Medium" documents which in turn is greater than the number of "Hard" documents. However, we want the model to preferentially learn from the "Hard" documents and "Medium" documents over the "Easy" documents. Hence, implementation of imbalanced learn should leave us with new probabilities p'_e, p'_m and p'_h which are the probabilities, for each minibatch, that the sample is from "Easy", "Medium" and "Hard". The new probabilities should follow $p'_h > p'_m > p'_e$. To achieve this, the new probabilities are chosen using inverse transform sampling. Therefore, the new probabilities are as follows

$$p'_e = 0.5 * (1 - p_e) \tag{3}$$

$$p'_m = 0.5 * (1 - p_m) \tag{4}$$

$$p'_h = 0.5 * (1 - p_h) \tag{5}$$

Because $p'_h > p'_m > p'_e$, in each epoch the same "Hard" document or "Medium" document will have to be seen multiple times. Hence, the documents from the "Hard" and "Medium" classes are randomly duplicated in the training dataset so that, for each epoch, when randomly sampling a minibatch, the probability that a sample is "Hard" is p'_h and the probability that a sample is "Medium" is p'_m. Refer to figures: 3 for the distribution of Easy, Medium, and Hard documents before and after class balancing.

3.4 Fine Tuning the Model

Fine-tuning the model is done by freezing the feature extraction layers and updating the weights of the later layers of the neural network during training. This is because neural networks abstract out information through their layers. The initial layers of a neural network are generally for feature extraction, where generic features independent of the current task are abstracted out. The layers that come after that work on these features and are much more specific to

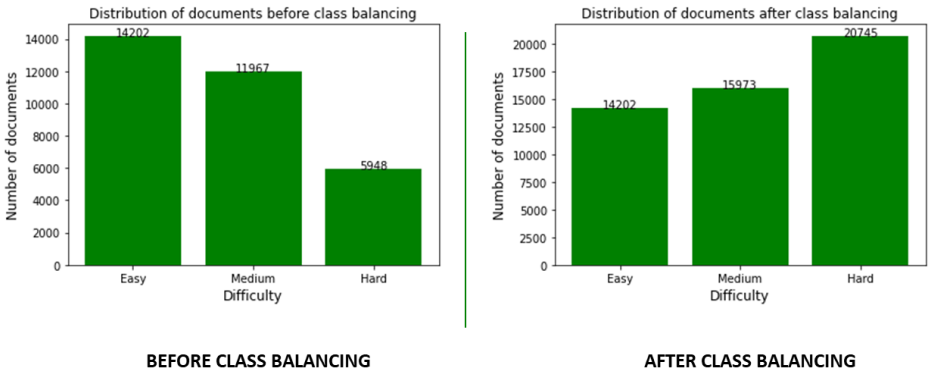


Fig. 3: Distribution of documents before and after class balancing

the task at hand. By freezing the initial feature extraction layers of the neural network, general features that are meaningful across various tasks can be extracted while the last few layers are fine-tuned for the particular task. Training the model is done on the expanded dataset containing duplicates of "Hard" and "Medium" documents. Each minibatch is randomly sampled from this expanded dataset.

The Adam[17] optimizer was used along with a Cosine Annealing[23] learning rate scheduler. For Adam $\beta_1 = 0.95$, $\beta_2 = 0.99$, $\epsilon = 10^{-6}$, and a weight decay of 0.001 were taken and the minimum learning rate parameter for Cosine Annealing was taken to be $4 * 10^{-8}$. Fine-tuning was run for a total of 20 epochs with an initial learning rate of 0.01.

4 Results

4.1 Performance of Fine-Tuned Model

On average, the state-of-the-art model was able to give an F1-score of 0.655, whereas the fine-tuned model is able to give an average F1-score of 0.886 on the Consortium dataset with an average per-document percentage increase in the F1-score of 46.55%. There is a more prominent increase in documents tagged "Hard", with an average increase of 117%, giving an average F1-score of 0.887107. "Medium" documents end up having the highest F1-score, followed by "Hard" and then by "Easy".

The model performs the best on Marathi with an F1-score of 0.951 and performs the worst on Tamil with an F1-score of 0.775. The per-document per-

centage increase in F1-score is highest for Gurumukhi which started off with the lowest value pre-fine-tuning. On the other hand, Tamil has the lowest percentage increase. We see that the three languages, Malayalam, Kannada, and Tamil, which had the highest F1 scores have the lowest percentage increases, with Kannada and Tamil being the languages with the lowest F1 scores post fine-tuning. This could be because of the larger number of documents with Hindi-like scripts in the dataset which bias the model during fine-tuning. It can also be argued that the percentage increase is lesser for these languages due to them starting off with a higher F1 score. However, post-fine-tuning none of the non-Hindi-like scripts (Malayalam, Tamil, Telugu, Oriya, Kannada) got an F1-score greater than 0.9, which 5 out of the 7 other Hindi-like scripts were able to achieve. Below Table 2 contains F1-scores for state-of-the-art and fine-tuned models along with the percentage increase in F1-score for all the languages.

Table 2: F1 scores for state-of-the-art and fine-tuned models along with the percentage increase with respect to each language

Language	F1 score - SOTA(db_resnet50) [21]				F1 score - Fine-tuned(db_resnet50 v2)			
	Easy	Medium	Hard	All	Easy	Medium	Hard	All**(%-increase)**
Assamese	0.77	0.65	0.46	0.66	0.88	0.87	0.78	0.87**(33.49)**
Bangla	0.78	0.61	0.44	0.65	0.91	0.91	0.84	0.90**(42.87)**
Gujarati	0.73	0.57	0.41	0.59	0.91	0.90	0.87	0.90**(57.53)**
Gurumukhi	0.81	0.55	0.41	0.46	0.88	0.93	0.91	0.91**(112.24)**
Hindi	0.77	0.57	0.42	0.51	0.87	0.89	0.90	0.90**(89.84)**
Kannada	0.78	0.60	0.36	0.75	0.85	0.83	0.62	0.85**(15.34)**
Malayalam	0.83	0.61	0.30	0.82	0.89	0.82	0.53	0.89**(13.19)**
Manipuri	0.76	0.64	0.47	0.68	0.89	0.88	0.71	0.88**(31.39)**
Marathi	0.78	0.62	0.37	0.72	0.95	0.94	0.89	0.95**(35.32)**
Oriya	0.72	0.60	0.46	0.57	0.91	0.89	0.87	0.89**(58.49)**
Tamil	0.82	0.52	0.31	0.80	0.79	0.43	0.29	0.77**(1.05)**
Telugu	0.74	0.59	0.40	0.62	0.88	0.88	0.79	0.87**(45.96)**
All	0.79	0.60	0.42	0.65	0.88	0.89	0.88	0.88**(46.54)**

4.2 Comparative Study

The issues mentioned in Section: 1 are resolved with our fine-tuned model. As seen in Table 3, precision, recall, and F1-score all show a considerable increase in our fine-tuned model compared to the state-of-the-art model. Refer to the GitHub page [11] for examples of document images comparing the performance of the state-of-the-art and fine-tuned models.

Table 3: Average Recall, Precision and F1-score over all documents for State-of-the-art and Fine-tuned models

Model	Recall	Precision	F1-score
State-of-the-art (db_resnet50) [21]	0.660	0.657	0.655
Fine-tuned (db_renset50 v2)	**0.894**	**0.884**	**0.886**

5 Conclusion

In this paper, we have shown that the performance of the state-of-the-art model for Indic documents has several issues. Using our fine-tuned model we are able to fix these issues. We have also come up with a semi-automatic approach to removing malformed ground truths present in the dataset. Through this, we contribute a robust model that performs well on printed Indic documents.

References

1. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character region awareness for text detection (2019). https://doi.org/10.48550/ARXIV.1904.01941 2
2. Bai, B., Yin, F., Liu, C.L.: A fast stroke-based method for text detection in video. In: 2012 10th IAPR International Workshop on Document Analysis Systems. pp. 69–73 (2012). https://doi.org/10.1109/DAS.2012.3 2
3. Bjerregaard, N.K., Cheplygina, V., Heinrich, S.: Detection of furigana text in images (2022). https://doi.org/10.48550/ARXIV.2207.03960 2
4. Chaudhuri, B.B.: On OCR of Major Indian Scripts: Bangla and Devanagari, pp. 27–42. Springer London, London (2010), https://doi.org/10.1007/978-1-84800-330-9_2 2
5. Del Gobbo, J., Herrera, R.M.: Unconstrained text detection in manga: a new dataset and baseline (2020). https://doi.org/10.48550/ARXIV.2009.04042 2
6. Google Cloud Document AI. https://cloud.google.com/document-ai 1
7. docTR. https://mindee.github.io/doctr/ 2
8. Du, Y., Li, C., Guo, R., Yin, X., Liu, W., Zhou, J., Bai, Y., Yu, Z., Yang, Y., Dang, Q., Wang, H.: Pp-ocr: A practical ultra lightweight ocr system (2020). https://doi.org/10.48550/ARXIV.2009.09941 1
9. EasyOCR. https://github.com/JaidedAI/EasyOCR 1
10. ABBYY FineReader. https://pdf.abbyy.com/ 1
11. GitHub page for comparative results and figures. https://vatsasree.github.io/ 2, 5, 8
12. Hamad, K., Kaya, M.: A detailed analysis of optical character recognition technology. International Journal of Applied Mathematics, Electronics and Computers **4**, 244–244 (12 2016). https://doi.org/10.18100/ijamec.270374 1
13. Islam, N., Islam, Z., Noor, N.: A survey on optical character recognition system (2017). https://doi.org/10.48550/ARXIV.1710.05703 1
14. Jawahar, C., Kumar, A., Phaneendra, A., Jinesh, K.: Building Data Sets for Indian Language OCR Research, pp. 3–25. Springer London, London (2010), https://doi.org/10.1007/978-1-84800-330-9_1 2

15. Keras OCR. https://github.com/faustomorales/keras-ocr 1
16. Kim, G., Son, J., Lee, K., Min, J.: Character decomposition to resolve class imbalance problem in hangul ocr (2022). https://doi.org/10.48550/ARXIV.2208.06079 2
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014). https://doi.org/10.48550/ARXIV.1412.6980 7
18. Kovács-V, Z.M., Guerrieri, R., Baccarani, G.: On Hand-Printed Character Recognition, pp. 37–48. Springer US, Boston, MA (1995), https://doi.org/10.1007/978-1-4899-1088-2_2 1
19. Kuang, Z., Sun, H., Li, Z., Yue, X., Lin, T.H., Chen, J., Wei, H., Zhu, Y., Gao, T., Zhang, W., Chen, K., Zhang, W., Lin, D.: Mmocr: A comprehensive toolbox for text detection, recognition and understanding (2021). https://doi.org/10.48550/ARXIV.2108.06543 1
20. Lehal, G.S.: A Complete Machine-Printed Gurmukhi OCR System, pp. 43–71. Springer London, London (2010), https://doi.org/10.1007/978-1-84800-330-9_3 2
21. Liao, M., Wan, Z., Yao, C., Chen, K., Bai, X.: Real-time scene text detection with differentiable binarization (2019). https://doi.org/10.48550/ARXIV.1911.08947 2, 8, 9
22. Liu, X., Fu, H., Jia, Y.: Gaussian mixture modeling and learning of neighboring characters for multilingual text extraction in images. Pattern Recognition **41**(2), 484–493 (2008). https://doi.org/https://doi.org/10.1016/j.patcog.2007.06.004 2
23. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts (2016). https://doi.org/10.48550/ARXIV.1608.03983 7
24. Lyu, M., Song, J., Cai, M.: A comprehensive method for multilingual video text detection, localization, and extraction. IEEE Transactions on Circuits and Systems for Video Technology **15**(2), 243–255 (2005). https://doi.org/10.1109/TCSVT.2004.841653 2
25. Mohammed, R., Rawashdeh, J., Abdullah, M.: Machine learning with oversampling and undersampling techniques: Overview study and experimental results. In: 2020 11th International Conference on Information and Communication Systems (ICICS). pp. 243–248 (2020). https://doi.org/10.1109/ICICS49469.2020.239556 2
26. Mosannafat, M., Taherinezhad, F., Khotanlou, H., Alighardash, E.: Farsi text detection and localization in videos and images. In: 2022 9th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS). pp. 1–6 (2022). https://doi.org/10.1109/CFIS54774.2022.9756472 2
27. Neeba, N., Namboodiri, A., Jawahar, C.V.and Narayanan, P.: Recognition of Malayalam Documents, pp. 125–146. Springer London, London (2010), https://doi.org/10.1007/978-1-84800-330-9_6" 2
28. O'Gorman, L., O'Gorman, L., Kasturi, R.: Document Image Analysis: An Executive Briefing. IEEE Computer Society Press, Washington, DC, USA, 1st edn. (1997) 1
29. PaddleOCR. https://github.com/PaddlePaddle/PaddleOCR/ 1
30. Pal, U., Chaudhuri, B.: Indian script character recognition: a survey. Pattern Recognition **37**(9), 1887–1899 (2004). https://doi.org/https://doi.org/10.1016/j.patcog.2004.02.003 1
31. Sankaran, N., Jawahar, C.V.: Recognition of printed devanagari text using blstm neural network. Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012) pp. 322–325 (2012) 2

32. Shivakumara, P., Huang, W., Tan, C.L.: Efficient video text detection using edge features. In: 2008 19th International Conference on Pattern Recognition. pp. 1–4 (2008). https://doi.org/10.1109/ICPR.2008.4761415 2
33. Tesseract-OCR. https://github.com/tesseract-ocr 1
34. Amazon Textract. https://aws.amazon.com/textract/ 1
35. Xu, L., Nagayoshi, H., Sako, H.: Kanji character detection from complex real scene images based on character properties. pp. 278–285 (09 2008). https://doi.org/10.1109/DAS.2008.34 2
36. Ye, Q., Doermann, D.S.: Text detection and recognition in imagery: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**, 1480–1500 (2015) 2
37. Zhou, J., Liang, Y., Chen, M.: A lightweight cnn for large-scale chinese character recognition. In: 2022 4th International Conference on Advances in Computer Technology, Information Science and Communications (CTISC). pp. 1–5 (2022). https://doi.org/10.1109/CTISC54888.2022.9849794 2