Optical Character Recognition for the International Phonetic Alphabet

Shu Okabe*[1,2] and Dejvi Zelo*[1] and Alexander Fraser[1,2,3]
[1]School of Computation, Information and Technology, Technische Universität München (TUM)
[2]Munich Center for Machine Learning
[3]Munich Data Science Institute
* Equal contribution
Corresponding authors: `{shu.okabe,dejvi.zelo}@tum.de`

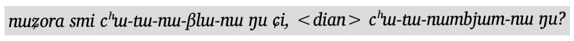
Abstract

As grammar books are increasingly used as additional reference resources specifically for very low-resource languages, a significant portion comes from scans and relies on the quality of the Optical Character Recognition (OCR) tool. We focus here on a particular script used in linguistics to transcribe sounds: the International Phonetic Alphabet (IPA). We consider two data sources: actual grammar book PDFs for two languages under documentation, Japhug and Kagayanen, and a synthetically generated dataset based on Wiktionary. We compare two neural OCR frameworks, Tesseract and Calamari, and a recent large vision-language model, Qwen2.5-VL-7B, all three in an off-the-shelf setting and with fine-tuning. While their zero-shot performance is relatively poor for IPA characters in general due to character set mismatch, fine-tuning with the synthetic dataset leads to notable improvements.

1 Introduction

Grammar books have been used in Natural Language Processing (NLP) as a main or additional resource to gain high-quality knowledge about languages, especially when they are low-resource. They have notably been used to improve large language models for Machine Translation (Tanzer et al., 2024; Hus and Anastasopoulos, 2024; Zhang et al., 2024), or to extract examples with linguistic annotations (or Interlinear Glossed Texts), such as in the ODIN (Lewis and Xia, 2010) or IMTVault (Nordhoff and Krämer, 2022) corpora.

However, most of such linguistic documents, when digitised, are only available as scans and need to go through an Optical Character Recognition (OCR) tool. For example, the ODIN dataset relied on the OCR of scanned PDFs of grammar books to extract linguistic annotations. It was, however, noisy due to the poor quality of the original scans, but also of the OCR tools, as noted in (Xia et al., 2014) and (Nordhoff and Krämer, 2022).


nɯʑora smi c^{h}ɯ-tɯ-nɯ-βlɯ-nɯ ŋu ɕi, <dian> c^{h}ɯ-tɯ-nɯmbjɯm-nɯ ŋu?

OCR	nɯʑora smi c^{h}ɯ–tɯ–nɯ–βlɯ-nɯ ŋu ɕi, <dian> c^{h}ɯ–tɯ–nɯmbjɯm–nɯ ŋu?
EN	Do you burn a fire, or do you get warm with an electric [radiator]?

Figure 1: An example sentence from the Japhug grammar book (Jacques, 2021), synthetically corrupted (top), with the reference IPA transcription (middle) and English translation (bottom).

In this work, we focus on a specific script that often appears in linguistic contexts: the International Phonetic Alphabet (IPA) (International Phonetic Association, 1999). IPA characters, which are used to transcribe speech by linguists, present specific challenges graphically from their unique symbols, as well as diacritics and similar characters. They occur in grammar books to transcribe words or sometimes to transcribe an oral language with no fixed writing convention (as in Figure 1).

In this article, we study the OCR quality specifically for IPA characters when used in a linguistic context. We focus on two realistic language scenarios where such characters are used in grammar books: (i) for any language (e.g., in the phonology chapter) and (ii) for a specific language transcribed with an IPA-based writing system. We thus aim at a first comprehensive study of OCR feasibility with *open-source* models for such IPA characters, in an effort to improve the digitisation of books that have only been scanned. We compare two neural OCR models, Tesseract (Smith, 2007) and Calamari (Wick et al., 2020), and a Large Vision-Language Model (LVLM), Qwen2.5-VL (Bai et al., 2025). We release the code material for the training and evaluation datasets, alongside the trained models, publicly.[1]

[1]`https://github.com/TUM-NLP/ocr-ipa`.

2 Related works

Few works performed OCR on IPA characters. The first is (Ashby, 2017), which converted a scanned English text written phonetically using IPA. The author used a customised ABBYY FineReader 12 Professional, which is a paid proprietary model. In this setting, the model only had a reduced set of IPA characters to recognise (i.e., those occurring in English). The more recent study on OCR for IPA characters is from (Li and Hill, 2023), where a list of words in Chinese characters was associated with the pronunciation of their translation in Burmish languages, using Transkribus.

To our knowledge, this work is the first study of OCR focusing on a broad range of IPA characters, especially to enhance the conversion of grammar book scans. We also focus on open-source models to better control the training step and ensure reproducibility.

OCR for low-resource languages Identifying IPA characters also shares similarities with the OCR task for low-resource languages, as the data and model coverage are scarce. The diacritics or similar characters add another layer of complexity, although the IPA is based on the Latin script. Rijhwani et al. (2020, 2021) successfully rely on post-correction to tackle the data scarcity for endangered languages. More generally, Agarwal and Anastasopoulos (2024) survey the OCR task for low-resource languages and notably recommend starting from an off-the-shelf system. We follow this approach in this work.

3 Datasets

As we need the gold transcription to train and evaluate, we rely on two types of datasets in our work: PDFs from actual grammar books and a synthetically generated dataset from the online dictionary Wiktionary. We only use images generated from the ground truth and not actual scans to control the proportion of noise. Besides, in this work, we assume that the PDF scans have already been preprocessed and segmented to obtain lines (i.e., we perform line-level OCR). General pipelines exist to clean and process the original images upstream.

3.1 PDFs from actual grammar books

The first source of IPA data is from grammar books of two low-resource languages.

Japhug Japhug is a Sino-Tibetan language spoken in Southwestern China (ISO code: `jya`; Glottolog: `japh1234`). The language has been extensively described in its grammar book (Jacques, 2021), where the linguist chose to transcribe it using an IPA-based system, notably for phonological purposes. For this resource, we hence focus on the example sentences or words in Japhug that illustrate the presented grammatical concepts (cf. Figure 1). The publisher, Language Science Press, releases the LaTeX source code of their books[2] with a CC-BY licence. This allows us to obtain the true transcription of the sentences or phrases, following a similar extraction methodology to (Nordhoff and Krämer, 2022; Okabe et al., 2022). Long strings are split into smaller chunks of 3 to 8 words, which gives us 13k samples.

Kagayanen Similarly, from the same publisher, we also consider the grammar book of Kagayanen[3] (Pebley and Payne, 2024), an Austronesian language spoken in the Philippines (`cgc`; `kaga1256`). Here, no full example sentences are written in IPA, but the transcriptions of some words are, especially in the Phonology section. The dataset is thus smaller in size (834 samples).

We can now represent both uses of IPA in grammar books: either for the transcription of the language altogether (`japhug`) or for the pronunciation of words (`kagayanen`).

3.2 Synthetic corpus from Wiktionary

Given the relatively small size of both datasets, we also rely on a synthetic corpus using a second resource: Wiktionary, a collaborative online dictionary for many languages.[4] We use the data scraped by Wikipron (Lee et al., 2020), which consists of around four million IPA transcriptions of words from the website for 307 languages (in 2025).[5] We note here that neither Japhug nor Kagayanen are in the Wikipron scrape. In Wiktionary, many words feature a rather simplified (broad) or detailed (narrow) IPA transcription; we use both to increase the final dataset size. Besides, we normalise the data by notably removing stress markers, syllable boundaries, and tone markers in this study.

[2] `https://github.com/langsci/295`. We note that we used the *first* (and now superseded) edition of the grammar book, as it was the only available version.

[3] `https://github.com/langsci/443`.

[4] Wiktionary has both the CC-BY-SA and GFDL licences.

[5] `https://github.com/CUNY-CL/wikipron/tree/master/data/scrape` (from early 2025).

Synthetic sentence generation As Wiktionary is a dictionary listing *words*, while grammar books can also feature full sentences transcribed in the IPA, we generate random combinations of words from a given language to approximate sentences in actual grammar books. We use the true distribution of words per line in the Japhug grammar book to sample realistic numbers of words. We also add punctuation marks that are actually present in linguistic grammar examples (but not in a word list), such as commas, periods, or brackets. To do so, we use probabilistic rules to reproduce grammar book sentences, where each symbol has a fixed probability (e.g., 0.01 in most cases) to appear before ([), between (-), or after (,) a unit. This step leads to longer and more challenging images to process, with around 1.1M pseudo-sentences.

Besides, we identified five different fonts that support IPA symbols (Charis SIL, Doulos SIL, Gentium, Andika, and Brill).[6] We increase the size of the synthetic dataset by randomly applying one typeface and one font style to words to diversify the images. This increases the dataset size threefold. We denote this dataset `synth`, which contains 3.4M samples.

3.3 Dataset summary

Finally, for all three datasets, we corrupted the images using traditional techniques, such as rotation (tilting), Gaussian noise injection, and contrast and brightness adjustments. All images are rendered to have a height of 70 px.

Due to their size, we mainly consider the `japhug` and `synth` datasets in our experiments and divide them into training, development, and test sets based on a 90:5:5 split. As it is considerably smaller, we only use the `kagayanen` set for evaluating the robustness of our models. Although the synthetic dataset is significantly larger than the `japhug` dataset, the latter contains actual and in-domain (i.e., from a grammar book) samples for our study.

Table 1 displays the number of samples ($N_{samples}$) and fonts (N_{fonts}) for all three datasets: `japhug`, `kagayanen`, and `synth`. Here, samples correspond to words and sentences, as they both occur in grammar books.

[6]The actual font used in the grammar books is Charis SIL.

dataset	train	data source	$N_{samples}$	N_{fonts}
`japhug`	✓	grammar	13k	1
`kagayanen`	✗	grammar	834	1
`synth`	✓	Wiktionary	3.4M	5

Table 1: General statistics of the three IPA datasets.

4 OCR-specific models

4.1 Base models and training

We compare two neural and open-source OCR frameworks: Tesseract and Calamari.

Tesseract Tesseract is an open-source OCR framework (Smith, 2007) recently used to carry out OCR for low-resource languages (Ignat et al., 2022; *inter alia*). We start from pre-trained LSTM-based models[7] that Tesseract 5 provides. We consider the English- and Latin-based models, as they both feature a large degree of overlap in characters with the IPA. Notably, the Latin model has a more extensive support for diacritics. We denote them `tess_eng` and `tess_lat`, respectively.

Calamari We choose Calamari, as Wick et al. (2020) report better performance than Tesseract on their benchmark. We start from the `idiotikon` model, which we identified as the best pre-trained variant of the model[8] for IPA recognition. It was originally trained to digitise a Swiss German dictionary, the *Schweizerisches Idiotikon*, and supports a larger codec including diacritics. We note that Calamari natively supports GPU, which leads to faster computation. We denote the base model `cal_ipa`.

Fine-tuning We then fine-tune all three base models on either `japhug` for the monolingual grammar book domain or `synth` for the synthetic multilingual training dataset. We mainly use the default hyperparameter setting. We mark models trained on the `japhug` training dataset with a `_Ja` ending and on the `synth` training dataset with `_Sy`. Appendix A presents more details for reproducibility.

4.2 Evaluation method

We evaluate our experiments with the usual character and word error rates (CER and WER, respectively). Both are based on the number of edits (insertions, deletions, or substitutions) needed between the reference and the prediction at their re-

[7]`https://github.com/tesseract-ocr/tessdata`.

[8]`https://github.com/Calamari-OCR/calamari_models`.

spective levels. For both frameworks, we will consider the model with the lowest average CER and WER on the japhug *and* synth test datasets as best.

4.3 Experimental results

test set	japhug		synth	
model	CER	WER	CER	WER
tess_eng	33.94	89.82	36.44	100.95
tess_eng_Ja	2.56	13.96	41.71	96.59
tess_eng_Sy	11.86	46.71	5.14	40.51
tess_lat	32.46	90.69	34.02	97.70
tess_lat_Ja	2.42	13.85	43.08	101.07
tess_lat_Sy	10.82	39.91	4.41	35.10
cal_ipa	52.36	94.47	57.12	102.87
cal_ipa_Ja	0.60	2.09	37.37	95.36
cal_ipa_Sy	1.41	6.28	0.20	2.45

Table 2: Comparison of the English-based, Latin-based Tesseract, and Calamari models before and after fine-tuning on japhug and synth.

Table 2 presents the results of the base Tesseract and Calamari models and their fine-tuned versions on the japhug and synth test datasets. First, we observe that the three off-the-shelf performances are rather poor, with notably high WER for both the Japhug and synthetic datasets. The larger character support of tess_lat seems to help only slightly; the codec mismatch leads to errors for all non-supported IPA characters. The base Calamari model also lags behind, with similar or worse error rates than the base Tesseract models (more than 18 points of difference for CER).

Fine-tuning drastically improves recognition performance, as the IPA symbols are now supported. Fine-tuning on the grammar book leads to the best in-domain score, but prevents it from generalising well on other languages (i.e., synth). We see this with the increase in CER on the synthetic corpus across all models trained on japhug. On the other hand, models trained on synth seem more robust, helped by the larger training data. Error rates drop significantly compared to the baseline performance for both datasets. Overall, the best model is cal_ipa_Sy, with its better generalisation, reaching less than 2 and 7 points of CER and WER, respectively. We also tried oversampling complex characters for both models in Appendix B.

5 LVLM: Qwen2.5-VL 7B

5.1 Model setting

As LVLMs emerge as a generalist tool to convert images into text, we also consider one of the latest models for the OCR task. We compared several open- and closed-source LVLMs across *reported* performance on two benchmarks, CC-OCR (Yang et al., 2024) and OCRBench v2 (Fu et al., 2025), specifically for the OCR task. Based on the performance and model size, we chose the open-source Qwen2.5-VL 7B (Bai et al., 2025) model. Appendix C details the model choice and the prompt.

Model fine-tuning As in the previous two models, the main challenge comes from the unseen IPA characters. We expand the tokeniser vocabulary of the model by adding all unique characters from the training dataset and initialise their weights randomly. We use a Quantised Low-Rank Adaptation (QLoRA; Dettmers et al., 2023) to fine-tune the model.

Dataset distillation As the synthetic dataset proved to be more beneficial in terms of generalisation, we only focus on fine-tuning using the synth dataset. Moreover, computational constraints meant that we could not fine-tune on the same entire training dataset. Hence, given its scale, we perform dataset distillation to reduce it, while ensuring that all characters, especially rare ones, are represented. The distillation leads to a dataset around half the original size, with around 1.7M samples; we denote it dist_synth.

5.2 Results

Table 3 compares the results of the zero-shot Qwen2.5-VL (qwen_zs) with its fine-tuned version on the distilled synth dataset (qwen_ft). We evaluate on two test sets: the same japhug as earlier and a subset extracted from the distilled dataset.

test set	japhug		dist_synth	
model	CER	WER	CER	WER
qwen_zs	28.36	84.41	35.42	98.23
qwen_ft	2.89	15.47	0.12	1.36

Table 3: Performance of the Qwen model before and after fine-tuning on the distilled synth dataset.

As expected, we observe the same trend as before: the zero-shot performance is hindered by the character set mismatch, whereas fine-tuning on our synthetic dataset proved to be effective, even when distilled. For in-domain synthetic entries, the error rate drops drastically, reaching less than 2 points of error, while for Japhug, which is not part of the fine-tuning dataset, the model also improves.

Although we cannot directly compare the numerical results for Qwen with the previous two OCR models due to dataset distillation and changed splits, we observe similar error rates, especially at the character level. We note that compared to the previous models, the computational cost is significantly higher for this model, as discussed in Appendix D.

6 Analysis

6.1 Qualitative analysis

Table 4 shows one entry from `japhug` which was wrongly transcribed by all three models.

ground truth	tɯɕk^{h}o+βzu
`tess_lat_Sy`	tɯɕkho ɬ βzu
`cal_ipa_OS`	tɯɕk^{h}o-ɬ-βzu
`qwen_ft`	tɯɕk^{h}o βzu

Table 4: Error analysis of a sample in the `japhug` dataset from the best models of each family (‘to do the action of drying in the sun’).

We see one of the main weaknesses of Tesseract, which is the superscript letters (‘h’ instead of ‘^{h}’). This is not the case for the other two models, which have no issue predicting the correct IPA symbol, in general. We, however, see that the ‘+’ character, which is not part of the IPA *per se*, but simply indicates that two units are to be added, has been predicted wrongly by all three models. Calamari, for instance, approximates using characters it has seen more often, such as the ‘-’ (indicating morpheme boundaries), and the ‘ɬ’. Such an issue would be impossible, had the output language been known, as the ‘ɬ’ sound doesn’t occur in such a position in the Japhug grammar book. Appendix E additionally reports the most missed IPA characters for the three best models on the `japhug` test set.

6.2 Unseen dataset: `kagayanen`

Our final goal is to carry out OCR on actual scans of grammar books, where the language and words might be unknown to the model. Hence, we also report the results of our three types of models on the smaller `kagayanen` dataset in Table 5. We recall that the Kagayanen language is not present in the `synth` dataset and is not related to Japhug.

model	CER (%)	WER (%)
`tess_lat_Ja`	60.92	117.56
`tess_lat_Sy`	28.67	97.86
`cal_ipa_Ja`	40.39	96.63
`cal_ipa_Sy`	**6.77**	**34.07**
`qwen_ft`	8.65	39.65

Table 5: Performance on the unseen `kagayanen` dataset.

As before, training on the larger synthetic corpus proves to be a better strategy overall, as it generalises better than with the `japhug` dataset. The fine-tuned Calamari models show better scores than the Tesseract models. They also outperform Qwen by around 2 points for CER and 6 points for WER.

7 Conclusion

We performed OCR for IPA characters in grammar books for the first time, in an effort to improve the digitisation of linguistic documents, notably in low-resource languages. We study the capacity of two neural OCR frameworks, Tesseract and Calamari, and an alternative and state-of-the-art LVLM approach with Qwen2.5-VL. Models were evaluated on two types of test sets: clean monolingual samples and synthetic multilingual pseudo-sentences.

Zero-shot performance was poor for all three model types due to the character set mismatch during pre-training and inference. This leads to systematic errors on the characters unique to the IPA, where models predict the closest known character instead. We then observe that fine-tuning on IPA datasets proved to be effective; using an actual grammar book as a training resource was not as helpful as a synthetic but large and multilingual dataset. We find that the best approach overall is based on Calamari trained with large synthetic data. While Qwen proved to be a competitive alternative in terms of accuracy, it is mainly hindered by its higher computational cost, at both the training and inference steps.

Future work will handle not only actual scans, but also standard texts together with IPA characters, as both co-occur in linguistic contexts (e.g., the phonology chapter or examples in a language transcribed with the IPA).

Limitations

The main limitation is the lack of real scans of grammar books in our evaluation. We only relied on clean PDFs that were artificially corrupted. This synthetic situation is due to the scarcity of scans with only IPA symbols and a verified reference. Our evaluation protocol means that some scan artefacts, such as blur, skew, or ink-bleed, are absent. Further efforts need to be made to curate such a dataset, potentially with manual annotation.

Moreover, we only considered two language grammar books in our study. As we needed high-quality transcriptions alongside the PDFs, the dataset choice was limited. We nonetheless observe that the improvement still holds for synthetic sentences in a large number of languages.

For the synthetic fine-tuning dataset (‘synth’), we relied on sentences made of randomly selected words from Wiktionary in a given language. This is due to the limited size of IPA-written *sentences* in most languages. Here, we preferred to focus on having a large number of languages represented (although not grammatically correct due to randomness) to cover all IPA *characters* across language families and data availability. This is also why we did not consider using existing grapheme-to-phoneme tools to massively convert sentences into the IPA (for supported languages). In our case, we have noisy sentences in terms of grammar, so the quality issue would be in the interaction *between* words (separated by white spaces) rather than *within* words (where errors could occur with an automatic tool). We also favoured better control of the data augmentation step to reach a large training dataset size.

Furthermore, in a real scenario, it is more likely to have non-IPA characters co-occurring, as mentioned in the conclusion. As we also saw that punctuation was already a challenge for the fine-tuned models, research must continue on the degree of specialisation for a more general use of the OCR tool for grammar books.

Finally, all our models assume that the input images have already been segmented into lines. External tools or pipelines such as OCRopus[9] can be used to carry out that step.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work was partly funded by the European Union (ERC, EPICAL, 101141712). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

[9]https://ocropus.github.io/.

References

Milind Agarwal and Antonios Anastasopoulos. 2024. A concise survey of OCR for low-resource languages. In *Proceedings of the 4th Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP 2024)*, pages 88–102, Mexico City, Mexico. Association for Computational Linguistics.

Michael Ashby. 2017. Recognition where it’s due: some experiments in optical character recognition (OCR) for phonetic symbols. *English phonetics: Journal*, (21):63–79.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc.

Ling Fu, Zhebin Kuang, Jiajun Song, Mingxin Huang, Biao Yang, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, Zhang Li, Guozhi Tang, Bin Shan, Chunhui Lin, Qi Liu, Binghong Wu, Hao Feng, Hao Liu, Can Huang, and 5 others. 2025. Ocrbench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning. *Preprint*, arXiv:2501.00321.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1332 others. 2023. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Jonathan Hus and Antonios Anastasopoulos. 2024. Back to school: Translation using grammar books. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20207–20219, Miami, Florida, USA. Association for Computational Linguistics.

Oana Ignat, Jean Maillard, Vishrav Chaudhary, and Francisco Guzmán. 2022. OCR improves machine translation for low-resource languages. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1164–1174, Dublin, Ireland. Association for Computational Linguistics.

International Phonetic Association. 1999. *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press.

Guillaume Jacques. 2021. *A grammar of Japhug*. Number 1 in Comprehensive Grammar Library. Language Science Press, Berlin.

Jackson L. Lee, Lucas F.E. Ashby, M. Elizabeth Garza, Yeonju Lee-Sikka, Sean Miller, Alan Wong, Arya D. McCarthy, and Kyle Gorman. 2020. Massively multilingual pronunciation modeling with WikiPron. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4223–4228, Marseille, France. European Language Resources Association.

William D. Lewis and Fei Xia. 2010. Developing odin: A multilingual repository of annotated language data for hundreds of the world's languages. *Literary and Linguistic Computing*, 25(3):303–319.

Shihua Li and Nathan Hill. 2023. Printed text recognition for lexical lists in chinese-international phonetic alphabet (ipa) glossing. *Journal of Open Humanities Data*.

Sebastian Nordhoff and Thomas Krämer. 2022. IMT-Vault: Extracting and enriching low-resource language interlinear glossed text from grammatical descriptions and typological survey articles. In *Proceedings of the 8th Workshop on Linked Data in Linguistics within the 13th Language Resources and Evaluation Conference*, pages 17–25, Marseille, France. European Language Resources Association.

Shu Okabe, Laurent Besacier, and François Yvon. 2022. Weakly supervised word segmentation for computational language documentation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7385–7398, Dublin, Ireland. Association for Computational Linguistics.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Carol J. Pebley and Thomas E. Payne. 2024. *A grammar of Kagayanen*. Number 8 in Comprehensive Grammar Library. Language Science Press, Berlin.

Shruti Rijhwani, Antonios Anastasopoulos, and Graham Neubig. 2020. OCR Post Correction for Endangered Language Texts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5931–5942, Online. Association for Computational Linguistics.

Shruti Rijhwani, Daisy Rosenblum, Antonios Anastasopoulos, and Graham Neubig. 2021. Lexically aware semi-supervised learning for ocr post-correction. *Transactions of the Association for Computational Linguistics*, 9:1285–1302.

Ray Smith. 2007. An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633.

Garrett Tanzer, Mirac Suzgun, Eline Visser, Dan Jurafsky, and Luke Melas-Kyriazi. 2024. A benchmark for learning to translate a new language from one grammar book. In *The Twelfth International Conference on Learning Representations*.

Christoph Wick, Christian Reul, and Frank Puppe. 2020. Calamari - A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition. *Digital Humanities Quarterly*, 14(2).

Fei Xia, William Lewis, Michael Wayne Goodman, Joshua Crowgey, and Emily M. Bender. 2014. Enriching ODIN. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3151–3157, Reykjavik, Iceland. European Language Resources Association (ELRA).

Zhibo Yang, Jun Tang, Zhaohai Li, Pengfei Wang, Jianqiang Wan, Humen Zhong, Xuejing Liu, Mingkun Yang, Peng Wang, Shuai Bai, LianWen Jin, and Junyang Lin. 2024. Cc-ocr: A comprehensive and challenging ocr benchmark for evaluating large multimodal models in literacy. *Preprint*, arXiv:2412.02210.

Kexun Zhang, Yee Choi, Zhenqiao Song, Taiqi He, William Yang Wang, and Lei Li. 2024. Hire a linguist!: Learning endangered languages in LLMs with in-context linguistic descriptions. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15654–15669, Bangkok, Thailand. Association for Computational Linguistics.

A Reproducibility

We used the default settings when fine-tuning. For the learning rate, we choose 0.0001 for Tesseract and 0.001 for Calamari. We fine-tune the Tesseract models by freezing the weights of the lower layers specialised for general feature-extraction and by training the task-specific upper layers only. All experiments were run on a single NVIDIA H100, except for the Tesseract models, which were on CPUs.

B Oversampling difficult characters

We also experimented with a further data augmentation strategy, where we oversampled characters in `japhug`, which proved to be difficult for the models trained on the `synth` dataset in the first evaluation phase. Concretely, we compute the proportion of errors for each character to find those which are responsible for around 80% of the character errors. We increase the `synth` training dataset with all the words where these characters occur at least once. This means that these samples are repeated twice in the dataset and denote it `synth_os`. We note here that the dataset varies depending on the considered model: for Tesseract, which struggled more, the dataset is larger (5.3M), whereas for Calamari, it is smaller (4.4M). Models trained on this dataset are marked with `_OS`.

model	synth		synth_os	
	CER	WER	CER	WER
`tess_eng`	36.44	100.95	38.85	102.86
`tess_eng_Sy`	5.14	40.51	6.51	49.58
`tess_eng_OS`	5.19	41.04	6.49	49.62
`tess_lat`	34.02	97.70	36.33	99.31
`tess_lat_Sy`	4.41	35.10	5.83	44.20
`tess_lat_OS`	4.45	35.69	5.80	44.38
`idiotikon`	57.12	102.87	56.04	102.75
`cal_ipa_Sy`	0.20	2.45	0.18	2.21
`cal_ipa_Sy_OS`	0.19	2.30	0.16	1.97

Table 6: Comparison of the effect of oversampling difficult characters (`_OS`) on both training and testing for the Tesseract and Calamari models.

Table 6 compares the prediction on the test split of the `synth` and the more difficult `synth_os` datasets. Oversampling has a limited impact: for Tesseract, training on difficult characters has led to very little improvement, if not worse error rates. For Calamari, the model was already well-trained, leading to negligible gains (less than 1 point). If we compare the scores across the two test datasets, we observe that the characters were actually difficult for the Tesseract model, whereas for Calamari, there is no major difference in test performance.

C Details on Qwen2.5-VL 7B

C.1 Comparing LVLMs

model	CC-OCR	OCRBench v2 (en)	TR (en)
	Qwen		
Qwen2.5-VL			
72B	**79.8**	**61.5**	—
7B	<u>77.8</u>	<u>56.3</u>	<u>68.8</u>
3B	74.5	54.3	—
Qwen2-VL			
72B	68.7	47.8	—
7B	—	—	**72.1**
	Closed-source LVLM		
Gemini-1.5-Pro	73.0	51.9	61.2
GPT-4o	66.6	46.5	61.2
Claude 3.5 Sonnet	62.7	45.2	62.2

Table 7: Benchmark performance of open- and closed-source LVLMs *reported* on CC-OCR and OCRBench v2 (English). TR stands for the text recognition task in the latter benchmark. '—' indicates an unreported score. The best score overall is in **bold**, while the second best is <u>underlined</u>.

Table 7 compares recent LVLMs on two benchmarks for OCR: CC-OCR (Yang et al., 2024) and OCRBench v2 (Fu et al., 2025). CC-OCR reports separate scores for four tasks: multi-scene OCR, multilingual OCR, document parsing, and key information extraction. The overall score comes from the averaged results over all four tasks. We report the overall and text recognition (TR) scores for English on OCRBench v2. The overall value is the average of the scores on the eight tasks of the benchmark. All scores are retrieved from the two benchmark articles and (Bai et al., 2025).

Qwen2.5-VL outperformed closed-source models such as Gemini-1.5-Pro (Gemini Team et al., 2023), GPT-4o (OpenAI et al., 2024), or Claude-3.5-Sonnet. Among the Qwen family, we chose its 7B version, as it offers a compromise between performance and computational cost. We note that the 72B is notably better than the smaller versions of the model.

C.2 Data structure for Qwen

```
{
"image": image_path,
"conversations": [
{
    "from": "human",
    "value": "<image>
    Please output only the text
       content from the image
       without any additional
       descriptions or formatting
       . The text is IPA.",
},
{"from": "gpt", "value":
   normalised_ground_truth_text
},
],
}
```

Figure 2: Image and text JSON structure used for Qwen. The conversation template uses the image file path and the ground truth.

Figure 2 presents how we structured the image, text, and prompt for Qwen.

C.3 Dataset distillation methodology

All words in the synth dataset are assigned a complexity score based on their character variety, χ, and ratio of diacritics, δ, as defined in Equation (1) for a word w. We give more weight to diacritics, as they are crucial and more difficult to identify.

$$\text{score}(w) = 0.4 \cdot \chi(w) + 0.6 \cdot \delta(w) \qquad (1)$$

To cover the entire set of characters, we keep all words containing rare characters (i.e., appearing less than 100 times in the training dataset) and the most complex words based on the computed score.

C.4 Hyperparameters for fine-tuning

per_device_train_batch_size	32
learning_rate	10^{-5}
optimiser	adamw_bnb_8bit
lora_rank	64
lora_alpha	64
lora_dropout	0.05

Table 8: Hyperparameters to fine-tune Qwen.

Table 8 lists the hyperparameters used to fine-tune Qwen with QLoRA.

D Computational cost

For an actual deployment in real life, another factor to take into account besides the output accuracy is the computational cost of the models, both for fine-tuning and inference. Table 9 reports the average *inference* time for the best fine-tuned variants of the three frameworks.

Tesseract	26 images/sec
Calamari	63 images/sec
Qwen2.5-VL-7B	0.65 images/sec

Table 9: Inference time for the three models.

On our test datasets, the best OCR approach for IPA in terms of inference is again the fine-tuned Calamari model. Despite the sometimes better performance achieved by Qwen2.5, the significantly higher computational load required by the larger number of parameters hinders its realistic use.

E Difficult characters

Table 10 reports frequently missed characters for the best models of Tesseract, Calamari, and Qwen on the japhug dataset. The main challenges seem to come from visually close characters (or combinations) and diacritics.

character	count	% of total	top misrecognition
ɯ (U+026F)	225	27.3%	u (U+0075)
ʰ (U+02B0)	214	26.0%	h (U+0068)
n (U+006E)	186	22.6%	m (U+006D)
́ (U+0301)	32	27.6%	í (U+00ED)
a (U+0061)	17	14.7%	ɑ (U+0251)
. (U+002E)	13	11.2%	(N/A)
a (U+0061)	248	82.7%	ɑ (U+0251)
g (U+0067)	10	3.3%	ɡ (U+0261)
ɯ (U+026F)	7	2.3%	ú (U+00FA)

Table 10: Difficult characters for each model and their error frequency on the japhug test dataset. Top: Tesseract; middle: Calamari; bottom: Qwen.