JOMNAM: Khmer Scene Text Annotation Tool

Pichphyrom RIN **Leangsreng SREAN** **Sovathara SENG** **Soklong HIM**

Cambodia Academy of Digital Technology, Phnom Penh, Cambodia
soklong.him@cadt.edu.kh

Abstract

The creation of large scale annotated Khmer datasets is a key challenge for advancing Artificial Intelligence and Natural Language Processing (NLP) in Cambodia. We present the Khmer Sense Text Annotation Tool, which integrates a YOLOv8 detection model trained on 12,000 images with Khmer Tesseract OCR and a human in the loop correction process. Evaluation using Intersection over Union (IoU), Precision, Recall, Confusion Matrix, and mean Average Precision (mAP) shows higher precision in detecting stacked characters and complex script structures while reducing manual labeling effort. The system demonstrates consistent improvements in both annotation speed and label accuracy compared to manual-only workflows. By focusing on Khmer's unique challenges, including the absence of word boundaries and character stacking, the tool enables more reliable dataset construction. This contribution not only accelerates resource development but also lays the foundation for future Khmer NLP advancements through deeper model integration.

Keywords: *Optical Character Recognition (OCR), Natural Language Processing (NLP), Annotation Tool, Text Recognition, Low-Resource Languages*

1 Introduction

The creation of annotated datasets is essential for developing reliable AI systems, yet building such resources for non-Latin scripts like Khmer remains challenging due to stacked characters, diacritics, and the absence of word boundaries. Traditional annotation tools such as VIA [1] or LabelMe [7] are primarily designed for general-purpose image annotation and are often used for diverse applications. However, they lack specific support for Khmer text, particularly for complex sense-text structures, making annotation slow and error-prone.

Recent surveys highlight the growing role of machine learning in auto-labeling tasks across images, audio, and text [2], but low-resource languages remain underserved. Advances in deep learning [3] and object detection techniques, including fully convolutional neural networks [6] and transfer learning [5], have shown strong performance in annotation-related tasks. Similarly, OCR systems have become essential for converting visual text into machine-readable form [4]. Nevertheless, these technologies have not been fully adapted to the unique challenges of Khmer script.

To address this gap, we propose JOMNAM, a model-assisted annotation tool specifically designed for Khmer sense-text annotation. JOMNAM integrates a fine-tuned YOLOv8 model with Khmer Tesseract OCR [12] and supports human-in-the-loop verification, providing both automation and accuracy. Unlike general-purpose tools, JOMNAM focuses on the particular needs of Khmer text, including stacked characters and diacritics, accelerating dataset creation and improving annotation consistency. By combining automation with expert oversight, JOMNAM contributes to the development of robust Khmer NLP resources and enables scalable, high-quality language technologies for low-resource scripts.

2 Related Work

Several open-source annotation tools have been developed to support dataset creation in computer vision and multimedia domains. LabelMe [7] provides polygon and bounding box annotation for object detection and image segmentation, while the VGG Image Annotator (VIA) [1] offers a lightweight, browser-based framework that requires no installation and supports multiple annotation shapes for images, audio, and video. More advanced systems such as CVAT [8], originally developed by Intel, extend these capabilities with industrial-grade features, including large-scale project management, team collaboration, and support for complex annotation tasks across video and 3D data. Although these tools demonstrate scalability and efficiency in visual domains, they are primarily optimized for images and high-resource languages, with limited or no support for complex scripts. In the case of Khmer, challenges such as stacked characters, diacritics, and the absence of explicit word boundaries demand specialized annotation frameworks [23][24][25][26]. Unlike general-purpose tools, a Khmer-focused system must integrate both text-specific handling and automatic labeling to reduce the burden of manual annotation. Our proposed tool builds on these insights by extending the strengths of open-source annotation environments while tailoring them to the linguistic and structural characteristics of Khmer sense text, thereby addressing a gap left by existing platforms. We also note complementary NLP and multimodal tools, including BRAT, WebAnno, INCEpTION, Prodigy, and ELAN [17][18][19][20], and recent web-based speech/audio annotation platforms such as Audino [22]. In multilingual modeling for low-resource languages, we reference mBART, mT5, and NLLB [13][14][15], as well as meta-learning for low-resource NMT [16].

3 Methodology

3.1 Data Preparation

Since there is no publicly available dataset for Khmer sense-text annotation, a new dataset was created as part of this work. In total, 12,000 text samples were collected, of which 6,000 were prepared during an earlier school project and an additional 6,000 were gathered specifically for this study. The dataset was designed to capture linguistic diversity, including variations in stacked characters, diacritics, and different writing contexts, which are essential challenges in Khmer text processing.

All images were carefully cleaned to remove duplicates, corrupted entries, and non-Khmer content. The final dataset was organized into sense-level categories according to the annotation guidelines defined in this work. Figure 1 illustrates randomly selected examples from the dataset.

Following data visualization, the dataset was divided into three subsets for experimentation: 70% for training, 20% for validation, and 10% for testing. This split ensures a robust training process while preserving sufficient data for hyperparameter tuning and final evaluation. Standard preprocessing steps, including text normalization, Unicode handling for stacked diacritics, and one-hot encoding of class labels, were applied. In addition, data augmentation techniques such as synthetic sentence generation and back-translation were explored to improve model generalization and address the limited size of labeled Khmer data.

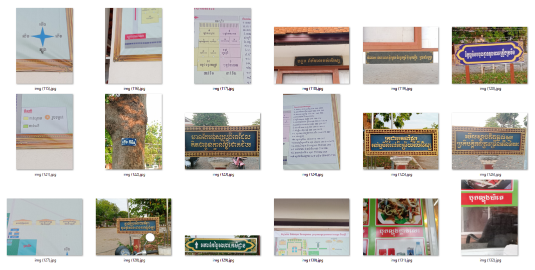

Figure 1: Example of Khmer Sense Text

3.2 Pre-trained Model

For this study, we adopt YOLOv8 as the pre-trained model to support Khmer sense-text annotation. YOLOv8 is an object detection framework that balances accuracy and efficiency [9]. It builds on the reputation of the YOLO family for real-time performance, integrating detection into a single neural network rather than multiple stages. This allows the model to achieve high detection speed and strong localization accuracy, making it suitable for tasks such as document analysis and text detection [10][11].

YOLOv8's architecture consists of three components: a backbone for extracting features, a feature pyramid network (FPN) for multi-scale fusion, and detection heads for bounding-box regression and classification. The FPN in particular enhances detection of small or complex objects, which is important for stacked characters and diacritics in Khmer script. Predictions are optimized through a compound loss that combines classification and regression terms.

A key advantage of YOLOv8 is its transfer learning capability, as pre-trained on large datasets such as COCO or ImageNet [5]. These can be fine-tuned for Khmer data, where annotated resources are limited [2]. By adapting YOLOv8 to our dataset, the model learns script-specific features, including irregular spacing and stacked glyphs not commonly present in Latin-based writing systems.

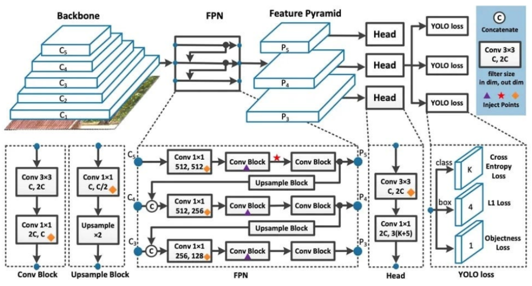

Figure 2: YOLOv8 architecture [28].

3.3 Model Evaluation

Model evaluation is a crucial phase in the machine learning workflow, as it demonstrates the model's ability to generalize to unseen data and ensures reliability for real-world annotation. In this study, we adopt both classification-based metrics and detection-based metrics to comprehensively assess YOLOv8's performance on Khmer sense-text annotation.

The evaluation metrics used include:

Accuracy: The proportion of correctly identified text regions out of all predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

Precision: The ratio of correctly detected text regions to all predicted regions.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Recall: The ratio of correctly detected text regions to all ground-truth regions.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

F1-Score: The harmonic mean of precision and recall, providing a balanced measure.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Intersection over Union (IoU): Measures the overlap between predicted bounding boxes.

$$\text{IoU} = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{5}$$

Mean Average Precision (mAP): The average precision computed across multiple IoU thresholds (e.g., 0.50 to 0.95). We follow the COCO-style evaluation for mAP at multiple IoU thresholds (0.50:0.95) [27].

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{6}$$

Here, TP refers to true positives (correctly detected text regions), TN to true negatives (non-text correctly identified), FP to false positives (wrongly predicted text), and FN to false negatives (missed text regions). These metrics provide a balanced view of both detection accuracy and robustness.

3.4 Experiments and Results

We conducted three experiments using different dataset sizes, denoted as Aksorv1, Aksorv2, and Aksorv3, to evaluate the effectiveness of the proposed system. All experiments were performed on a workstation equipped with an NVIDIA RTX 4060Ti GPU (8GB VRAM), 32GB RAM, and Intel Core i5-14400F CPU (14th Gen). Each dataset was split into 70% for training, 20% for validation, and 10% for testing. The same hyperparameter settings were used across experiments to ensure fairness. Model performance was evaluated using mAP@0.5:0.95, mAP@0.5, Recall, runtime, and parameter count.

As shown in Table 1, scaling up the dataset improved performance across all metrics. Aksorv3 achieved the highest results with an mAP@0.5 of 0.8615, recall of 0.8197, and mAP@0.5:0.95 of 0.5981, clearly outperforming Aksorv1 and Aksorv2. Although training time increased with larger datasets, the accuracy gains justified the additional computation, highlighting the importance of dataset scale for Khmer text detection.

To analyze training dynamics, Figure 3 shows the loss and evaluation curves for Aksorv3. The plots demonstrate stable convergence with consistent improvements in precision, recall, and mAP throughout training, confirming the robustness of the fine-tuned YOLOv8 model.

As shown in Table 1, the performance improves when scaling to a larger dataset. In particular, Aksorv3 achieves the best results with mAP@0.5:0.95 of 0.59813, mAP@0.5 of 0.86154, and recall of 0.81974, outperforming

Table 1: Performance comparison of YOLOv8 fine-tuning on different Khmer text datasets.

Model	Dataset Size	mAP@0.5:0.95	mAP@0.5	Recall	Run Time (H)	Parameters (M)
Aksorv1	6023	0.58531	0.83507	0.81660	2.38	2.6
Aksorv2	7823	0.57226	0.84611	0.80619	2.53	2.6
Aksorv3	**13846**	**0.59813**	**0.86154**	**0.81974**	5.43	2.6

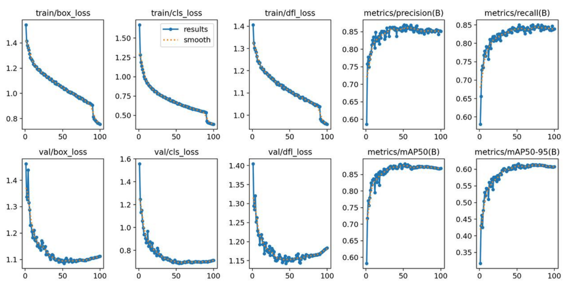

Figure 3: Multi-plot figure of Aksor V3

both Aksorv1 and Aksorv2. Although training time increased with dataset size (from 2.38h to 5.43h), the accuracy gains justify the cost, demonstrating the importance of larger annotated resources for Khmer text detection.

4 System Design

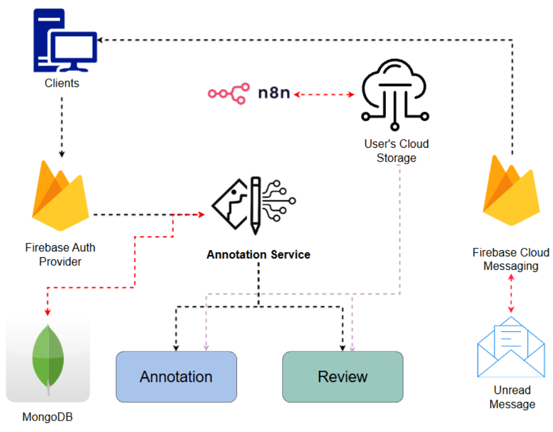

Figure 4: System flow diagram of JOMNAM.

The JOMNAM system, illustrated in Figure 4, is a modular, cloud-centric architecture designed to support secure and efficient annotation of Khmer sense text. Users access the system through authenticated client applications using Firebase Authentication. Annotation tasks are managed within the system, with model-assisted suggestions and human-in-the-loop review to ensure high-quality outputs. Structured project metadata and annotations are persisted in MongoDB, while associated media files are stored in the user's cloud storage, reducing server overhead. Automation is integrated via n8n, which monitors storage changes and triggers necessary processes. Finally, Firebase Cloud Messaging (FCM) delivers real-time notifications to client applications, keeping users informed of task updates and system events.

4.1 Key Features

JOMNAM is designed as a centralized model-aided annotation platform to address the unique challenges of Khmer sense-text annotation. It combines project management with model-assisted annotation, using a fine-tuned YOLOv8 model for text region detection and Khmer Tesseract OCR for recognition. Annotators review and correct model-generated labels through a human-in-the-loop process, which reduces manual effort while maintaining high quality. The system also supports full Unicode normalization for Khmer script and is containerized with Docker to simplify setup and deployment.

4.2 Data Storage

Unlike traditional annotation platforms that manage and persist all media files within a central server, JOMNAM adopts a lightweight storage design. User-uploaded images are saved directly to their own local storage either Google Drive, One Drive, while the backend only keeps references (e.g., file IDs, paths, or hashes) in a structured database for project and annotation management. This approach reduces server overhead, preserves user privacy, and gives annotators full control over their data. By decoupling media storage from metadata management, the system ensures scalability, portability, and easier integration with existing user workflows.

4.3 Server Side

The server side of JOMNAM is implemented using a Node.js and Express backend running inside Docker containers. Unlike traditional architectures that rely on additional layers such as

NGINX or Flask-based services, JOMNAM provides a lightweight yet extensible API layer that directly manages user authentication, project workflows, and annotation requests.

4.4 Client Side

The client side of JOMNAM is developed using React, providing a modular and responsive interface for annotation. React's component-based structure enables smooth integration with the backend API and ensures efficient rendering of project dashboards and annotation panels. Importantly, the system is designed to be flexible: users can either access a centralized deployment or host the tool locally, allowing them to customize and modify the interface to meet their own requirements.

4.5 Comparison with Existing Tools

To address feedback regarding comparative performance, we conducted a small evaluation between JOMNAM and the VGG Image Annotator (VIA), one of the most widely used manual annotation tools. The comparison focused on annotation time and correction effort for Khmer scene-text images.

Manual annotation speed is essentially the same across VIA and JOMNAM, as both rely on human-drawn bounding boxes. However, when using model-assisted annotation, JOMNAM reduces average annotation time by approximately **4–5 seconds per image**, due to automatic bounding-box proposals from YOLOv8.

It is worth noting that while model-assisted detection speeds up the process, the Khmer Tesseract OCR still produces low-confidence text predictions, requiring human verification. As a result, the primary time savings come from faster region detection rather than text transcription.

5 Discussion

Our experiments demonstrate that modern deep learning models can be adapted to Khmer when carefully tuned with script-specific adjustments. The integration of YOLOv8 with Khmer OCR effectively addresses the challenges of stacked characters and diacritics, while human-in-the-loop verification provides a practical balance between automation and accuracy. These findings highlight that annotation efficiency can be substantially improved without sacrificing quality, making JOMNAM a valuable tool for low-resource language settings.

Nevertheless, some limitations remain. Performance is tied to the size and diversity of the training data, and while larger datasets improved accuracy, collecting such resources is time-consuming. In addition, although our system is modular and extensible, reliance on pre-trained models may still leave gaps for highly domain-specific texts. Finally, annotation quality depends on user expertise, suggesting that further optimization of the correction interface could enhance usability.

6 Limitations

Despite promising results, JOMNAM has several limitations. First, its performance is still constrained by the size and diversity of available training data, suggesting that larger and more varied datasets are required to further improve accuracy. Second, the system currently struggles with handwritten Khmer text, which remains a major challenge for OCR and detection models. Finally, certain complex scene conditions, such as noisy backgrounds or highly stylized fonts, remain difficult to handle and will require additional model tuning.

7 Conclusion

This paper introduced JOMNAM, a Khmer scene-text annotation tool that integrates YOLOv8, Khmer Tesseract OCR, and human-in-the-loop verification to reduce manual effort and improve annotation consistency. Experiments on a 12,000 sample dataset showed reliable detection of complex Khmer scripts, confirming the effectiveness of our approach.

8 Future Work

Future work will focus on addressing the limitations identified in our discussion. Expanding the dataset with more diverse and domain-specific samples will strengthen model robustness. We also aim to improve the annotation interface to further reduce reliance on expert annotators, and to explore semi-supervised and active learning strategies for better scalability. Finally, extending the system beyond text to include audio and multimodal annotation will broaden its applicability for low-resource languages.

Table 2: Comparison of annotation tools

Tool	Open Source	Automation Annotation	Store Dataset	Private Storage	Khmer Model Support
LabelMe	✓	×	×	×	×
VIA	✓	×	✓	✓	×
CVAT	✓	✓	✓	✓	×
BRAT	✓	×	✓	✓	×
INCEpTION	✓	×	✓	✓	×
Jomnam	✓	✓	✓	✓	✓

Acknowledgment

We would also like to thank the past JOMNAM team members, including Non Sorany, Chan Sovandara, Sambo Sopheakline, Poch Sreypov and Sin Panharong for their valuable contributions and early efforts on the project.

References

[1] A. Dutta and A. Zisserman, "The VIA Annotation Software for Images, Audio and Video," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 2276–2279. doi: 10.1145/3343031.3350535.

[2] S. Zhang, O. Jafari, and P. Nagarkar, "A Survey on Machine Learning Techniques for Auto Labeling of Video, Audio, and Text Data," *arXiv:2109.03784*, Sep. 2021. doi: 10.48550/arXiv.2109.03784.

[3] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to Machine Learning, Neural Networks, and Deep Learning," *Transl. Vis. Sci. Technol.*, vol. 9, no. 2, p. 14, Feb. 2020. doi: 10.1167/tvst.9.2.14.

[4] M. Law *et al.*, "Heatmap-based Object Detection and Tracking with a Fully Convolutional Neural Network," 2021.

[5] C. Borngrund, U. Bodin, and F. Sandin, "Machine Vision for Construction Equipment by Transfer Learning with Scale Models," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8. doi: 10.1109/IJCNN48605.2020.9207577.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *arXiv:1411.4038*, Mar. 2015. doi: 10.48550/arXiv.1411.4038.

[7] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: A Database and Web-Based Tool for Image Annotation," *Int. J. Comput. Vis.*, vol. 77, no. 1–3, pp. 157–173, 2008. doi: 10.1007/s11263-007-0090-8.

[8] CVAT: "Computer Vision Annotation Tool," Intel, 2017–2024.

[9] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," 2023.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv:2004.10934*, 2020.

[12] R. Smith, "An Overview of the Tesseract OCR Engine," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, 2007, pp. 629–633. doi: 10.1109/ICDAR.2007.4376991.

[13] Y. Liu *et al.*, "mBART: Multilingual Denoising Pre-training for Neural Machine Translation," *arXiv:2001.08210*, 2020.

[14] C. Raffel *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 1–67, 2020.

[15] M. Artetxe, A. Costa-jussà, N. Shankar, *et al.*, "No Language Left Behind: Scaling Human-Centered Machine Translation," *arXiv:2207.04672*, 2022.

[16] J. Gu, Y. Wang, Y. Zhao, V. O. K. Li, and K. Cho, "Meta-Learning for Low-Resource Neural Machine Translation," in *Proc. EMNLP*, 2018, pp. 3622–3631.

[17] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, "BRAT: a Web-based Tool for NLP-Assisted Text Annotation," in *Proc. EACL (System Demonstrations)*, 2012, pp. 102–107.

[18] R. Eckart de Castilho, E. M. Rehm, I. Gurevych, *et al.*, "WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations," in *Proc. ACL*, 2014.

[19] R. Eckart de Castilho, S. Eger, and I. Gurevych, "INCEpTION: A Semantic Annotation Platform for Rapid Concept Creation," in *Proc. ACL System Demonstrations*, 2018.

[20] M. Honnibal and I. Montani, "Prodigy: A Modern Annotation Tool for Machine Learning," 2017.

[21] H. Brugman and A. Russel, "Annotating Multimedia/Multi-modal Resources with ELAN," in *Proc. LREC*, 2004.

[22] P. Kumar, M. S. Shukla, A. Jain, and V. N. Mishra, "Audino: A Modern Annotation Tool for Audio and Speech," in *Proc. Int. Conf. Intelligent Human Computer Interaction (IHCI)*, 2020, pp. 1–5.

[23] C. Chea, Y. K. Thu, D. Ding, M. Utiyama, and E. Sumita, "Khmer Word Segmentation Using Conditional Random Fields," in *Proc. O-COCOSDA*, 2016, pp. 154–159.

[24] S. Sry, A. S. Nguyen, C. H. H. Chhun, T. Enomoto, and S. Matsumoto, "A Review of Khmer Word Segmentation and Part-of-Speech Tagging and an Experimental Study Using Bidirectional LSTM," *Int. J. Asian Lang. Process.*, vol. 32, no. 2, pp. 35–51, 2022.

[25] S. Nom, S. Kang, and S. Kim, "KhmerST: A Low-Resource Khmer Scene Text Detection and Recognition Benchmark," in *Proc. Asian Conf. Comput. Vis. (ACCV)*, 2024.

[26] S. Sok, Y. Tian, and L. Jin, "Addressing the Attention Drift Problem for Khmer Long Textline," *Int. J. Document Anal. Recognit. (IJDAR)*, Springer, 2025.

[27] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft COCO: Common Objects in Context," in *Proc. ECCV*, 2014, pp. 740–755.

[28] Ultralytics, "YOLOv8 Architecture and Components (Backbone/Neck/Head)," 2023–2025.