ACET 2025

RESEARCH | SHARE | CONNECT

Connecting ASEAN Researchers for Digital Research

Organized by:

CADT

IDRI

Supported by

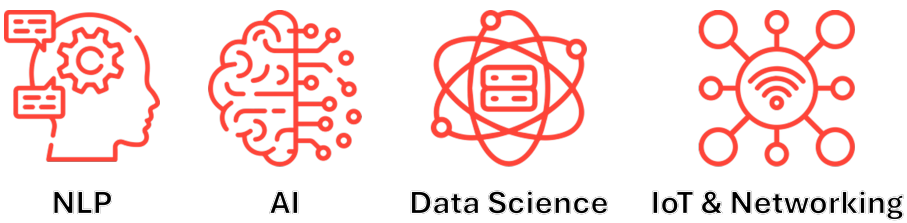
Cambodia Academy of Digital Technology (CADT)

Institute of Digital Research and Innovation (IDRI)

ASEAN Conference on Emerging Technologies 2025 (ACET 2025)

November 13 - 14, 2025

Phnom Penh, Cambodia



Acknowledgment

We are delighted to welcome you to the ASEAN Conference on Emerging Technologies 2025 (ACET 2025). ACET has served as a pivotal platform for fostering innovation, collaboration, and knowledge-sharing among researchers, academicians, and professionals both locally and across the region. The conference theme underscores the transformative potential of cutting-edge technologies such as Natural Language Processing (NLP), Data Science, Networking & the Internet of Things (IoT), and Artificial Intelligence (AI) in addressing current challenges while simultaneously unlocking opportunities for the future.

The quality of the research presented have been truly inspiring, reflecting the dedication and commitment of our global community to pushing the boundaries of technological innovation. The diversity of perspectives and expertise shared during this conference has undoubtedly enriched our understanding of the latest developments in the relevant fields.

We express our profound gratitude to our esteemed keynote speakers and presenters. Their invaluable insights and expertise were pivotal in making this conference a resounding success. Our sincere thanks also go to our dedicated reviewers, volunteers, and sponsors. Their tireless efforts and unwavering support ensured the seamless execution of the event, creating an enriching and memorable experience for all attendees. We urge all participants to stay connected, continue to build upon the networks formed here, and persist in their vital contributions to the progress and advancement of emerging technologies.

Once again, thank you all for your support and valuable contributions. We look forward to welcoming you again next year for another successful and inspiring edition of ACET.

Yours sincerecly,

ACET Committee

Organizing Committee

Reviewer Team

Prof. Dominique Vaufreydaz
Dr. Sandra Castellanos-Paez
Dr. Dona Valy
Dr. Sa Math
Dr. Sovuthy Cheab
Dr. Hongly Va
Mr. Pisal Chanty
Dr. Masugi Inoue
Dr. Hour Kaing
Dr. Kimsrornn Khon
Dr. Soky Kak
Dr. Phutphalla Kong
Dr. Porchourng Chour
Dr. Sotheara Leng
Dr. Kazutaka Kikuta
Dr. Rina Bouy
Dr. Sophea Chhun
Dr. Rottana Ly
Dr. Lihour Nov
Dr. Väänänen Daavid
Dr. Bonpagna Kann

Publication Team

Dr. Sovuthy Cheab
Mr. Pisal Chanty
Mr. Leangsiv Han
Dr. Lihour Nov
Mr. Vathna Lay
Mr. Mengchhoung Ang
Dr. Mangseang Hor
Mr. Ratha Yeu

Technical Support Team

Dr. Phutphalla Kong
Mr. Sophal Thear
Ms. Sreyleak Sam
Dr. Hongly Va
Ms. Sokchovy Monirath
Mr. Phannet Pov
Mr. Soklong Him
Ms. Nab Mat
Ms. Sophin Chheng

Event, Publicity and Logistic Team

Mr. Hieng Mao
Mr. Natt Korat
Mr. Ponleur Veng
Mrs. Davin Ly
Mr. Soklay Heng
Mr. Sopagna Heang
Ms. Sofy Thy
Mrs. Khmery Vann
Mr. Sokleap Som
Ms. Kimhouy Yann
Mr. Hangnypolen Heng

Keynote Speech 01: Educational Model for Cultivating Digital Talent in Cambodia: A CADT Case Study

Cambodia's National Artificial Intelligence Strategy 2025–2030 sets a clear direction for the nation's AI future — one that emphasizes responsible innovation, inclusive growth, and collaboration across government, academia, and industry. Yet, realizing this vision depends on the ability of researchers to connect their scientific pursuits with the country's strategic needs.

This keynote highlights how research institutions and innovators can align their AI efforts with the six national strategic priorities — from human resource development and data ecosystems to digital government, sectoral applications, and ethical AI. It explores practical ways for researchers to translate ideas into deployable solutions, contribute to national datasets and models, and engage in cross-sector partnerships that bridge research and policy.

Drawing from Cambodia's leadership role in ASEAN's AI governance and ongoing regional collaborations, the talk invites the research community to pursue "research with purpose" — research that not only advances algorithms and publications, but also strengthens Cambodia's digital sovereignty, resilience, and inclusive development.

Dr. Sovann En, Director of Department of Digital Government Transformation

Keynote Speech 02: Towards Responsible and Inclusive AI China's Exploration in Multimodal

Dr. Yuerui Feng, Engineer of China Academy of Information and Communications Technology

This address delineates China's commitment to fostering responsible and inclusive artificial intelligence, focusing on the transformative potential of multimodal large language models. It explores the global evolution of AI towards greater reasoning and autonomy, spotlighting key trends such as the shift in computing demand, the critical role of high-quality data, and the rise of open-source ecosystems and foundation super models. The presentation further highlights the instrumental role of the China Academy of Information and Communications Technology (CAICT) in benchmarking and ethical standardization, and introduces the China–ASEAN AI Industry Innovation Center as a cornerstone for regional collaboration. Through these initiatives, China aims to promote an open, cooperative, and sustainable global AI ecosystem for shared benefits.

Keynote Speech 03: Human Perception in the Age of Intelligent Systems: Accessible Design and Inclusive Innovation

Dr. Kyle Keane, Senior Lecturer in Assistive Technology in School of Computer Science at University of Bristol

Dr. Kyle Keane is a blind multidisciplinary researcher at the intersection of artificial intelligence, human perception, and inclusive design. Currently Senior Lecturer in Assistive Technology in the School of Computer Science at the University of Bristol, Kyle draws from over a decade of teaching and program development experience at MIT to explore multisensory interaction and engagement with complex information. Dr. Keane's technical contributions include developing neural network systems for automatic production of audio-tactile graphics optimized for human perceptual interpretability, with publications spanning cognitive science, quantum physics, perception science, and assistive technology engineering. Dr. Keane coordinates global assistive technology initiatives that fundamentally restructure how technologies are developed with and for disabled communities, centering self-determination and knowledge sovereignty for disabled communities.

Dr. Keane is a founding member of an international co-design network linking the Cambodian Academy of Digital Technologies, Penn State University, MIT, and the University of Bristol, where the experiences and priority needs of refugees and conflict-displaced people with disabilities directly inform collaborative prototyping and development cycles. This approach explicitly rejects extractive research methodologies, instead ensuring that communities with disabilities retain ownership of both problem identification and solution development. Through an inclusive model of assistive technology innovation, this reciprocal exchange of knowledge and value ensures that engineering solutions are driven by user needs, aligned with local expertise, and informed by lived experiences.

In this talk, Dr. Keane will explore the intricacies of human spatial perception, focusing on auditory and haptic localization through innovative multi-speaker arrays and multi-channel haptic systems. He will also discuss his efforts in building a global network of makerspaces and hackathons that promote assistive technology through community-led, sustainable innovation. This keynote will highlight how understanding and designing for 'the world at a distance' can transform our interactions with our environments, each other, and ourselves.

Keynote Speech 04: The Wrestling Between Public Health and Privacy: The Draft National Health Insurance Data Management Act in Taiwan

Ms. Rosalind Liu, Project Manager at Open Culture Foundation

Taiwan started implementing the National Health Insurance (NHI) program in 1995. The program covers nearly 99.9% of Taiwan's population of approximately 23 million people and its database contains detailed healthcare utilization records, including demographic information, diagnosis codes, medical procedures, medications, and healthcare expenditures, spanning over two decades.

Undoubtedly, the National Health Insurance Database (NHID) is one of the world's most comprehensive health claims databases and presents high research value. It became a crucial resource for epidemiological research, health services research, drug safety monitoring, and health policy evaluation until the Constitutional Court mandated privacy protections enhancement in 2022.

According to the judgement, the administration agency was required to amend the National Health Insurance Act or other relevant laws, or enact a dedicated law to explicitly stipulate the matter before August in 2025. The Draft National Health Insurance Data Management Act was submitted by the Executive Yuan in May. Unfortunately, the legislation process is still pending due to the struggle between public health and personal privacy.

In this session, the speaker will introduce the current arguments on the Draft NHID Management Act in Taiwan and also share about the latest revision of Taiwan's Personal Data Protection Act, which happened on Nov 4th, 2025.

Keynote Speech 05: Data Governance in Fintech – Legal Challenges and Opportunities

The rapid growth of ASEAN's FinTech sector, poised to underpin a projected $2 trillion digital economy, is critically dependent on robust data governance. This presentation examines the pivotal legal challenges and opportunities shaping this landscape. It identifies a fragmented regulatory patchwork across member states, a significant trust deficit stemming from cybersecurity threats, and operational complexities as major barriers to regional expansion and innovation. These challenges increase compliance costs and hinder the effectiveness of data-driven services like AI-powered credit scoring.

Mr. Teck Kheong Looi, Principal Consultant (ASEAN), Public Policy Asia Advisors

Conversely, the presentation proposes a forward-looking framework for turning data governance into a competitive advantage. Key solutions include leveraging the ASEAN Digital Economy Framework Agreement (DEFA) to harmonize standards, promoting innovation-first approaches like regulatory sandboxes, and treating data as a shared asset through open finance. By advocating for policy agility, strategic investment in trust, and regional unity, the presentation argues that effective data governance is not a barrier but the essential foundation for building a secure, inclusive, and globally trusted FinTech ecosystem in ASEAN.

List of Papers

I. MAIN TRACK

II. STUDENT TRACK

ACET 2025

MAIN TRACK - PAPERS

Towards Explainable Khmer Polarity Classification

Marry Kong[†] **Rina Buoy**[†] **Sovisal Chenda**[†] **Nguonly Taing**[†]
[†]Techo Startup Center, Cambodia
Corresponding author: rina.buoy@techostartup.center

Abstract

Khmer polarity classification is a fundamental natural language processing task that assigns a positive, negative, or neutral label to a given Khmer text input. Existing Khmer models typically predict the label without explaining the rationale behind the prediction. This paper proposes an explainable Khmer polarity classifier by fine-tuning an instruction-based reasoning Qwen-3 model. The notion of explainability in this paper is limited to self-explanations, which the model uses to rationalize its predictions. Experimental results show that the fine-tuned model not only predicts labels accurately but also provides reasoning by identifying polarity-related keywords or phrases to support its predictions. In addition, we contribute a new Khmer polarity dataset consisting of short- to medium-length casual, romanized, and mixed-code Khmer expressions. This dataset was constructed using both heuristic rules and human curation and is publicly available through a gated Hugging Face repository[1]. The fine-tuned Qwen-3 models are also made available in the same Hugging Face account.

Keywords: Polarity Classification, Explainability, LLM, Khmer Dataset.

1 Introduction

Polarity classification is a classical natural language processing task that assigns a discrete label to a given input text, typically from the set positive, negative, or neutral [1]. It is also commonly referred to as sentiment analysis. Fundamentally, polarity classification is a specific instance of general text classification or categorization, where the assigned label reflects sentiment [2]. Other examples of text classification tasks include spam detection, language identification, and authorship attribution [2].

Although Khmer is a low-resource language, the task of Khmer text classification has been relatively well studied. Various approaches, using both classical machine learning and deep learning, have been proposed for Khmer text classification. Similarly, several methods [1; 3; 4; 5; 6] for Khmer sentiment and polarity classification have been developed to assign sentiment labels to given Khmer text inputs. However, all of these methods are essentially black boxes, providing only the final label without any justification or explanation for the prediction. Understanding the model's reasoning is one of fundamental aspects of model explainability. It should be noted that the notion of explainability in this paper is limited to self-explanations, which the model uses to rationalize its predictions.

Thus, this paper proposes an explainable Khmer polarity classifier that not only predicts labels accurately but also provides justification or reasoning for its predictions. This is achieved by fine-tuning an instruction-based reasoning Qwen-3 [7] model. Experimental results demonstrate that the fine-tuned model delivers both accurate labels and coherent reasoning. Furthermore, to address the existing dataset gap, we introduce a new casual Khmer polarity dataset constructed through a combination of heuristic rules and human curation. The dataset comprises short- to medium-length casual, romanized, and mixed-code Khmer expressions. To the best of our knowledge, this is the first Khmer dataset of its kind. Our contributions are as follows:

1. We propose the first explainable Khmer polarity classifier that not only predicts labels accurately but also provides justification or reasoning for its predictions.

[1] rinabuoy/khmerpolarity_nonreasoning

2. We contribute a new Khmer polarity dataset consisting of short- to medium-length casual, romanized, and mixed-code Khmer expressions.

3. We make the resulting fine-tuned models publicly available to the community.

2 Related Work

2.1 Khmer Text Classification

Khmer is classified as a low-resource language due to limited research attention and the scarcity of datasets available for training and evaluation. Nonetheless, the task of Khmer text classification has been relatively well explored. Phann et al. [3] studied the classification of local news into nine categories using a range of classical machine learning methods, such as logistic regression, Naïve Bayes, and support vector machine (SVM) with term-frequency inverse document-frequency (TF–IDF) features. The SVM model with an radial basis function (RBF) kernel achieved the best performance. Jiang et al. [8] employed pretrained Khmer language models, including BERT [9] and ELECTRA [10], for Khmer news classification, reporting a top accuracy of 70.6% across eight news categories. Rifat and Imran [5] proposed a similar pretraining and fine-tuning approach using BERT for both sentiment and news classification, significantly outperforming the baseline FastText-based model. Similarly, Buoy et al. [4] utilized pretrained word embeddings (FastText) together with neural architectures such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs) for Khmer text classification. Compared with a baseline pipeline using TF–IDF features and an SVM classifier, their models achieved improved accuracies on both multiclass and multilabel classification tasks.

In the area of sentiment detection, Prom et al. [6] proposed a method that combines BERT-based contextual features with a bidirectional long short-term memory (BiLSTM) classifier. Their approach achieved an accuracy of 86%, outperforming traditional machine learning algorithms in classifying Khmer text sentiments into negative, neutral, and positive categories. Similarly, Ye et al. [1] constructed a new Khmer polarity dataset and experimented with a range of classical machine learning techniques and text feature extractors.

2.2 Prompt-Based Sentiment Classification

Instruction-tuned large language models can be prompted to perform sentiment classification using zero-shot or few-shot prompting techniques, without the need for fine-tuning or gradient updates [11]. Nonetheless, for low-resource languages such as Khmer, their performance often suffers from hallucination and does not consistently align with user expectations. A prompt-based polarity detection was proposed for Czech language, using both zero-shot and few-shot prompting techniques [12]. The authors highlighted that the prompt-based method outperformed traditional finetuning especially on a limited training dataset. Also, the authors suggested that additional pretraining on target domain can enhance performance in a zero-shot prompting case.

2.3 Parameter-Efficient Fine-tuning

Fine-tuning a large model such as Qwen-3 by updating all parameters requires a substantial memory footprint. To address this, parameter-efficient fine-tuning approaches are recommended, especially in resource-constrained environments (e.g., a single GPU). Among these, the low-rank adaptation (LoRA) method [13] is one of the most widely used. LoRA approximates linear layer matrices with low-rank matrices containing fewer trainable parameters. Instead of updating all parameters, LoRA injects only trainable low-rank updates, reducing the number of trainable parameters by approximately 90% while maintaining performance. QLoRA [14] further reduces memory usage by using 4-bit quantized pretrained models.

3 Proposed Solution

This paper proposes an explainable Khmer polarity classifier. In this context, explainable means that the classifier not only predicts labels accurately but also provides

justification or reasoning for its predictions. This notion is also known as self-explanation. This is achieved by fine-tuning an instruction-tuned large language model (LLM) with reasoning capability. For this purpose, we adopted the instruction-tuned Qwen3 models, which unify both a thinking mode for complex tasks and a non-thinking mode for simpler tasks within the same architecture. Due to resource constraints, we experimented only with the Qwen3-1.7B, Qwen3-4B, and Qwen3-8B variants, as these models are multilingual and support the Khmer language. Qwen-3 is the latest release of the Qwen model family, including Qwen-2 [15] and Qwen-2.5[16], with enhanced multilingual (Khmer included) instruction understanding and translation.

To fine-tune an instruction-tuned Qwen3 model into an explainable Khmer polarity classifier, we design a prompt that activates the model's thinking mode by guiding it to identify polarity-related keywords or phrases before predicting the final polarity label. These identified keywords or phrases serve as the basis for explaining the model's predictions. The reasoning prompt template is shown in Figure 1. As illustrated, the polarity-related keywords or phrases are enclosed between *<think>* and *</think>* tokens, which trigger the model to use this information as reasoning prior to concluding the final label during training.

When the polarity-related keywords or phrases are not available, the thinking mode is deactivated by applying the non-reasoning prompt template, as shown in Figure 2. This allows training to incorporate both datasets that include explicit polarity cues and those that do not, within the same model.

During inference, the thinking mode is always activated using the inference prompt template (Figure 3). Consequently, the model identifies any polarity-related keywords or phrases as part of its reasoning process before arriving at the final label. In this way, the model's predictions are inherently self-explainable.

We apply LoRA fine-tuning to reduce memory requirements. The LoRA configurations are shown in Table 1, and the number of LoRA trainable parameters, along with the total model parameters for each Qwen3 variant, are presented in Table 2. As provided in the table, the fine-tuning is limited to only the self-attention and feed-forward layers.

Table 1. LoRA configurations.

Parameter	**Value**
r	32
α	32
dropout	0
bias	none
modules	projection layers

Table 2. The LoRA trainable parameters. *Params*: model parameters.

Model	**LoRA Params**	**Full Params**
Qwen3-1B	34M	1.7B
Qwen3-4B	66M	4B
Qwen3-8B	80M	8B

4 Datasets

4.1 Khmer Polarity (KP) Dataset

The dataset [1] comprises 10,000 manually labeled Khmer text inputs collected from online news articles. Each input is assigned one of three possible polarity labels: positive, negative, or neutral. The dataset is characterized by relatively long text inputs written in formal language. In addition, it includes keywords or phrases associated with the final label. However, the authors of this dataset used only the input texts and labels to train multiple Khmer polarity classifiers with various classical machine learning approaches, such as k-nearest neighbors (kNN) and support vector machines (SVM), and text feature extractors. As a result, their trained models lack explainability and justification. A few training samples from this dataset are shown in Table 3. For evaluation, the dataset is split into training (9,000) and test (1,000) sets.

4.2 Casual Khmer Polarity (CKP) Dataset

To complement the above KP dataset, we constructed a new casual Khmer polarity dataset [17] of approximately 16,500 texts,

```
    instruction = 'Classify the given text as positive, neutral, or negative:\n '
    conversations.append([
        {"role" : "user",      "content" :   instruction + {text}},
        {"role" : "assistant", "content" : f'<think> Because the input texcontains the
following {reasoning} </think>\n'+ label},
    ])
```

Figure 1. The reasoning prompt template for polarity classification. *text*: Khmer text input. *reasoning*: any polarity-related keywords or phrases. *label*: negative, positive, or neutral.

```
instruction = 'Classify the given text as positive, neutral, or negative:\n '
conversations.append([
    {"role" : "user",      "content" :   instruction + {text}},
    {"role" : "assistant", "content" : f'<think>\n\n</think>\n'+ label},
])
```

Figure 2. The non-reasoning prompt template for polarity classification. *text*: Khmer text input. *label*: negative, positive, or neutral.

```
instruction = 'Classify the given text as positive, neutral, or negative:\n '
conversations.append([
    {"role" : "user",      "content" :   instruction + {text}},
    {"role" : "assistant", "content" : '<think>'},
])
```

Figure 3. The reasoning prompt template for polarity classification during inference. *text*: Khmer text input.

Table 3. A few samples from the KP dataset, showing the Khmer text inputs, polarity-related keywords or phrases, and labels.

Input Text	**Reasoning**	**Label**
យុវនារីណាដែលកាន់សៀវភៅពេលបច្ចុប្បន្ន គឺនឹងក្លាយទៅជាម្តាយដ៏ល្អនាពេលអនាគត។ (Any young woman who holds a book today will become a good mother in the future.)	ម្តាយដ៏ល្អ (good mother)	positive
ឧទាហរណ៍ មនុស្សថ្លង់ ឬពិបាកស្តាប់ អាចស្តាប់មិនឮថានគរបាលនិយាយអ្វីខ្លះ ។ (For example, people who are deaf or hard of hearing may not be able to hear what the police are saying.)	ពិបាកស្តាប់/ស្តាប់មិនឮ (inaudible)	negative
រដ្ឋាភិបាលកម្ពុជាបានប្រកាសថា កាស៊ីណូទាំងអស់នៅក្នុងប្រទេសនឹងត្រូវបានអនុញ្ញាតឱ្យដំណើរ ការឡើងវិញ ប៉ុន្តែមានតែជាមួយល្បែង ស្ពាល និងម៉ាស៊ីនលេងល្បែងតែប៉ុណ្ណោះ។ (The Cambodian government has announced that all casinos in the country will be allowed to reopen, but only with slot machines and slot games.)	ប្រកាស (announce)	neutral

focusing on casual Khmer texts that are often used on social media, such as Facebook. These casual texts include Khmer, romanized, and mix-coded expressions. Data cleaning was applied by converting all English text to lowercase, removing punctuation and irrelevant symbols, eliminating emojis and special characters to reduce noise, and correcting misspellings to handle informal or noisy language effectively. A heuristic method first labeled comments automatically using sentiment-related keywords, followed by manual review and correction to ensure accuracy and handle misclassifications caused by context or sarcasm. We adopted the same labeling scheme (i.e., positive, negative, and neutral). Nonetheless, this dataset does not provide any identified polarity keywords or phrases. A few training samples from this dataset is provided in Table 4. For evaluation, this dataset is split into train (14,850) and test (1,650) sets.

Thus, by leveraging the proposed reasoning and non-reasoning prompting templates together with the KP dataset (with reasoning) and the CKP dataset (without reasoning), we fine-tune an explainable Khmer polarity classifier that can self-explain its predictions.

5 Experiments and Results

5.1 Experimental Setup

We used the Unsloth[2] fine-tuning framework, which enables memory-efficient training and inference on a single GPU. In all experiments, training was conducted for two full epochs on the combined data (i.e., KP and CKP) using a linear learning rate schedule with an initial rate of 2×10^{-4} and a batch size of eight. The maximum context length was set to 2,048. Gradient accumulation was set to four, resulting in an effective batch size of 32 per gradient update. A weight decay of 0.01 was applied. We used instruction-tuned 4-bit quantized Qwen-3 model weights. For computing resources, training was performed on a single NVIDIA L4 GPU provided by Google Colab.

[2]https://unsloth.ai/

5.2 Results and Discussion

In this section, we present the experimental results followed by key analyses. We begin with a quantitative assessment using a standard classification accuracy metric, and then provide a qualitative assessment of reasoning and explainability.

5.2.1 Quantitative Assessment of Classification

Table 5 presents an accuracy comparison between our fine-tuned Qwen-3 models and existing classical machine learning methods on the KP dataset. As shown, our fine-tuned models with the enabled thinking mode achieved an accuracy of 84%, compared to 60% by the best-performing tuned SGD model. The same table shows that the default Qwen-3 models (i.e., without fine-tuning) achieve an accuracy of 0% as they are unable to sufficiently understand Khmer language. As shown in the same table, When the thinking mode was not enabled (i.e., baseline), the model performance significantly degraded. In summary, the findings highlight the robustness of our proposed fine-tuned models.

However, on the CKP dataset in Table 6, our fine-tuned models performed comparably to existing deep learning–based models (i.e., RNN and CNN). Specifically, our fine-tuned Qwen-8B model achieved a classification accuracy of 87%, compared to 88% by the CNN model. The model's slight under-performance compared to the dedicated CNN model is likely due to the fact that fine-tuning was applied only to the self-attention and feed-forward layers, while the embedding layers, which lack sufficient Khmer language understanding, were not updated. Like on the KP dataset, the default Qwen-3 models achieved an accuracy of 0% and without the thinking mode (i.e., baseline), the model performance significantly degraded.

In summary, across both datasets, our proposed fine-tuned models either outperformed or achieved comparable performance to existing approaches. Moreover, our models provide reasoning and justification for their predictions (see next Section 5.2.2), a feature absent in previous methods.

Table 4. A few samples from the CKP dataset, showing the Khmer text inputs, and labels.

Input Text	Reasoning	Label
ម្ហូបនៅកន្លែងគាត់ឆ្ងាញ់ញុមបានទៅម្តងហើយ។ (The food at his place is delicious. I've been there once.)	-	positive
ល់ម៉ោងមើលទៀតហើយ ។ (It's time to watch again.)	-	neutral
អ្នកកំពង់សោមធ្វើអត់ស្អាតទេ។ (The people of Kampong Som cannot prepare it well.)	-	negative

Table 5. Assessment of classification accuracy on the KP dataset. **bold**: highest. *italic*: second highest. Baseline: w/o thinking.

Model	Accuracy
kNN (Bigram) [1]	0.48
Decision Tree [1]	0.54
Random Forest [1]	0.60
SVM [1]	0.59
SGD [1]	0.58
SGD Tuning [1]	0.60
Qwen-1.7B (w/o FT)	0.00
Qwen-4B (w/o FT)	0.00
Qwen-8B (w/o FT)	0.00
Qwen-1.7B (Baseline)	0.77
Qwen-4B (Baseline)	0.74
Qwen-8B (Baseline)	*0.78*
Qwen-1.7B (Ours)	**0.84**
Qwen-4B (Ours)	**0.84**
Qwen-8B (Ours)	**0.84**

5.2.2 Qualitative Assessment of Reasoning and Explainability

In this section, we present a qualitative assessment of the model's reasoning and explainability. As described in Section 3, our fine-tuned models are trained to identify polarity-related keywords or phrases in the input text before producing the final label prediction. Table 7 provides some example Khmer text inputs and English translations, the model's derived polarity-related key words or phrases, and the final labels. The table shows that the model can not only give the correct labels but also extracts some polarity-related key words. These polarity-related key words are the basis or reasoning to understand or support why the model predicts what it predicts.

Table 6. Assessment of classification accuracy on the CKP dataset. **bold**: highest. *italic*: second highest. Params : trainable only. Baseline: w/o thinking. FT: fine-tuning. FT: fine-tuning.

Model	Params	Accuracy
XGBoost [17]	NA	0.85
RNN [17]	3.91M	0.87
CNN [17]	2.97M	**0.88**
Qwen-1.7B (w/o FT)	0M	0.00
Qwen-4B (w/o FT)	0M	0.00
Qwen-8B (w/o FT)	0M	0.00
Qwen-1.7B (Baseline)	34M	0.81
Qwen-4B (Baseline)	66M	0.85
Qwen-8B (Baseline)	80M	0.82
Qwen-1.7B (Ours)	34M	0.83
Qwen-4B (Ours)	66M	0.86
Qwen-8B (Ours)	80M	*0.87*

6 Future Work

Future work should include the following tasks:

1. The fine-tuning experiments were limited to the Qwen-3 model family only. Thus, future work should include other Khmer-supporting LLMs, such as GPT-0SS [18] or Sea-Lion [19].

2. The assessment of model explainability and reasoning is qualitative. Thus, future work should include other evaluation metrics, such as LLM as a judge.

3. The accuracies on the CP and CKP datasets are still limited because of the limited data quality and quantity. Thus, future work should focus on constructing a large-scale, high-quality Khmer polarity dataset.

Table 7. Qualitative assessment of the model reasoning and explainability based on the fine-tuned Qwen-8B model.

Input Text	Reasoning	Label
ម្ហូបនៅហ្នឹងឆ្ងាញ់ គ្រប់មុខបងជាពិសេសសម្លកកូរ។	ម្ហូបនៅហ្នឹងឆ្ងាញ់/សម្លកកូរ	positive
(The food here is delicious,		
especially the stir-fried soup.)	(dellicious food/stir-fried soup)	-
ស្បែកជើងខ្ញុំទិញឲ្យកូនៗហើយស្រួលពាក់ណាស់ចាំ ទៅទិញទៀត។	ស្រួលពាក់ណាស់	positive
(I bought the shoes for my children		
and they are so comfortable to wear.		
I can't wait to buy more.)	(comfortable to wear)	-
ប្រទេសកម្ពុជាស្ថិតនៅក្នុងតំបន់អាស៊ីអាគ្នេយ៍។	ស្ថិត	neutral
(Cambodia is located in Southeast Asia.)	(is located)	-
សេះនេះមានខ្មៅ។	មាន	neutral
(This horse is black.)	(is)	-
ម្ហូបមិនសូវមានរស់ជាតិឆ្ងាញ់។	មិនសូវមានរស់ជាតិឆ្ងាញ់	negative
(The food is not very tasty.)	(not very tasty)	-
គាត់ចូលចិត្តនិយាយអាក្រក់ពីអ្នកផ្សេង។	ចូលចិត្តនិយាយអាក្រក់	negative
(He likes to talk badly about others.)	(talk badly)	-

7 Conclusion

We propose the first explainable Khmer polarity classifier, which not only predicts the final label but also provides reasoning to justify its predictions. In this way, it becomes possible to explain why the model predicts what it does. Experimental results show that the fine-tuned Qwen-3 models either outperform or perform comparably to existing approaches on the KP and CKP datasets, respectively, for the Khmer polarity classification task.

Acknowledgment

We would like to acknowledge Ms. Udam Sonita for her contribution to the construction of the CKP dataset.

References

[1] Sokheng Khim, Ye Kyaw Thu, and Sethserey Sam. Sentiment polarity classification for khmer. In *2023 18th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, pages 1–6, 2023.

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd edition, 2025. Online manuscript released August 24, 2025.

[3] Raksmey Phann, Chitsutha Soomlek, and Pusadee Seresangtakul. Multi-class text classification on khmer news using ensemble method in machine learning algorithms. *Acta Informatica Pragensia*, 12, 03 2023.

[4] Rina Buoy, Nguonly Taing, and Sovisal Chenda. Khmer text classification using word embedding and neural networks, 2021.

[5] Md Rifatul Rifat and Abdullah Al Imran. *Incorporating Transformer Models for Sentiment Analysis and News Classification in Khmer*, pages 106–117. 12 2021.

[6] Sokleng Prom, Panharith Sun, Neil Ian Cadungog-Uy, Sa Math, and Tharoeun Thap. A bilstm-based sentiment analysis scheme for khmer nlp in time-series data. In *2024 International Conference on Information Science and Communications Technologies (ICISCT)*, pages 133–138, 2024.

[7] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu,

Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[8] Shengyi Jiang, Sihui Fu, Nankai Lin, and Yingwen Fu. Pretrained models and evaluation data for the khmer language. *Tsinghua Science and Technology*, 27(4):709–718, 2022.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[10] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[12] Jakub Šmíd and Pavel Přibáň. Prompt-based approach for Czech sentiment analysis. In Ruslan Mitkov and Galia Angelova, editors, *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 1110–1120, Varna, Bulgaria, September 2023. INCOMA Ltd., Shoumen, Bulgaria.

[13] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[14] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. *URL https://arxiv.org/abs/2305.14314*, 2, 2023.

[15] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[16] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu,

Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[17] Sonita Udam. Khmer profanity detection. Bachelor's thesis, Royal University of Phnom Penh, 2025.

[18] OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025.

[19] Raymond Ng, Thanh Ngan Nguyen, Yuli Huang, Ngee Chia Tai, Wai Yi Leong, Wei Qi Leong, Xianbin Yong, Jian Gang Ngui, Yosephine Susanto, Nicholas Cheng, Hamsawardhini Rengarajan, Peerat Limkonchotiwat, Adithya Venkatadri Hulagadri, Kok Wai Teng, Yeo Yeow Tong, Bryan Siow, Wei Yi Teo, Wayne Lau, Choon Meng Tan, Brandon Ong, Zhi Hao Ong, Jann Railey Montalan, Adwin Chan, Sajeban Antonyrex, Ren Lee, Esther Choa, David Ong Tat-Wee, Bing Jie Darius Liu, William Chandra Tjhi, Erik Cambria, and Leslie Teo. Sealion: Southeast asian languages in one network, 2025.

Fabric Detection in Real-World Manufacturing Using YOLOv5-Transformer Models

Makara Mao[1] **Keovichear Ouk**[1] **Hongly Va**[2] **Min Hong**[3]
[1]Department of Information Technology and Engineering, Royal University of Phnom Penh, Cambodia
[2]Department of Computer Science, Cambodia Academy of Digital Technology, Cambodia
[3]Department of Computer Software Engineering, Soonchunhyang University, Asan-si, 31538, Republic of Korea
mao.makara@rupp.edu.kh

Abstract

Detecting fabric defects, especially in textiles with complex textures, presents significant challenges due to the intricate nature of fabric patterns. Among various object detection methods, the YOLO algorithm is renowned for its real-time performance and accuracy. By treating object detection as a single regression problem, YOLO predicts bounding boxes and class probabilities from an entire image in one pass. This paper proposes a novel approach for fabric detection using the YOLOv5-Transformer model, which integrates Transformer architecture to enhance defect detection in textiles. YOLOv5, a fully convolutional neural network, strikes an optimal balance between speed and accuracy in end-to-end detection tasks. Leveraging the latest advancements in deep learning, YOLO achieves high detection speeds without significantly compromising precision, making it ideal for real-world applications. Our proposed YOLOv5-Transformer model surpasses other YOLOv5 variants, achieving an accuracy of 82.9%, representing a 5.6% improvement over YOLOv5s and YOLOv5n and a 2.7%–3.2% improvement over other versions YOLOv5m, YOLOv5l, YOLOv5x. Comparative performance metrics are also presented, including processing time on GPU, precision, recall, and F1 score.

Keywords: *YOLOv5, Object Detection, Fabric Dataset, Textile Materials.*

1 Introduction

With the rapid growth of artificial intelligence (AI) technology, there has been increasing interest in applying AI solutions to various industrial challenges, particularly fabric manufacturing. The textile industry faces significant challenges in fabric quality control due to the intricate patterns, textures, and designs involved [1]. Traditionally, manual inspection has been used for defect detection, but it is labor-intensive, time-consuming, and prone to human error. To overcome these limitations, computer vision techniques have emerged as powerful tools for automating fabric defect detection, significantly improving accuracy and efficiency [2]. Among these techniques, object detection algorithms like You Only Look Once (YOLO) have gained prominence due to their real-time performance and ability to handle large-scale data [3]. YOLO's strength lies in treating object detection as a single regression problem, predicting object locations and classifications in a single pass through the network. However, detecting fabric defects remains challenging due to textiles' complex textures and subtle irregularities [4].

Convolutional neural networks (CNNs) have become the dominant model in computer vision, excelling in tasks like image classification, object detection, and semantic segmentation [5]. Existing CNN-based object detection models can be divided into one-stage and two-stage detectors.

One-stage detectors, such as the YOLO family of models and single-shot multi-box detectors (SSD), treat object detection as a straightforward regression problem. These models achieve fast inference speeds by directly predicting bounding boxes and class labels in a single step, making those ideal for real-time applications [6]. However, it may sacrifice some accuracy, especially when dealing with minor or overlapping objects. Two-stage detectors, such as Faster R-CNN [7], offer higher accuracy by separating the detection process into two stages: the region proposal network (RPN) identifies candidate object regions,

and a second stage refines these proposals and classifies the objects [8][9]. While more accurate, two-stage models are often slower and more computationally intensive.

This paper presents the application of The development of the YOLOv5-Transformer model, which combines YOLOv5's object detection capabilities with the Transformer's multi-scale information fusion, and a detailed comparison of performance metrics, including processing time on (CPU and GPU), precision, recall, F1 score, and frames per second (FPS) across different models.

2 Related Work

Object detection is a critical task in computer vision that involves both localizing objects within an image and classifying them into predefined categories. The algorithm typically returns bounding boxes, confidence scores, and class labels for detected objects [8]. Recent advances in deep learning have led to the development of highly efficient and accurate object detection models, categorized primarily into one-stage and two-stage detectors.

In addition to real-world data, simulation datasets have become essential for training and evaluating these models, particularly when obtaining diverse or labeled real-world data is challenging. Simulated datasets offer controlled environments to replicate varying conditions, such as lighting changes, object occlusions, and different viewpoints, ensuring robust model performance across scenarios [9].

Processing these datasets involves data augmentation techniques, including scaling, rotation, and noise addition, which help improve model generalization. Proper pre-processing tasks, including normalization and converting annotations into formats such as COCO or YOLO, ensure compatibility with detection frameworks. Leveraging simulations accelerates algorithm development and provides a cost-effective way to test object detectors under complex or rare conditions without relying entirely on physical data collection [10].

One-stage detectors, such as YOLO and single-shot multi-box detectors (SSD), prioritize speed by directly predicting object locations and classifications from the image in a single network pass [11]. YOLOv5, in particular, is a widely used model for its high speed and relatively high accuracy. It uses convolutional neural networks (CNNs) to extract spatial features from the image, enabling fast real-time detection [12].

YOLOv5 employs a feature pyramid network (FPN) to detect objects at different scales, making it suitable for multi-scale object detection tasks. It divides the image into grid cells, predicting bounding boxes and confidence scores for each cell. Despite its speed, YOLOv5 may encounter difficulties with small or occluded objects due to its reliance on local image features and relatively limited receptive field in the CNN layers.

Another one, two-stage detectors like Faster R-CNN are designed for higher accuracy, though they may be slower compared to one-stage detectors [13]. These models generate region proposals (potential object locations) and then refine those proposals through a second-stage classification and bounding box regression.

The advantage of two-stage detectors lies in their ability to provide more precise localization, as the second stage refines the predictions made in the first stage. While two-stage detectors typically offer higher accuracy, they are computationally more expensive due to the added complexity of region proposal generation and refinement [14].

The proposed YOLOv5-Transformer model builds upon the advantages of the YOLOv5 architecture and Transformer networks. By combining the fast inference speed of YOLOv5 with Transformers enhanced global context capture ability, this model aims to balance speed and accuracy, particularly for detecting torn fabric defects [15].

YOLOv5 Backbone: The backbone consists of convolutional layers that extract hierarchical features from the input image. These features are passed through neck architecture, like the FPN, which can detect objects at different scales. Transformer Integration: The Transformer network is integrated into the detection pipeline to enlarge the receptive field beyond the local region CNNs covers. Transformers excel at modeling long-range dependencies in the image, enabling the model to effectively capture regional and global features [16].

This is especially important for fabric defect detection, where defects might span across large areas or appear at multiple scales. Two-Stage Refinement: After the initial predictions

by the YOLOv5 backbone, a second stage refines the predictions by leveraging the global context modeled by the Transformer.

This refinement stage helps improve the accuracy of bounding box localization, especially for challenging cases like overlapping or minor defects on complex fabric textures [17].

3 Methodology

In this section, we present a comprehensive overview of the model architecture of YOLOv5 in Figure 1. Our innovative model integrates YOLOv5 with a Transformer model to enhance accuracy and speed without compromising efficient object detection performance.

Specifically, we have incorporated Transformer layers into the YOLOv5 backbone to augment the model's capacity to capture global contextual information and long-range dependencies within the image.

This hybrid architecture combines the rapid, one-stage detection of YOLOv5 with the attention mechanisms of Transformers, enabling the model to discern objects in intricate, crowded scenes better. By integrating the Transformer model, our objective is to achieve enhanced precision in detection, particularly in demanding scenarios, while capitalizing on the real-time capabilities of YOLOv5.

Transformers have become increasingly popular in computer vision due to their capability to capture global context and long-range dependencies, making them particularly valuable for object detection.

The Transformer model divides an image into small patches, each flattened into a token and passed through multiple self-attention layers. Through this process, the self-attention mechanism computes relationships between patches, enabling the model to comprehend global dependencies and focus on different parts of the image.

Multi-head attention is applied in parallel to capture diverse aspects of the image, and the results are then processed through feed-forward layers with layer normalization to stabilize the training process. This approach assists the model in accurately predicting object classes and bounding boxes using learnable object queries.

In our integrated architecture, the features extracted from YOLOv5's backbone are segmented into patches and passed through Transformer layers to enhance the model's capacity to capture global context. The Transformer outputs are combined with YOLOv5's feature pyramid network (FPN) to enhance multi-scale detection, enabling the model better to detect small, occluded, or overlapping objects.

This hybrid approach combines the speed and efficiency of YOLOv5 with the attention-driven accuracy of the Transformer, resulting in a powerful and context-aware object detection model.

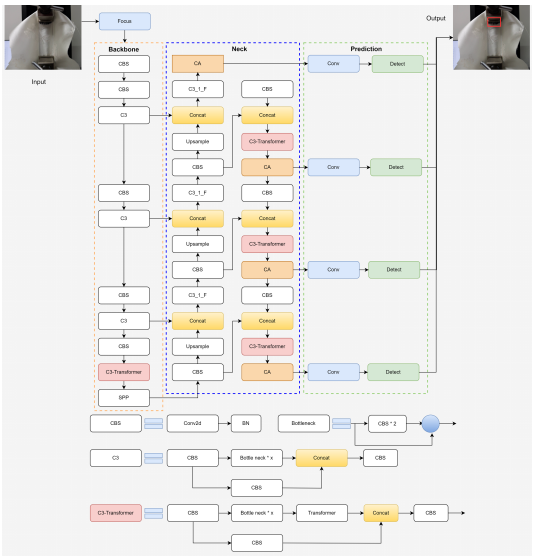


Figure 1. The architecture of the Transformer model.

3.1 Image Labeling

We utilized LabelMe, a versatile graphical image annotation tool depicted in Figure 2, to meticulously create polygonal annotations for object detection tasks. LabelMe's flexibility makes it well-suited for preparing datasets for different YOLO versions. By manually tracing polygons around objects in the images, we could accurately annotate object boundaries, which is crucial for training precise YOLOv5 models. These detailed annotations enable the model to detect objects with great accuracy. Once the annotations were completed, LabelMe generated JSON files containing the annotation data, which we converted into the required format for YOLOv5.

LabelMe's integration with Python also facilitated the seamless incorporation of data augmentation techniques. We utilized scripts to apply transformations such as rotation, scaling, and color adjustments directly to the annotated im-

ages and their corresponding JSON files. This process created an augmented dataset, enhancing the model's robustness and generalization capabilities. This integration has simplified the process of preparing and augmenting the dataset, ultimately contributing to the exceptional performance of YOLOv5 in various object detection scenarios.

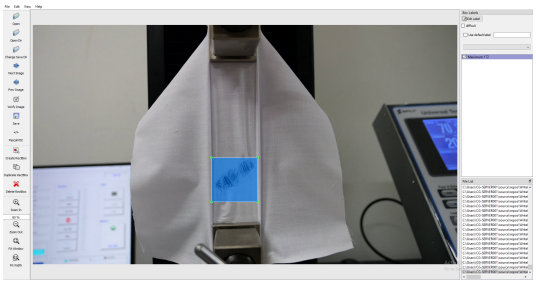


Figure 2. Labeling on Image by using the LabelMe tool.

3.2 Dataset

The dataset discussed in this section, captured using the LUMIX GH6 camera, is a crucial resource for fabric category experimentation. It consists of 7,620 images, the dataset was divided into 80% for training, 10% for validation, and 10% for testing to ensure balanced model evaluation, each categorized into one of five distinct fabric groups: Cotton Fabric Plain, Fabric Wide Hanbok Fabric Nobang DTP, Cotton Yarn-Dyed Check Stripe Plain Fabric, Hanbok Fabric, and Cotton Blend Plain Fabric. This rich diversity in fabric types facilitates comprehensive training and testing of object detection models on YOLOv5-Transformer.

3.3 Data Augmentation

To enhance the diversity and robustness of the fabric dataset, we employed a comprehensive set of data augmentation techniques, as illustrated in Figure 3. These techniques included horizontal and vertical flips to mirror the images, random rotations within a range of -45 to +45 degrees, Gaussian blurring to simulate out-of-focus conditions, and brightness adjustments varying from -22% to +22% for different lighting scenarios.

Additionally, we introduced random noise to mimic real-world imaging variations and darkened images to simulate low-light environments. We converted images to grayscale to focus on texture and pattern rather than color. Cropping was also applied to emphasize different parts of the fabric, aiding the model in learning from diverse perspectives. Our experiment focused on the transformations within the green polygons highlighted in Figure 3. Specifically, we examined random rotations and brightness adjustments to simulate real-world variations and enhance the model's accuracy in classifying fabrics under diverse conditions.

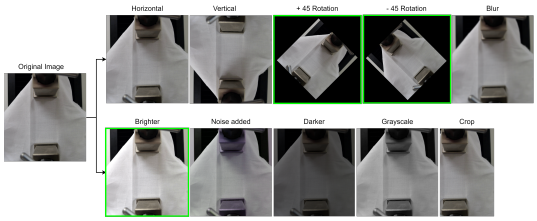


Figure 3. The techniques for applying the data augmentation.

4 Results and Discussion

In this section, we discuss the fabric prediction model based on YOLOv5-Transformer. We used a batch size of 32 for training to balance computational efficiency and model accuracy. A weight decay of 0.0005 was also employed to regularize the model and mitigate overfitting by penalizing large weights. We standardized the input image size to 320 x 320 pixels.

Training was carried out over 100 epochs to allow the model ample iterations to learn from the data. A learning rate of 0.0001 was chosen to ensure steady and reliable convergence. We opted for the stochastic gradient descent (SGD) optimizer due to its effectiveness in large-scale machine-learning tasks. We configured eight workers to parallelize data loading and expedite the training process.

Table 1. Experiment environment.

Component	Specification
Operating System	Windows 10 (64-bit)
Python Version	3.12.4
PyTorch Version	2.3.1
CUDA Toolkit	11.8
cuDNN Version	8.9.7
CPU	AMD Ryzen 9 5900X
GPU	NVIDIA GeForce RTX 4060
RAM	128 GB DDR4

In our training experiments, we compared the time performance of YOLOv5 and YOLOv5-Transformer on both CPU and GPU, as shown in Table 2. In addition to inference time, we now include the number of floating-point operations (GFLOPs) and model parameters for fair comparison.

The proposed YOLOv5-Transformer model maintains a moderate computational cost (43.3 GFLOPs, 20.4 M parameters), which is lower than YOLOv5m (47.9 GFLOPs, 21.2 M) and substantially lighter than YOLOv5x (203.8 GFLOPs, 86.7 M). Despite the integration of Transformer layers, our model achieved faster run-time on both CPU and GPU.

This efficiency improvement can be attributed to two main factors: (1) the Transformer block effectively enhances feature representation without significantly increasing convolutional complexity, thereby reducing redundant computations during feature extraction; and (2) the optimized feature fusion and reduced number of convolutional layers in the neck lead to fewer sequential operations, improving parallel processing efficiency on GPUs.

Consequently, YOLOv5-Transformer achieves a 69.61% CPU and 57.34% GPU speed improvement over YOLOv5x, while maintaining high accuracy. These results confirm that our design enhances computational efficiency without sacrificing model precision.

Table 3 compares the performance of various YOLOv5 variants and the proposed YOLOv5-Transformer model based on GPU latency and frames per second (FPS).

The proposed YOLOv5-Transformer achieves a high mean average precision (mAP) of 82.9%, representing an improvement of 2.2–4.4% over other YOLOv5 models, while maintaining a competitive FPS of 100.33 and GPU latency of 3183 ms.

Although the YOLOv5-Transformer has higher GFLOPs (43.3) than YOLOv5s (15.8) and YOLOv5n (4.1), its FPS remains comparable. This seemingly counterintuitive result arises because FLOPs alone do not fully reflect real-time performance. The Transformer block enhances feature representation without sub-stantially increasing memory transfer or kernel launch overhead, allowing more efficient parallelization on GPUs.

Moreover, our model employs optimized tensor operations and fewer sequential convolutional layers in the neck, reducing pipeline bottlenecks. As a result, the GPU utilization is higher and inference time remains stable, even with moderately increased computational complexity.

5 Conclusions

In this paper, we propose the YOLOv5-Transformer algorithm for detecting torn paths on fabric datasets from a material testing machine captured by a camera. This work aims to improve the existing YOLOv5 algorithm.

Our approach combines a convolutional network and a Transformer to design a new model within YOLOv5 and validate several improved measures to enhance YOLOv5's performance in fabric detection. Specifically, our proposed YOLOv5-Transformer module integrates the local observation capabilities of ConvNext and the global analysis capabilities of the Transformer, making a more significant contribution to improving detection accuracy compared to the original YOLOv5 module.

Additionally, we integrate the YOLOv5-Transformer module to reduce interference from background information, allowing the network to focus more effectively on valuable areas and further enhance detection accuracy. The proposed model achieved a 4.4% higher mAP compared to YOLOv5s on the fabric dataset.

While the YOLOv5-Transformer achieves the highest mAP, further exploration of other variants, fine-tuning hyperparameters, and incorporating advanced feature aggregation techniques could further improve accuracy while maintaining efficiency, surpassing other comparative to other existing papers such as Teacher Network, Improved YOLOv5s, FD-YOLOv5, and YOLOv5. The YOLOv5-Transformer demonstrated a 4.4% higher mAP compared to YOLOv5s, a 2.2% improvement over YOLOv5l, and a 2.6% increase over YOLOv5x. These results reflect the robustness of our proposed algorithm.

Furthermore, we plan to investigate the potential of our model on more complex datasets, such as 3D objects and multiple objects in a single image, using two-stage detectors. We also plan to expand the fabric dataset by adding more categories for each type of fabric. Additionally, we aim to compare our model with more versions of

Table 2. Comparison results of the model's performance by using CPU and GPU.

Models	GFLOPs	Parameters (M)	CPU (ms)	GPU (ms)
YOLOv5s	15.8	7.2	28,960	1,988
YOLOv5n	4.1	1.9	13,081	1,749
YOLOv5m	47.9	21.2	68,405	3,121
YOLOv5l	107.7	46.5	130,105	4,506
YOLOv5x	203.8	86.7	212,994	4,629
YOLOv5-Transformer	43.3	20.4	66,121	3,003

Table 3. Comparison of YOLO models and our proposed model on the fabric dataset using GPU.

Models	GFLOPs	P (%)	R (%)	mAP0.5 (%)	mAP (%)	GPU (ms)	FPS
YOLOv5s	15.8	99.7	100	99.5	78.5	1,988	101.88
YOLOv5n	4.1	99.7	100	99.5	78.5	1,749	100.82
YOLOv5m	47.9	99.9	100	99.5	80.7	3,121	84.28
YOLOv5l	107.7	99.8	100	99.5	80.5	4,506	110.92
YOLOv5x	203.8	99.9	100	99.5	80.3	4,629	102.91
YOLOv5-Transformer	43.3	99.9	100	99.5	82.9	3,183	100.33

the YOLO family and work on improving frames per second (FPS) performance for real-time systems.

Acknowledgment

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (NRF-2022R1I1A3069371), was funded by BK21 FOUR (Fostering Outstanding Universities for Research) (No.: 5199990914048).

References

[1] D. Zheng, "Pattern-driven color pattern recognition for printed fabric motif design," *Color Research & Application,* pp. 207–221, 2021.

[2] Q. Liu, C. Wang, Y. Li, M. Gao, and J. Li, "A fabric defect detection method based on deep learning," *IEEE Access,* vol. 10, pp. 4284–4296, 2022.

[3] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: Challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications,* vol. 82, pp. 9243–9275, 2023.

[4] M. M. Khodier, S. M. Ahmed, and M. S. Sayed, "Complex pattern Jacquard fabrics defect detection using convolutional neural networks and multispectral imaging," *IEEE Access,* vol. 10, pp. 10653–10660, 2022.

[5] Y. Amit, P. Felzenszwalb, and R. Girshick, "Object detection," in *Computer Vision,* Cham, Switzerland: Springer International Publishing, pp. 875–883, 2021.

[6] P. Adarsh, P. Rathi, and M. Kumar, "YOLO v3-Tiny: Object detection and recognition using one stage improved model," in *Proc. 6th Int. Conf. Advanced Computing and Communication Systems (ICACCS),* Coimbatore, India, pp. 687–694, 2020.

[7] W. Zhang, Q. Zhu, Y. Li, and H. Li, "MAM Faster R-CNN: Improved Faster R-CNN based on malformed attention module for object detection on X-ray security inspection," *Digital Signal Processing,* vol. 139, p. 104072, 2023.

[8] E. Kim, J. Lee, H. Jo, K. Na, E. Moon, G. Gweon, B. Yoo, and Y. Kyung, "SHOMY: Detection of small hazardous objects using the You Only Look Once algorithm," *KSII Transactions on Internet & Information Systems,* vol. 16, pp. 1–16, 2022.

[9] S. Ros, P. Tam, I. Song, S. Kang, and S. Kim, "A survey on state-of-the-art experimental simulations for privacy-preserving

federated learning in intelligent networking," *Electronic Research Archive,* vol. 32, pp. 1333–1364, 2024.

[10] S. Teng, J.-Y. Kim, S. Jeon, H.-W. Gil, J. Lyu, E. H. Chung, K. S. Kim, and Y. Nam, "Analyzing optimal wearable motion sensor placement for accurate classification of fall directions," *Sensors,* vol. 24, p. 6432, 2024.

[11] A. Kumar, Z. Zhi-Jie, and H. Lyu, "Object detection in real time based on improved single shot multi-box detector algorithm," *EURASIP Journal on Wireless Communications and Networking,* vol. 1, p. 204, 2020.

[12] N. O. Lynda, N. A. Nnanna, and M. M. Boukar, "Remote sensing image classification for land cover mapping in developing countries: A novel deep learning approach," *Int. J. Comput. Sci. & Netw. Security,* vol. 2, pp. 214–222, 2022.

[13] H. Wang and N. Xiao, "Underwater object detection method based on improved Faster R-CNN," *Applied Sciences,* vol. 13, p. 2746, 2023.

[14] D. Demetriou, P. Mavromatidis, P. M. Robert, H. Papadopoulos, M. F. Petrou, and D. Nicolaides, "Real-time construction demolition waste detection using state-of-the-art deep learning methods: Single-stage vs two-stage detectors," *Waste Management,* vol. 167, pp. 194–203, 2023.

[15] Z. Zhao and Y. Zhao, "YOLOv5s-Transformer: Improved YOLOv5 network for real-time detection of cigarette smoking based on image processing," in *Proc. 4th Int. Seminar on Artificial Intelligence, Networking and Information Technology (AINIT),* Nanjing, China, pp. 672–675, 2023, IEEE.

[16] Z. Zhang, Z. Lei, M. Omura, H. Hasegawa, and S. Gao, "Dendritic learning-incorporated vision transformer for image recognition," *IEEE/CAA Journal of Automatica Sinica,* vol. 11, pp. 539–541, 2024.

[17] K. S. Kumar and M. R. Bai, "LSTM-based texture classification and defect detection in a fabric," *Measurement: Sensors,* vol. 26, p. 100603, 2023.

A Study on ML-related Models for PM2.5 Air Quality Forecasting in Phnom Penh City

Makara Roeum[1] **Watcharaphong Yookwan**[2] **Lihour Nov**[1]
[1]Department of Computer Science, Cambodia Acedemy of Digital Technology, Phnom Penh, Cambodia
[2]Faculty of Informative Burapha University, Chonburi, Thailand
makara.roeum@cadt.edu.kh

Abstract

Air pollution from fine particulate matter ($PM_{2.5}$) is a major public health concern in Phnom Penh, Cambodia, driven by rapid urbanization, traffic congestion, and industrial activity. Accurate forecasting of $PM_{2.5}$ concentrations is essential for issuing early warnings and supporting policy interventions. This study evaluates six machine learning and deep learning models Light Gradient Boosting Machine (LightGBM), eXtreme Gradient Boosting (XGBoost), Extra Trees, Deep Neural Networks (DNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (Bi-LSTM) using OpenAQ data from May to August 2025. To mitigate extreme outliers, a logarithmic transformation was applied to positively skewed variables, improving stability and predictive reliability. Among the models, Extra Trees achieved the best performance, with RMSE of 0.0607, MAE of 0.0468, and R^2 of 0.9962. These findings demonstrate that well-optimized ensemble tree-based models can outperform complex deep learning approaches under local data constraints, providing an efficient and reliable solution for $PM_{2.5}$ forecasting and supporting timely public health interventions in Cambodia.

Keywords: *Air Quality, PM2.5, Forecasting, Machine Learning, Deep Learning, Phnom Penh*

I. Introduction

Air pollution from fine particulate matter ($PM_{2.5}$) is a leading environmental risk factor for respiratory and cardiovascular diseases, particularly in rapidly urbanizing cities where industrial emissions and traffic congestion are increasing. Accurate short-term forecasts of $PM_{2.5}$ concentrations enable public health agencies to issue timely warnings, help residents reduce exposure, and support evidence-based policy decisions.

Phnom Penh, Cambodia's capital, has experienced episodic and seasonally elevated $PM_{2.5}$ concentrations in recent years. Figure 1 illustrates a notable episode on 23 January 2025 when air quality reached the "Very Unhealthy" category. In this study, we analyze local air quality measurements collected from May to August 2025 to evaluate forecasting approaches that are appropriate for the city's environmental context and data constraints.

Figure 1 Phnom Penh Faces "Very Unhealthy" Air Quality Amid Rising $PM_{2.5}$ Levels on January 23, 2025 (Source: Kiripost).

A range of machine learning and deep learning methods has been applied to $PM_{2.5}$ forecasting. Tree-based ensembles such as XGBoost and LightGBM provide fast training, built-in handling of missing values, and interpretable feature importances but require explicit feature engineering to capture temporal dependencies [1], [2]. Recurrent neural networks including LSTM and Bi-LSTM are effective at capturing sequential dependencies and often achieve strong short-term predictive performance, but they require larger datasets and careful hyperparameter tuning [3], [4]. Deep neural networks can reach very high R^2 values on some datasets but are computationally intensive and less interpretable [5]. Hybrid approaches that combine meteorological model outputs with ensemble methods have demonstrated robust 1–2 day forecasts when such external inputs are available [6]. These studies highlight complementary strengths across model classes, but their performance is highly dependent on data quality, local conditions, and feature selection.

Among these ensemble methods, the Extremely Randomized Trees (Extra Trees) algorithm has shown promising results for environmental data forecasting. Extra Trees is an ensemble learning method based on decision trees, similar to Random Forest, but it introduces a higher degree of randomization during feature selection and split threshold determination. This randomization helps reduce variance and overfitting while maintaining low bias, making it robust to noisy or highly variable datasets such as those encountered in air quality monitoring.

Furthermore, Extra Trees is computationally efficient and requires minimal hyperparameter tuning, offering a strong balance between accuracy, interpretability, and training cost. These characteristics make it particularly suitable for practical air quality forecasting where data are limited or irregularly distributed.

Despite substantial progress in other regions, studies specifically targeting Phnom Penh remain limited [7], [8]. To address this gap, the present work evaluates representative high-performing models from prior studies including XGBoost, LightGBM, Extra Trees, optimized LSTMs, Bi-LSTM, and a deep neural network on a Phnom Penh dataset covering the May–August period. The analysis also investigates the effect of log transformation on skewed variables, a widely used technique for reducing variance instability and improving regression performance [9]. Through this comparative evaluation, the study provides new insights into which modeling strategies are most suitable for short-term $PM_{2.5}$ forecasting in Phnom Penh under local environmental and data constraints, while offering practical evidence to guide model selection and future research in similar urban contexts.

II. Methodology

A. Dataset

The dataset was obtained from OpenAQ and consists of air quality measurements recorded in Phnom Penh between May 1 and August 31, 2025, which corresponds to the rainy season in Cambodia. Each observation corresponds to an hourly measurement, yielding a total of 2,780 rows per parameter. The raw dataset contained both dynamic measurements and static metadata fields. Since the metadata (such as location identifiers, latitude/longitude) remained constant for a single monitoring site and contributed no predictive information, they were discarded. The dataset was reshaped from long to wide format using the `pivot_table` function in Pandas, with each parameter represented as a column indexed by its timestamp.

Five parameters were retained for analysis: PM1, $PM_{2.5}$, relative humidity, temperature, and UM003 (Table I). All values were rounded to two decimal places for consistency across measurements. The dataset was chronologically sorted by `datetimeLocal`, which served as the index for subsequent time series analysis. To preserve temporal dependencies, the dataset was split into training (May–July, 75%), validation (first half of August, 13%), and testing (second half of August, 12%). Collecting data exclusively during the rainy season is relevant, as precipitation and associated meteorological conditions can significantly influence $PM_{2.5}$ concentrations and temporal patterns.

Summary statistics are provided in Table II. PM1, $PM_{2.5}$, and UM003 show very high maximum values compared to their medians, indicating the presence of extreme outliers. In contrast, temperature and relative humidity vary within narrower ranges and appear more stable.

To assess distributional properties, skewness was computed using Equation (1). PM1, $PM_{2.5}$, and UM003 exhibit high positive skewness values (Table IV), confirming the presence of heavy-tailed distributions. This justifies the application of a log transformation to stabilize variance and reduce the influence of extreme outliers [9].

$$\text{Skewness} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{3/2}} \qquad (1)$$

The target variable for forecasting was $PM_{2.5}$, while the input features included PM1, relative humidity, temperature, and UM003. This setup ensured that models were trained on representative historical data, tuned on unseen validation data, and evaluated on an independent test set.

B. Investigated ML-related Models

In this study, we investigated both tree-based and deep learning models for PM2.5 prediction. The input features consisted of main environmental variables (*pm1*, *um003*, *temperature*, *relative humidity*) along with lagged PM2.5 values at 1, 3, 6, 12, and 24 hours to capture temporal dependencies [10]. For deep learning models, sequences of length 24 hours were created to exploit time-series patterns.

Tree-Based Models

Three tree-based models were tuned using GridSearchCV with 5-fold cross-validation [11], [12]:

- **Extra Trees Regressor** with hyperparameters: $n\_estimators = 300$, $max\_depth = None$, $min\_samples\_split = 5$.
- **XGBoost Regressor** with hyperparameters: $n\_estimators = 200$, $max\_depth = 6$, $learning\_rate = 0.05$.
- **LightGBM Regressor** with hyperparameters: $n\_estimators = 100$, $num\_leaves = 63$, $learning\_rate = 0.05$.

The cross-validated mean squared errors were 0.00724, 0.007762, and 0.008366 for Extra Trees, XGBoost, and LightGBM, respectively.

Deep Learning Models

For deep learning, we implemented DNN, LSTM, and Bi-LSTM models. Input sequences of 24 hours were flattened for DNN and kept as 3D sequences for LSTM/Bi-LSTM. Hyperparameters were tuned manually, and early stopping was applied to prevent overfitting [13]. The best configurations and validation losses were:

- **DNN:** units = 32, dropout = 0.3, learning rate = 0.001, validation loss = 0.1965.

Table I Description of Dataset Parameters

Parameter	Unit	Description
PM1	μg m^{-3}	Particulate matter with diameter ≤ 1 μm
$PM_{2.5}$	μg m^{-3}	Fine particulate matter with diameter ≤ 2.5 μm
Relative Humidity	%	Ratio of water vapor to saturation at given temperature
Temperature	°C	Ambient air temperature
UM003	particles cm^{-3}	Ultrafine particles with diameter < 0.3 μm

Table II Summary Statistics of Air Quality

Statistic	PM1	PM2.5	RH	Temp	UM003
Mean	8.14	13.25	66.21	26.91	666.45
Std	8.76	13.07	5.73	1.51	492.75
Min	0.00	0.00	45.96	23.22	69.39
25%	1.70	3.56	62.58	25.81	353.63
50%	5.03	9.09	67.27	26.78	544.18
75%	12.27	20.75	70.43	27.86	842.49
Max	131.82	229.16	79.56	32.11	9607.53

Table III Interpretation of skewness values

Skewness Value	Interpretation
-0.5 to 0.5	Approximately symmetric distribution
-1 to -0.5 or 0.5 to 1	Moderate skewness
< -1 or > 1	Highly skewed distribution

Table IV Skewness scores of dataset parameters

Parameter	Skewness
PM1	2.883
$PM_{2.5}$	3.585
Relative Humidity	-0.659
Temperature	0.423
UM003	5.083

- **LSTM:** units = 64, dropout = 0.1, learning rate = 0.001, validation loss = 0.1509.
- **Bi-LSTM:** units = 64, dropout = 0.2, learning rate = 0.001, validation loss = 0.1552.

Only tree-based models used GridSearchCV for systematic hyperparameter tuning; deep learning models relied on manual search. All models were trained and evaluated on the validation set, with final testing performed on unseen data.

C. Evaluation Metrics

To assess predictive performance, models were evaluated using three widely adopted regression metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the Coefficient of Determination (R^2). MAE computes the average magnitude of residuals without considering their direction, providing an intuitive measure of prediction accuracy; lower values indicate better performance. RMSE, in contrast, squares residuals before averaging and then applies a square root, thereby penalizing large errors more heavily than MAE. This makes RMSE more sensitive to outliers, which is particularly important in air quality forecasting where extreme pollution events may occur. Finally, R^2 quantifies the proportion of variance in the observed data that is explained by the model. Values closer to 1 indicate stronger explanatory power, while values below 0 suggest that the model performs worse than a simple mean baseline.

The three metrics are formally defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4}$$

where y_i represents the observed values, $\hat{y}_i$ the predicted values, $\bar{y}$ the mean of observed values, and n the number of samples. Together, these metrics provide a balanced evaluation: MAE offers interpretability, RMSE emphasizes large deviations, and R^2 provides a normalized measure of overall model fit.

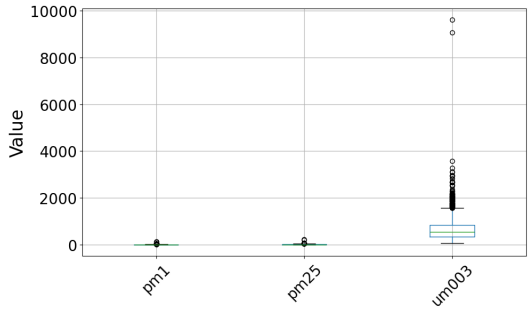
III. Results and Discussion

A. Data Preprocessing

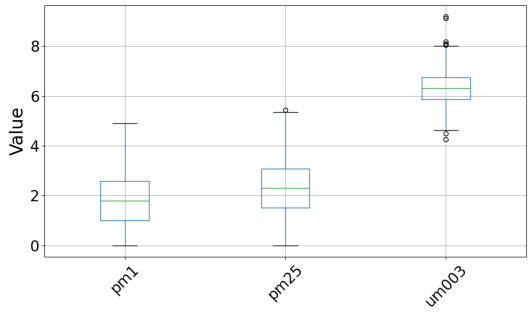
To stabilize variance and reduce the disproportionate influence of extreme observations, a logarithmic transformation was applied to positively skewed variables, defined as $x' = \log(1 + x)$ [9]. This transformation is commonly used in environmental time series where pollutant concentrations exhibit long-tailed distributions.

Figs. 2 demonstrate the impact of the log transformation on distributional symmetry. Before transformation , PM_1, $PM_{2.5}$, and UM003 exhibited heavy right skew with extreme outliers. After transformation, the distributions became more symmetric and compact, making them better suited for both statistical analysis and model training. By mitigating skewness and compressing extreme spikes, the log transformation enhances model robustness, particularly for metrics such as RMSE that are sensitive to large deviations.

The effect of this transformation is further illustrated in Fig. 3, which presents the correlation heatmap of the log-transformed parameters. Compared to the raw correlations, the log scale slightly reduced the magnitude of certain associations, indicating that extreme values in the raw data had previously inflated correlation coefficients. This adjustment provides a more reliable representation of relationships among features.



(a) Before log transformation



(b) After log transformation

Figure 2 Box plots of PM_1, $PM_{2.5}$, and UM003 before and after log transformation, showing improved symmetry and reduced skewness.

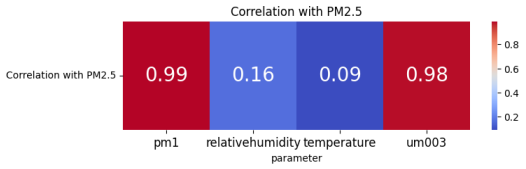


Figure 3 Correlation heatmap of log-transformed parameters, showing more reliable relationships compared to raw correlations.

B. Discussion

The developed $PM_{2.5}$ forecasting framework demonstrated strong predictive performance across both tree-based and deep learning architectures. Among all evaluated models, the ExtraTrees regressor achieved the lowest error (RMSE = 0.8039, MAE = 0.4906, and R^2 = 0.9939), outperforming other ensemble and neural network models. The superior performance of tree-based methods highlights their robustness in handling multivariate sensor data and temporal lag features without requiring extensive hyperparameter tuning or complex network architectures.

The results in Table V indicate that the ExtraTrees model consistently outperformed all other algorithms in this study, achieving an R^2 above 0.99 and significantly lower error metrics. These findings demonstrate that ensemble-based approaches

Table V Model performance comparison. Lower MAE/RMSE and higher R^2 indicate better performance.

Model	RMSE	MAE	R^2
ExtraTrees	**0.0607**	**0.0468**	**0.9962**
XGBoost	0.0647	0.0495	0.9957
LightGBM	0.0691	0.0530	0.9951
DNN	0.6163	0.4852	0.6145
LSTM	0.6004	0.4602	0.6341
Bi-LSTM	0.5663	0.4338	0.6744

are more efficient in capturing complex temporal dynamics of air pollutants compared to deep learning models, which required greater computational effort but yielded higher residual variance.

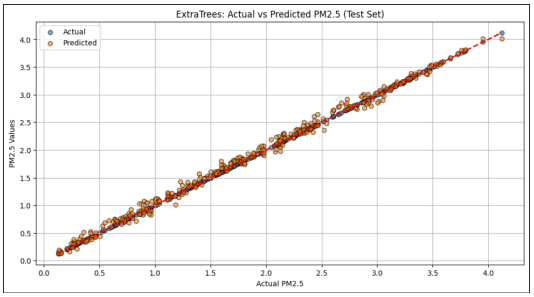


Figure 4 Scatter plot of actual vs. predicted $PM_{2.5}$ values on the test set.

To further contextualize the performance, Table VI presents a regional comparison between this study and the $PM_{2.5}$ forecasting system proposed by Minh et al. (2021) in Ho Chi Minh City, Vietnam. Their approach combined meteorological simulations from the Weather Research and Forecasting (WRF) model with the ExtraTrees regressor, achieving RMSE = 7.68 μg m^{-3}, MAE = 5.38 μg m^{-3}, and R^2 = 0.68. In contrast, the ExtraTrees model in this study achieved RMSE = 0.80 μg m^{-3} and R^2 = 0.9939, reflecting a substantial improvement in predictive accuracy and model stability.

It is important to note that this comparison is illustrative rather than direct, as the two studies employed different feature sets, input sources, and forecasting objectives. While Minh et al. (2021) integrated meteorological forecasts from the WRF model with machine learning inputs, the present study relied solely on measured sensor data and engineered temporal lag features such as differencing and a 24-hour sequence window. Moreover, Minh et al. (2021) focused on regional-scale forecasting, whereas the current work emphasizes localized, high-frequency prediction for an urban monitoring site in Phnom Penh. Despite these methodological differences, both cities share a tropical monsoon climate, allowing for a meaningful regional performance comparison.

The substantially lower error metrics observed in this study indicate that localized, data-driven models can outperform hybrid meteorological–machine

Table VI Comparison of $PM_{2.5}$ forecasting performance between this study and Minh et al. (2021).

Study	RMSE (μg m^{-3})	MAE (μg m^{-3})	R^2
This Study (Phnom Penh, 2025)	0.80	0.49	0.9939
Minh et al. (2021, Ho Chi Minh City)	7.68	5.38	0.68

learning systems when high-resolution, site-specific observations are available. This finding underscores the potential of lightweight, sensor-based forecasting frameworks for operational deployment in Southeast Asian urban environments, particularly in regions with limited access to meteorological modeling resources.

These findings reinforce that highly complex deep learning architectures are not always necessary to achieve state-of-the-art forecasting accuracy. Instead, well-optimized ensemble models such as ExtraTrees can provide a strong balance between precision, interpretability, and computational efficiency. For Cambodia, this result has direct implications for evidence-based policymaking and public health protection, as reliable and accessible air quality forecasts can support early warnings and mitigation measures for pollution episodes.

C. Limitations and Future Work

The dataset used in this study spans only four months (May–August 2025), which may limit the models' ability to capture seasonal and long-term variations in air pollution. Air quality in Phnom Penh can be influenced by distinct dry and wet season patterns, transboundary pollution, and festival-related activities. As such, the restricted temporal coverage may reduce the generalizability of the models to other periods or years. Future studies will integrate data from multiple years and sources, including meteorological and satellite-derived variables, to enhance model robustness and adaptability.

In addition, an analysis of error cases revealed that model performance tended to decline during abrupt changes in weather conditions, such as sudden rainfall or shifts in wind direction. These conditions can alter $PM_{2.5}$ concentrations rapidly and nonlinearly, posing challenges for data-driven models trained primarily on smooth temporal patterns. Future work could explore hybrid physical–statistical modeling frameworks or ensemble correction strategies that dynamically adjust predictions under such transient conditions.

IV. Conclusion

This study highlights the potential of machine learning to advance air quality forecasting in Phnom Penh, Cambodia, with a specific focus on $PM_{2.5}$ prediction. By applying logarithmic transformation to mitigate skewness, we improved data quality and model stability, ensuring more reliable learning outcomes. Among the models evaluated, Extra Trees consistently delivered superior performance, surpassing both other ensemble techniques and deep learning approaches. Its combination of predictive accuracy, robustness, interpretability, and computational efficiency makes it a highly practical choice for real-world forecasting systems in resource-constrained settings.

References

[1] B. Pan, "Application of xgboost algorithm in hourly $PM_{2.5}$ concentration prediction," in *IOP Conference Series: Earth and Environmental Science*, vol. 113. IOP Publishing, 2018, p. 012127.

[2] J. Zhong, X. Zhang, K. Gui, Y. Wang, H. Che, X. Shen, L. Zhang, Y. Zhang, J. Sun, and W. Zhang, "Robust prediction of hourly $PM_{2.5}$ from meteorological data using lightgbm," *National Science Review*, vol. 8, no. 10, p. nwaa307, 2021.

[3] H. D. Tran, H.-Y. Huang, J.-Y. Yu, and S.-H. Wang, "Forecasting hourly $PM_{2.5}$ concentration with an optimized lstm model," *Atmospheric Environment*, vol. 315, p. 120161, 2023.

[4] M. Zhang, D. Wu, and R. Xue, "Hourly prediction of $PM_{2.5}$ concentration in beijing based on bi-lstm neural network," *Multimedia Tools and Applications*, vol. 80, no. 16, pp. 24 455–24 468, 2021.

[5] Y. Liang, J. Ma, C. Tang, N. Ke, and D. Wang, "Hourly forecasting on $PM_{2.5}$ concentrations using a deep neural network with meteorology inputs," *Environmental Monitoring and Assessment*, vol. 195, no. 12, p. 1510, 2023.

[6] V. T. T. Minh, T. T. Tin, and T. T. Hien, "$PM_{2.5}$ forecast system by using machine learning and wrf model: a case study of ho chi minh city, vietnam," *Aerosol and Air Quality Research*, vol. 21, no. 12, p. 210108, 2021.

[7] P. Sokharavuth, S. Thiv, C. Nara, C. Him, S. Sokyimeng, D. K. Henze, R. Holmes, J. C. I. Kuylenstierna, C. S. Malley, E. Michalopoulou *et al.*, "Air pollution mitigation assessment to inform cambodia's first clean air plan," *Environmental Research*, vol. 220, p. 115230, 2023.

[8] C. Samhaiy, S. Chhin, and R. Yim, "Impacts of air pollution on richness and abundance of bird species in phnom penh urban habitats," *Insight: Cambodia Journal of Basic and Applied Research*, vol. 6, no. 1, pp. 23–39, 2024.

[9] M. B. Johansen and P. A. Christensen, "A simple transformation independent method for outlier definition," *Clinical Chemistry and Laboratory Medicine*, vol. 56, no. 9, pp. 1524–1532, 2018.

[10] B. Shao, "Application of xgboost algorithm in hourly pm2.5 concentration prediction," *IOP Conference Series: Earth and Environmental Science*, vol. 113, p. 012127, 2018.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[12] B. Bischl, M. Binder, M. Lang *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices and open challenges," *arXiv preprint arXiv:2107.05847*, 2021.

[13] L. Prechelt, "Early stopping–but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

Classification of Rice Leaf Disease Using Convolutional Neural Network Models

Sokhey Kim **Hongly Va**
Cambodia Academy of Digital Technology, Phnom Penh, Cambodia
Sokhey.kim@cadt.edu.kh

Abstract

Rice is a crucial crop globally, particularly in Asia, where it serves as a staple food. However, various diseases severely affect rice crop yields, and without proper detection, these diseases can spread, leading to a substantial decline in production. In extreme cases, diseases can result in a total crop loss, threatening food security. Deep learning, particularly Convolutional Neural Networks (CNNs), has become the standard method for image identification and classification tasks. Accurate diagnosis of rice diseases is essential to mitigate these impacts, yet current diagnostic methods are often inefficient, requiring specialized equipment. This study developed a deep learning–based automatic rice disease diagnosis method using an ensemble of CNN models. The method, built on deep learning, utilized a dataset of 17,500 images covering seven types of rice diseases, including bacterial blight, hispa, leaf blast, and others. The Ensemble Model, which combined several submodels, was the core of this method. Validation showed that EfficientNet-B0, DenseNet-121, and MobileNetV2 were the most effective submodels, achieving an overall accuracy of 96%. The Ensemble Model minimized confusion between disease types, reducing misdiagnosis and enhancing disease recognition accuracy, making it a reliable tool for rice disease detection.

Keywords: *Rice leaf diseases, Deep learning classifier, CNNs, Ensemble learning*

1 Introduction

Rice is one of the most important staple crops worldwide, and its yield depends on multiple factors, including soil type, weather conditions, irrigation facilities, geography, seed selection, and biological threats [1], [2]. Among these threats, plant diseases are a major cause of yield reduction and economic loss, particularly in Asia, where rice is a dominant food source [3]. Bacterial, fungal, and viral infections, including Bacterial Blight, Hispa, Leaf Blast, Sheath Blight, Tungro, and Brown Spot, frequently damage rice crops, leading to poor grain quality, lower production, and, in severe cases, complete crop failure. Protecting rice crops from these diseases is therefore critical for food security.

Traditional diagnosis of rice diseases relies on visual inspection, expert knowledge, and reference guides [4]. While these methods are helpful, they are time-consuming, labor-intensive, and prone to human error. They also often require trained specialists or specialized equipment that may not be accessible to farmers in rural areas. Consequently, automated rice disease detection methods have received increasing attention as a faster and more reliable alternative.

Early studies applied computer vision and machine learning techniques for crop disease detection, including handcrafted feature extraction and classifiers such as support vector machines (SVMs) and random forests [5]. Although these approaches achieved moderate success, they relied heavily on manual feature engineering, which limited their generalization to diverse field conditions.

More recently, deep learning has emerged as a powerful alternative for classifying plant diseases. CNNs in particular have demonstrated strong performance by automatically learning hierarchical features from raw images [6]. Models such as ResNet, DenseNet, and EfficientNet have achieved high accuracies across various crops. However, relying on a single CNN architecture may still lead to overfitting or misclassification, especially for visually similar diseases. To overcome these limitations, ensemble learning, which combines predictions from multiple models, has been proposed as an effective strategy to improve robustness and reduce errors [7].

In this study, we focus on the development and evaluation of deep learning models for rice

disease classification. A total of 63,889 images were initially collected from public sources, including Kaggle, Mendeley, and Roboflow, covering healthy leaves and ten rice diseases. To ensure balanced representation and avoid class bias, undersampling was applied, resulting in a final dataset of 17,500 images evenly distributed in seven categories (six major rice diseases and healthy). Three CNN architectures, including EfficientNet-B0, DenseNet-121, and MobileNetV2, were trained and evaluated, and their predictions were integrated through ensemble learning. The experimental results show that the ensemble model consistently outperformed individual CNNs, achieving an overall accuracy of 96%.

The main contributions of this paper are as follows.

- We trained a deep learning network to diagnose seven different types of rice disease, including healthy samples.
- We evaluated three state-of-the-art CNN architectures, such as EfficientNet-B0, DenseNet-121, MobileNetV2, for rice disease detection.
- We designed an ensemble learning approach that integrates multiple CNN models to improve robustness and demonstrated a high accuracy rate of 96%, which is considered a good result.

The remainder of this paper is organized as follows. In the next section, we review related work that is pertinent to our own. Section III discusses the methodology of a model. Section IV describes the experiment and the analysis of detecting rice leaf disease. Finally, the discussion and conclusion are summarized at the end of this paper.

2 Related Work

To better position our approach, we review related research in rice disease detection and ensemble learning. Prior works can be broadly grouped into three categories: (1) traditional computer vision and machine learning techniques, (2) deep learning based approaches for plant disease detection, and (3) ensemble learning strategies.

Early work in crop disease detection relied on handcrafted feature extraction combined with classical classifiers. Techniques such as color analysis, texture descriptors, and shape features were commonly applied, followed by classifiers like SVM, k-nearest neighbors (KNN), and random forests [4], [5]. While these methods achieved moderate success, they required domain expertise for feature design and performed poorly in complex field conditions where lighting and backgrounds varied significantly.

The emergence of CNNs has revolutionized image-based plant disease detection by eliminating the need for manual feature engineering. CNNs have been widely used to classify diseases in crops such as rice, maize, wheat, and tomato. Deep learning has also been applied to other rice-related tasks. For example, Deng et al. [8] used CNN-based models to automatically detect productive tillers in rice, illustrating the broader applicability of deep learning techniques in rice phenotyping and crop analysis. In the domain of disease detection, Deng et al. [6] proposed an in-field rice disease detection system using deep CNNs, demonstrating strong performance under real-world conditions. Other studies have evaluated architectures such as ResNet, Inception, DenseNet, and EfficientNet for leaf disease classification, achieving higher accuracy compared to traditional machine learning methods [9]. Despite these advances, individual CNNs often struggle to distinguish visually similar diseases, resulting in misclassification.

Ensemble learning has been increasingly adopted to address the limitations of single CNN models. By combining predictions from multiple classifiers, ensemble approaches enhance robustness, mitigate overfitting, and yield more reliable results. For instance, Deng et al. [6] employed ensemble CNNs to classify rice diseases, achieving higher accuracy than individual networks. Similarly, Feng et al. [7] demonstrated that ensemble methods improved yield prediction in alfalfa using hyperspectral imagery. In plant disease detection, ensemble strategies such as majority voting, soft probability averaging, and stacking have consistently shown advantages over single models.

While these approaches are promising, several challenges remain. Many datasets are imbalanced, with limited samples for certain diseases, which reduces generalization ability. Fur-

thermore, most prior studies emphasize the performance of single models without fully exploiting ensemble methods to address interclass similarity and misclassification among visually similar rice diseases. These limitations motivate our work on developing a balanced dataset and integrating multiple CNN architectures, such as EfficientNet-B0, DenseNet-121, and MobileNetV2 to an ensemble model to enhance classification accuracy and robustness.

3 Methodology

3.1 Data acquisition

Deep learning requires a large number of training images to achieve good results. In this study, a total of 63,889 rice leaf images were collected from public sources, including Kaggle, Mendeley, and Roboflow. The datasets included healthy leaves and ten rice diseases such as Bacterial Blight, Brown Spot, Rice Hispa, Leaf Blast, Sheath Blight, Tungro, Leaf Scald, Narrow Brown Spot, Neck Blast, and Leaf Smut [10], [11]. Table 1 summarizes eleven common rice diseases, their causal agents, symptoms, and impacts on the plant, while Figure. 1 shows the distribution of the initially collected samples. To ensure a well-rounded dataset, we incorporated two types of images. **White Background Images,** these images were taken in controlled settings with consistent lighting. This makes it easier to see the rice plants and their symptoms. The uniform background helps in training machine learning models by reducing distractions. **Field Background Images,** were captured in real agricultural environments. They show the rice crops in their natural conditions, with various backgrounds and lighting. This variety helps the model to learn how diseases appear in real life, making it more effective for practical use. However, to balance the dataset of each class, we apply undersampling, a technique that reduces the total number of images of some classes, resulting in a balanced dataset of 2,500 images with equal representation of healthy crops and each disease [12]. Each category, including healthy crops and six rice diseases, contains 2,500 images, with 1,700 field background images and 800 white background images per class. Therefore, the final dataset used for model training and evaluation comprised 17,500 images across seven categories (six rice diseases and one healthy category), with equal class representation.

3.2 Data preprocessing

Image Processing is performed to reduce overfitting of deep learning models during training and validation. Before the model reads the images, the pixels in the image are normalized. Then, we applied random affine transformations, including rotations, shifts, and shearing, to introduce the models to different perspectives. Brightness adjustments, Gaussian blur, and flipping were also applied to help the models learn from various conditions. Additionally, the images were resized to 224×224 as required by the architectures of selected deep learning models [13]. Figure 2 shows the processing of data preprocessing step by step of the models.

3.3 Model development

3.3.1 Convolutional neural network Models

The structure of the CNN has a crucial influence on the performance of the final model. To evaluate the effectiveness of different architectures, we selected three state-of-the-art CNN models, including EfficientNet-B0 [14], DenseNet-121 [15], and MobileNetV2 [16]. These models were chosen due to their balance between accuracy and computational efficiency, making them suitable candidates for agricultural applications, including potential deployment on mobile devices.

3.3.2 Ensemble learning

Ensemble learning combines multiple base models to enhance robustness and mitigate the risk of misclassification by leveraging complementary decision boundaries. In this study, the predictions of EfficientNet-B0, DenseNet-121, and MobileNetV2 were combined using a voting-based ensemble strategy [17]. Specifically, we implemented soft voting, in which the predicted class probabilities from each model were averaged, and the class with the highest probability was selected as the final output. This approach reduces bias from individual models and provides more stable performance compared to single CNNs.

3.3.3 Fine-tuning of the models

Fine-tuning was applied to optimize the CNN models for the rice disease dataset. Transfer learning was used to initialize each model

Table 1: Overview of rice viral diseases, including their causes, symptoms, and impact.

Disease Name	Cause	Description	Affected Plant Part
Healthy	None	Normal green leaves with no disease symptoms	Whole Plant
Bacterial Blight	Bacterial	Yellowish-brown streaks on leaves, leading to wilting and drying	Leaf
Neck Blast	Fungal	Collapse and decay of the neck area, affecting grain formation	Panicle
Leaf Scald	Fungal	Browning and wilting of leaf tips; reduced vigor	Leaf
Narrow Brown Leaf Spot	Fungal	Narrow, elongated brown lesions on leaves; leads to leaf death	Leaf
Leaf Smut	Fungal	Dark, smutty spores on leaves; affects photosynthesis	Leaf
Leaf Blast	Fungal	Gray-centered lesions with brown borders, affecting plant growth	Leaf
Sheath Blight	Fungal	Oval, water-soaked lesions on leaf sheaths, leading to plant lodging	Leaf Sheath, Leaf
Brown Spot	Fungal	Small brown spots with yellow halos, reducing grain quality	Leaf, Grain
Hispa	Parasitic	Small brown spots with yellow halos, reducing grain quality	Leaves
Tungro	Virus	Stunted growth and yellow-orange discoloration	Leaves, Whole Plant

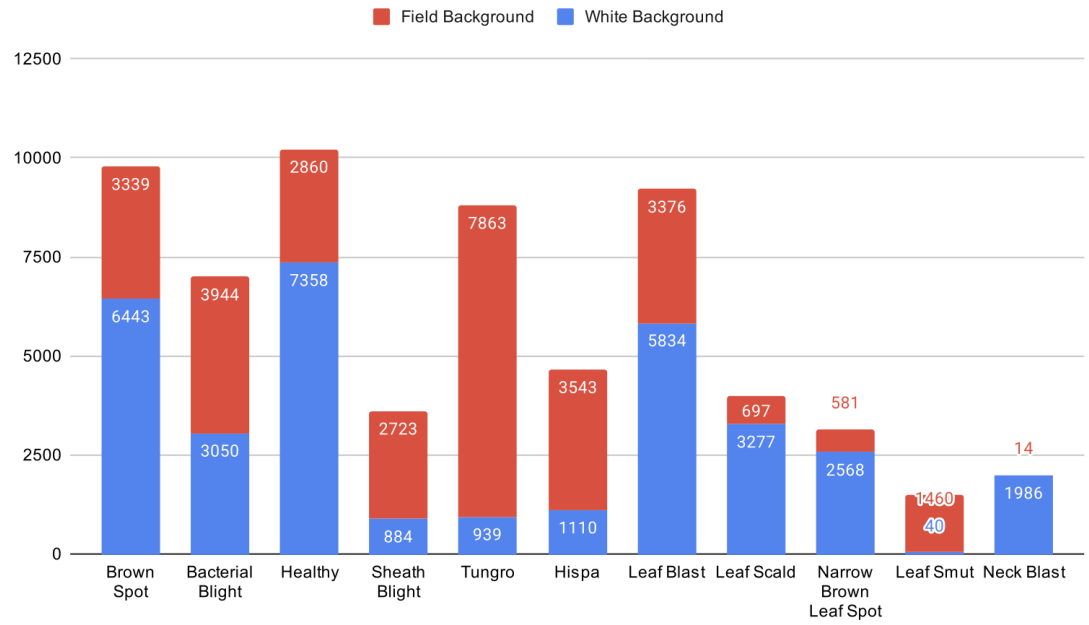


Figure 1: Overview of all the datasets before cleaning. The data consists of 11 types of rice disease, including healthy samples.

with pre-trained ImageNet weights, enabling the knowledge learned from large-scale image classification to be adapted to the agricultural domain [18]. We then fine-tuned all layers of EfficientNet-B0, DenseNet-121, and MobileNetV2 on our dataset. Each model was trained for 20 epochs with early stopping criteria to prevent overfitting. This fine-tuning process enabled the networks to adapt to the specific visual features of rice diseases while reducing training time and computational cost.

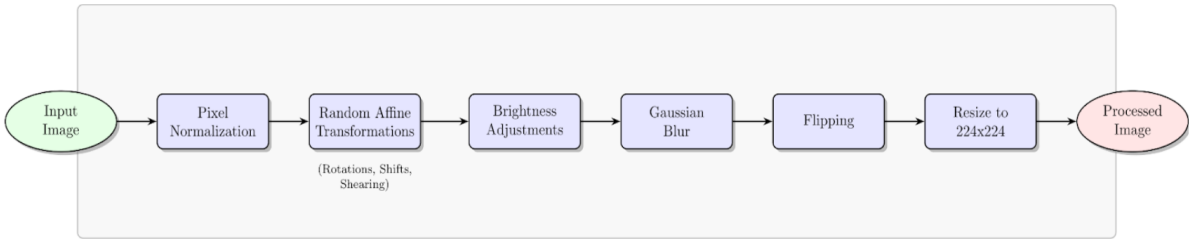


Figure 2: Image preprocessing pipeline used in the models.

4 Experimental results and analysis

This section mainly focuses on the empirical study of the proposed method.

4.1 Evaluation metrics

To evaluate the performance of the proposed method, we used four standard classification metrics, such as Accuracy, Precision, Recall (Sensitivity), and F1-score.

Accuracy measures the overall proportion of correctly classified samples and is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

Precision reflects the classification accuracy of the classifier, indicating all samples predicted to be positive examples. The proportion of correctly predicted samples.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall reflects the recall ability of the classifier, indicating that all positive samples are correctly predicted in the sample proportion tested.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F1-score is the harmonic mean of Precision and Recall, providing a balanced measure of both metrics:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

4.2 Experimental dataset

The experiments were conducted using the proposed model applied to the rice disease dataset. The dataset was divided into training, validation, and testing subsets with a ratio of 7:2:1. This split was applied across all seven categories to ensure balanced representation. As a result, 70% of the samples (12,250 images) were used for training, 20% (3,500 images) for validation, and the remaining 10% (1,750 images) for testing. This stratified partitioning provided sufficient samples in each category for effective training and evaluation.

4.3 Results and analysis

All experiments were implemented in Python (3.10.11) using TensorFlow GPU 2.10 with CUDA 12.8.0 acceleration. Training was conducted on a system equipped with an AMD Ryzen 7 6800H CPU, an NVIDIA RTX 3060 GPU (6 GB), and 16 GB of RAM. The models were trained using the Adam optimizer with a learning rate of 0.0001 and a batch size of 32 for a total of 30 epochs. Specifically, 10 epochs were used for initial training followed by 20 epochs of fine-tuning. Sparse categorical cross-entropy loss was employed to optimize multi-class classification across the seven rice disease categories. The detailed software and simulation parameters are summarized in Table 2.

4.3.1 Models performance

As summarized in Table 3, the ensemble model outperformed all individual CNNs, achieving the highest accuracy, precision, recall, and F1-score (0.96). Table 4 further shows that this advantage extended across nearly all disease classes, with particularly strong results for Sheath Blight and Bacterial Blight (F1-scores of 0.98), while reducing errors in more challenging classes such as Brown Spot and Healthy.

- **EfficientNet-B0** achieved the highest single-model validation accuracy 0.95. Its confusion matrix (Figure 4a) shows high true positives for Bacterial Blight (240),

Table 2: Simulation software and parameters.

Software/Parameters	Value
Operating System	Windows 11, 64-bit
Programming Language	Python version = 3.10.11, TensorFlow-GPU = 2.10
CPU	Ryzen 7 6800H
GPU	NVIDIA RTX 3060 (6GB)
RAM	16 GB
Batch Size	32
Optimizer	Adam
Learning Rate	0.0001
Epochs	30 (10 initial trainings + 20 fine-tuning)

Table 3: Evaluation of the ensemble model's overall performance on the test set.

Model	Accuracy	Precision	Recall	F1-score
EfficientNet-B0	0.95	0.95	0.95	0.95
DenseNet-121	0.93	0.92	0.92	0.92
MobileNetV2	0.90	0.87	0.88	0.87
Ensemble Model	**0.96**	**0.96**	**0.96**	**0.96**

Healthy (237), and Sheath Blight (245). Misclassifications included eight Brown Spot instances labeled as Leaf Blast and ten Healthy samples classified as Hispa. The classification heatmap (Figure 3a) reports precision, recall, and F1-scores between 0.90 and 0.98.

- **DenseNet-121** obtained the second-highest accuracy 0.93. As shown in Figure 4b, it correctly identified most Sheath Blight (244) and Healthy (231) samples, but misclassified 16 Brown Spot images as Bacterial Blight and eight as Leaf Blast. The heatmap Figure 3b confirms strong results for Sheath Blight and Tungro (F1 = 0.96), while Brown Spot lagged (F1 = 0.90).

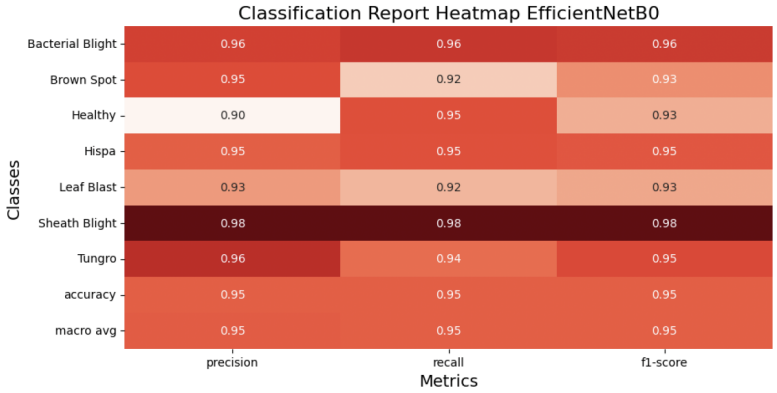
- **MobileNetV2** achieved an accuracy of 0.90. Its confusion matrix (Figure 4c) shows correct predictions for Sheath Blight (239) but notable errors, such as 38 Healthy samples labeled as Hispa and 22 Leaf Blast samples as Brown Spot. The heatmap (Figure 3c) shows an F1 score of 0.87, with strong results for Sheath Blight 0.95 and Tungro 0.96, but lower performance on Healthy (0.81) and Hispa 0.84. These findings illustrate MobileNetV2's trade-off between computational efficiency and predictive accuracy, relevant for mobile deployment scenarios.

- **Ensemble learning model** combined EfficientNet-B0, DenseNet-121, and MobileNetV2 using soft voting. As shown in Figure 4d, it achieved balanced results with minimal misclassifications (e.g., one sample of bacteria blight was classified as brown spot, and one healthy sample was classified as hispa). The heatmap (Figure 3d) indicates consistently high precision 0.91–0.99 and recall 0.90–0.99, leading to per-class F1 scores between 0.90 (Brown Spot) and 0.98 (Sheath Blight). These results confirm that the ensemble delivers the most robust and stable performance in all categories.

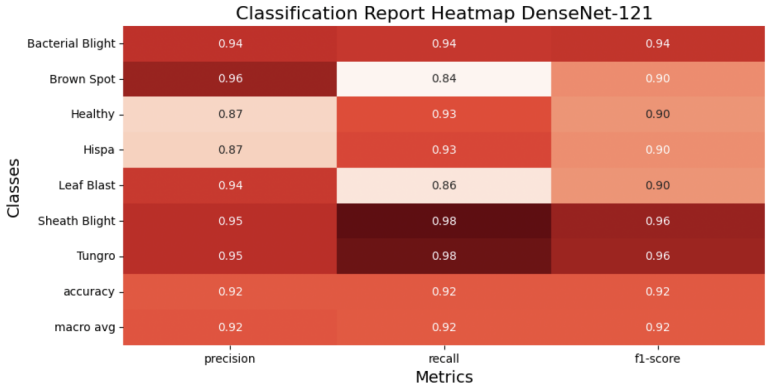
4.3.2 Training and validation visualization

Figure 5 shows the comparison of the accuracy of all three models, including EfficientNet-B0, DenseNet-121, and MobileNetV2 of training and validation loss.

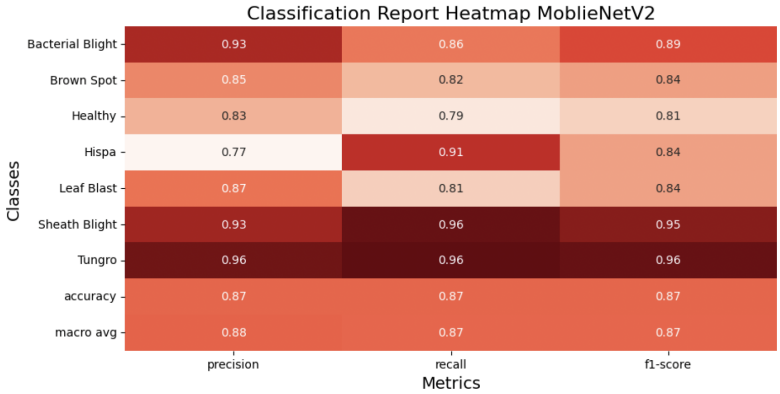
During training, as shown in Figure 5a and Figure 5b, EfficientNet-B0 consistently achieved the strongest results among the evaluated models. It reached approximately 95% training accuracy by the final epochs and maintained the lowest training loss, indicating its effectiveness in capturing complex patterns within the dataset without overfitting. DenseNet-121 also demonstrated robust learning behavior, showing steady improvements in accuracy alongside a decreasing loss curve. Although its loss values were slightly higher than those of EfficientNet-B0, the dense connectivity of its layers supported effective feature reuse and efficient learning of challenging patterns. In contrast, MobileNetV2 ex-



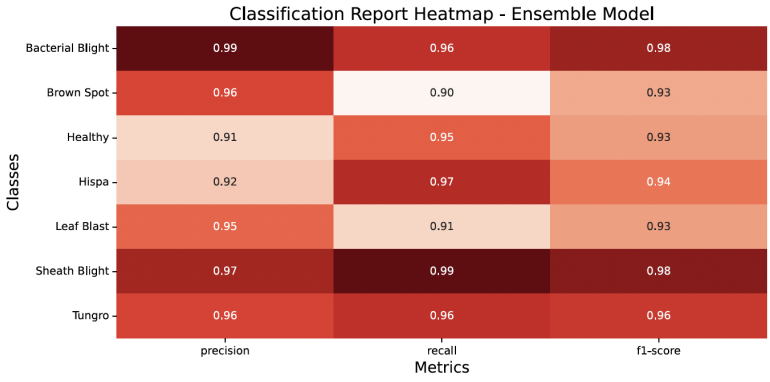
(a) EfficientNet-B0



(b) DenseNet-121



(c) MobileNetV2

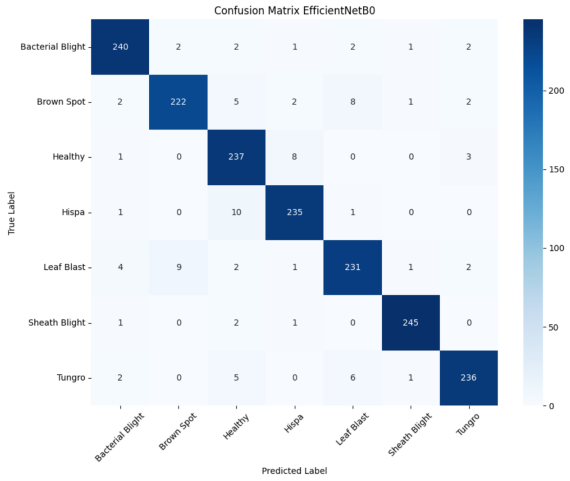


(d) Ensemble Model

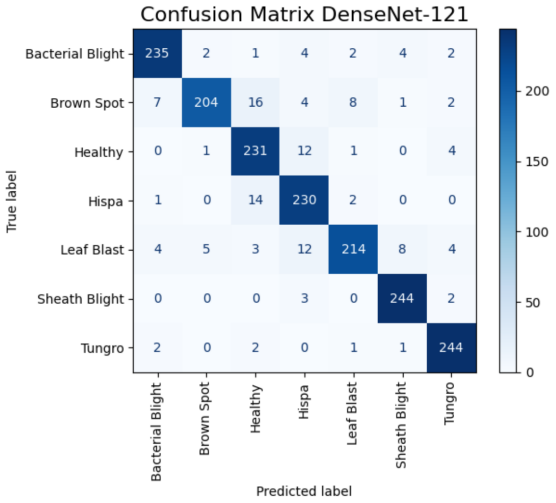
Figure 3: Heatmaps of the classification reports for individual models and the ensemble model.

Table 4: F1-scores of each model across all disease classes, with the ensemble model demonstrating the most balanced performance.

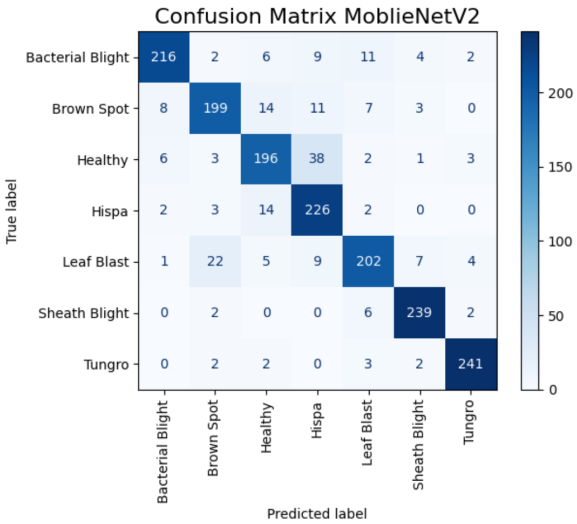
Disease	EfficientNet-B0	DenseNet-121	MobileNetV2	Ensemble Model
Bacterial Blight	0.96	0.94	0.88	**0.98**
Brown Spot	0.93	0.90	0.85	**0.93**
Healthy	0.94	0.91	0.81	**0.93**
Hispa	0.95	0.92	0.84	**0.94**
Leaf Blast	0.94	0.91	0.86	**0.93**
Sheath Blight	0.98	0.96	0.95	**0.98**
Tungro	0.96	0.96	0.96	**0.96**



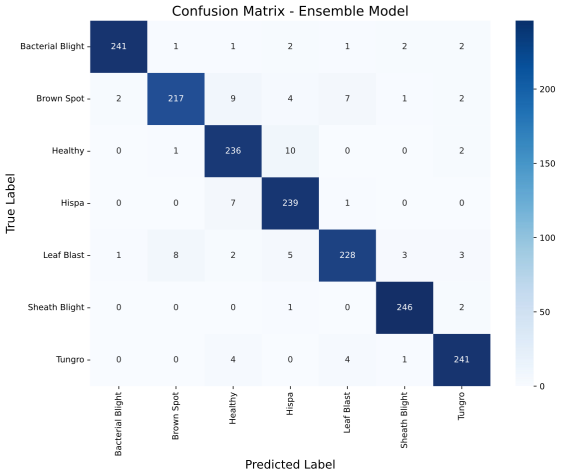
(a) Confusion matrix of EfficientNet-B0



(b) Confusion matrix of DenseNet-121



(c) Confusion matrix of MobileNetV2



(d) Confusion matrix of Ensemble Model

Figure 4: Confusion matrices of the individual models and the ensemble model on the test set: (a) EfficientNet-B0, (b) DenseNet-121, (c) MobileNetV2, and (d) Ensemble Model. The ensemble demonstrates fewer misclassifications.

hibited a slower increase in training accuracy and a higher loss, especially during the initial epochs.

Figure 5c and Figure 5d present the validation accuracy and loss for each model, illustrating their ability to generalize to unseen data. EfficientNet-B0 again achieved the strongest results, maintaining the highest validation accuracy and the lowest validation loss. This consis-

tency across both training and validation phases confirms its robustness and ability to capture complex patterns without overfitting. DenseNet-121 displayed a similar trend, attaining commendable validation accuracy and a gradual decrease in validation loss. These results demonstrate its capacity to generalize effectively to new data while maintaining stable learning dynamics. In contrast, MobileNetV2 performed well but lagged behind the other models in validation accuracy and showed a higher validation loss, mirroring its training behavior. This outcome reinforces that MobileNetV2's design prioritizes computational efficiency over representational depth, making it suitable for lightweight deployment but less optimal for highly complex classification tasks without further fine-tuning or architectural adjustments.

Overall, the training and validation curves highlight the strengths and weaknesses of individual CNNs, confirming EfficientNet-B0 as the most reliable base model, DenseNet-121 as a stable alternative, and MobileNetV2 as a lightweight but less expressive option, further justifying the use of an ensemble strategy to achieve balanced and robust performance.

5 Discussion

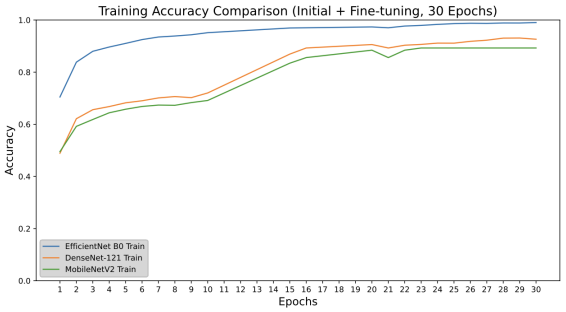
The experimental results demonstrate that CNNs are effective for rice disease detection, with EfficientNet-B0 and DenseNet-121 showing strong performance, and the ensemble model further improving overall robustness. In this section, we will discuss key observations, practical implications, and limitations of our work.

EfficientNet-B0 consistently achieved the highest accuracy among the single models 95%, reflecting its efficient scaling and ability to capture complex visual features even in relatively lightweight configurations. DenseNet-121 followed closely 92%, leveraging dense connections to promote feature reuse, which helped in capturing subtle lesion characteristics. MobileNetV2, while computationally efficient and suitable for mobile deployment, performed lower 87% due to its reduced representational capacity. This highlights the trade-off between computational efficiency and classification accuracy, which is critical when selecting models for field deployment.

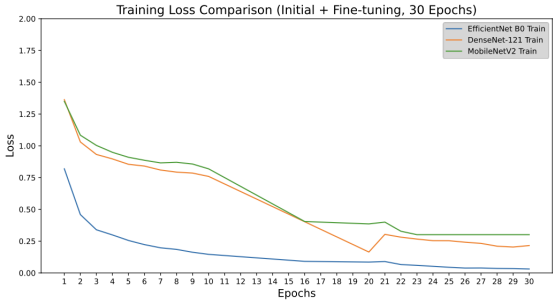
The ensemble model achieved the best overall accuracy of 96%, outperforming all individual CNNs. Its strength lies in combining the complementary decision boundaries of different models, which reduces the misclassification of visually similar diseases, such as Brown Spot vs. Leaf Blast and Healthy vs. Hispa. These confusions were evident in the confusion matrices of the individual models, but significantly reduced in the ensemble predictions.

Despite promising results, some limitations remain. First, our dataset was restricted to seven categories, while rice crops are affected by a broader range of diseases. Extending the dataset to include additional disease types would improve generalizability. Second, the dataset balancing was achieved through undersampling, which reduced diversity in some classes. It is important to note that this balancing process did not involve artificial data generation or oversampling. As described in Section 3.1 (Data Acquisition), we applied undersampling by reducing the number of images in overrepresented classes to achieve an equal distribution of 2,500 samples per class, consisting of approximately 1,700 field background and 800 white background images. This approach ensured that each category contributed equally to training without introducing synthetic samples. However, undersampling inherently reduces data diversity, which may affect model generalization. Future work will explore advanced data augmentation and class-weighted learning strategies to preserve diversity while maintaining balance. Additionally, further verification revealed signs of model overfitting, likely caused by the limited dataset size and the use of a single dataset split (70/20/10) without cross-validation or an external test set. Although ensemble learning was adopted to improve robustness and reduce the risk of overfitting, its effectiveness is still constrained when data diversity is limited. While several techniques, such as data augmentation, dropout, and early stopping were applied to reduce overfitting, these measures were insufficient to fully prevent it.

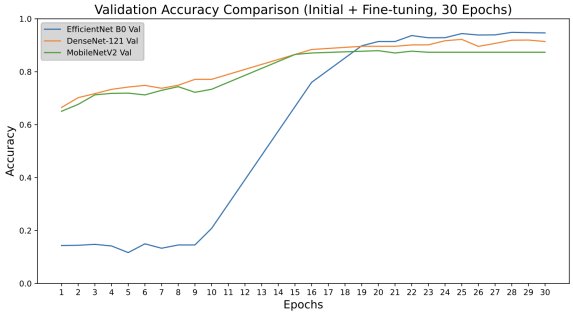
Future work will incorporate k-fold cross-validation, larger and more diverse datasets, and external validation to ensure better model generalization and reliability. Third, while the ensemble achieved strong results, its inference speed may be slower compared to lightweight single models, requiring further optimization for real-time deployment. Moreover, exploring optimization-based ensemble strategies, such as



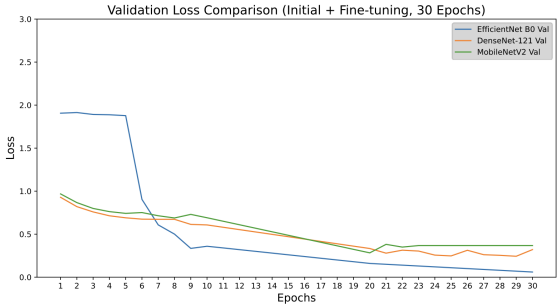
(a) Training accuracy curve.



(b) Training loss curve.



(c) Validation accuracy curve.



(d) Validation loss curve.

Figure 5: Comparison of the three models, EfficientNet-B0, DenseNet-121, and MobileNetV2 using training and validation metrics: (a) Training accuracy, (b) Training loss, (c) Validation accuracy, and (d) Validation loss.

the Ensemble Genetic Algorithm and Convolutional Neural Network (EGACNN) [19], may provide an effective way to automatically optimize model combinations and improve generalization. This represents a promising direction for integrating metaheuristic optimization with deep learning in agricultural image analysis.

6 Conclusion

This paper presented a deep learning framework for detecting rice diseases using convolutional neural networks. By integrating EfficientNet-B0, DenseNet-121, and MobileNetV2, the ensemble model achieved superior performance, with an overall accuracy and F1-score of 96%. Compared to individual CNNs, the ensemble approach reduced misclassifications between visually similar diseases, demonstrating greater robustness and reliability for field use.

Beyond accuracy, this work emphasizes the development and evaluation of an effective deep learning–based ensemble model for rice disease classification. The findings highlight the potential of combining multiple CNN architectures to enhance performance and provide a foundation for future research in precision agriculture.

However, some limitations remain. The dataset used in this study was limited to seven categories and balanced through undersampling, which reduced sample diversity. Future research should expand to a broader range of rice diseases, adopt advanced augmentation or generative techniques for data balancing, and explore optimization-based ensemble methods such as the Ensemble Genetic Algorithm and Convolutional Neural Network (EGACNN) to further improve model generalization and robustness.

In conclusion, the proposed ensemble approach represents a promising step toward practical, AI-driven solutions for precision agriculture. With further development and refinement, such models have the potential to support more accurate and reliable plant disease management, ultimately contributing to sustainable agricultural practices and global food security.

Acknowledgment

The authors would like to sincerely thank the students Chiep Cheaminh, Sang Haksou, Van Hoklin, Eng Sovansoupor, Pen Povrajana, and Touch Livita for their valuable assistance in data collection and model development. The authors

also gratefully acknowledge the support provided by Cambodia Academy of Digital Technology (CADT), whose resources and encouragement made this work possible.

References

[1] J.-J. Dethier and A. Effenberger, "Agriculture and development: A brief review of the literature," *Economic Systems*, vol. 36, no. 2, pp. 175–205, 2012.

[2] P. A. Seck, A. Diagne, S. Mohanty, and M. C. S. Wopereis, "Crops that feed the world 7: Rice," *Food Security*, vol. 4, no. 1, pp. 7–24, 2012.

[3] S. Savary, A. Ficke, J.-N. Aubertot, and C. Hollier, "Crop losses due to diseases and their implications for global food production losses and food security," *Food Security*, vol. 4, no. 4, pp. 519–537, 2012.

[4] P. K. Sethy, N. K. Barpanda, A. K. Rath, and S. K. Behera, "Image processing techniques for diagnosing rice plant disease: a survey," *Procedia Computer Science*, vol. 167, pp. 516–530, 2020.

[5] L. C. Ngugi, M. Abelwahab, and M. Abo-Zahhad, "Recent advances in image processing techniques for automated leaf pest and disease recognition–A review," *Information Processing in Agriculture*, vol. 8, no. 1, pp. 27–51, 2021.

[6] R. Deng *et al.*, "Automatic diagnosis of rice diseases using deep learning," *Frontiers in Plant Science*, vol. 12, p. 701038, 2021.

[7] L. Feng *et al.*, "Alfalfa yield prediction using UAV-based hyperspectral imagery and ensemble learning," *Remote Sensing*, vol. 12, no. 12, p. 2028, 2020.

[8] R. Deng *et al.*, "Deep learning-based automatic detection of productive tillers in rice," *Computers and Electronics in Agriculture*, vol. 177, p. 105703, 2020.

[9] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Computers and Electronics in Agriculture*, vol. 153, pp. 46–53, 2018.

[10] Loki4514, "Rice Leaf Diseases Detection," *Kaggle*, 2021. [Online]. Available: https://www.kaggle.com/datasets/loki4514/rice-leaf-diseases-detection

[11] Mtech MIT, "Paddy leaf disease detection Dataset," *Roboflow Universe*, Jul. 2023. [Online]. Available: https://universe.roboflow.com/mtech-mit/paddy-leaf-disease-detection

[12] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: overview study and experimental results," in *Proc. 11th Int. Conf. on Information and Communication Systems (ICICS)*, 2020, pp. 243–248.

[13] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer Nature, 2022.

[14] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2019, pp. 6105–6114.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 4700–4708, 2017.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[17] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proc. 21st Int. Conf. on Machine Learning (ICML)*, 2004, p. 18.

[18] J. Chen, J. Chen, D. Zhang, Y. Sun, and Y. A. Nanehkaran, "Using deep transfer learning for image-based plant disease identification," *Computers and Electronics in Agriculture*, vol. 173, p. 105393, 2020.

[19] Hussain, Wajahat and Mushtaq, Muhammad Faheem and Shahroz, Mobeen and Akram, Urooj and Ghith, Ehab Seif and Tlija, Mehdi and Kim, Tai-hoon and Ashraf, Imran, "Ensemble genetic and CNN model-based image classification by enhancing hyperparameter tuning," *Nature Publishing Group UK London*, vol. 15, p. 1003, 2025.

A Preliminary Study on Khmer Sign Language Recognition Using Neural Networks

Ponleur Veng **Nab Mat** **Kimhuoy Yann** **Vichhika Sina**
Sokleap Som **Rottana Ly**
Cambodia Academy of Digital Technology, Phnom Penh, Cambodia
ponleur.veng@cadt.edu.kh

Abstract

People with hearing impairments often face challenges in both social interactions and their language development, especially when they communicate with individuals who have little or no knowledge about sign language. Although sign language research has rapidly progressed we still need to working on for different linguistic across countries. This paper provides the preliminary study of Khmer Sign Language (KSL) recognition with different neural networks approaches. We also compare the performance where we train with RGB video frame and pose keypoint extraction to find accuracy yet computing efficient approach. We collected a dataset consisting of 6 deaf participants, featuring 20 signs across 580 videos, with each video recorded at a frame rate of 30 frames per second. The dataset was collected using digital cameras and smartphones in different environments to ensure the model's robustness across different devices and conditions. We trained several architectures and show that SlowFast achieved the best performance with 92.81% accuracy, 92.50% F1, 92.81% recall, and 93.57% precision. The keypoint pipeline with R3D-18 also performed competitively (92.05% accuracy, 92.56% F1, 92.05% recall, 95.21% precision), suggesting a promising trade-off for scenarios with tighter computed budgets. For this research has shown that SlowFast with RGB frame provide higher accuracy while pose keypoint show the better scalability for training. For the future work will expand the dataset and class number improve accuracy and generalizability, especially in real-world scenarios.

Keywords: *Sign Language Recognition, Transformer Model, Video Vision Transformer Model, SlowFast, R3D-18, LSTM, Bi-LSTM, GRU, Velocity features, Low-resource dataset*

1 Introduction

Sign language serves as an essential means of communication for people who are deaf or hard of hearing, enabling interaction and social integration both within their communities and with hearing individuals who understand sign language. Globally, it is estimated that more than 70 million people experience deafness, with more than 80% residing in developing countries, where access to resources and support is often limited [1, 2]. In 2020, approximately 1,700 newborns were born deaf, further highlighting the global prevalence of hearing loss [3]. In Cambodia, of a total population of 16 million people, there are 1.5 million deaf and hearing impaired. Approximately 3.5% of 1.5 million deaf people are profoundly deaf [4]. These figures highlight the critical importance of research aimed at improving communication and accessibility for deaf and hard-of-hearing individuals worldwide.

However, in Cambodia, support for the deaf for accessibility and communication in education and employment is limited. Although Krousar Thmey provides education for hearing-impaired children and the Maryknoll Deaf Development Program offers training for adults, most deaf individuals lack access to essential services [5]. In addition, another main challenge is when students start to learn new words at school and their family members and relatives cannot understand that new sign language vocabulary. In 2019, children with disabilities were three times less likely to attend school than other children [6], highlighting the challenges facing the deaf community and the urgent need for technologies to improve communication and a better education approach.

We aim to contribute to the community by providing (i) better communication and support for self-learning, we propose sign language

Table 1. Dataset Overview

Attribute	Value
Number of Videos	580
Participants	6 deaf participants
Location	NISE
Number of Signs	20
Frame Rate (fps)	30
Resolution (pixels)	1920×1080
Recording Devices	Digital cameras and smartphones

recognition by providing accurate recognition, which students can use to acquire new sign language independently to help this community with a better education experience and social integration, and (ii) compare different techniques for effective training and results by choosing two different approaches. Model training with RGB frame video with several architectures such as the SlowFast network [7], Channel Separated Convolutional Networks (CSN) [8], ViViT [9], and another approach pose keypoint extraction with mediapipe with velocity techniques and then apply the deep learning model such as LSTM [10], Bi-LSTM [11], and GRU [12] for better comparison of effective training applied to the larger dataset.

We collected our own data set that starts with a small data set from 6 deaf participants consisting of 20 signs with 580 videos, captured in different environments and lighting conditions to reflect real-world scenarios.

The paper is structured as follows. Section 2 provides a review of related work in sign language recognition, Section 3 outlines the methodology used for KSL recognition, Section 4 presents the experimental setup, Section 5 presents the results, and Section 6 concludes with a discussion of the findings and potential future directions for research.

2 Related Work

Sign language recognition has gained significant attention due to its potential to bridge the communication gap for deaf and hard-of-hearing individuals. The Word-Level American Sign Language Dataset (WLASL), introduced by Li et al. [13], stands as the largest word-level ASL dataset, featuring over 21,000 video samples across 2,000 glosses performed by 119 signers. This dataset's inclusion of significant inter-signer variability and annotations for dialectal variations has positioned it as a benchmark for large-scale word-level sign recognition.

In recent years, advancements in machine learning and deep learning, particularly with transformer-based architectures like ViViT [9] and spatiotemporal models like the SlowFast Network, have enabled significant progress in the computer vision field [7, 14, 15].

The SlowFast network has emerged as a powerful architecture for advancing sign language recognition tasks. Ahn [16] leveraged a two-pathway SlowFast network for Continuous Sign Language Recognition (CSLR), introducing Bi-directional Feature Fusion (BFF) and Pathway Feature Enhancement (PFE) to achieve state-of-the-art performance on datasets such as PHOENIX14 [17] and CSL-Daily [18]. Hassan [19] extended the application of SlowFast Networks to dynamic sign language recognition on the WLASL dataset and achieved a 79.34% in top-1 accuracy. Similarly, Radhakrishnan [20] demonstrated the effectiveness of the SlowFast model in word-level sign language detection on the MSASL dataset [21], achieving a 92.35% increase in top-1 accuracy. These studies highlight the versatility and robustness of SlowFast architectures across diverse sign language recognition tasks.

In 2018, Tran et al. proposed another approach for video classification called Channel-Separated Convolutionals (CSN) [8]. A new method was presented to add factorizing on the channel, while factorizing simply on the spatial and temporal dimensions. This approach was evaluated on Kinetics-400 dataset [22], achieving 76.6% in top-1 accuracy without any pre-trained dataset applied. By applying pre-trained dataset with Sport1M [23] and evaluate on the same data, the CSN model increased the top-1

accuracy by 1.8%.

In 2020, Camgoz et al. [24] proposed a transformer-based model joint end-to-end sign language recognition and translation, using Connectionist Temporal Classification (CTC) loss. Their approach achieved state-of-the-art results on the RWTH-PHOENIX-Weather-2014T dataset [25], improving both recognition and translation accuracy, with a significant boost in BLEU scores for translation tasks. In the same year, De Coster et al. [26] proposed a method for Sign Language Recognition using Transformer Networks. They combine OpenPose-based feature extraction with end-to-end feature learning using CNNs for sign language recognition [27]. Applying the multi-head attention mechanism [28] from transformers, they recognize isolated signs in the Flemish Sign Language corpus, achieving 74.7% accuracy on a 100-class vocabulary, significantly outperforming previous methods. Three years later, Kothadiya et al. [29] proposed SIGNFORMER, a Vision Transformer model for static Indian sign language recognition. It divides signs into positional embedding patches processed by a transformer with self-attention layers. The model achieved 99.29% accuracy with minimal training epochs, outperforming convolution-based architectures and showing effectiveness under various augmentations.

Keypoint-based sign language recognition has been developing by utilizing 2D/3D landmarks (hand, face, and body) that emphasize motion dynamics and are essential to the recognition task. For Indian SLR, Subramanian et al. [30] presented a MediaPipe-optimized GRU (MOPGRU) in 2022. After being normalized and denoised, holistic landmarks are modeled using a modified GRU cell whose update gate is conditioned by the reset gate. This results in significant gains over the video corpus's vanilla LSTM/GRU baselines. This study has demonstrated that recurrent design on posture sequence can compete with more complex vision backbones while maintaining real-time compatibility.

Building on similar pose-first pipelines, Bhadouria et al. present an LSTM-based SLR system that ingests MediaPipe landmarks extracted from videos and reports strong recognition with a compact temporal model—underscoring that recurrent architectures remain competitive when features are cleanly factorized into trajectories of keypoints [31].

3 Methodology

3.1 Dataset Collection

To work on the Khmer Sign Language Recognition task, a small corpus was developed for the training and evaluation. The dataset consists of 580 videos recorded by 6 deaf participants at the National Institute for Special Education (NISE). The signs represent everyday concepts and actions, including directional terms, locations, and objects commonly encountered in daily life. For better organization, the terms can be grouped into categories and listed as follows. Under Locations and Actions, we begin with "Where", "When", "Market", "Buy", and "Location". In the Directions category, we list "Left", "Right", "North", and "South". Next, under Objects, we have "Pen", "Blue Pen", "Red Pen", "Pencil", "Book", "Line", and "Eraser". Lastly, in the People category, we list "Teacher", "Director", "Female Director", and "Deputy Director". This structure emphasizes the most important terms first, with clear grouping and Khmer translations in parentheses for better understanding.

The data was recorded at 30 frames per second with a resolution of 1080 x 1920 pixels, using both cameras and smartphones across various environments and lighting conditions. This diverse dataset is well-suited for training and evaluating sign language recognition models in real-world scenarios. A sample of the proposed dataset is shown Table 1.

3.2 RGB video frame with Neural Network Approaches

Figure 2 illustrates the Khmer Sign Language Recognition System workflow from video data using the Sign Recognition models (CSN, SlowFast and ViViT). The process begins with the input of raw video data, which undergoes preprocessing steps to prepare it for model input. These steps include frame extraction, resizing, and normalization to ensure that the video frames are consistent in size and values before they are passed to the model. Once the preprocessing was done, it fed processed data into the recognition models. The extracted features are then used for Classification, where the model assigns a specific label (sign) based on the learned

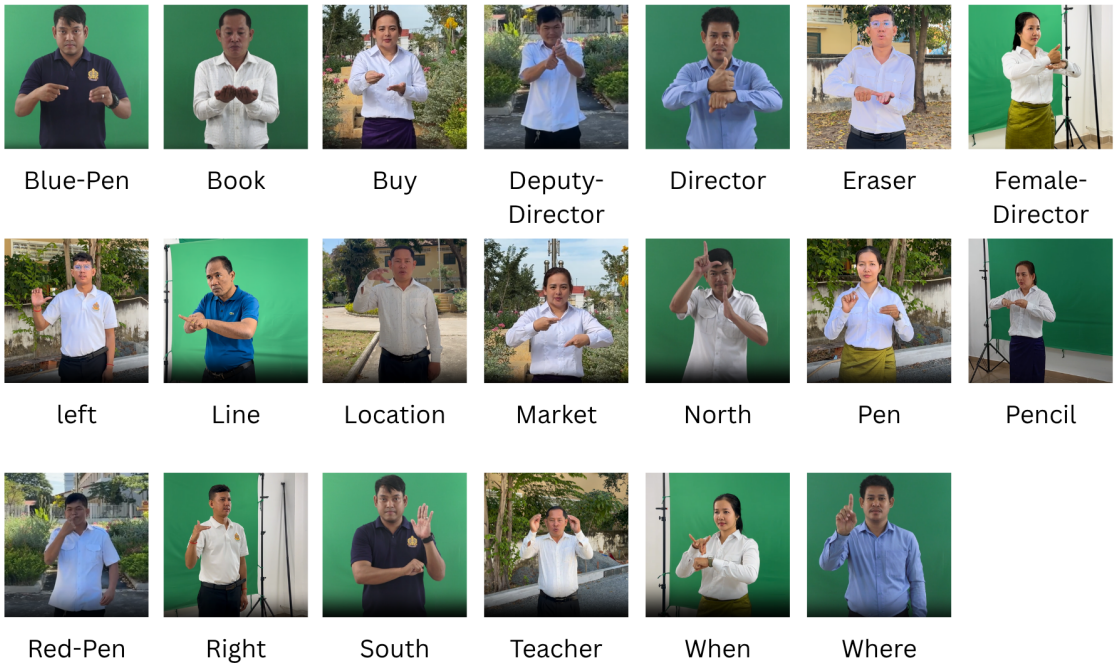


Figure 1. Sample Video Khmer Sign Language Dataset

patterns from the KSL dataset. This stage generates the prediction for the input video, which is the recognized sign. Finally, the output from the classification stage is presented as the recognized sign, corresponding to the KSL sign, providing valuable feedback for the user. For KSL recognition tasks, 4 models were selected for the experiment: Video Vision Transformers (ViViT), SlowFast Network, and Channel-Separated Convolutional Networks (CSN), and R3D-18.

3.2.1 Video Vision Transformers (ViViT)

This model proposed by Arnab et al. [9], apply transformer architectures to video data by dividing video frames into spatial patches and learning long-term temporal dependencies. This method surpasses convolutional approaches in datasets like Kinetics-400 due to its attention-based mechanism. ViViT's ability to model complex spatial-temporal interactions makes it suitable for capturing the nuances of sign gestures.

3.2.2 The SlowFast network

Introduced by Feichtenhofer et al. [7], processes video inputs at two different frame rates—one fast pathway for motion-sensitive features and one slow pathway for fine-grained spatial details. This dual-pathway architecture excels in capturing both dynamic and static elements of actions, making it particularly effective for sign language gestures that combine subtle hand movements with facial expressions. Recent work demonstrates its strong performance on datasets such as Charades and AVA, highlighting its adaptability to video-based tasks.

3.2.3 Channel-Separated Convolutional Networks (CSN)

This neural network has proposed by Tran et al. [8] which designed for tasks like video analysis. In these networks, the layers that process information are split into two types: one type focuses on mixing information across different channels (like colors or features) without looking at the surrounding space, and the other type focuses on analyzing patterns in the local space (like shapes or movements) without mixing channel information. This separation makes the network more efficient and specialized. Traditional networks combine both tasks in one step, but CSNs handle them separately. This idea has been proposed in Xception [32] and MobileNet [33] for image classification and R(2+1)D [34] video classification, but CSNs specifically aim to separate channel-related processing from spatial and temporal processing (movement-related).

3.2.4 ResNet 3D (R3D)

3D Residual Networks (R3D), proposed by Hara et al. [35] are another version of the 2D ResNet design [36], are CNNs for video that replace 2D spatial operations with 3D spatio-temporal kernels (time × height × width). In R3D, each residual block jointly models appearance and motion in a single 3D convolution while preserving skip connections, allowing deeper networks to train stably. This lets the model learn direct spatio-temporal features from short clips rather than single frames, which is effective for action and sign-language recognition. Unlike factorized designs such as R(2+1)D [37], which split spatial and temporal processing into separate steps, R3D performs them together in one operation—yielding a strong, simple baseline. A common lightweight instance is R3D-18, widely used as a backbone in this research.

3.3 Pose Key Point with Deep Learning Approach

3.3.1 Training Process

Similarly, from the previous training to training with the keypoint process with deep learning models, we start from the video frame. There are some slight differences in the preprocessing dataset step: we do all frame extraction from the video and the sampling from thoes frames. Then we will apply the keypoint extraction with mediapipe [38]. In order to capture the movement of the keypoint for each word, we apply the velocity technique [39]. The next step is the training process with deep learning models (LSTM, Bi-LSTM, and GRU) and end with the evaluation method.

3.3.2 Long Short-Term Memory (LSTM)

The model achitecture was introduced by Hochreiter and Schmidhuber in 1997 [10], the canonical origin of LSTM. It is a gated RNN that combats vanishing/exploding gradients by adding a persistent cell state and three gates (input, forget, and output). With this can help the model carry information over long time spans—crucial for signs whose meaning depends on motion over many frames.

3.3.3 Bidirectional Long short-Term Memory (Bi-LSTM

A bidirectional RNN [11] processes the sequence in forward and backward time and concatenates the two hidden states, giving access to past and future context at each frame. The principle was also introduced by Schuster and Paliwal in 1997; swapping the vanilla recurrent cell for LSTM yields Bi-LSTM, now standard for sequence labeling and SLR.

3.3.4 Gated Recurrent Unit (GRU)

This model is a lighter gated RNN [12] that merges the input/forget logic into update and reset gates—fewer parameters than LSTM, often similar accuracy. It was introduced in the RNN Encoder–Decoder work by Cho et al. (2014) and Chung et al. (2014) and was competitive with the LSTM model as well.

3.4 Evaluation Matrice

After the classification task to show the accuracy of each classification model, this task should be evaluated with evaluation matrices. In this paper, we choose the evaluation suitable for classification tasks such as , Precision, Recall, and F1-Score, Accuracy [40].

Let TP, FP, TN, and FN denote true positives, false positives, true negatives, and false negatives, respectively.

3.4.1 Precision

The precision is the ratio $TP/(TP + FP)$ where TP is the number of true positives and FP the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{1}$$

3.4.2 Recall

The recall is the ratio $TP/(TP+FN))$ where TP is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2}$$

3.4.3 F1-Score

The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.

$$F1 == \frac{2TP}{2TP + FP + FN}. \tag{3}$$

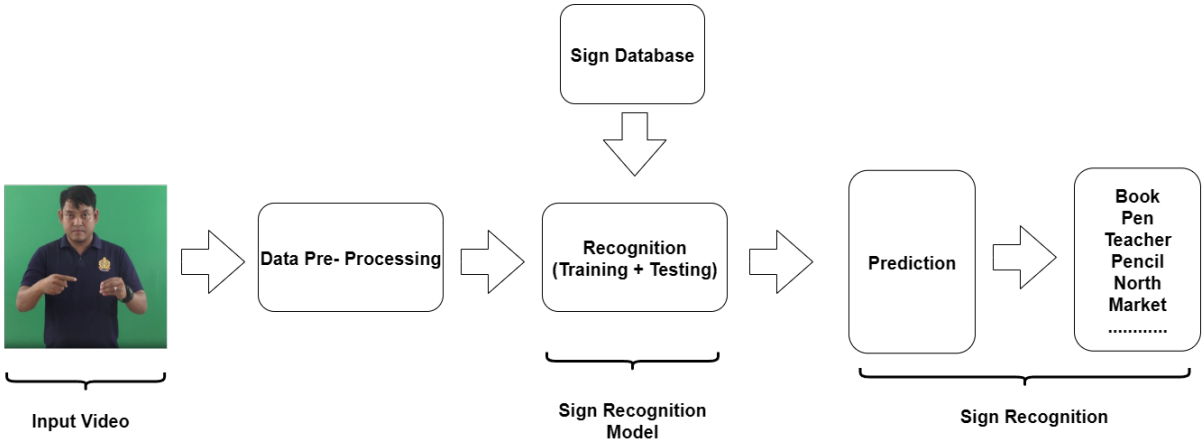


Figure 2. Flow Chart Frame Video Training Approach

3.4.4 Accuracy

In multilabel classification, accuracy function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

$$\text{Subset Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(\hat{\mathbf{y}}_i = \mathbf{y}_i)\,, \quad (4)$$

where $\hat{\mathbf{y}}_i$ and $\mathbf{y}_i$ are the predicted and true *label sets* for sample i, and $\mathbb{1}(\cdot)$ is the indicator function.

4 Experimental Setup

4.1 Data Pre-processing and Augmentation

We resize the resolution of all original video frames such that the diagonal size is 224 pixels for ViViT and 256 pixels for SlowFast, R3D-18, and CSN. For SlowFast training, R3D-18, and CSN, we center crop and apply normalization with the mean value of [0.45, 0.45, 0.45] and standard derivation value of [0.225, 0.225, 0.225]. For ViViT training, we randomly crop 224x224 pixels and apply normalization and random horizontal flipping. Note that, we randomly selected 30 frames.

4.2 Implementations Detail

All the experiments are implemented in Pytorch and pre-trained weight with Kinetics-400 datasets [22] are used for all models. We train all the models with Adam Optimizer [41]. Cross-Entropy loss function [42] is used in our experiments. All the models are trained with 50 epochs on each subset.

We split the sample of our dataset into the training and testing following the ratio of 70% and 30% of the total sample, respectively. To make sure the dataset is generalize for both training and testing set by manually for each class.

5 Result and Discussion

Table 2 indicates the performance of the two different approaches working on KSL 20 classes of the dataset: (i) RGB frame video architecture (SlowFast, CSN, ViViT, R3D-18), (ii) the pose keypoint with the sequence model (LSTM, Bi-LSTM, GRU, ResNet-3D-18). The evaluated methods are precision, recall, F1-score, and accuracy.

On the RGB side, SlowFast emerged as the top performer, achieving an accuracy of 92.81% and an F1-score of 92.50%. Its success can be credited to its unique dual-pathway architecture, which processes both fast and slow temporal features simultaneously. This design allows it to capture fine-grained motion details while also understanding broader temporal patterns, making it particularly well-suited for the complexities of sign language recognition. CSN came in second, with an accuracy of 82.87% and an F1-score of 81.20%. While its channel-separated convolution approach effectively reduces redundancy and improves feature extraction, it falls short of SlowFast's performance because it lacks the explicit temporal modeling capabilities that make SlowFast so effective. ViViT, a transformer-based model, achieved an accuracy of 75.13%, showing that while transformers are powerful for processing sequences,

Table 2. The results of KSL recognition using different neural network approaches.

Architecture	Precision	Recall	F1-Score	Accuracy
RGB Video–Based Models				
CSN	86.00%	82.87%	81.20%	82.87%
SlowFast	**93.57%**	**92.81%**	**92.50%**	**92.81%**
ViViT	82.68%	75.13%	75.31%	75.13%
R3D-18	88.35%	84.09%	83.72%	84.09%
Pose Keypoint–Based Models (and Fusion)				
Pose Keypoint + LSTM	90.00%	88.00%	87.00%	88.00%
Pose Keypoint + Bi-LSTM	87.00%	86.00%	86.00%	86.00%
Pose Keypoint + GRU	84.00%	80.00%	80.00%	81.00%
Pose Keypoint + R3D-18	**95.21%**	**92.05%**	**92.56%**	**92.05%**

they may not be as effective for recognizing fine-grained motion in sign language. This could be because transformers typically require large amounts of data to perform well and lack the built-in spatial and temporal processing advantages of CNN-based models.

For the pose-keypoint approach, models ingest normalized joint trajectories (with velocity) and focus on explicit motion cues while being robust to background and lighting. LSTM out perform the accuracy result amount of RNNs with an accuracy of 88% and 87% of F1 score. followed by Bi-LSTM (accuracy of 86% and 86% with F1 score) and GRU accuracy of 81% and 80% of F1 score. This result suggests that making motion explicit via velocity helps, and that a well-regularized Bi-LSTM can be competitive when clips provide sufficient context.

The best overall for the performance is pose keypoint with ResNet-3D-18, which come up with an accuracy of 92.05% and an F1 score of 92.56% althought it is slighly beaten SLowFast on F1 score, with the precision of 95.21% showing that it has the fewest false positives.

These results highlight that models explicitly designed for motion modeling, such as **SlowFast** with RGB video, outperform others by efficiently capturing both spatial and temporal features in sign language recognition tasks. However, pose keypoint with **ResNet-3D-18** also performs strongly by combining motion with limited appearance cues; in our summary (Table 3), this configuration attains the best F1 with the highest precision while keeping medium training and inference cost. In resource-constrained or cluttered backgrounds, the **Pose Keypoint + LSTM/Bi-LSTM/GRU** family is attractive: it models *motion only*, is *low/low* for training and inference compute, and is notably robust to background, though it may miss appearance-dependent distinctions (e.g., subtle hand texture or mouthing). **RGB + R3D-18** remains a solid, balanced baseline (*medium* train/infer) but lacks the dual-rate temporal pathway that gives **SlowFast** its edge. Finally, **CSN/ViViT** capture both motion and appearance but typically require medium/high compute and careful data/tuning; when such resources are available, they can be highly competitive. Overall, the choice hinges on data scale, hardware budget, and how much appearance information is needed; a pragmatic path is to start with keypoints (for efficiency and robustness) and scale to RGB+3D CNNs as resources permit.

6 Limitation

This experiment is still working on the small dataset with the limitation of the signer and variety of the environment. For the first experiment, we start working on the splitting, which follows the 70/30 train-test protocol rather than a signer-independent schema, which can lead to overestimating performance on an unseen signer.

We did not include a separate validation set or K-fold cross-validation, which could lead to suboptimal hyperparameters and early stopping. For the model we rely on the kinetic-400 pretraining and a single keypoint encoding (MediaPipe landmarks), and for evaluation we only apply accuracy, precision, recall, and F1 score, without per-class evaluation analysis and other robustness techniques.

7 Conclusion

To conclude, among all the experiments that we have conducted, the pose keypoint with

Table 3. Model families, compute, and behavior.

Approach	Captures	Train	Infer	Notes
RGB Video + SlowFast	Motion + appearance	High	High	Strong RGB-only
RGB Video + R3D-18	Motion + appearance	Med	Med	Solid baseline
CSN / ViViT	Motion + appearance	Med/High	Med/High	Needs data/tuning
Pose Keypoint + LSTM/Bi-LSTM/GRU	Motion only	Low	**Low**	Robust to background
Pose Keypoint + R3D-18	Motion + appearance	Med/High	**Med/High**	Best F1; highest precision

the ResNet-3D-18 model performed better for Khmer Sign Language recognition, with effective evidence, even if it was slightly beaten by the SlowFast model with RGB video. Additionally, when it comes to increasing the number of datasets, it will be beneficial for computing resource limitations. Our results demonstrate the effectiveness of the pose keypoint with the ResNet-3D-18 in sign language recognition, achieving an accuracy of 92.05%, an F1 score of 92.56%, a recall of 92.05%, and a precision of 95.21%. This work addresses the gap in KSL recognition and contributes to improving accessibility and a better learning approach for the deaf community in Cambodia. To further improve the system's accuracy and adaptability, especially in real-world situations, future research will investigate expanding the dataset, adding more sign modifications, and experimenting with different model architectures. For deaf people in Cambodia, the creation of a KSL recognition system will be very helpful, enabling improved communication and enhancing educational opportunities and inclusivity.

8 Acknowledgment

This research was initiated and supervised by Dr. Ye Kyaw Thu and was made possible through the collaboration between the National Institute for Special Education (NISE) and the Cambodia Academy of Digital Technology (CADT). We would like to thank the National Institute for Special Education (NISE) for providing essential resources and contributing to the data collection, as well as for manually verifying the dataset. We also extend our gratitude to the Cambodia Academy of Digital Technology (CADT) for offering the necessary support, including research materials and opportunities.

References

[1] A. Firth, *Deafness and Hard of Hearing*. Berkeley, CA: Apress, 2024, pp. 147–182. [Online]. Available: https://doi.org/10.1007/979-8-8688-0152-5_5

[2] J. S. Izquierdo-Condoy, L. E. Sánchez Abadiano, W. Sánchez, I. Rodríguez, K. De La Cruz Matías, C. Paz, and E. Ortiz-Prado, "Exploring healthcare barriers and satisfaction levels among deaf individuals in ecuador: A video-based survey approach," *Disability and Health Journal*, vol. 17, no. 3, p. 101622, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1936657424000530

[3] M. A. De Rosa, M. T. Bernardi, S. Kleppe, and K. Walz, "Hearing loss: Genetic testing, current advances and the situation in latin america," *Genes*, vol. 15, no. 2, 2024. [Online]. Available: https://www.mdpi.com/2073-4425/15/2/178

[4] A. Derrington, "The (Lack of) Deaf Culture in Cambodia — Ashley Derrington — ashleyderrington.com," https://www.ashleyderrington.com/blog/post-5#:~:text=Of%20those%2016%20million%2C%20roughly,deaf%20person%20in%20the%20world, [Accessed 29-01-2025].

[5] C. J. Waterworth, M. Marella, M. F. Bhutta, R. Dowell, K. Khim, and P. L. Annear, "Access to ear and hearing care services in cambodia: a qualitative enquiry into experiences of key informants," *The Journal of Laryngology 38; Otology*, vol. 138, no. 1, p. 22–32, 2024.

[6] B. Baghdasaryan, G. Ghawi, T. Godfrey-Faussett, and U. H. Castillo, "Paving the pathway: Inclusive education for children with disabilities in cambodia," *UNICEF Innocenti-Global Office of Research and*

Foresight,, 2024.

[7] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," 2019. [Online]. Available: https://arxiv.org/abs/1812.03982

[8] D. Tran, H. Wang, L. Torresani, and M. Feiszli, "Video classification with channel-separated convolutional networks," 2019. [Online]. Available: https://arxiv.org/abs/1904.02811

[9] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," 2021. [Online]. Available: https://arxiv.org/abs/2103.15691

[10] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks," 9 2019. [Online]. Available: https://arxiv.org/abs/1909.09586

[11] R. Mayya, V. Venkataraman, A. P. R, and N. Darapaneni, "A Novel Bi-LSTM And Transformer Architecture For Generating Tabla Music," 4 2024. [Online]. Available: https://arxiv.org/abs/2404.05765

[12] R. Dey and F. M. Salem, "Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks," 1 2017. [Online]. Available: https://arxiv.org/abs/1701.05923

[13] D. Li, C. R. Opazo, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," 2020. [Online]. Available: https://arxiv.org/abs/1910.11006

[14] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Comput. Surv.*, vol. 54, no. 10s, Sep. 2022. [Online]. Available: https://doi.org/10.1145/3505244

[15] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 87–110, 2023.

[16] J. Ahn, Y. Jang, and J. S. Chung, "Slowfast network for continuous sign language recognition," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 3920–3924.

[17] J. Forster, C. Schmidt, T. Hoyoux, O. Koller, U. Zelle, J. Piater, and H. Ney, "Rwth-phoenix-weather: A large vocabulary sign language recognition and translation corpus," 05 2012.

[18] Q. Zhu, J. Li, F. Yuan, J. Fan, and Q. Gan, "A chinese continuous sign language dataset based on complex environments," 2024. [Online]. Available: https://arxiv.org/abs/2409.11960

[19] A. Hassan, A. Elgabry, and E. Hemayed, "Enhanced dynamic sign language recognition using slowfast networks," in *2021 17th International Computer Engineering Conference (ICENCO)*, 2021, pp. 124–128.

[20] S. Radhakrishnan, N. C. Mohan, M. Varma, J. Varma, and S. N. Pai, "Cross transferring activity recognition to word level sign language detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022, pp. 2446–2453.

[21] H. R. V. Joze and O. Koller, "Ms-asl: A large-scale data set and benchmark for understanding american sign language," *arXiv preprint arXiv:1812.01053*, 2018.

[22] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," 2017. [Online]. Available: https://arxiv.org/abs/1705.06950

[23] B. Varadarajan, G. Toderici, S. Vijayanarasimhan, and A. Natsev, "Efficient large scale video classification," 2015. [Online]. Available: https://arxiv.org/abs/1505.06250

[24] N. C. Camgoz, O. Koller, S. Hadfield, and R. Bowden, "Sign language transformers: Joint end-to-end sign language recognition and translation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[25] N. C. Camgoz, S. Hadfield, O. Koller,

H. Ney, and R. Bowden, "Neural sign language translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7784–7793.

[26] M. De Coster, M. Van Herreweghe, and J. Dambre, "Sign language recognition with transformer networks," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, Eds. Marseille, France: European Language Resources Association, May 2020, pp. 6018–6024. [Online]. Available: https://aclanthology.org/2020.lrec-1.737/

[27] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," 2017. [Online]. Available: https://arxiv.org/abs/1611.08050

[28] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[29] D. R. Kothadiya, C. M. Bhatt, T. Saba, A. Rehman, and S. A. Bahaj, "Signformer: Deepvision transformer for sign language recognition," *IEEE Access*, vol. 11, pp. 4730–4739, 2023.

[30] B. Subramanian, B. Olimov, S. M. Naik, S. Kim, K.-H. Park, and J. Kim, "An integrated mediapipe-optimized GRU model for Indian sign language recognition," *Scientific Reports*, vol. 12, no. 1, 7 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-15998-7

[31] A. Bhadouria, P. Bindal, N. Khare, D. Singh, and A. Verma, "Lstm-based recognition of sign language," in *Proceedings of the 2024 International Conference on Contemporary Computing (IC3)*, ser. IC3 '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 508–514.

[32] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2017. [Online]. Available: https://arxiv.org/abs/1610.02357

[33] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[34] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," 2018. [Online]. Available: https://arxiv.org/abs/1711.11248

[35] K. Hara, H. Kataoka, and Y. Satoh, "Learning Spatio-Temporal Features with 3D Residual Networks for Action Recognition," 8 2017. [Online]. Available: https://arxiv.org/abs/1708.07632?utm_source=chatgpt.com

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 12 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[37] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A Closer Look at Spatiotemporal Convolutions for Action Recognition," 11 2017. [Online]. Available: https://arxiv.org/abs/1711.11248

[38] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A Framework for Building Perception Pipelines," 6 2019. [Online]. Available: https://arxiv.org/abs/1906.08172

[39] R. Messing, C. Pal, and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 104–111.

[40] scikit-learn developers, "Scikit-learn: Machine learning in python — performance metrics (precision_score, recall_score, accuracy_score, f1_score)," https://scikit-learn.org/stable/modules/model_evaluation.html, scikit-learn, accessed: 2025-10-01.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[42] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," 2023. [Online]. Available: https://arxiv.org/abs/2304.07288

Baseline Deep Learning Model for Khmer Sign Language Recognition with a Small Dataset

Sokleap Som **Rottana Ly** **Nab Mat**
Cambodia Academy of Digital Technology, Phnom Penh, Cambodia
sokleap.som@cadt.edu.kh

Abstract

People with hearing impairments in Cambodia often face challenges in daily communication and learning, especially when others cannot understand Khmer Sign Language. These difficulties can limit social interaction and access to education. To address this problem, this research proposes a Khmer Sign Language recognition approach using deep learning to support better communication and self-learning tools for the Deaf community. A dataset containing 100 sign classes was collected, with 20 fully annotated classes used for training, each including 28 to 31 samples recorded in varied environments at the National Institute for Special Educat ion (NISE). The data were processed into 30 fps and 1920 $\times$ 1080 resolution to ensure temporal smoothness and clear motion capture. Two models based on 3D ResNet (R3D-18) were trained and compared: one using raw RGB frames and another using keypoints extracted by MediaPipe. The RGB-based model achieved 88.35% precision, 84.09% recall, 83.72% F1-score, and 84.09% accuracy. The keypoint-based model achieved 95.21% precision, 92.05% recall, 92.56% F1-score, and 92.05% accuracy, showing that focusing on body and hand landmarks improves robustness and generalization on small datasets. This research provides a foundation for Khmer Sign Language recognition using limited data. Future work will expand the dataset, explore more classes, and improve inference for real-time applications while applying Early-Stopping to prevent overfitting.

Keywords: *Khmer Sign Language Recognition, Deep Learning, 3D ResNet-18, MediaPipe, Keypoint-based learning*

1 Introduction

The Khmer Sign Language is the main mode of communication for people who are deaf or hard of hearing in Cambodia. However, limited technological support makes it difficult for deaf people to interact with hearing people unfamiliar with sign language, leading to social isolation and reduced educational opportunities. Although organizations such as Krousar Thmey and the Deaf Development Programme have provided support, accessible digital tools remain scarce.

Recent advances in deep learning have led to significant progress in Sign Language Recognition (SLR), especially through 3D Convolutional Neural Networks (3D CNNs) capable of learning spatio-temporal features from video sequences [1], [2]. Advanced architectures such as SlowFast [3] and X3D [4] improve motion understanding, while Transformer-based models such as ViViT [5] and Sign Language Transformers [6] achieve strong results through attention mechanisms. However, these models are based on large-scale datasets, which are unavailable for underrepresented languages such as Khmer Sign Language [7], [8].

To address data scarcity, keypoint-based methods use pose estimation frameworks such as MediaPipe to extract skeletal landmarks that capture sign movement while filtering out background noise [9], [10]. Combining MediaPipe with CNN-LSTM or Transformer models has achieved promising results on small datasets [11], [12], [13]. Building on these insights, this research presents a baseline R3D-18 model trained on RGB and keypoint data to establish a foundational benchmark for Khmer Sign Language recognition.

This research contributes by creating a new Khmer Sign Language dataset that provides one of the first structured collections for this language. It also establishes a baseline deep learning model using both RGB and keypoint input, offering a reference for future studies. The results and methods can help future researchers build larger models and improve technologies that support the Deaf community in Cambodia.

This paper is organized as follows. Section 2 reviews existing research and related work on Sign Language Recognition and Keypoint-Based approaches. Section 3 presents the proposed methodology, including dataset collection, preprocessing, and model architecture of R3D-18. Section 4 describes the experimental setup and evaluation metrics used in this research. Section 5 discusses the experimental results and performance comparisons between the RGB and keypoint-based models. Finally, Section 6 concludes the article and outlines directions for future work.

2 Related Work

Research in sign language recognition has advanced rapidly with deep learning models that capture spatio-temporal patterns from video data. Early studies using 3D CNNs such as 3D ResNet and R(2+1)D demonstrated strong capabilities to model motion dynamics across frames [1], [2], [14]. Later architectures like SlowFast [3] and X3D [4] enhanced temporal efficiency and accuracy, becoming standard baselines for video classification tasks, including SLR.

Transformer-based models such as ViViT [5] and Sign Language Transformers [6] further improved recognition through long-range attention, but their performance depends on extensive labeled data. This limits their use in low-resource settings, where annotated datasets are scarce [7]–[15]. Consequently, baseline models trained on small datasets remain essential to establish feasible starting points for further research and deployment.

To overcome small-data constraints, keypoint-based SLR has emerged as an effective approach, where skeletal landmarks replace full RGB frames to emphasize motion cues and reduce noise. MediaPipe has gained popularity for its lightweight and real-time extraction of hand, face, and body landmarks [10]. Recent studies integrating MediaPipe with CNN-LSTM [12], Transformer [11], and hybrid networks [16], [13] show that models driven by landmarks generalize better and train faster with limited data, aligning with the goal of this research.

3 Methodology

This section describes three main components: data collection, data annotation, and model selection. First, the data collection process outlines how Khmer Sign Language videos were recorded and organized. The data annotation process explains how gestures were labeled and validated to ensure linguistic accuracy. Finally, the model selection subsection discusses the 3D ResNet-18 (R3D-18) [1] architecture used for RGB and keypoint-based inputs [17], [12], [10], including its configuration and training parameters.

3.1 Data Collection

A Khmer Sign Language dataset was a task that was cooperated between Cambodia Academy of Digital Technology (CADT) and the National Institute of Special Education (NISE) in Phnom Penh to record the data as videos for this research. The complete dataset contains 100 classes, but only 20 fully annotated classes were used to ensure consistency and reliable labels. The remaining 80 were excluded due to missing annotations, as unlabeled data could introduce ambiguity and hinder supervised learning.

The videos were recorded in 1920 $\times$ 1080 resolution as Full HD at 30 fps with the digital camera. Although sometimes NISE helps record videos using the smartphone which might have slightly different resolution but we used the tool that helps all videos resolution to be consistency. This resolution was chosen to ensure clear visualization of finger and hand movements while keeping a balance between detail and computational cost. A higher frame rate (30 fps) helps capture smooth temporal motion without excessive storage usage. Although both cameras and smartphones were used, quality differences were minimal after preprocessing since all videos were standardized in resolution and frame rate.

The 20 annotated signs span four categories: *Locations and Actions* (“Where”, “When”, “Market”, “Buy”, “Location”), *Directions* (“Left”, “Right”, “North”, “South”), *Objects* (“Pen”, “Blue Pen”, “Red Pen”, “Pencil”, “Book”, “Ruler”, “Eraser”), and *People* (“Teacher”, “Director”, “Female Director”, “Deputy Director”). Six deaf participants recorded the signs using cameras and smartphones at 1920 $\times$ 1080 resolution and 30 fps in varied environments. A sample of the dataset is illustrated in Figure 1, and a summary is shown in Table 1.

Figure 1. Sample of the proposed Khmer Sign Language dataset.

Table 1. Dataset Overview

Total Classes:	100
Classes Used:	20 Annotated
Videos Used:	591 Videos
Participants:	6 Deaf Participants
Location:	NISE
Frame Rate:	30 fps
Resolution:	1920 × 1080
Record Device:	Camera and Smartphone

3.2 Data Annotation

Data Annotation was conducted manually by trained experts, with additional validation steps to ensure the highest level of linguistic accuracy and consistency. Each video in the dataset was carefully reviewed and labeled according to its corresponding sign class, ensuring that it contained a single complete gesture without ambiguity or overlap. By including only verified and fully validated samples, we preserved the integrity and uniformity of the dataset, which is particularly crucial when evaluating the performance of the baseline model on small datasets. This meticulous process not only guarantees reliable annotations but also provides a solid foundation for reproducible research and meaningful comparisons across different computational models.

3.3 3D ResNet-18 (R3D-18)

The R3D-18 model is an extended version of ResNet-18 that replaces 2D convolutions with 3D ones, allowing it to learn spatial and temporal features of video frames [1]. This ability is important for Sign Language Recognition, where hand motion and movement over time carry meaning. Compared to larger models such as I3D [2], SlowFast [3], or Transformer-based networks [5], R3D-18 provides a good balance between accuracy and efficiency, making it suitable for small datasets.

With this lightweight model, it is powerful enough to learn motion features without overfitting the 591-video dataset around 27–31 samples per class. Its 3D filters capture motion patterns across 30-frame sequences while keeping the spatial structure clear. Using pre-trained R3D-18 weights from large video datasets also improves generalization through transfer learning.

For the keypoint-based version, the MediaPipe framework was used to extract 543 keypoints from the body, hands, and face, giving the model simplified inputs that are less sensitive to lighting or background changes. Previous studies have shown that such landmark-based features work better in small datasets and improve robustness.

Both the RGB and keypoint models were

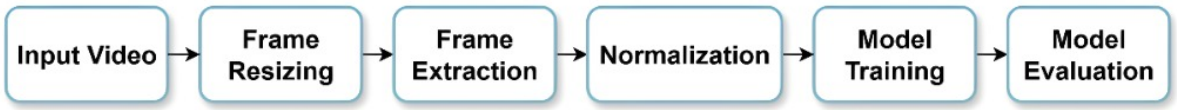


Figure 2. Preprocessing workflow for RGB input.

Figure 3. Preprocessing workflow for MediaPipe keypoint input.

trained with the same settings: Adam optimizer (learning rate $1e^{-4}$), batch size 16, and 5 epochs. Five epochs were chosen because overfitting appeared beyond that point, giving a good trade-off between accuracy and training time. The RGB model achieved 84.09% accuracy, while the keypoint model reached 92.05%, demonstrating that focusing on landmarks gives better results on small data. After applying early stopping and model checkpointing (Figure 6), the keypoint-based model improved further to 94.32% accuracy, 95.43% precision, and 94.28% F1-score.

During training, validation loss was monitored each epoch. If it did not improve for 5 epochs (*EARLY_STOPPING_PATIENCE* = 5), training was stopped automatically to prevent overfitting. Whenever the validation loss improved, the model weights (*state_dict*) were saved in *BEST_MODEL_PATH*. This process also works with *nn.DataParallel* models. Only the weights were saved, but the optimizer and epoch number can also be stored if training needs to be resumed later. The final performance after applying these methods is shown in Table 4.

4 Experimental Setup

4.1 Data Preprocessing

Each video was resized to 224 × 224 pixels and normalized to [0, 1]. Two modalities were explored: RGB, preserving the appearance features Figure 2; and MediaPipe, extracting 2D keypoint landmarks for the hands, body, and face Figure 3. The natural variation in the recording conditions provided sufficient diversity for model training.

4.2 Implementation and Evaluation Setup

The experiments were conducted in PyTorch During early development, models were trained for 5 epoch using cross-entropy loss, Adam Optimizer ($1e^{-4}$), and a batch size of 16. The dataset was split into 70%, 15%, and 15% for training, validation, and testing under a signer-dependent scheme. This ratio was chosen to maintain a balanced trade-off between learning and generalization, providing enough data for model fitting while keeping unseen samples for fair evaluation. The RGB inputs were normalized pixel frames, while the keypoint inputs were coordinate-normalized landmarks.

When applying early stopping and model checkpointing during extended 14 epoch experiments, training was conducted in Kaggle's GPU environment using *nn.DataParallel* for multi-GPU processing. Note that *nn.DataParallel* operates in a single-node, single-process configuration and may incur CPU-to-GPU scatter overhead. When scaling to multiple GPUs, it is advisable to proportionally increase the batch size and save both the optimizer state and the current epoch to allow resuming interrupted training. The effect of applying early-stopping and model checkpointing during 14-epoch runs is to ensure the robustness of the model, and whenever the validate does not improve, save the model and select the best model at the best epoch in Figure 6.

5 Results and Discussion

The results are summarized in Table 2. The RGB-based R3D-18 achieved an accuracy of 84.09%, while the keypoint-based model reached 92.05%. The keypoint model also demonstrated higher precision and F1-score, indicating stronger consistency between classes.

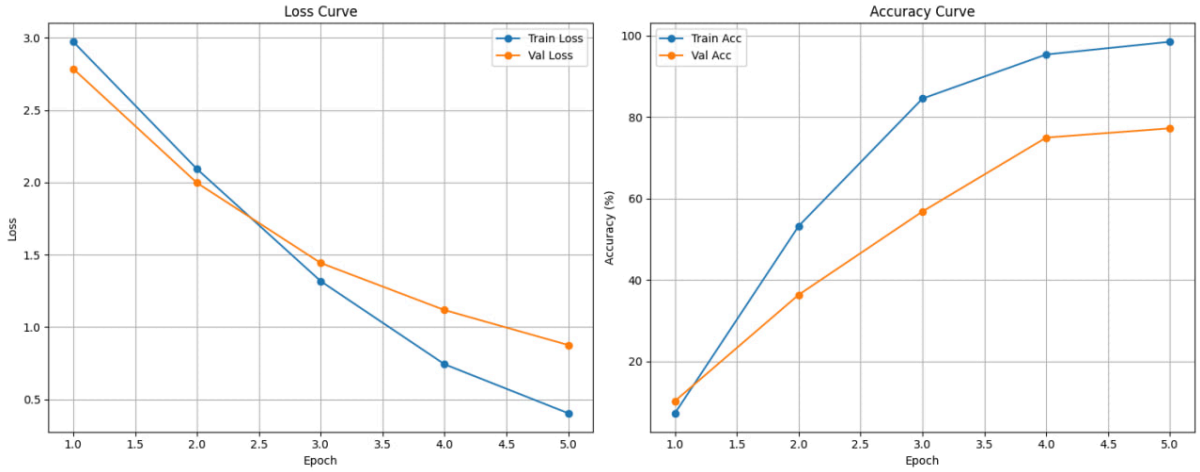
To further enhance performance, early stopping and model checkpointing were applied dur-

Table 2. Performance Comparison Between RGB and MediaPipe Models

Architecture	Precision	Recall	F1-Score	Accuracy
R3D-18 (RGB)	88.35%	84.09%	83.72%	84.09%
R3D-18 (Keypoint)	**95.21%**	**92.05%**	**92.56%**	**92.05%**

Table 3. Sample of Most Confused Class Pairs

True Label	Predicted Label	Count
Ruler	Eraser	2
Director	Female Director	2



(a) **(b)**

Figure 4. Training and validation curves for the Keypoint-based model. (**a**) Loss Curve: Consistent, steep reduction in both Training and Validation Loss, confirming successful convergence. (**b**) Accuracy Curve: Substantial increase in both Training and Validation Accuracy, demonstrating significant predictive improvement.

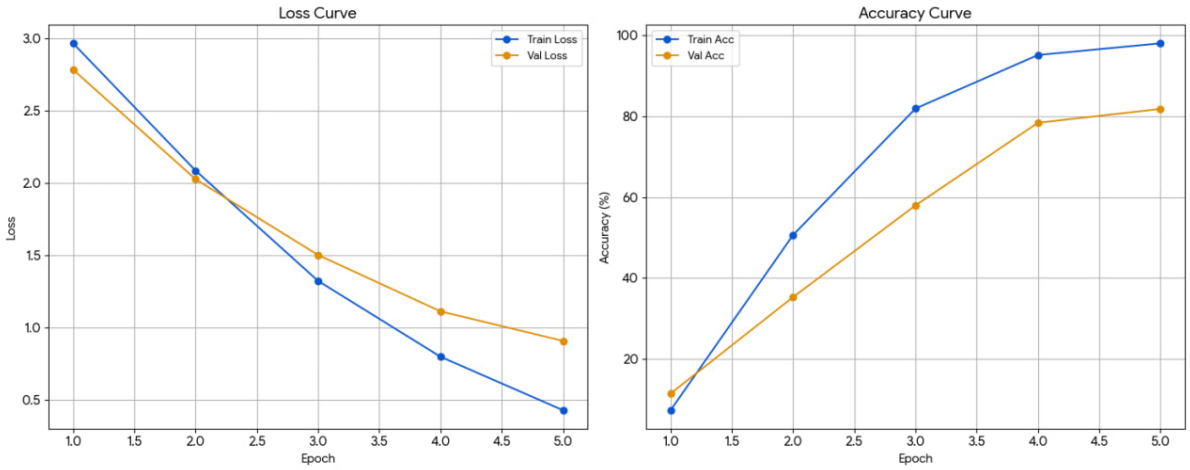
Table 4. Early-Stopping and Model Checkpointing Performance - 14 Epoch

Architecture	Precision	Recall	F1-Score	Accuracy
R3D-18 (Keypoint)	**95.43%**	**94.32%**	**94.28%**	**94.32%**

ing a 14-epoch training process. This approach helped prevent overfitting and automatically saved the best-performing model based on validation loss. As shown in Table 4, the keypoint-based model improved to 94.32% accuracy, with 95.43% precision and 94.28% F1-score. The training curves in Figure 6 also demonstrate smoother convergence and stable validation trends, confirming that early stopping effectively reduced unnecessary epochs while preserving generalization.

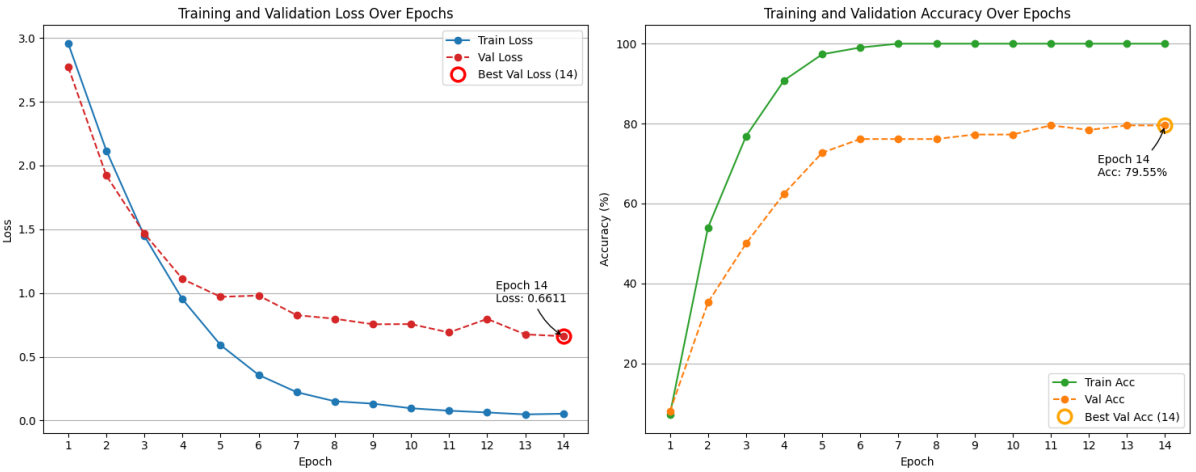
Some confusion remained, particularly between semantically and visually similar classes (Table 3). For example, "Director" and "Female Director" have nearly identical initial movements but differ subtly near the end of the gesture, making it difficult for the model to distinguish between them. Figure 9 provides a qualitative comparison showing similar hand trajectories that caused this confusion.

These findings confirm that, while RGB provides richer appearance information, it also introduces noise that hinders small-data learning. Keypoint-based features emphasize essen-



(a) **(b)**

Figure 5. Training and validation curves for the RGB-based model. (**a**) Loss Curve: Consistent, strong reduction in both Training and Validation Loss. (**b**) Accuracy Curve: Rapid, continuous increase in both Training and Validation Accuracy.



(a) **(b)**

Figure 6. Applied early stopping for the Keypoint-based model. (**a**) Loss Curve: Optimal checkpoint selected at Epoch 14 (Best Val Loss: 0.6611). (**b**) Accuracy Curve: Peak generalization reached at Epoch 14 (Best Val Acc: 79.55%).

tial motion cues and yield higher accuracy and generalization for Khmer Sign Language recognition.

6 Conclusion

To conclude, a baseline deep learning framework for Khmer Sign Language recognition was developed using a small annotated dataset. Two variants of the R3D-18 model were evaluated: RGB and MediaPipe keypoint-based. The keypoint model outperformed the RGB model, achieving an accuracy of 92.05%, as shown in Figures (4-7), while the RGB model reached an accuracy of 84.09% in Figures (5-8). This performance gap can be explained by the fact that the RGB model is more sensitive to lighting, background variations, and visual noise, whereas the keypoint model focuses on skeletal landmarks, capturing motion and structure more effectively under limited data conditions.

This study is limited by the small size dataset and the signer-dependent setup.

These results demonstrate that landmark-based representations enhance robustness and

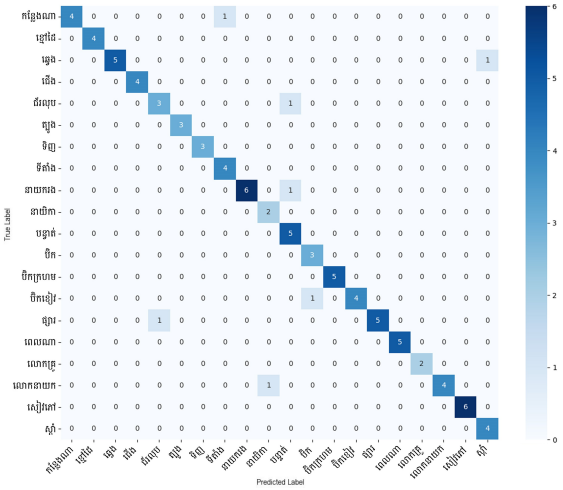


Figure 7. Confusion Matrix (Keypoint).

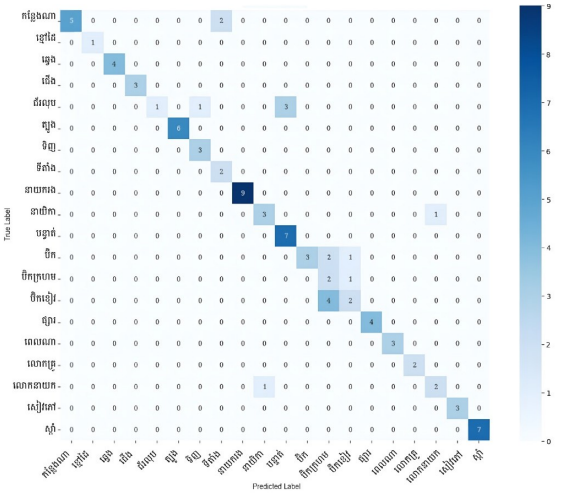


Figure 8. Confusion Matrix (RGB).

generalization for small datasets. Future work will add labels for all 100 classes, test the model with different signers, and use techniques like early stopping and saving model checkpoints with extensive testing to ensure robustness. We will also explore Transformer-based architectures and attention mechanisms over time to further improve accuracy. This research establishes a foundation for the recognition of Khmer Sign Language and contributes to the development of inclusive technologies that support communication, education, and accessibility for the Deaf community in Cambodia.

Acknowledgment

We sincerely thank the teachers and staff of the National Institute of Special Education (NISE), Cambodia, for their valuable support in the data collection process. Their assistance in recording and verifying Khmer Sign Language videos was essential to this research, and we also appreciate all participants for their time and contribution to the dataset.

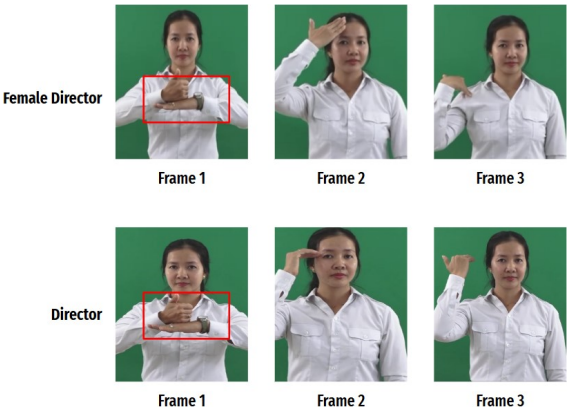


Figure 9. Similarity between "Male Director" and "Female Director" gestures, illustrating their visual resemblance.

References

[1] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6546–6555.

[2] J. a. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4724–4733.

[3] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6202–6211.

[4] C. Feichtenhofer, "X3d: Expanding architectures for efficient video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 203–213.

[5] A. Arnab, M. Dehghani, G. Heigold, C. S. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," in *Proceedings of the IEEE/CVF International Con-*

ference on Computer Vision (ICCV), 2021, pp. 6836–6846.

[6] N. C. Camgöz, O. Koller, S. Hadfield, and R. Bowden, "Sign language transformers: Joint end-to-end sign language recognition and translation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 023–10 033.

[7] N. Adaloglou, T. Chatzis, I. Papastratis, A. Stergioulas, G. T. Papadopoulos, V. Zacharopoulou, G. J. Xydopoulos, K. Atzakas, D. Papazachariou, and P. Daras, "A comprehensive study on deep learning-based methods for sign language recognition," *arXiv preprint arXiv:2007.12530*, 2020. [Online]. Available: https://arxiv.org/abs/2007.12530

[8] M. Madhiarasan and P. P. Roy, "A comprehensive review of sign language recognition," *arXiv preprint arXiv:2203.02387*, 2022.

[9] O. Özdemir, I. Baytaş, and L. Akarun, "Multi-cue temporal modeling for skeleton-based sign language recognition," *Frontiers in Neuroscience*, vol. 17, 2023.

[10] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7291–7299.

[11] H.-H. Li and C.-C. Hsieh, "Dynamic hand gesture recognition using mediapipe and transformer," *Engineering Proceedings*, vol. 8, no. 1, pp. 1–6, 2025.

[12] A. Fitriani, M. P. Utama, and D. T. Iswanto, "Dynamic sign language recognition using mediapipe library and modified lstm method," *International Journal of Advanced Science Engineering Information Technology (IJASEIT)*, vol. 13, no. 5, pp. 19 401–19 408, 2023.

[13] M. H. Uddin, M. S. Islam, and M. R. Islam, "A deep learning-based bangla sign language recognition system using mediapipe and cnn-lstm architecture," *Sensors*, vol. 23, no. 21, pp. 1–18, 2023.

[14] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6450–6459.

[15] N. Sarhan, S. Kammoun, and M. Hammami, "Unraveling a decade: A comprehensive survey on isolated sign language recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2023, pp. 2312–2321.

[16] M. H. Ali, N. M. Noor, and H. A. Jalab, "Next-gen dynamic hand gesture recognition: Mediapipe, inception-v3 and lstm-based enhanced deep learning model," *Electronics*, vol. 13, no. 16, 2024.

[17] C. Lugaresi, J. Tang, H. Nash, C. McClanahan *et al.*, "Mediapipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

Estimating Sand Volume Change and Analyzing Nearshore Dynamics with Airborne LiDAR Series Data for Coastal Monitoring: A Case study in Chanthaburi

Vichhika Sina[1] **Phutphalla Kong**[1] **Kasorn Galajit** [2] **Surasak Boonkla**[2] **Jessada Karnjana**[2]
[1]Cambodia Academy of Digital Technology, Phnom Penh, Cambodia
[2]National Electronics and Computer Technology Center, Pathum Thani, Thailand
vichhika.sina@student.cadt.edu.kh

Abstract

Coastal erosion is a global challenge that requires accurate and reliable monitoring of sediment dynamics. This study presents an effective approach for estimating sand volume changes and analyzing coastal dynamics using time-series airborne LiDAR data. Digital Elevation Models (DEMs) derived from LiDAR surveys were used to quantify elevation changes, and an offset compensation method was applied to reduce systematic measurement errors. Calibration using known reference box volumes confirmed that the offset-corrected results closely matched ground-truth measurements, improving volumetric accuracy. Seasonal analysis revealed distinct patterns of sediment transport along the coastline. From October 2024 to April 2025, all sections experienced notable sediment deposition, indicating beach accretion during the dry season. Conversely, from April to August 2025, significant erosion occurred likely due to intensified wave energy and monsoonal influences during the rainy season. The results demonstrate the reliability of airborne LiDAR, when combined with error correction and DEM differencing, as a powerful tool for high-resolution coastal monitoring and for supporting evidence-based shoreline management.

Keywords: *Coastal monitoring, Sand volume change, Airborne LiDAR, Coastal erosion, Nearshore analysis*

1 Introduction

Coastal erosion is a global challenge that threatens ecosystems, infrastructure, and human settlements in near-shore environments [1]. Effective coastal monitoring and management are therefore essential to understand the dynamics of the shoreline and mitigate associated risks [2], [3]. A critical component of such monitoring is the accurate estimation of the change in sand volume, since fluctuations in sediment distribution directly influence erosion and deposition processes [4]. Airborne light detection and ranging (LiDAR) has emerged as a powerful tool for topographic mapping in coastal regions for its ability to provide high resolution three-dimensional representations of the terrain [5], [6], [7].

LiDAR surveys provide valuable data for quantifying coastal morphological changes over time. Studies have shown that beach erosion and deposition patterns are spatially variable, with changes often concentrated at specific locations rather than uniformly distributed [8], [9]. For example, analysis of a 63 km stretch of Assateague Island revealed that the island ends experienced the most significant erosion and deposition [9]. LiDAR data can be used to calculate various coastal metrics, including shoreline change, volume change, and subaerial volume change [10]. These metrics allow for comprehensive monitoring of dune systems, as demonstrated in a study of Formby coast, which found significant sand erosion on the beach and frontal dunes, while inner dunes gained sand over a 21-year period [11]. The high-density data provided by LiDAR enables accurate assessment of shore changes at both small and large scales [8]. Adam P. Young et al. [12] uses airborne LiDAR to quantify volumetric change from seacliff and gully erosion in the Oceanside Littoral Cell, California. The results suggest that sea cliffs are the dominant source of beach-sand-sized sediment compared to gullies and rivers during this relatively dry 6-year period. Faik Ahmet Sesli et al. [13] conducted a study to determine coastline change as a basic element of Integrated Coastal Zone Management, reporting a significant coastal retreat in the study area over a two-year period. Recent work by Glenn M.

Suir et al. [14] focuses on combining LiDAR-derived elevation model(DEM) with vegetation metrics to assess hurricane-induced spatial and temporal changes in both terrain elevation and the coastal dune vegetation that stabilizes it. Recent regional studies have applied LiDAR and remote sensing for coastal monitoring in Southeast Asia. Phua et al. [15] used airborne LiDAR and UAV data in Malaysia to assess coastal topography and vegetation dynamics, while Latif and Yong [16] analyzed shoreline changes in Brunei Darussalam using spatial and geomorphological methods. However, while these studies successfully quantified shoreline change, they did not incorporate volumetric validation using known-reference structures.

Even though considerable research has been conducted on coastal change detection using LiDAR data, there remains a lack of methods that validate estimated volumetric changes against known-volume reference structures. In this work, we propose an approach to estimate sand volume change and analyze coastal dynamics over time using airborne LiDAR time-series data. A compensation method is introduced by using known box volumes as ground-truth references. This approach enhances confidence in the derived volume change results and supports more reliable coastal monitoring and management applications.

This study is part of ASEAN IVO framework that aims to generate scientific insights to support policy development and decision-making that aimed to mitigate the issue of coastal erosion.

2 Study area and Data acquisition

2.1 Study Area

The data was gathered along a 3.5-kilometer section of coastline in Chanthaburi province, Thailand. This area features bamboo fences installed to combat coastal erosion. The goal of these fences are to reduce wave energy and to trap sediment on the muddy beach, which receives sediment from both land and sea to create enough stable ground for mangrove forestation, a natural defense against further erosion.

This particular site was selected for its dynamic coastal characteristics and environmental significance. This region experiences continuous shoreline changes influenced by wave action, tidal processes, and sediment transport, making it suitable for analyzing coastal dynamics. This coastline represents a typical estuarine environment along the eastern Gulf of Thailand, where issues of erosion and sedimentation are evident. The area's manageable size, and accessibility also facilitate accurate data collection and analysis. Furthermore, the area's ecological and socioeconomic importance, particularly its aquaculture activities and mangrove ecosystems, highlights the need for a deeper understanding of coastal processes to support sustainable management and the mitigation of erosion impacts.

Figure 1. Study area located in Khlung, Chanthaburi Province, Thailand, showing the four divided coastal sections.

The study area is divided into four sections based on geomorphological characteristic and presence of bamboo fences. Section 1 represents the northernmost part of the coast, characterized by a gently sloping beach with dense mangrove vegetation and relatively stable shoreline conditions. Section 2 includes a moderately eroded area adjacent to small coastal developments, where partial bamboo fencing structures are present to reduce wave energy. Section 3 is located near a more exposed shoreline with less vegetation coverage and exhibits more

noticeable sediment redistribution, indicating active erosion and deposition processes. Section 4 lies at the southern end of the study site, where the beach is protected by continuous bamboo fences and experiences dynamic sediment accumulation near the structures, suggesting their influence on local sediment trapping and shoreline stabilization.

2.2 Data Acquisition

The data in this study were collected using UAV LiDAR, a laser-based remote sensing technique that is widely used as one of the most advanced technologies currently available. Conceptually, LiDAR operates in a way similar to RADAR or SONAR, but instead of radio or sound waves, it relies on laser light. The system transmits rapid pulses of laser beams toward the Earth's surface, where they are refracted back to the sensor. With ground sample distance about 2 cm and high point density, the survey provided a highly detailed representation of the terrain. Positioning was supported by a GNSS base station and corrected using the National CORS Data Center, while an inertial measurement unit (IMU) helped determine sensor orientation [17]. The dataset is stored as a 3D point cloud as shown in Figure 2. However, as with most LiDAR systems, water surfaces remain a challenge due to their low reflectivity, which reduces the number of points recorded [18].

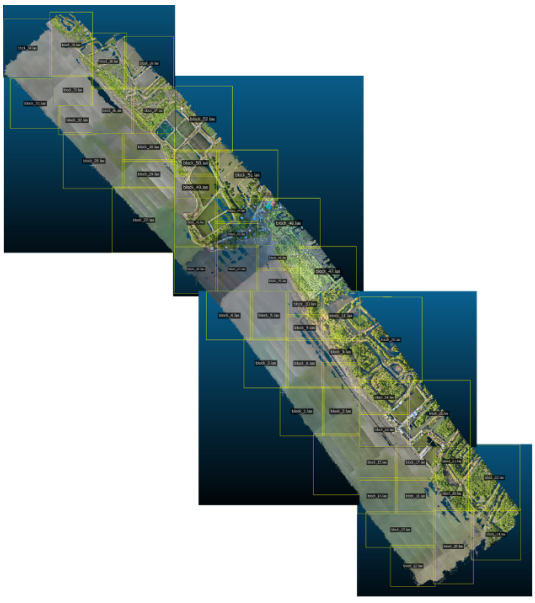


Figure 2. Full scan survey of the study area.

3 Methodology

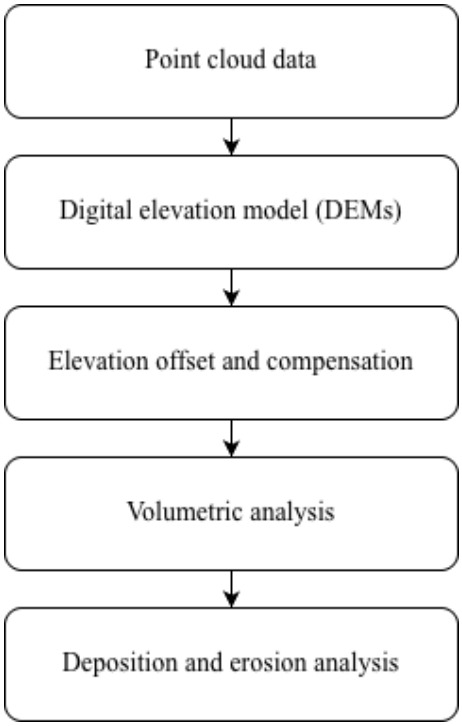


Figure 3. Coastal change analysis framework.

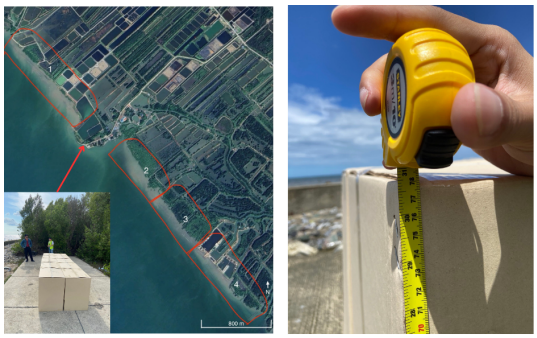


Figure 4. Field setup: (a) boxes placement in the study area and (b) verification of the box's height using tape measure.

This study employed two airborne LiDAR surveys conducted over the study site within a monthly interval. The primary survey was collected in April 2025, while the secondary surveys were obtained between October 2024 and September 2025. The surveys covered areas in both square and rectangular shapes. All raw point cloud data were preprocessed and segmented into four areas of interest

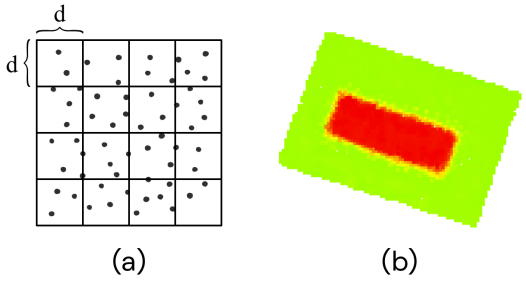


Figure 5. (a) Creation of a Digital Elevation Model (DEM) and (b) box area subsection between April-May, 2025.

using a custom-built point cloud cutting tool, designed to account for the geomorphological characteristics of the beach.

Following preprocessing, the regions of interest were further divided into box areas and sand areas using CloudCompare [19]. To ensure accuracy in volumetric estimation, non-relevant features such as trees and vegetation were removed manually using segmentation tool in CloudCompare, leaving only the sand-covered surfaces for analysis.

The analysis workflow is illustrated in Figure 3. Next, the point cloud data were converted into Digital Elevation Models (DEMs) by dividing the area into a series of grids with a selected grid size d. The elevation value (z-value) of each grid cell was calculated by averaging the z-value of all 3D points contained within it. These processes were implemented using Python scripts. To capture temporal changes in coastal morphology, DEM differencing was applied, whereby the secondary DEM was subtracted from the primary DEM. This produced elevation change surfaces as shown in Figure 5(b), which served as the basis for subsequent volumetric calculations. Let p is a point in 3D space with (x, y, z) as a coordinate system. x is latitude, y is longitude and z is the elevation. The point cloud is divided into grid with size d, as shown in Figure 5(a). Let $(\bar{Z}_{i,j})$ is the average elevation of points within the grid (i, j). Then,

$$\Delta Z_{i,j} = \bar{Z}^1_{i,j} - \bar{Z}^2_{i,j}, \tag{1}$$

where

$\bar{Z}^{\mathbf{1}}_{i,j}$ is from primary DEM,

$\bar{Z}^{\mathbf{2}}_{i,j}$ is from secondary DEM,

i denotes the i^{th} row,

j denotes the j^{th} row.

Therefore, the volume change is

$$V = \sum_{i=1}^{M} \sum_{j=1}^{N} \Delta Z_{i,j} \cdot d^2_{i,j}, \tag{2}$$

where

V is the total volume,

$d^2_{i,j}$ is an area of the grid cell (i, j), and

M and N are the number of rows and columns in the grid.

Equations (1) and (2) are the simple idea of volume estimation; however, the problem is that we cannot know the ground truth of how much the actual volume change is. In addition, the accuracy of the estimation is sensitive to the parameter d or grid resolution and the point cloud density (points/m^2). Therefore, we propose to use a known-volume object placed on an area that we can control and use the estimated volume change as a reference. In this paper we propose to use a square boxes structures with a total known volume of 2.79 m³ and height of 0.78 centimeters as shown in Figure 4. Cropped subsections ΔZ containing these boxes as illustrated in Figure 5(b), were analyzed to compare the estimated volumes derived from DEM differencing with the actual physical volumes. We sampled points in the area on the top of the box and calculated the height average. The similar approach was applied to an area on the road surface around the boxes. We use histogram plots to check the elevation distribution of the road surface and surface on the top of the box. We take the advantage of using the square boxes with sharp edges. That indicates a distinct elevation difference between the road surface and the top of the boxes. The histogram is expected to exhibit two primary peaks, corresponding respectively to the road surface and the upper surface of the boxes, as shown in Figure 6. The left peak represents the ground surface, which is centered around zero, consistent with the assumption that no elevation change occurs on the road. Based on the fact that there should be no change on the road area. Therefore, we move all the pillars to the left

with some value, which we call an offset value. The offset value as corresponds to the difference between average height ($\bar{Z}$) and actual height of the box(red) area, as in (3). This correction factor was then applied across the sand area grids, as in (4) to improve the reliability of the sand volume estimations. When applying the offset value to the sand area with different grid size, we use the concept of Constant of proportionality [20] to calculate corresponding offset value.

$$\text{offset} = \text{actual height} - \text{average height}, \quad (3)$$

$$\mathrm{Z}^{\text{new}} = \mathrm{Z}^{\text{old}} + \text{offset} \quad (4)$$

Finally, volumetric change analysis was performed to quantify erosion and deposition processes across the selected area and time frame. In addition, spatial analyses were conducted using DEM differencing plots to visually interpret and validate the distribution of erosion and deposition within the sand area.

4 Results and Discussion

4.1 Offset value and compensation

In this section, box volume estimation method was conducted compare with the ground-truth. All experiments were carried out with the same grid size of 0.07 meters to generate the DEMs for volume calculation. Table 1 presents a comparison between the actual box volume, the initial estimation without correction, the refined estimation with the offset adjustment, and the results obtained using CloudCompare software. The initial estimation produced a volume of 3.61 m³, which considerably overestimated the true volume of 2.79 m³. After applying the offset correction, the estimated volume is 2.91 m³, close to the ground-truth value. In contrast, the CloudCompare estimation yielded 3.60 m³, which remained significantly higher than the actual volume.

To quantitatively assess the effectiveness of the offset compensation, the absolute error between the estimated and actual box volumes was calculated before and after applying the correction. The initial estimation showed an absolute error of 0.82 m³, indicating a notable overestimation of the true volume. After applying the offset adjustment, the error was significantly reduced to 0.12 m³, representing an improvement of approximately 85% in estimation accuracy. This substantial reduction demonstrates that the offset compensation method effectively minimized systematic bias and enhanced the reliability of volumetric calculations.

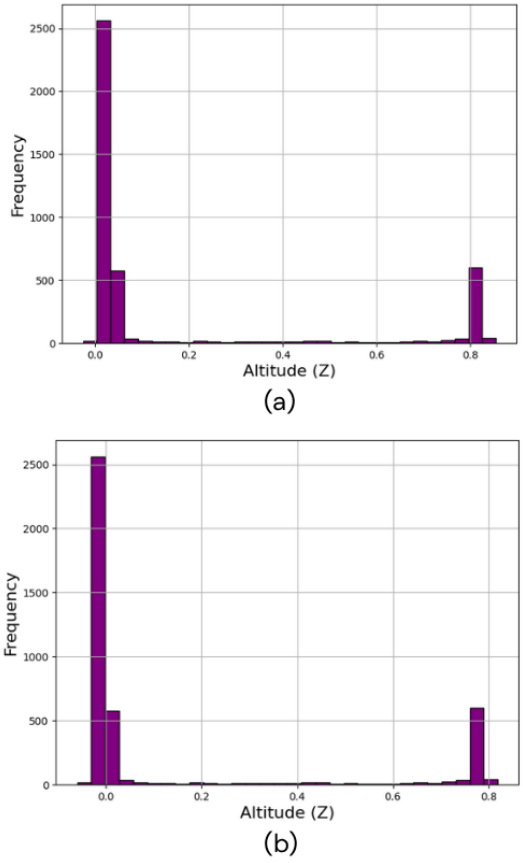


Figure 6. Histograms of delta altitude (Z) values in the box area: (a) before offset correction and (b) after offset correction.

To further evaluate the offset compensation approach, elevation histograms of the box area were examined before and after applying the correction. As shown in Figure 6, the distribution of delta altitude (Z) values shifted slightly to the left after offset adjustment. The highest frequency bin remained centered around zero, corresponding to the ground surface, which confirms the stability of the reference level. More importantly, the secondary peak on the right side of the histogram, representing the top surface of the boxes, was observed at approximately 0.78 m after offset correction.

Table 1. Volumetric change estimation comparison.

Actual (m^3)	Initial Estimation (m^3)	Estimation w/t offset (m^3)	Estimation by CloudCompare (m^3)
2.79	3.61	**2.91**	3.6

This value is consistent with the known physical height of the boxes, indicating that the applied offset improved the alignment between the estimated and actual height. Overall, the results demonstrate that the offset compensation successfully reduced systematic errors and improved the reliability of volume calculations derived from DEM differencing.

4.2 Volumetric Analysis

As observed from Table 2, the volumetric change from October 2024 to August 2025 reveals distinct patterns of sand deposition and erosion across the four study sections. During the first observation period (October 2024-April 2025), a substantial net deposition of +93,725 m³ was observed. This pronounced accretion suggests that the calmer sea conditions and lower wave energy typical of the dry season favor sediment accumulation along the coast. Among the four sections, Section 2 recorded the highest net increase (+35,395 m³), indicating that this area may act as a natural sediment sink during periods of low hydrodynamic activity.

In contrast, the second observation period (April-August 2025) exhibited a notable decrease in net deposition, with an overall gain of only +22,325 m³ and significantly higher erosion volumes (-6,452 m³). This seasonal shift can be attributed to intensified wave action, stronger longshore currents, and increased rainfall runoff during the monsoon season, which promote sediment transport and coastal erosion. Section 2 again showed a marked response, recording the highest erosion (-4,326 m³), suggesting that this area is particularly sensitive to hydrodynamic changes.

Overall, the seasonal analysis underscores the dynamic nature of sediment distribution along the coastline, where the balance between deposition and erosion is strongly governed by climatic patterns. These findings highlight the importance of continuous monitoring to better understand seasonal shoreline evolution and to support effective coastal management strategies.

4.3 Deposition and Erosion Analysis

During the first period (October-April), a noticeable net deposition was observed across all four sections. Section 2 exhibited the highest deposition followed by Section 4.The spatial plots in Figure 7 support this observation, showing broad yellow-to-green zones representing positive elevation changes, especially in Sections 2 and 4, indicating large-scale sand deposition.Whereas, in the second period (April-August), noticeable erosion occurred in several sections, particularly Section 1 and Section 2, likely due to increased wave energy and coastal run-off associated with the monsoon-driven currents. The maps in Figure 8 visually confirm this trend, with increased dark purple regions indicating erosion zones, particularly along the upper parts of Sections 1 and 2.

5 Conclusion

This study demonstrated an effective approach for estimating sand volume change and analyzing nearshore dynamics using a time series of airborne LiDAR data. By employing Digital Elevation Model (DEM) differencing and an offset compensation technique, the proposed method effectively reduced systematic LiDAR errors and improved the accuracy of volumetric estimations. Validation using known box volumes confirmed that the offset-corrected results closely matched ground-truth measurements, reinforcing the reliability of the proposed approach.

The volumetric and seasonal analyses revealed distinct patterns of deposition and erosion influenced by monsoonal conditions. From October to April, the site exhibited substantial net deposition, whereas from April to August, increased erosion and sediment redistribution were observed. These findings provide valuable insights into the temporal and spatial variability of coastal sediment dynamics and emphasize the role of seasonal processes in shaping shoreline change. Overall, the results highlight the potential of airborne LiDAR as a reliable tool for

Table 2. Summary of seasonal volumetric change from October, 2024 to August, 2025.

Period	Sections	Net-Change (m^3)	Deposition (m^3)	Erosion (m^3)
October, 2024 - April, 2025	Section 1	+19704	+19704	-0
	Section 2	+35395	+35522	-127
	Section 3	+10382	+10646	-264
	Section 4	+28244	+28276	-32
	Total	+93725	+94148	-423
April, 2024 - August, 2025	Section 1	+1074	+2541	-1467
	Section 2	+2186	+6513	-4326
	Section 3	+3503	+4131	-627
	Section 4	+15561	+15593	-32
	Total	+22325	+28777	-6452

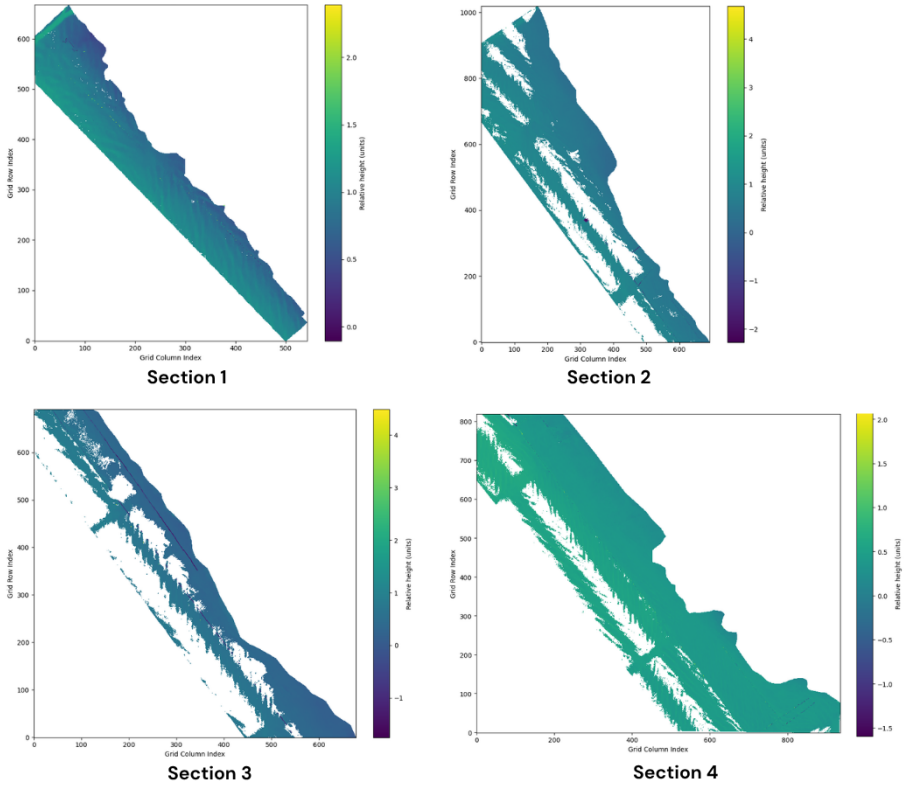


Figure 7. Deposition and erosion pattern of the beach from October, 2024 to April, 2025.

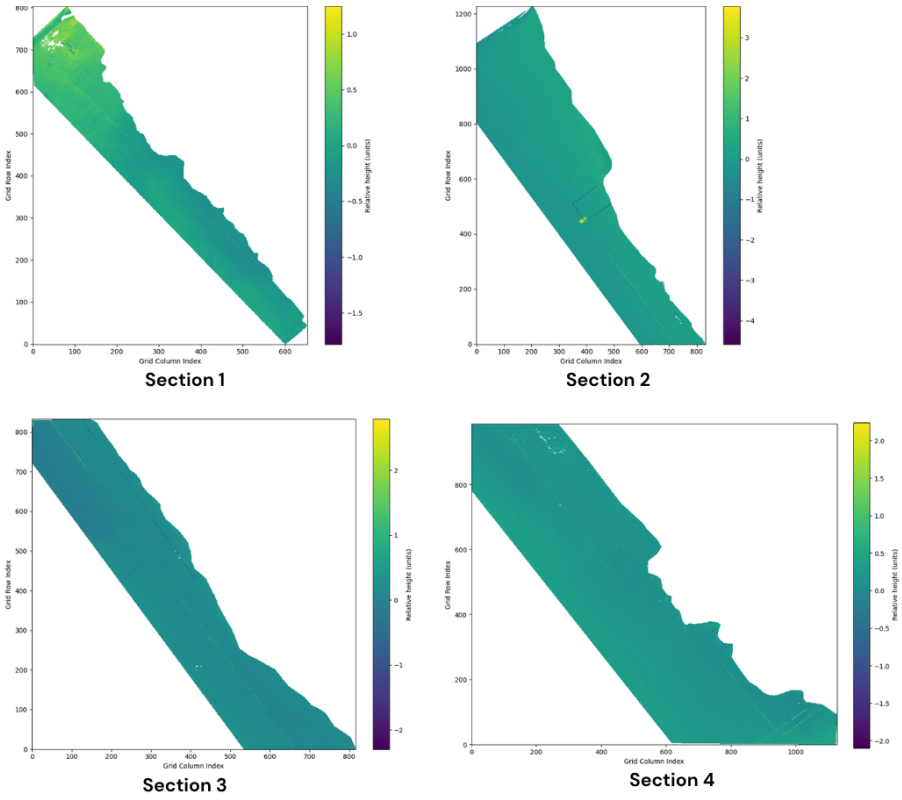


Figure 8. Deposition and erosion pattern of the beach from April, 2025 to August, 2025.

quantitative coastal monitoring and long-term management, particularly when integrated with appropriate error correction and spatial analysis techniques.

Acknowledgment

The ASEAN IVO project (https://www.nict.go.jp/en/aseanivo/index.html), titled "Coastal Erosion Monitoring Platform Based on Wireless Sensor Networks and 3D Point Clouds from Airborne LiDAR" was involved in the production of the contents of this work and financially supported by NICT (https://www.nict.go.jp/en/index.html).

References

[1] D. B. Angnuureng, B. Charuka, R. Almar, O. A. Dada, R. Asumadu, N. A. Agboli, and G. T. Ofosu, "Challenges and lessons learned from global coastal erosion protection strategies," *iScience*, vol. 28, no. 4, p. 112055, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2589004225003153

[2] R. S. Dewi and W. Bijker, "Dynamics of shoreline changes in the coastal region of sayung, indonesia," *The Egyptian Journal of Remote Sensing and Space Science*, vol. 23, no. 2, pp. 181–193, 2020.

[3] P. Contestabile and D. Vicinanza, "Coastal vulnerability and mitigation strategies: From monitoring to applied research," p. 2594, 2020.

[4] R. C. Carvalho and R. Reef, "Quantification of coastal change and preliminary sediment budget calculation using sfm photogrammetry and archival aerial imagery," *Geosciences*, vol. 12, no. 10, p. 357, 2022.

[5] U. Okyay, J. Telling, C. L. Glennie, and W. E. Dietrich, "Airborne lidar change detection: An overview of earth sciences applications," *Earth-Science Reviews*, vol. 198, p. 102929, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0012825218306470

[6] Y.-C. Lin, Y.-T. Cheng, T. Zhou, R. Ravi, S. M. Hasheminasab, J. E. Flatt, C. Troy, and A. Habib, "Evaluation of uav lidar for

mapping coastal environments," *Remote Sensing*, vol. 11, no. 24, p. 2893, 2019.

[7] J. Wang, L. Wang, S. Feng, B. Peng, L. Huang, S. N. Fatholahi, L. Tang, and J. Li, "An overview of shoreline mapping by using airborne lidar," *Remote Sensing*, vol. 15, no. 1, p. 253, 2023.

[8] K. Zhang, D. Whitman, S. Leatherman, and W. Robertson, "Quantification of beach changes caused by hurricane floyd along florida's atlantic coast using airborne laser surveys," *Journal of Coastal Research*, vol. 21, no. 1, pp. 123–134, 2005.

[9] G. Zhou and M. Xie, "Coastal 3-d morphological change analysis using lidar series data: a case study of assateague island national seashore," *Journal of Coastal Research*, vol. 25, no. 2, pp. 435–447, 2009.

[10] Q. Robertson, L. Dunkin, Z. Dong, J. Wozencraft, and K. Zhang, "Florida and us east coast beach change metrics derived from lidar data utilizing arcgis python based tools," in *Beach Management Tools-Concepts, Methodologies and Case Studies*. Springer, 2017, pp. 239–258.

[11] A. M. A. Mahmoud, E. Hussain, A. Novellino, P. Psimoulis, and S. Marsh, "Monitoring the dynamics of formby sand dunes using airborne lidar dtms," *Remote Sensing*, vol. 13, no. 22, p. 4665, 2021.

[12] A. P. Young and S. A. Ashford, "Application of airborne lidar for seacliff volumetric change and beach-sediment budget contributions," *Journal of Coastal Research*, vol. 22, no. 2, pp. 307–318, 2006.

[13] F. A. Sesli and M. Caniberk, "Estimation of the coastline changes using lidar." *Acta Montanistica Slovaca*, vol. 20, no. 3, 2015.

[14] G. M. Suir, S. Jackson, C. Saltus, and M. Reif, "Multi-temporal trend analysis of coastal vegetation using metrics derived from hyperspectral and lidar data," *Remote Sensing*, vol. 15, no. 8, p. 2098, 2023.

[15] Y. Zhang and X. Hou, "Characteristics of coastline changes on southeast asia islands from 2000 to 2015," *Remote Sensing*, vol. 12, no. 3, p. 519, 2020.

[16] M. M. H. A. Latif and G. Y. Yong, "Coastal erosion in the unprotected and protected sections at berakas: A comparative study in brunei darussalam," *Southeast Asia: A Multidisciplinary Journal*, vol. 21, no. 1, pp. 45–62, 2021.

[17] D. Mongus and B. Žalik, "Parameter-free ground filtering of lidar data for automatic dtm generation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 67, pp. 1–12, 2012.

[18] W. Xiao, "Detecting changes in trees using multi-temporal airborne lidar point clouds," Master's thesis, University of Twente, 2012.

[19] CloudCompare, "CloudCompare (version 2.13.0) [GPL software]," http://www.cloudcompare.org, retrieved on November 3, 2025.

[20] L. M. Ahl, "Research findings' impact on the representation of proportional reasoning in swedish mathematics textbooks," *Journal of Research in Mathematics Education*, vol. 5, no. 2, pp. 180–204, 2016.

STUDENT TRACK - PAPERS

A Two-Stage Approach to Khmer Scene Text Recognition using Faster R-CNN and TrOCR

Sethisak San **Mitona Chan** **Eklim Sek**
School of Digital Technology, American University of Phnom Penh, Phnom Penh, Cambodia
2023471san@aupp.edu.kh

Abstract

Khmer scene text recognition presents significant challenges due to the complex structure of Khmer's script, which includes stack consonants, special diacritics, and a lack of consistent word spacing. In this work, we propose a two-stage approach to detect and recognize Khmer scene text. We first utilize Faster R-CNN, a deep learning-based object detection model, to identify text region coordinates in various images. Subsequently, the detected regioned are processed by TrOCR, a Transformer-based model, for character recognition. Our model is trained and evaluated on a combination of real-world and synthetic data, including the KhmerST dataset, the 62k Khmer Printed Dataset, and the Khmer Annotation dataset. Our approach achieves strong detection performance, with a high recall rate of 91.4% and a precision of 70.1%, indicating robust text localization. The recognition stage yields a Character Error Rate (CER) of 18.3%. On a manually curated real-world test set, the detection model achieves a precision of 62.2%, recall of 55.5%, demonstrating reasonable generalization. Our analysis shows that while the detector effectively localize the text, recognition errors are primarily linked to the script's inherent complexity and inconsistencies in text-line segmentation.

Keywords: *Khmer text recognition; scene text detection; OCR; Faster R-CNN; TrOCR; deep learning; computer vision*

1 Introduction

Text detection from natural images is not an easy task, particularly for complex scripts such as Khmer. Text may appear in different forms, including printed text, handwritten text, and scene text. Printed text is easier to detect because it is often shown in clear space and font size. On the other hand, handwritten style is more difficult due to the styles, sizes, and alignment of

Figure 1. Printed text display in clean background with consistent space and font use.

the word. Regarding the text of the scene, it may appear in various sizes, fonts, orientations, and backgrounds in real-world scenarios, as illustrated in Figure 1-3. Furthermore, the characteristics of the Khmer language, stacked consonants, diacritical marks, and the absence of obvious word spaces make it even more difficult to detect [1] . Therefore, Khmer scripts present unique challenges that make scene text recognition far more difficult than widely studied languages like English or Chinese [2]. Moreover, unlike Latin scripts, Khmer is not space-separated between words consistently, which complicates segmentation. Such language features combined with real-world issues such as low-quality images, non-uniform lighting, noisy environments, and inconsistent fonts pose very challenging tasks to effective text detection and recognition [1, 3].

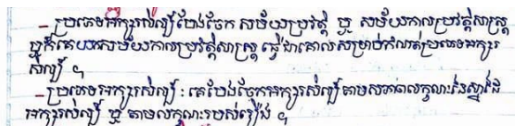


Figure 2. Handwritten shows in different writing styles, sizes, spacing, and stroke.

Early OCR engines, like Tesseract [4], have not been very successful when applied to Khmer, while the TRBA model [2], in which ResNet [5] is combined for feature extraction, BiLSTM [6] for sequence modeling, and Transformer-based attention [7], achieves significantly better performance [2]. Most of the existing work has been done on printed text or synthetic data, whereas

a. lighting variation b. shadow and reflection c. complex background

Figure 3. Scene text affect by lighting variation, shadows, reflections, and complex background.

natural scene images include variations such as lighting, blur, and background noise, in which can significantly impact the OCR performance.

To tackle these difficulties, our project comes up with a two-stage approach. In the first stage, the model learns intermediate concepts, which then serve as the feature space for the second stage [8]. Specifically, we first employ an object detection method named Faster R-CNN (Region-based Convolutional Neural Network) [9] for detecting and localizing text areas in real images, in which make use of FPN (Feature Pyramid Network) [10]. In the second stage, we utilize TrOCR, a Transformer-based OCR model, to detect and convert recognized text areas into readable digital text that can be processed by computers.

2 Related Work

2.1 Seq2Seq with Attention on Khmer OCR

Khmer Optical Character Recognition (OCR) conducted by [11] introduced one of the pioneering end-to-end sequence-to-sequence (Seq2Seq) frameworks employing attention mechanisms for recognizing printed text. The encoder architecture comprised convolutional residual blocks in conjunction with Gated Recurrent Units (GRUs) [12], whereas the decoder utilized an attentive GRU to produce character sequences directly from complete text-line images, eliminating the need for segmentation or manually engineered feature extraction. In order to train the model, the researchers created approximately 92,000 synthetic images sourced from the Asian Language Treebank (ALT) [13] by employing seven widely used Khmer fonts, alongside further augmentations including noise, dilation, and rotation. Evaluation on test dataset with 3000 images shows the model achieved a Character Error Rate (CER) of only 1% and a Word Error Rate (WER) of 9%, in which surpassed the Tesseract OCR that's reported to achieved a CER of 3% and a WER of 26% [11].

Building on this foundation, Buoy et al. [14] further improved this by increased the dataset to over three million synthetic text-line images in various Khmer fonts. Unlike the earlier research, this study applied with augmentation by adding noise, blur, and complex background during training in order to make the model more generalized in real life situations. The result with the dataset of 6400 augmentation validation images achieved a CER of 0.7% while Tesseract only achieved with CER of 35.9%. Even with de-augmented data, the new model still have better accuracy with 0.24% vs 1.6% for Tesseract. The main difference between these two studies is in scale and robustness; with the 2021 study showing that Seq2Seq with attention could work for Khmer OCR, while the 2022 study improved it by enlarging the dataset, using more complex augmentation, and achieving top performance in noisy and font-variable conditions [14].

2.2 Khmer STDR

Previous studies on Khmer Optical Character Recognition (OCR) mostly focused on printed text with sequence-to-sequence (Seq2Seq) models with attention. However, these systems only worked well for printed text in most situations. They did not address recognizing scene text, which is when text is found in natural settings with distortions, blur, and background noise. Nom et al. [1] introduced KhmerST, the first dataset for Khmer scene text. This dataset contained 1,544 images that the researchers have annotated; 997 indoors and 547 outdoors. Unlike fake printed datasets, KhmerST shows real-world differences, including raised or flat text, poor lighting, far or partly hidden signs, and complex backgrounds. Each image is annotated with polygonal bounding boxes at text-line level, producing 3,463 cropped regions for recognition tasks. The authors also did experiment on both detection and recognition task with the YOLO family and the Transformer model. Several YOLO architectures were fine-tuned; among them, YOLOv8 achieved the best result for the text detection task, with a recall of 0.832 and mAP50 of 0.899. For text recognition, the researchers investigated the performance of two transformer models; TrOCR and Tesseract. TrOCR achieved a Character Error Rate (CER)

of 1.01% and Word Error Rate (WER) of 2.24%, while Tesseract fall behind with 1.30% CER and 4.75% WER.

2.3 Cross-Lingual Learning for Khmer STDR

In high-resource languages like English and Chinese, have shown great performance in both text detection and text recognition, however low-resource language like Khmer remain a problem in the field. To overcome these limitations, Nom et al., [15] proposed CLL framework by using high resource language fine-tune on Khmer languages . The authors uses YOLOv11 [16] and TrOCR on the WildKhmerST [17]for training and KhmerST [1] for evaluation. In this study, the authors compares both training from scratch and fine-tuning models pretrained on high resources language. For text detection, they used YOLOv11 and compared a model that built from scratch to the improved one that using weights pretrained on the COCO dataset [18]. The improved YOLOv11 performed better (precision 0.840, recall 0.809, mAP50 = 0.889) and needed much less computing power than the model built from scratch (precision 0.807, recall 0.801, mAP50 = 0.852). For recognition, they used TrOCR and found that they could significantly reduce the error rate if they upgraded the pretrained Latin script models with the Khmer data. They achieved CER of 0.17–0.18 with WER of 0.33–0.35 compare to the CER 0.40–0.44 and WER 0.55–0.59 if they were trained from scratch [15].

3 Methodology

Scene text detection and recognition have been widely studied, but Khmer presents unique challenges due to the complexity of its script [1]. To obtain desirable outcomes, a two-stage approach is preferred over end-to-end models, with Stage 1 focusing on precise text detection and Stage 2 on recognition. This separation enhances detection accuracy in complex backgrounds and allows independent optimization, making it better suited for low-resource environments [19].

Among detection models, EAST [20] provides real-time performance through direct regression of text boxes without complex post-processing. However, its coarse outputs often miss fine details critical for Khmer script. CRAFT [21] achieves high accuracy by predicting pixel-level character regions and affinity links, enabling detection of curved or irregular text, but it requires character-level annotations, which are scarce for Khmer datasets.

To address these challenges, Faster R-CNN, proposed by [9], offers a robust two-stage detection framework. It first generates region proposals, followed by refined classification and bounding box regression. When paired with a Feature Pyramid Network (FPN) backbone [10], the detector gains powerful multi-scale feature extraction capabilities. This is particularly advantageous for detecting small and intricate elements, such as Khmer diacritics.

The Faster R-CNN + FPN architecture strikes an effective balance between accuracy and annotation simplicity. Unlike CRAFT, it only requires word-level bounding boxes, reducing the burden of intensive annotation. The FPN further enhances detection of small glyphs across multiple resolutions, making this combination highly suitable for Khmer scene text detection. When integrated with a strong recognition network, this approach provides a practical and scalable solution for Khmer scene text recognition in real-world settings.

After selecting a detector, the next step is to choose a recognizer capable of accurately handling the complexities of Khmer text. Traditional OCR engines like Tesseract rely on image preprocessing and LSTM-based sequence modeling [22], which work well for clean printed text but struggle with noisy, irregular scene text. Deep learning-based recognizers such as CRNN [23], RARE [24], ASTER [25], and SAR [26] have advanced this field by integrating CNNs, recurrent layers, spatial transformers, and attention mechanisms. While these models achieve strong performance on regular text, they face challenges with highly complex scripts like Khmer, which contain stacked glyphs and intricate diacritics.

The Transformer-based OCR model, TrOCR, proposed by Li et al. [27], introduces a fully transformer-based encoder-decoder architecture. The encoder is a vision transformer (ViT) that processes image patches, while the decoder is a pre-trained language transformer similar to RoBERTa. TrOCR leverages large-scale pre-training on synthetic and real-world image-text pairs, significantly improving performance after fine-tuning on target datasets. According to the KhmerST benchmark, TrOCR substantially out-

performs Tesseract for Khmer scene text recognition [1]. This demonstrates the effectiveness of modern transformer-based architectures for low-resource languages such as Khmer.

By comparing all of these different models, we find TrOCR to be the most suitable one as its global self-attention makes it excellent at modeling long-range dependencies between base consonants and diacritics, which is crucial for Khmer language.

3.1 Pipeline and Detailed Architecture

Given the previously mentioned challenges of Khmer script recognition [1], we propose a two-stage architecture that combines Faster R-CNN with a Feature Pyramid Network (FPN) for text detection and TrOCR (Transformer-based OCR) for text recognition. This hybrid approach leverages the strengths of advanced object detection and sequence-to-sequence modeling, enabling precise localization and robust recognition of Khmer text in complex, real-world scenes. The system operates as a two-step pipeline (1) Text Detection: Faster R-CNN with FPN identifies and localizes text regions within the input image, producing bounding boxes that tightly enclose words or lines of text. (2) Text Recognition: Each detected region is cropped and passed to the TrOCR model, which encodes the visual features and decodes them into a sequence of Khmer characters as can be observed from Figure 4.

3.1.1 Text Detection with Faster R-CNN and FPN

Standard Faster R-CNN can struggle with small-scale features, such as the fine details and diacritical marks that are common in Khmer text, often resulting in incomplete detection. To address this limitation, we integrate a Feature Pyramid Network (FPN) into the Faster R-CNN framework. The FPN enhances multi-scale feature representation by combining high-resolution, low-level features with high-level semantic features. This design improves the model's ability to detect both large and small text instances in diverse scene settings. The detector outputs a set of bounding boxes, each representing a localized text region. These bounding boxes are then cropped and resized to a consistent size before being forwarded to the recognition stage.

3.1.2 Text Recognition with TrOCR

For the recognition step, we employ TrOCR, a Transformer-based, sequence-to-sequence OCR model. TrOCR reframes text recognition as a translation task, mapping visual features from cropped text regions directly to a sequence of text tokens. This eliminates the need for explicit character segmentation, which is particularly difficult for Khmer due to its connected characters and complex positional rules.

TrOCR is composed of two main components: (1) A Vision Transformer (ViT)-based encoder for visual feature extraction. (2) A Transformer-based decoder for autoregressive text generation.

For Vision Transformer (ViT) Encoder, The cropped text region is first divided into non-overlapping patches of size 16×16 pixels. Each patch is linearly embedded into a vector of dimension 768 (hidden_size). Positional embeddings are added to preserve the spatial relationships among patches. The resulting sequence of patch embeddings is processed through 12 Transformer layers, each equipped with 12 self-attention heads, to generate a rich representation of the visual input.

For the Transformer Decoder, the decoder operates autoregressively, generating one Khmer token at a time. At each step, it leverages cross-attention mechanisms to align the encoded visual features from the ViT encoder with the partially generated output sequence. The decoder also consists of 12 layers but uses 16 attention heads and a hidden dimensionality of 1024 (d_model), providing greater capacity for handling complex text sequences.

To effectively recognize Khmer text, TrOCR is configured with several important adjustments: Vocabulary Size: A tokenizer is designed with a vocabulary of 50,265 tokens, covering all Khmer characters, numbers, and punctuation marks. Input Size: Cropped text regions are resized to 384×384 pixels before being processed by the encoder, ensuring consistent input dimensions. Output Generation: The decoder continues generating tokens until it outputs an end-of-sequence (<EOS>) token, which signifies completion of the transcription. A Two-Stage Approach workflow process proceeds as follows: (1) Input Scene Image: A full-scene image containing Khmer text is provided to the system. (2) Text Detection: Faster R-CNN with FPN gener-

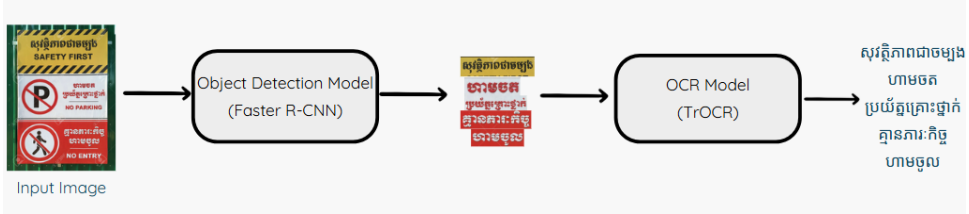


Figure 4. Two-Stage Approach OCR pipeline integrating detection and recognition

ates bounding boxes for text regions at multiple scales. (3) Cropping: Each bounding box is used to extract and normalize individual text regions. (4) Recognition: TrOCR converts each cropped region into a sequence of Khmer characters.

3.2 Dataset

3.2.1 Training and Validation Datasets

To test and train our Khmer scene text recognition system, we gathered and combined data from three public sources: KhmerST [1], the 62k Khmer Printed Dataset [28] (synthetic images), and the Khmer Annotation Dataset [29]. The datasets provide a good balance of Khmer text samples — ranging from clean synthetic text lines to actual scene text in natural scenes. By combining these datasets, we could train our model to recognize both the visual pattern of Khmer script and learn to deal with the issues of scene variability, font differences, and image noise.

• KhmerST Dataset (GitLab) [1]: This dataset contains 1,544 real-world Khmer text images from diverse settings, including signs, banners, and advertisements. It provides line-level polygon annotations converted to bounding boxes for detection and cropped regions for OCR training, featuring varied lighting, angles, and backgrounds to simulate real-world scenarios.

• 62K Khmer Printed Dataset (Hugging Face) [28]: This dataset consists of 62,300 synthetically generated images of printed Khmer text. The images were generated with different font sizes and color variations.

• Khmer Annotation Dataset (Kaggle) [29]: This Khmer Annotation data is a collection of images of Khmer text with XML-style annotations consisting of 3,376 image of printed Khmer text. It is a record with an image file and relevant width and height, word-level annotations with both textual content written in Khmer script and coordinates (x1, x2, y1, y2) of a bounding box localizing each word in the image.

3.2.2 The KST-Wild Test Set

Figure 5. Sample images from our KST-Wild test set. The images showcase the diversity of challenges, including varied fonts, complex backgrounds, reflections, and non-horizontal text orientations.

For the final, rigorous evaluation of our model's performance, we prepared a separate test set, which we refer to as the Khmer Scene Text In-the-Wild (KST-Wild) set. This dataset comprises of 310 real-scene images collected specifically to simulate challenging, real-world use cases and was not used during training or validation. The collection process was to include the following: (1) Real-world text from street signs, billboards, shops, posters, vehicles, and labels. (2) A wide range of font styles, background textures, and image conditions (e.g., tilted text, low lighting, occlusion). (3) Only real-scene images (no synthetic text).

To standardize training data, we normalized text areas and manually annotated images using the VGG Image Annotator (VIA), a lightweight web-based tool for creating boxes and bounding shapes [30]. This dataset serves as the benchmark for our final reported metrics.

Table 1. Summary of Datasets Used

Source	Images	Type	Use
Merged for Training/Validation Pool			
KhmerST [1]	1,544	Scene	Train/Val
62K Printed [28]	62,300	Synthetic	Train/Val
Khmer Annot. [29]	3,376	Printed	Train/Val
Total for Split	**67,220**	**Mixed**	**80/10/10**
Held-out Final Test Set			
KST-Wild (Ours)	**310**	**Wild**	**Final Eval.**

4 Experiments and Results

4.1 Experimental Setup

TrOCR Training: We utilized Amazon SageMaker Studio on Amazon Web Services (AWS) to train the TrOCR model on two datasets: a 62K-sample Khmer Printed Text dataset and the Khmer Scene Text (KhmerST) dataset. Training was conducted on an `ml.g5.4xlarge` instance, equipped with one NVIDIA A100 Tensor Core GPU, 64 vCPUs, and 256 GB of RAM. The process required approximately 8.3 hours to complete.

Object Detection Training: For the text detection component, we trained a Faster R-CNN model on the KhmerST dataset using an `ml.g4dn.8xlarge` instance, which provides one NVIDIA Tesla T4 GPU and 128 GB of RAM. The training process completed in approximately 30 minutes.

Pipeline Integration: Since text detection and recognition are distinct tasks, we integrated both models into a unified pipeline for practical use. The detection model first analyzes an input image and outputs bounding box coordinates for regions containing text. These coordinates are then passed to the OCR model, which extracts the text from each detected region. To enhance usability and accessibility, we implemented a simple graphical interface using Gradio, allowing end-users to interact with the pipeline without requiring technical expertise.

4.2 Evaluation Protocols

4.2.1 Object Detection

The object detection task is commonly evaluated using precision, recall, and F1-Score. Precision measures the proportion of correctly predicted bounding boxes among all predicted boxes, while recall measures the proportion of correctly detected objects among all ground-truth objects. while F1-score gives us an average balanced score between recall and precision. Together, these metrics evaluate whether the model is correctly identifying target objects and minimizing missed detections.

Formally, they are defined as:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 - score = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{3}$$

where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives.

These evaluation metrics follow the standard definition used in the PASCAL VOC Challenge [31].

4.2.2 Optical Character Recognition

To evaluate the OCR task, we utilize several complementary metrics.

Character Error Rate (CER) is computed as the Levenshtein edit distance at the character level divided by the total number of characters [32].

Word Error Rate (WER) extends this concept to the word level, capturing insertions, deletions, and substitutions at the word granularity [33, 34].

Together, these metrics provide a robust, multi-level assessment of OCR performance.

4.3 Results and Discussion

4.3.1 Object Detection Result

From the table illustrated, Faster R-CNN have shown impressive results achieving an impressive score of 91.4% recall score indicating that

Table 2. Hyperparameter Settings for the Detection and Recognition Models

Parameter	Faster R-CNN	TrOCR
Model Base	ResNet-50 w/ FPN	microsoft/trocr-base-handwritten
Optimizer	SGD	AdamW
Learning Rate	0.005	1×10^{-5} to 2×10^{-5}
Batch Size	8	16
Epochs	10	5
Weight Decay	0.0005	0.01
Input Size	Variable	384×384 px
IoU Threshold	0.5	–
Max Token Length	–	128 char

Table 3. Performance Comparison of Object Detection and OCR Models

Task	Metric	Score
Object Detection (Faster R-CNN)	Precision	70.1%
	Recall	91.40%
	F1-score	79.40%
Object Detection (Faster R-CNN), KST-Wild	Precision	62.24%
	Recall	55.51%
	F1-score	58.43%
OCR (TrOCR), KST-Wild	CER	18.27%
	WER	54.43%

Figure 6. Sample results of output from Faster R-CNN

the model is able to detect most of the actual relevant objects in the scene text images. it has also shown a relatively good score of 70.1% for the precision score which indicates that the model's output are relatively correct but may output false positive; this gives us a F1-Score of 79.40%. However, on the manually collected dataset, the model have achieved a slightly lower score of F1-score at 58.43%. From these results, we can infer that the model is capable of predicting and outputting bounding box around text areas in scene-text images; however, those bounding box are not localizing the bounding box as accurately as the ground truth in real world images. However, for the task of optical character recognition in scene-text scenarios, this result is acceptable as the model have shown that that it is capable of detecting all of the relevant text region area in the image even if those box may not exactly match the annotated bounding box which could be drawn subjectively.

4.3.2 Optical Character Recognition Result

From Table 3, the **KST-Wild** dataset achieved an accuracy of 18.27% in terms of CER and 54.43% in terms of WER. These metrics indicate that, despite the model producing seemingly reasonable predictions in certain cases, the overall transcription accuracy remains relatively low. This discrepancy can be attributed to the behavior of the object detection stage, which crops detected text regions before passing them to the recognition model. In several instances, ground truth annotations contain multi-line text regions, whereas the object detection model separates these into multiple individual image segments. When these segmented regions are processed independently by TrOCR, only partial text—often corresponding to a single line—is recognized, while the remaining lines are omitted. This mismatch between annotation granularity and recog-

Table 4. Recognition Performance Compared with the KhmerST Baseline. Models Were Evaluated on the KhmerST Dataset

Model / Pipeline	**CER (%)**	**WER (%)**
KhmerST Baseline (YOLOv8 + TrOCR) [1]	1.01	2.24
Tesseract (as reported in [1])	1.30	4.75
Our Model (Faster R-CNN + TrOCR)	**5.12**	**12.11**

nition output significantly increases both CER and WER.

In addition, the relatively low performance can be largely attributed to the unique structural characteristics of the Khmer script. Khmer words are commonly formed by combining a primary consonant with one or more subconsonants, resulting in complex compound character structures. The recognition model frequently struggles to correctly identify these composite characters, leading to misrecognitions that substantially degrade overall accuracy. These findings underscore the inherent challenges of OCR for complex scripts such as Khmer, particularly in handwritten or noisy real-world settings.

5 Conclusion

This study presented a two-stage Khmer scene text recognition pipeline that integrates Faster R-CNN with Feature Pyramid Networks for text detection and a Transformer-based OCR model (TrOCR) for recognition. The proposed approach demonstrates that a detection-first architecture is effective for localizing Khmer text in complex real-world scenes, achieving strong recall and robust generalization across varied backgrounds and lighting conditions. While recognition performance on the KST-Wild dataset remains moderate due to challenges such as stacked consonants, inconsistent spacing, and line-level segmentation mismatches, the results highlight the feasibility of applying transformer-based OCR models to low-resource, structurally complex scripts like Khmer. The findings indicate that recognition accuracy is primarily constrained by limited training data, short fine-tuning duration, and imperfect alignment between detection and recognition granularity. Future work will focus on improving text-line grouping strategies, expanding real-world annotated datasets, incorporating language-model–aware decoding, and extending training at scale to fully exploit the capabilities of Transformer-based OCR for Khmer scene text recognition.

Acknowledgment

We extend our heartfelt gratitude to our advisor and professor Dr. Ly Rottana, for his support and guidance on this research from the very beginning. We also would like to extend our appreciation to all the authors and researchers who have worked on similar projects in the past for allowing our research to be built upon their works.

References

[1] V. Nom, S. Bakkali, M. M. Luqman, M. Coustaty, and J.-M. Ogier. Khmerst: A low-resource khmer scene text detection and recognition benchmark. *arXiv:2410.18277*, 2024.

[2] S. Keo, M. Coustaty, S. Bakkali, and M. Rossinyol. State-of-the-art khmer text recognition using deep learning models. In *Proc. Int. Conf. Adv. Eng. Technol. (ACET)*, Phnom Penh, Cambodia, 2024.

[3] D. Valy, M. Verleysen, and S. Chhun. Text recognition on khmer historical documents using glyph class map generation with encoder–decoder model. In *Proc. Int. Conf. Pattern Recognit. Appl. Methods (ICPRAM)*, pages 749–756, 2019.

[4] Wikipedia contributors. Tesseract. Wikipedia, The Free Encyclopedia, n.d. Accessed: Sep. 15, 2025.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[6] Z. Huang, W. Xu, and K. Yu. Bidirectional LSTM-CRF models for sequence tagging. *arXiv:1508.01991*, 2015.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez,

L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv:1706.03762*, 2017.

[8] S. Mani, W. R. Shankle, M. B. Dick, and M. J. Pazzani. The two-stage machine learning model for guideline development. 1999.

[9] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 28, pages 91–99, 2015.

[10] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2117–2125, 2017.

[11] R. Buoy, S. Kor, and N. Taing. An end-to-end khmer optical character recognition using sequence-to-sequence with attention. *arXiv:2106.10875*, 2021.

[12] K. Cho, B. van Merrienboer, C. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. *arXiv:1406.1078*, 2014.

[13] H. Riza, M. Purwoadi, Gunarso, T. Uliniansyah, A. A. Ti, S. M. Aljunied, L. C. Mai, V. T. Thang, N. P. Thai, V. Chea, R. Sun, S. Sam, S. Seng, K. M. Soe, K. T. Nwet, M. Utiyama, and C. Ding. Introduction of the asian language treebank. In *Proc. Conf. Oriental Chapter Int. Comm. Coord. Standardization Speech Databases Assess. Tech. (O-COCOSDA)*, pages 1–6, 2016.

[14] R. Buoy, N. Taing, S. Chenda, and S. Kor. Khmer printed character recognition using attention-based seq2seq network. *HCMCOU J. Sci., Eng. Technol.*, 12(1):3–16, 2022.

[15] V. Nom, S. Keo, S. Bakkali, M. M. Luqman, M. Coustaty, and J.-M. Ogier. Cross-lingual learning for low-resource khmer scene text detection and recognition. In *Proc. Int. Conf. Document Anal. Recognit. (ICDAR) Workshops*, Wuhan, China, September 2025.

[16] G. Jocher and J. Qiu. Ultralytics YOLO11, 2024. Version 11.0.0.

[17] V. Nom, S. Keo, S. Bakkali, M. M. Luqman, M. Coustaty, M. Rossinyol, and J.-M. Ogier. Wildkhmerst: A comprehensive dataset and benchmark for khmer scene text detection and recognition in the wild. In *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Wuhan, China, September 2025.

[18] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft COCO: Common objects in context. *arXiv:1405.0312*, 2015.

[19] U. Pal, A. Halder, P. Shivakumara, and M. Blumenstein. A comprehensive review on text detection and recognition in scene images. *Artif. Intell. Appl.*, 2(4):1–5, 2024.

[20] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. EAST: An efficient and accurate scene text detector. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5551–5560, 2017.

[21] Y. Baek, B. Lee, S. Yun, D. Han, S. Kim, and J. Kim. Character region awareness for the detection of text in the wild. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 9365–9374, 2019.

[22] R. Smith. An overview of the tesseract ocr engine. In *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, pages 629–633, 2007.

[23] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(11):2298–2304, 2017.

[24] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai. Robust scene text recognition with automatic rectification. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4168–4176, 2016.

[25] B. Shi, X. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai. ASTER: An attentional scene text recognizer with flexible rectification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(9):2035–2048, 2019.

[26] C. Li, X. Tian, C. Shen, C. He, and L. Zhang. Show, attend and read: A simple and strong baseline for irregular text recog-

nition. In *Proc. AAAI Conf. Artif. Intell.*, volume 33, pages 8610–8617, 2019.

[27] M. Li, Y. Lu, F. Zhai, H. Yin, Y. Liu, X. Li, D. Yu, and W. Gao. TrOCR: Transformer-based optical character recognition with pre-trained models. *arXiv:2109.10282*, 2021.

[28] SoyVitou. 62k-images-khmer-printed dataset. Hugging Face, n.d. Dataset.

[29] Keatchakravuth. Khmer annotation. Kaggle, n.d. Dataset.

[30] A. Dutta and A. Zisserman. The VIA annotation software for images, audio and video. In *Proc. ACM Int. Conf. Multimedia*, pages 2276–2279, 2019.

[31] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010.

[32] OCR-D. Ocr-d evaluation specification: ocrd_eval. Online, 2023. Accessed: Sep. 10, 2025.

[33] VISAi. Evaluation metrics for automatic speech and ocr. Online, 2024. Accessed: Sep. 10, 2025.

[34] Cloudraft. A comprehensive guide to ocr. Online, 2024. Accessed: Sep. 10, 2025.

A Study on Enhancing Performance of Marker-Based Augmented Reality for Low-end Smartphone

Yanghai Pov **Monioudom Mao** **Lihour Nov**
Department of Computer Science of Cambodia Academy of Digital Technology,
Phnom Penh, Cambodia
Yanghai.Pov@student.cadt.edu.kh

Abstract

Augmented reality (AR) lets people interact with virtual objects in the real world using their smartphones. However, most AR systems need powerful devices to run smoothly, which leaves out many users who have older or low-end phones. This research focuses on making marker-based AR work well on these less powerful smartphones. By using simple and fast algorithms to detect patterns and smart techniques to show 3D models efficiently, the system can recognize images accurately and run smoothly even on budget devices. Tests show that the system keeps good performance and image recognition accuracy, proving that AR can be made more accessible to a wider audience. This work aims to help bring the benefits of AR to more people around the world, regardless of the device they use.

Keywords: *Marker-Based Augmented Reality (AR), Features from Accelerated Segment Test(FAST), Oriented FAST and Rotated BRIEF (ORB)*

1 Introduction

Augmented Reality (AR) is a technology that overlays virtual objects onto the physical world using digital devices such as smartphones or tablets [1], [2]. This integration allows users to interact with both digital and real-world elements simultaneously, creating immersive experiences. It is increasingly applied in various fields such as education, entertainment, healthcare, and retail due to its ability to enhance visual and interactive content. AR systems generally work by capturing the real environment through a camera, detecting visual markers or features, understanding the spatial layout, and rendering 3D content in real time. The rendered virtual objects must align accurately with the real-world environment to ensure a stable and convincing experience. These processes require considerable computational resources, including real-time image processing and graphics rendering, which can be challenging for low-end mobile devices.

There are several types of AR, including Marker-Based AR, Markerless AR (i.e., GPS/Sensor-Based AR), Simultaneous Localization and Mapping (SLAM)-based AR, and Projection-Based AR. Among them, Marker-Based AR is the most lightweight and device-friendly[1]. It uses printed patterns (called markers) to help the system identify specific locations in the camera view, making it easier to place and track virtual content. Marker-Based AR is widely used in mobile applications, especially on devices with limited hardware. However, running AR applications on low-end smartphones remains difficult due to their limited CPU power, smaller RAM, weaker GPUs, and lower-quality cameras. These limitations often cause reduced frame rates, slow marker detection, and unstable rendering [3]. As AR becomes more widely adopted, it is important to develop lightweight and optimized solutions to ensure acceptable performance on these types of devices.

To address this issue, many researchers have explored efficient computer vision algorithms such as Features from Accelerated Segment Test (FAST) and Oriented FAST and Rotated BRIEF (ORB). FAST is a corner detection algorithm that quickly identifies keypoints in an image [4]. ORB builds on FAST by adding rotation-invariant descriptors using BRIEF (Binary Robust Independent Elementary Features) for feature matching [5]. Together, these algorithms provide a fast and effective method for real-time tracking on low-end hardware, without sacrificing too much accuracy. Several studies have shown that combining FAST and ORB improves AR performance. Rublee et al. introduced an optimized Marker-Based AR method using ORB and planar surface recognition [5]. It also used

OpenGL ES 2.0 with vertex rendering optimizations to improve 3D rendering performance. The result showed up to 90% faster recognition and stable frame rates of 16–24 FPS on low-end smartphones [6], [8]. Another study compared Marker-Based AR applications on different mobile devices and found that image resolution and processing speed were key factors affecting performance. Lower resolutions helped improve speed on older phones without significantly affecting detection quality [7], [9].While these studies have successfully improved AR performance on smartphones, most of them focused on mid- to high-end devices or did not evaluate performance differences across hardware levels. Additionally, they often optimized either the feature detection or the rendering process individually, rather than integrating both. As a result, their solutions may still struggle to deliver smooth, real-time AR performance on low-end smartphones with limited CPU and memory capacity.

To fill these gaps, this paper proposes an optimized Marker-Based AR framework called ModAR, which integrates lightweight computer vision and rendering techniques tailored for low-end smartphones. Specifically, the system combines the FAST and ORB algorithms to enhance marker detection speed and stability while minimizing computational load. The proposed method aims to improve real-time performance by maintaining a stable frame rate above 15 FPS, reducing latency, and ensuring smooth operation without requiring additional sensors or external hardware.

2 Methodology

This section will explain how the ModAR system works to recognize a pattern and overlay a 3D model onto it in real time, especially on low-end smartphones. ModAR is based on a marker-based AR (MAR) approach, where a known image called a pattern is used to place virtual objects in the real world through the phone's camera [8]. The system assumes the pattern is a flat, rectangular image. A top-down (nadir) image of this pattern is needed to initiate the AR process. In addition to the pattern image, the 3D model and camera calibration data (i.e., the camera's internal parameters) are required to ensure accurate projection.

The center of the pattern is defined as the origin point $(0, 0, 0)$ in the world coordinate system shows in **Figure 1**. The X and Y axes lie on the surface of the pattern, while the Z axis extends perpendicularly outward from the pattern toward the camera. To simplify calculations and maintain consistency across devices, the four corners of the pattern are normalized relative to the image dimensions. As a result, their X and Y values range from -1 to 1, and their Z values are set to 0. This normalization allows for easy transformation between image coordinates and real-world coordinates [9]. This setup provides a stable spatial reference frame, ensuring that virtual 3D objects are anchored correctly and appear aligned with the physical pattern when viewed through the camera in real time.

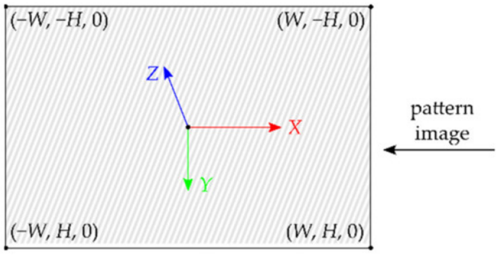


Figure 1: World coordinate system setup for the marker pattern [8].

2.1 Feature Extraction and Image Matching

To recognize the pattern in the real world, ModAR starts by detecting special visual features unique points like corners or edges—in the reference pattern image. During runtime, it looks for those same features in each new frame captured by the phone's camera. To do this efficiently, ModAR uses feature detection and description methods such as ORB, which combines the FAST algorithm for finding keypoints and the BRIEF algorithm for describing them, and BRISK with AGAST, where AGAST detects the keypoints and BRISK describes them in a way that works even when the image is rotated or scaled. After detecting features in both images, the system compares them using Hamming distance, which measures how different the binary descriptors are. To improve accuracy, it performs a cross check only keeping matches that are confirmed in both directions. It then filters out bad matches by removing those with large differences, and applies RANSAC (Random Sample

Consensus) to keep only the best matches that fit a realistic transformation of the image, helping the system determine where and how the pattern appears in the real world [8].

2.2 Pattern Recognition and Camera Pose Estimate

Once RANSAC finds at least 8 good matches, the pattern is considered detected, and a 3D model can be placed on it. RANSAC estimates a homography matrix that describes how the pattern image appears in the real-world camera view. This estimation is refined using the Levenberg–Marquardt algorithm, which improves accuracy. To understand how the camera is positioned and oriented relative to the pattern, the system calculates the camera's 6 degrees of freedom (6-DOF) pose—its position (X, Y, Z) and rotation in 3D space. This is done using a projection transformation matrix, which relates real-world points to their position in the image. The rotation matrix is improved using singular value decomposition (SVD) to ensure it is mathematically valid. Then it's converted into a more compact form using the Rodrigues formula, which helps during optimization [8].

2.3 3D Render

The 3D model used in ModAR is stored in the `.obj` file format, which includes information about the model's shape and its textures. To display the model, the system first sends the model data to the GPU using a vertex buffer. For every frame, it applies a set of transformation matrices to place and display the model correctly. These transformations are combined using (1).

$$\text{Final Position} = \text{Projection} \times \text{View} \times \text{Model} \times \text{Vertex} \quad (1)$$

To make the rendering more efficient, the system compresses texture files using techniques like ETC1 and PVRTC, which reduce memory usage. It also uses a method called frustum culling, which ensures only the parts of the model that are visible in the camera view are drawn—saving processing power. Additionally, when the same model needs to appear multiple times or is animated, the system uses geometry instancing. This allows it to draw many copies of the same model with just one command, which helps improve performance[9]. The entire rendering process runs on a lightweight graphics engine that uses OpenGL ES 2.0, making it suitable even for mobile devices with limited hardware[10].

2.4 AR Integration

For AR to work properly, the virtual model must align correctly with the real world. This is called registration. The system ensures the 3D model sits at the center of the pattern, facing the camera along the Z-axis. Key elements needed for AR include the estimated camera position and orientation, the camera calibration matrix, the pattern image, and the 3D model and rendering functions. To coordinate rendering and detection processes, semaphores (synchronization tools) are used. These help avoid overlapping memory usage and wasted computing time. When a pattern is detected, the AR system calculates a model matrix based on the camera's pose and applies the appropriate projection matrix to render the model in real-time. The projection matrix is adapted to match OpenGL requirements, ensuring the 3D model appears correctly on screen. Clipping planes, field of view, and aspect ratio are also considered to avoid rendering errors due to depth miscalculations. Finally, shaders are used to define how the model looks (lighting, color, texture), and the rendering engine builds the final AR scene by combining all these components[10].

3 Implementation

The ModAR prototype uses OpenCV (C++ via Android NDK) for computer vision and OpenGL ES 2.0 for graphics through the Android SDK. Java communicates with the native C++ code via JNI, with the C++ compiled into a shared library using CMake. The 3D engine handles vertices, models, textures, and shaders[8].

Pattern detection and 3D rendering run simultaneously on separate threads, sharing camera position, orientation, and recognition status. Semaphores ensure this shared data is safely managed.

For each camera frame, the app runs cameraPoseEstimation, which detects features, matches them to the pattern, and calculates the camera's position and orientation. To reduce flickering from motion blur or noise, the model is displayed only after two consecutive frames have enough matches [9].

The 3D rendering runs on its own thread and is initialized when the graphics engine starts. The renderer manages things like depth testing (to correctly show which objects are in front), culling back faces (to improve performance), and setting up the viewport when the screen changes. It loads 3D models, textures, and sets up the camera using various helper classes. Lighting and model-view matrices are also prepared to display the scene properly[10].

The Java side of the app includes classes like JavaCameraView, which handle camera operations getting the camera's projection matrix, intrinsic parameters, and capturing frames to send to the pose estimation process. The AndroidCameraView class converts camera frames into a format usable by the native code. Finally, the MainARActivity manages permissions, sets up views, and controls the overall AR experience.

Table 1: Specifications of low-end and mid-range test devices.

Specification	Low-End	Mid-Range
Model	Galaxy Ace 2	ZTE Blade A5
Year	2012	2019
CPU	800 MHz dual-core	1.6 GHz octa-core
Chipset	ARM Cortex-A9	Spreadtrum SC9863A
GPU	ARM Mali 400	IMG8322
Storage	4 GB	16 GB
Memory	768 MB RAM	2 GB RAM
Camera	5 MP	13 MP
Video	720×1280, 30 FPS	1080×1920, 30 FPS
Screen Res.	480×800 px	720×1440 px

4 Experimental Results and Discussion

4.1 Experimental Setup

To evaluate ModAR's performance, a series of experiments were conducted using two Android smartphones: one older low-end device and one affordable mid-range device. The aim was to understand how hardware specifications impact the system's ability to detect patterns and render 3D models in real time. Both devices supported OpenGL ES 2.0 and ran at least Android 4.4 KitKat (API level 19). The specifications of these devices are presented in **Table 1**. The testing used six pattern images, whose dimensions and resolutions are presented in **Table 2** and four 3D models included with the ModAR prototype, presented in **Table 3**. The pattern images were everyday objects such as book covers, a board game box, a framed wall painting, and a graffiti mural, with resolutions ranging from 0.07 to 0.25 megapixels. The camera resolution was 0.92 MP (720×1280 pixels) on the low-end device and 2.07 MP (1080×1920 pixels) on the mid-range device, both running at 30 frames per second. The 3D models came from various sources like image-based photogrammetry and laser scanning, and they varied in size and complexity. "POTTERY" and "BUST" were small models, "STATUE" was a medium-sized model with vertex color instead of texture, and "CHURCH" was a large, highly detailed model. This setup allowed for a detailed performance assessment of ModAR's detection and rendering processes across different device capabilities[8].

Table 2: Dimensions and resolution of the pattern images used in the experiments.

Pattern Image	Dimensions (px)	Resolution (MP)
BOTERO	348 × 717	0.25
SCYTHE	300 × 245	0.07
SURVEYING	310 × 450	0.14
JEFFERS	250 × 294	0.07
PAINTING	336 × 252	0.08
GRAFFITI	400 × 347	0.14

Table 3: Size, faces, and vertices of the 3D models used in the experiments.

3D Model	Size (KB)	Faces / Vertices
POTTERY	167	3296 / 1661
BUST	903	8767 / 4555
STATUE	1690	21,453 / 42,712
CHURCH	87	14,645,744 / 7,335,148

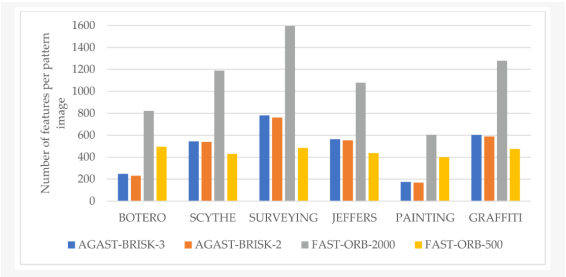


Figure 2: Number of feature points for each pattern image used in the experiments extracted by the tested detectors with different parameterization [8].

Table 4: Parameterization evaluated in the experiments performed using the AGAST detector and the BRISK descriptor.

Experiment	**AGAST-BRISK-3**	**AGAST-BRISK-2**
AGAST detection threshold score	30	30
Number of octaves	3	2
Scale applied to the pattern used for sampling the neighborhood of each feature	1	1

Table 5: Average computational time and percentage of CPU usage for ModAR prototypes with low-end device.

Task	**AGAST-BRISK-3**		**AGAST-BRISK-2**		**FAST-ORB-2000**		**FAST-ORB-500**	
	Time (s)	CPU (%)	Time (s)	CPU (%)	Time (s)	CPU (%)	Time (s)	CPU (%)
Feature detection	2.425	49.47	2.296	47.73	0.086	12.51	0.069	12.02
Feature description	2.337	45.91	2.229	46.69	0.230	32.09	0.189	34.29
Image matching	0.029	0.53	0.026	0.43	0.020	2.82	0.016	3.20
Homography estimation	0.089	1.49	0.063	1.31	0.098	19.37	0.035	6.30

4.2 Image Matching and Pattern Recognition

Two main types of algorithms for detecting features in images such as AGAST-BRISK and FAST-ORB were tested. There are four different setups, including AGAST-BRISK-2, AGAST-BRISK-3, FAST-ORB-500, and FAST-ORB-2000. Among these, FAST-ORB-2000 performed the best overall. It offered a good balance between speed and accuracy, worked well on both low-end and mid-range phones, and was able to recognize patterns even if they were rotated or smaller. FAST-ORB-2000 took a bit longer—around 0.4 seconds per frame on the low-end device and 0.35 seconds on the mid-range but gave more reliable and consistent pattern detection[8]. The AGAST-BRISK parameterization evaluated in the experiments is shown in **Table 4**. The number of feature points extracted for each pattern by the different detectors is illustrated in **Figure 2**.

The CPU and time usage results showed that the AGAST-BRISK methods used nearly 95% of the CPU, which is very high and not efficient for mobile devices. In contrast, the FAST-ORB methods only used about 50% of the CPU, making them much more efficient for real-time use. The observed efficiency of FAST-ORB compared to AGAST-BRISK can be explained by its algorithmic design. FAST detects corners using simple intensity comparisons around a pixel, avoiding the complex multi-scale keypoint detection required by AGAST. ORB then uses the lightweight BRIEF descriptor, which encodes features as binary strings for fast Hamming-distance matching. In addition, FAST-ORB does not require the extensive sampling of multiple scales and orientations like AGAST-BRISK, which significantly reduces computation. In contrast, AGAST-BRISK involves more computationally intensive operations for both detection and description, increasing CPU load. This combination makes FAST-ORB significantly lighter and more suitable for low-end devices. The pattern recognition was almost perfect, reaching close to 100% accuracy, as long as the pattern appeared in the camera frame at least half as large as its original size.This information is presented in **Table 5** for the low-end device and in **Table 6** for the mid-range device. This means the system works best when the pattern was reasonably visible and not too small[8].

The best method for pattern recognition in ModAR is FAST-ORB-2000 because it is fast, stable, and highly accurate. The AR system performs well with small and medium-sized 3D models, even on older smartphones. However, when using large 3D models, the performance decreases due to limitations in the GPU and memory. To ensure smooth operation on low-end devices, optimization techniques such as instancing and grouping Vertex Buffer Objects (VBOs) are crucial. These optimizations help the system run efficiently and provide a better AR experience on less powerful phones.

The AR system ran at 16 to 24 frames per second (FPS), which is generally acceptable for real-time use. Smaller and medium-sized models, like “POTTERY” and “STATUE,” were ren-

Table 6: Average computational time and percentage of CPU usage for ModAR prototype with mid-range device.

Method	AGAST-BRISK-3		AGAST-BRISK-2		FAST-ORB-2000		FAST-ORB-500	
	Time (s)	CPU (%)	Time (s)	CPU (%)	Time (s)	CPU (%)	Time (s)	CPU (%)
Feature detection	1.926	49.52	1.851	47.70	0.069	14.21	0.057	10.33
Feature description	1.795	45.50	1.656	46.51	0.194	37.46	0.155	35.25
Image matching	0.027	0.53	0.023	0.41	0.070	13.94	0.012	3.40
Homography estimation	0.080	1.50	0.050	1.28	0.023	3.89	0.026	5.30

dered quickly and smoothly on both devices. However, the large "CHURCH" model caused noticeable slowdowns, especially on the low-end phone. To improve performance, optimization techniques such as instancing which allows drawing many copies of the same object in a single call and organizing the models into a single memory buffer called a Vertex Buffer Object (VBO). These optimizations improved rendering performance by up to 3.7 times on low-end devices[8].

GPU performance and CPU profiling results are presented in **Figure 3**, and memory profiling for the large 3D model "CHURCH" is shown in **Figure 4**. These results highlight that while CPU usage was significant, the main challenge lay in GPU memory and rendering. By applying optimization strategies, it became possible to achieve smoother frame rates and reduce resource consumption, even on constrained hardware. This demonstrates the importance of efficient GPU memory management for making marker-based AR systems practical on low-end smartphones. Furthermore,these findings confirm that targeted optimizations can significantly extend the usability of AR applications on budget devices without requiring additional hardware support.

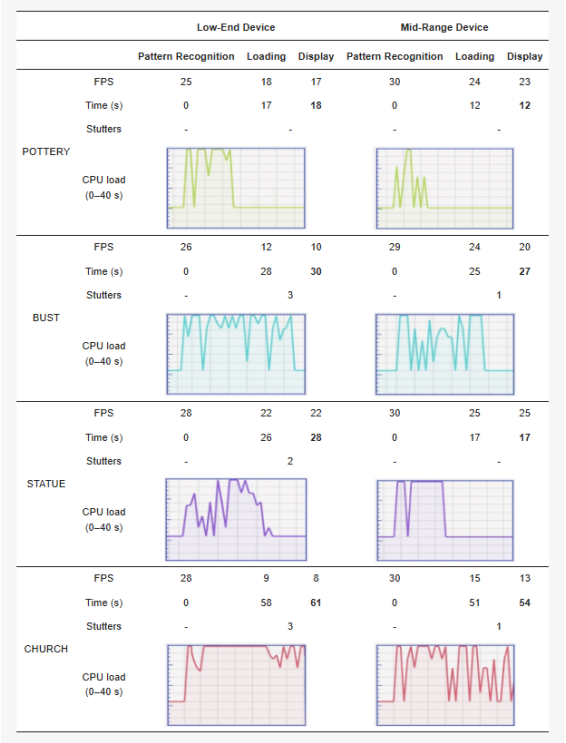


Figure 3: GPU performance and CPU profiling results during the graphics computations of four AR sessions [8].

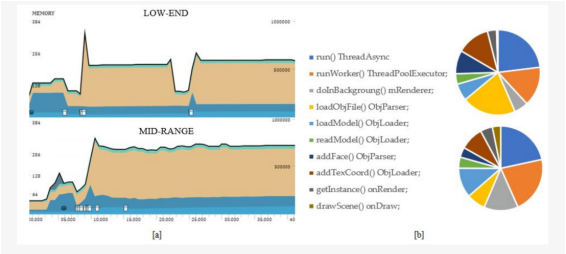


Figure 4: Memory profiling of the "CHURCH" 3D Model: [a] 40s timeline upon pattern recognition with the memory allocation of the main memory types; [b] graphics objects and classes with the most heap count [8].

5 Conclusion

In conclusion, this review demonstrates that marker-based augmented reality can work effectively on low-end smartphones despite their limited processing power and hardware. Efficient algorithms such as FAST, ORB, OpenGL ES 2.0, geometry instancing, and frustum culling enable AR systems to achieve good accuracy and smooth performance. The research also highlights the importance of designing AR systems that account for device diversity and user needs. Focusing on lightweight methods and performance improvements can help marker-based AR reach everyone, bridging the digital divide and

expanding opportunities for users globally.

References

[1] R. Azuma, "A survey of augmented reality," *Presence: Teleoperators Virtual Environ.*, vol. 6, no. 4, pp. 355–385, Aug. 1997, doi: 10.1162/pres.1997.6.4.355.

[2] S. R. Fan and S. Y. Lin, "Augmented reality for mobile devices: A survey," *IEEE Access*, vol. 8, pp. 136–153, 2020, doi: 10.1109/ACCESS.2020.2965539.

[3] M. Billinghurst, A. Clark, and G. Lee, "A survey of augmented reality," *Found. Trends Hum. Comput. Interact.*, vol. 8, no. 2–3, pp. 73–272, 2015, doi: 10.1561/1100000049.

[4] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2006, pp. 430–443, doi: 10.1007/11744023_34.

[5] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2011, pp. 2564–2571, doi: 10.1109/ICCV.2011.6126544.

[6] X. Xu, D. Chen, J. Ren, and B. Dang, "Research on augmented reality method based on improved ORB algorithm," *J. Phys. Conf. Ser.*, vol. 1453, no. 1, p. 012024, Jan. 2020, doi: 10.1088/1742-6596/1453/1/012024.

[7] S. Irawan, M. Suhartono, R. R. Suryanegara, and D. Suryani, "Performance evaluation of marker-based augmented reality applications on low-end and high-end mobile devices," in *Proc. Int. Conf. ICT Smart Soc. (ICISS)*, 2019, pp. 1–6, doi: 10.1109/ICISS48059.2019.8969765.

[8] S. Verykokou, C. Ioannidis, and G. Kambourakis, "Mobile augmented reality for low-end devices based on planar surface recognition and optimized vertex rendering," *J. Phys. Conf. Ser.*, vol. 1453, no. 1, p. 012024, Jan. 2020, doi: 10.1088/1742-6596/1453/1/012024.

[9] M. Fernández, R. Martínez, L. Pérez, A. León, and D. Díaz, "Performance evaluation of marker-based augmented reality applications on mobile devices," in *Proc. 16th Int. Conf. Ubiquitous Comput. Commun. (IUCC)*, Granada, Spain, Dec. 2017, pp. 623–630, doi: 10.1109/IUCC.2017.00117.

[10] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*, Newton, MA, USA: O'Reilly Media Inc., 2017.

IoT-enabled Real-Time Water Level Monitoring System for Early Flash Flood Detection

Dariya Kong **Sovanmonich Chea** **Hengtong Lim** **Sovanndara Am** **Lihour Nov**
Department of Telecommunications and Networking of Cambodia Academy of Digital Technology, Phnom Penh, Cambodia
dariya.kong@student.cadt.edu.kh

Abstract

Flash flood is a significant natural hazard in Cambodia, causing severe damage to infrastructure and posing a substantial risk to human life. Its sudden and unpredictable occurrence, intensified by climate change, underscores the need for reliable real-time monitoring and early warning systems. This paper presents an Internet of Things (IoT)-based solution that employs multiple sensors to capture critical environmental data, including water level, rainfall, and water flow. The system is designed with an emphasis on accuracy, durability for outdoor conditions, and cost-effectiveness, making it suitable for deployment in vulnerable communities. Sensor data are transmitted to a cloud platform for storage and analysis, while a real-time notification mechanism delivers early alerts to users, enabling them to respond promptly to potential flood events. The proposed prototype provides a practical and affordable approach to enhancing community preparedness, awareness, and disaster risk reduction in flood-prone regions.

Keywords: *Flash flood, IoT, early warning*

1 Introduction

Flooding remains one of Cambodia's most destructive natural hazards, affecting urban and rural populations during the rainy season. The 2022 floods affected 14 provinces, demonstrating the country's vulnerability to sudden flood events [1]. Urban areas are particularly prone to street-level flooding when intense rainfall overwhelms drainage systems, while rural and riverine regions can experience rapid river surges that threaten communities near tributaries such as the Stoeng Prek Thnaot Basin [2], [3]. In Phnom Penh city, intense rainfall, specifically flash flood, often overwhelms drainage infrastructure and causes street-level inundation, disrupting transportation and damaging property [1]. Figure 1 shows urban flash flooding in Phnom Penh that illustrates the rapid onset and local impacts of such events.

Figure 1: Street-level flash flooding in Phnom Penh, Cambodia [4].

Additionally, riverine and basin-scale flooding are also recurrent problems, especially in tributaries such as the Stoeng Prek Thnaot Basin. Hydrological studies of the Prek Thnaot basin highlight changing water-balance regimes and increased extremes under climate-change scenarios, underscoring the need for localized monitoring and early warning [3], [7]. Existing flood monitoring solutions in Cambodia and neighbouring countries rely primarily on manual water-level measurements or basic sensor networks [2], [5]. Although these approaches provide some awareness, they often lack real-time monitoring, predictive capabilities, and cost-effectiveness, which limits their usefulness for widespread deployment. To address these gaps, this paper proposes an integrated, real-time flash flood monitoring system based on Internet of Things (IoT) technology. The system collects key environmental data such as water level, rainfall, and water flow and delivers real-time alerts through a cloud-based notification mechanism.

The objective of this study is to develop a practical, durable, and cost-effective IoT system suitable for communities in flood-prone areas, with a pilot implementation focused on the Sto-

eng Prek Thnaot Basin. The development prototypes compose of Node MCUs and various sensors to keep track on the current water level. The system will provide alert notifications via mobile app when serious flooding is happening from the real-time IoT system.

2 Methodology

Prototype systems utilize ESP32 Node MCU, sensors, and cloud platforms to enable near-real-time data acquisition and notification [2], [8], [9]. Reviews of digital innovations for flash-flood early warning observe two clear trends including, increased adoption of cloud-backed IoT sensor networks for situational awareness, and expanding use of AI/ML for prediction [6], [10]. These studies suggest that a pragmatic approach is to pair robust, low-cost sensing with cloud messaging and simple local alerts, reserving predictive analytics for later enhancement. Despite progress, important gaps remain for practical deployment in Cambodia, specifically many prototypes are validated only under controlled conditions rather than diverse field settings and the assumption of good internet connectivity. This consideration reduces reliability in areas with intermittent service, and the issues of cost and long-term durability are not always fully addressed [2], [5].

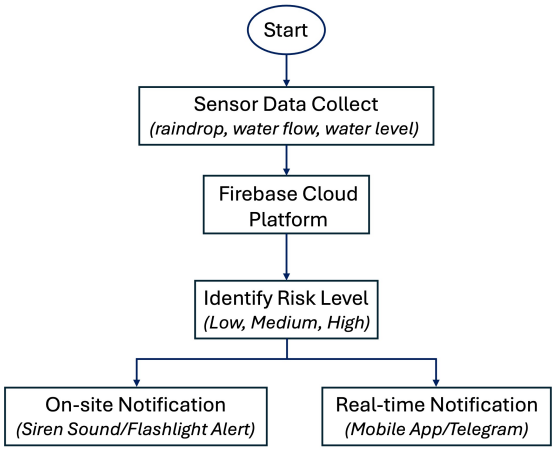


Figure 2: A flow diagram of the system operational process.

The proposed flood monitoring system is designed to provide real-time and reliable alerts for flood-prone areas by combining on-site data collection, wireless transmission, cloud integration, and two-way notification methods. Figure 2 shows the operational design of our system to handle detection and notification tasks. The transmitter node integrates three main sensors including raindrop sensor, water flow sensor, and an ESP32-CAM for capturing water level images. The receiver node serves as a gateway to relay the sensing data to be stored and processed with Firebase Cloud Messaging to enable near real-time notification via mobile or web applications. Simultaneously, the receiver triggers offline alerts, such as buzzers or flashing lights, to ensure immediate warnings even without internet access. This dual-alert mechanism guarantees timely and reliable flood notifications to users under all conditions.

3 Design and Implementation

The system is designed to provide real-time monitoring of flood-prone areas and deliver early alerts to users. It consists of two main units a transmitter and a receiver shown in Figure 3. The transmitter collects environmental data using sensors, while the receiver processes the data, stores it in the cloud, and triggers alerts. Wireless communication between the units is handled via the ESP-NOW protocol, and Firebase is used for cloud-based monitoring and online notifications. The core components of the proposed system include the ESP32 microcontroller, which serves as the central controller for data acquisition and wireless communication, and an ESP32-CAM module for capturing water-level images. Environmental sensing is carried out using a raindrop sensor to detect rainfall and a water flow sensor to measure stream or drainage velocity. The receiver unit integrates with Firebase for cloud storage and notification services, while local alert devices such as buzzers or flashing lights ensure immediate offline warnings. A rechargeable battery or solar power source sustains continuous outdoor operation, making the system suitable for deployment in remote or flood-prone areas.

To demonstrate the cost-effectiveness of the proposed prototype, Table 1 summarizes the component-level cost breakdown in USD. The total cost of the system is approximately $33, which is significantly lower than previously reported IoT-based flood monitoring systems costing $100–$150 per unit [2; 8]. This cost-efficient design supports the claim that the proposed system is low-cost and practical for community-

Table 1: Cost breakdown of the proposed IoT flood monitoring prototype (USD).

Component	Quantity	Unit Cost (USD)	Total Cost (USD)
ESP32 NodeMCU	1	5	5
ESP32-CAM	1	10	10
Raindrop Sensor	1	2	2
Water Flow Sensor	1	5	5
Battery / Power	1	8	8
Enclosure / Box	1	3	3
Total			**33**

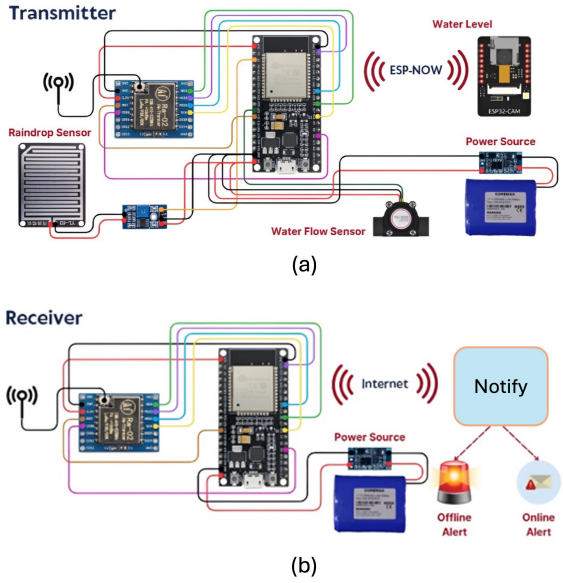


Figure 3: Design schematic: (a) Transmitter node, (b) Receiver node.

level deployment.

The system continuously monitors environmental conditions at flood-prone locations. The receiving sensor data will be transferred to the Firebase cloud database through an internet connection, enabling real-time remote monitoring. From this data, we can analyze the risk levels and generate appropriate online notifications via Firebase Cloud Messaging on mobile or web applications. Simultaneously, the receiver triggers offline alerts, such as buzzers or flashing lights, to provide immediate warnings even without internet connectivity. This dual-alert mechanism ensures timely and reliable flood notifications under diverse operating conditions. Figure 4 shows the complete design of the prototype enclosed in a plastic box for outdoor deployment purposes.

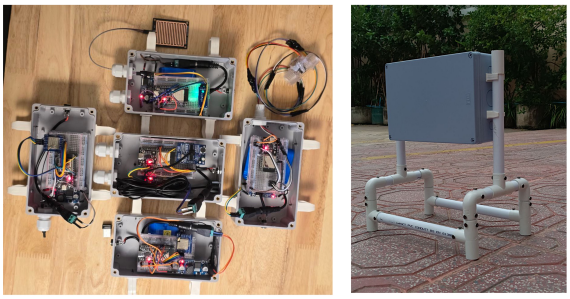


Figure 4: Prototype development: (a) sensor node boxes, (b) complete prototype.

4 Results and Discussion

The prototype was developed and tested to confirm that all core sensors were functioning correctly. Test scenarios simulated heavy rainfall, rising water levels, and varying flow rates in local drainage channels. In all cases, the system generated accurate and timely alerts, demonstrating its effectiveness for real-time monitoring and early warning in both urban and rural areas. The results confirm that the system provides real-time situation awareness and reliable early flood warnings.

Figure 5 presents the web-based dashboard interface designed for real-time monitoring of environmental data from the sensor nodes. The dashboard provides a centralized platform where water level, rainfall, and flow rate measurements are visualized through graphical charts and numerical indicators. This visualization facilitates user-friendly access to critical information, enabling stakeholders to track flood conditions remotely and respond effectively. Furthermore, the dashboard integrates alert notifications, thereby enhancing situational awareness and supporting timely decision-making in flood-prone areas.

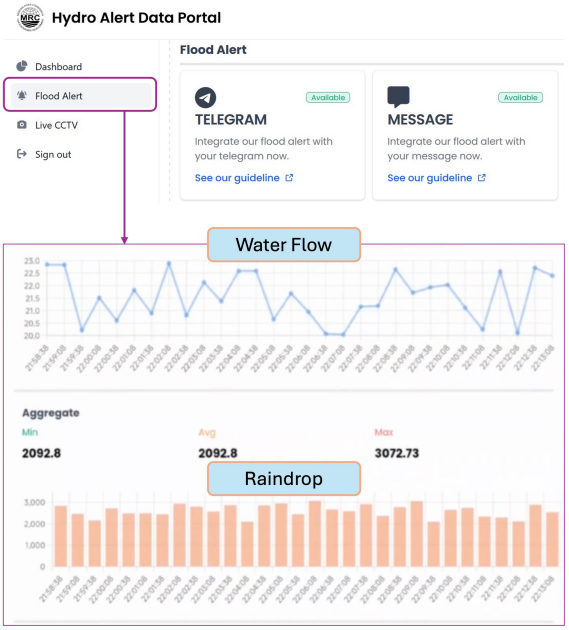


Figure 5: Web dashboard illustrating the feature and visualization of data monitoring for sensor node.

5 Conclusion

This study presented an IoT-based flash flood monitoring and early warning system that integrates an ESP32 microcontroller, ESP32-CAM, rainfall sensor, a water flow sensor, and cloud services through Firebase. The prototype demonstrated reliable data acquisition, real-time transmission, and a dual-alert mechanism that ensures both online and offline notifications, making it suitable for deployment in flood-prone areas. Its low cost, durability, and continuous operation capability highlight its potential as a practical solution for community-level disaster preparedness.

6 Future Works

Future improvements include integrating predictive algorithms for enhanced forecasting, and optimizing power consumption to extend deployment duration and improve sustainability for outdoor operation. The enhancement of the system through predictive analytics and broader system integration. Specifically, artificial intelligence (AI) and machine learning (ML) techniques will be incorporated to enable forecasting capabilities, moving beyond the current threshold-based detection approach. By leveraging historical hydrological records alongside real-time sensor inputs, the system can generate more accurate flood predictions and provide earlier warnings. These advancements will improve system resilience and strengthen its role as a comprehensive solution for flood risk reduction.

Acknowledgment

The project was funded by the Institute of Digital Technology of the Cambodia Academy of Digital Technology. Additionally, the authors would like to thank Mr. ChannUdam Ray, Ms. Panhanyta Phal, Mr. Whatanak Kean, and Mr. Doung Ngob for their contribution to this project.

References

[1] S. R. Phy, "Flood hazard and management in Cambodia: A review of activities, knowledge gaps, and research direction," Climate, vol. 10, no. 11, p. 162, Nov. 2022.

[2] C. Prakash, et al., "An IoT-based system for monitoring and forecasting flash floods," J. Earth Syst. Sci., vol. 132, no. 3, pp. 1–12, 2023.

[3] L. Saravuth, Cambodia. Los Angeles, CA, USA: Public Works Research Institute, 2006. [Online]. Available: https://www.pwri.go.jp/icharm/training/2006-pdf/saravuth_cambodia.pdf. (Accessed: Sep. 16, 2025)

[4] EAC News, "Phnom Penh streets flooded after heavy rainfall," EAC News, Aug. 2023. [Online]. Available: https://eacnews.asia/home/details/23588. (Accessed: Sep. 16, 2025)

[5] Prek Thnot River Basin. Los Angeles, CA, USA: Public Works Research Institute, 2007. [Online]. Available: https://www.pwri.go.jp/icharm/training/pdf2007/07_proposal_report/cambodia.pdf. (Accessed: Sep. 16, 2025)

[6] G. Al-Rawas, A. Haddad, and S. M. Al-Saadi, "A critical review of emerging technologies for flash flood early warning," Water, vol. 16, no. 14, pp. 1–20, Jul. 2024.

[7] S. R. Phy, "Integration of hydrological and flood inundation models for assessing flood events in the lower Prek Thnot River Basin

under climate change," J. Hydrol. Eng., vol. 27, no. 5, pp. 1–12, May 2022.

[8] G. A. L. Kumaran and J. Lias, "IoT-based flood monitoring system using ESP32," Evolution in Electrical and Electronic Engineering (EEEE), vol. 5, no. 1, pp. 470–478, Apr. 2024.

[9] R. Boudville, "IoT-enabled monitoring: ESP32 and Firebase implementations for water-level and environmental monitoring," UiTM Institutional Repository, 2025. [Online]. Available: https://ir.uitm.edu.my/112679/1/112679.pdf. (Accessed: Sep. 16, 2025)

[10] H. Zhang, et al., "Developing real-time IoT-based public safety alert and response systems," Sci. Rep., vol. 15, no. 11234, pp. 1–10, 2025.

[11] Khmer Times, "Cambodia ranks 4th globally for flood risk, 23rd for heat," Khmer Times, May 28, 2025. [Online]. Available: https://www.khmertimeskh.com/501690816/cambodia-ranks-4th-globally-for-flood-risk-23rd-for-heat/. (Accessed: Sep. 16, 2025)

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.

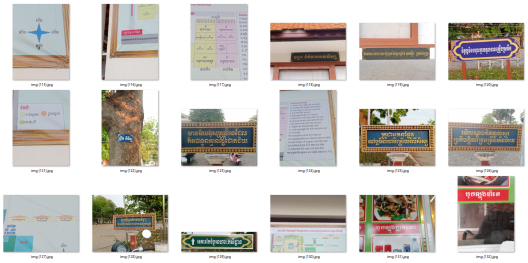


Figure 1: Example of Khmer Sense Text

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.

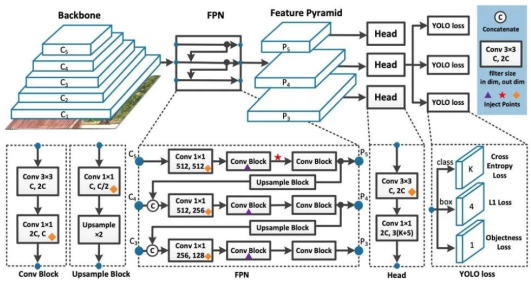


Figure 2: YOLOv8 architecture [28].

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

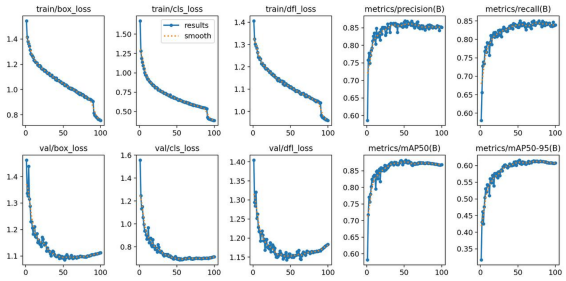


Figure 3: Multi-plot figure of Aksor V3

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

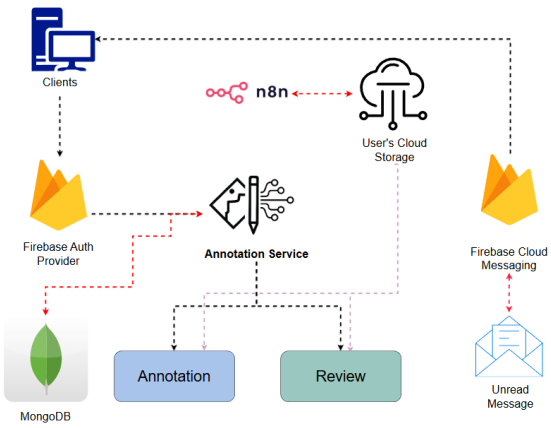


Figure 4: System flow diagram of JOMNAM.

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.

A Comparative Analysis of Unimodal and Multi-Modal Architectures for the Robust Recognition of Khmer Consonant Hand Signs

Sereyratanak Heng **Sethisak San**
School of Digital Technology, American University of Phnom Penh, Phnom Penh, Cambodia
2023472heng@aupp.edu.kh

Abstract

Automated systems for under-resourced languages like Khmer Sign Language require real-world robustness, a factor often overlooked in favor of accuracy on clean data. This study evaluates deep learning models on both a large, curated dataset of 33 Khmer consonant signs and a manually created "Challenge Set" featuring realistic degradations. We systematically compared unimodal (Vision-Only, Skeleton-Only) and various multi-modal fusion architectures (MLP, LSTM, Attention). Our findings were decisive and counter-intuitive. A unimodal Skeleton-Only (LSTM) model was the most robust, achieving 81% accuracy on the Challenge Set. In stark contrast, all multi-modal fusion models, which combined skeletal data with features from a pretrained EfficientNetB0, underperformed significantly, with an advanced attention model collapsing to just 11% accuracy. We identify this failure as a critical case of "Modality Mismatch," where the brittle vision model produces erroneous, high-confidence features ("Confident Garbage") that degrade the fusion process. This work proves that for applications with a significant domain shift, a simpler, more robust unimodal model can be decisively superior to a complex multi-modal system, challenging the assumption that more data is always better.

Keywords: *Deep Learning, Machine Learning (ML), Convolutional Neural Network (CNN), Mediapipe, Sign Language, Classification*

1 Introduction

A person who is not able to hear as well as someone with normal hearing is said to have hearing loss, usually called deaf or hard-of-hearing. Those people use Sign language as the primary means of communication, yet the lack of widespread understanding often create significant barriers in education, healthcare and daily life. This communication gap can lead to social isolation, reduced opportunities, and limited access to essential services. Despite the serious of this issue, it is often overlooked and the public awareness is remaining low. However, only a small percentage of people are capable of using sign language, most of whom are either members of the deaf community or professionals working in the interpreting field. Hard-of-hearing refers to people with hearing loss ranging from mild to severe. It can start from one ear or both, leading to difficulty in hearing for conversation, speech, or loud sounds. The use of sign language by deaf and hard-of-hearing people plays a vital role in the interaction of communities around the world. According to the World Federation of the Deaf, there are approximately 70 million Deaf individuals worldwide (2024) [1] . There are more than 300 distinct sign languages around the world, as each county has its own grammar and lexicon that is inspired by their spoken language (UN, 2024) [2].

Figure 1. On the International Day of the Deaf at the Ministry of Social Affairs [3]

The Institute of Statistics of the Ministry of Planning has stated that there are 19,993 deaf or hard-of-hearing people, which is equivalent to 2.9% of the total population in Cambodia reported by the census [3].

According to the World Health Organization

(WHO, 2025) [4], the causes of hearing loss and deafness can be classified based on different stages of life. **Prenatally,** they may be due to genetic factors or infections such as rubella. **Perinatal** risks include birth asphyxia, jaundice, low birth weight, and related complications. In **childhood and adolescence**, chronic ear infections and meningitis are key contributors. In **adulthood and older age**, hearing loss may stem from chronic diseases, lifestyle factors, or age-related conditions. Across all stages, factors such as ear trauma, noise exposure, ototoxic drugs or chemicals, poor nutrition, and progressive genetic conditions can also lead to hearing impairment.

The deaf and hard-of-hearing (DHH) community share various experience of discrimination, stigma and prejudice from hearing people with other linguistically and culturally minority hearing groups in the United States and organizational networks [5]. For example, a DHH person may have trouble communicating with their hearing family members, suffer from bullying at school, or encounter a conflict between their own values as a DHH individual and the values and expectations of others in their environment [6].

Hearing loss impacts multiple dimensions of life, including individual, social, and societal levels. At the individual level, it leads to limitation in communication and speech development, along with adverse effects on cognitive functioning. On the social level, DHH individual frequently experience isolation, loneliness and stigma. For the societal level, DHH people encounter barriers in accessing education and employment, resulting in reduced workforce participation and increase economic burden. From a public health perspective, hearing loss substantially increases years lived with disability (YLDs) and contributes to disability-adjusted life years (DALYs), highlighting its serious global impact (World Health Organization, 2021) [7].

Assistive devices include hearing aids, and cochlear implants provide important support for people with DHH. However, these solutions are not universally accessible or effective for everyone because the World Health Organization estimates that only about 3% of the need for hearing aids has been provide, leaving a large portion of people without adequate support [8]. Olkin

Figure 2. UN providing hearing aid [8]

(2002) highlighted that many professional training sites lack adequate support for deaf individuals, noting that about 80% were missing essential tools like teletypewriter (TTY) systems, which hinder equal participation for trainees with hearing loss. [9].

In order to improve sign language learning for all groups, Dr. Naa claims that machine learning and related technologies can be used to translate sign language into words and vice versa [10].

Despite the growing advancement of sign language recognition worldwide, Khmer Sign Language (KSL) remains a low-resource language, constrained by limited datasets and insufficient technological support. Currently, no robust KSL recognition system exists in Cambodia that can effectively translate gestures into text. This scarcity creates substantial barriers to accessibility and inclusion for the deaf community.

This research aims to address this gap by developing machine learning models capable of translating KSL into text. The proposed system seeks to advance sign language recognition in low-resource contexts, such a system is essential for promoting more equitable communication and participation in Cambodian society, with particular potential to enhance accessibility in education and other critical domains.

2 Literature review

2.1 Object detection

Sign language recognition has seen significant advancements in recent years with the integration of deep learning and real-time object detection techniques. Buttar et al. (2023) developed a hybrid approach for American Sign Language (ASL) recognition that effectively com-

bines YOLOv6 [11]—a state-of-the-art real-time object detection model—for static hand gesture detection, with Long Short-Term Memory (LSTM) [12] networks and MediaPipe [13] to recognize dynamic gestures [14]. This comprehensive solution bridges the gap between static and dynamic sign recognition, enhancing the overall system's flexibility and performance. In the domain of few-shot object detection, several innovative approaches have emerged to address the challenge of limited labeled data. The Meta-DETR framework proposed by Zhang et al. (2022) eliminates the need for traditional region proposal methods by leveraging image-level few-shot learning [15]. It is capable of detecting a wide variety of object categories such as animals, vehicles, household items, and tools, even with minimal training examples. This model emphasizes recognizing novel or unseen classes by learning semantic relationships between base and new categories. Similarly, Zhang et al. (2021) introduced a method for accurate few-shot object detection by incorporating support-query mutual guidance and hybrid loss functions. Their approach enhances detection performance for rare object classes by effectively exploiting the relationship between support and query images [16].

2.2 Sign Language Recognition

Focusing on sign language recognition in different linguistic contexts, Shenoy et al. (2021) developed a real-time Indian Sign Language (ISL) recognition system that uses a smartphone camera to identify 33 static hand poses and 12 dynamic gestures [17]. By applying grid-based feature extraction and a k-Nearest Neighbors (k-NN) classifier, their system achieved high accuracies of 99.7% for static poses and 97.23% for dynamic gestures, significantly aiding communication for individuals with hearing and speech impairments. In a comparative study, Kondo et al. (2024) analyzed the performance of Vision Transformers (ViT) [18] versus Convolutional Neural Networks (CNN) [19] for Japanese Sign Language recognition [20]. Their findings revealed that ViT models, particularly those using angular features, outperform traditional CNNs, offering valuable insights into the potential of transformer-based architectures in sign language applications. Daniels et al. (2021) proposed a recognition system for Indonesian Sign Language (BISINDO) that utilizes the YOLOv3 object detection algorithm [21]. The system demonstrated excellent performance, achieving 100% accuracy on static image data and 72.97% accuracy on dynamic video data, underscoring the promise of real-time object detection models for sign language recognition. Additionally, Dong et al. (2022) introduced Incremental-DETR, an extension of the DETR [22] architecture that integrates fine-tuning with self-supervised learning [23]. This method allows the model to learn new object classes with minimal labeled data while maintaining performance on previously learned base classes. It addresses the issue of catastrophic forgetting and overfitting, making it a robust solution for few-shot learning scenarios. Recent studies have shown that MediaPipe Holistic provides a reliable framework for extracting multimodal landmarks of the hands, face, and body, which significantly improves the performance of continuous sign language recognition models [24].

These studies collectively illustrate the rapid evolution of sign language and object detection technologies. They highlight the importance of integrating real-time detection, few-shot learning, and advanced deep learning models to build robust, accurate, and adaptable recognition systems.

3 Methodology

Our methodology is designed to rigorously test our model performance in not only the ideal scenarios but also real-world scenarios as well. In order to achieve this, we created two distinct datasets, a dual-stream data preprocessing pipeline, and a collection of model architectures for comparative analysis.

3.1 Dataset and Evaluation Strategy

Dataset classes

Since the Khmer alphabet consists of 33 letters, we created 33 separate classes, each class representing a different alphabet. The majority of hand shapes in the Khmer Sign Alphabet are characterized more by the position of the fingers than by their specific orientation. For instance, even when the hand is slightly rotated, the particular finger arrangement for "kor" keeps the hand shape visually different from other letters. For references to take images of each alphabet sign we used an app call 'Khmer

Figure 3. Simple Classes of Khmer Alphabet Sign Language from Krousar Thmey [25]

Sign Language' that is developed through collaboration between Research Triangle Institute (RTI) and Ministry of Education, Youth and Sport (MoEYS) under leadership and technical support from Department of Information and Technology (DIT), Special Education Department (SED), and National Institute for Special Education (NISE) [26]. In addition, the Khmer Sign Language chart developed by Krousar Thmey was used as a reference for standardizing the representations of 33 consonants (figure 3) [25]. These resources together served as the foundation for building a structured dataset, which can be applied to machine learning models for sign-to-text translation.

3.1.1 Primary Training Dataset

As no large-scale public dataset is available for Khmer Consonant Hand Signs, a primary dataset was manually collected and curated for this study. To facillitate the rapid collection of a large volume of data, we captured frames from a video stream and convert them into individual images. This method allowed us to assemble a comprehensive dataset of over 17,000 images distributed across 33 classes each representing a unique Khmer consonant. The data was collected under controlled and uniform condition, characterized by consistent lighting, a single person, and recorded at the same background. Standard visual references were use to ensure the accuracy of the performed signs. This dataset served as the foundation for training and validating our models in a controlled environment.

3.1.2 The "Challenge Set": A Real-World Robustness Benchmark

To evaluate model robustness beyond the controlled environment of the primary dataset, we also curated a "Challenge Set". This smaller but more difficult dataset was specifically collected and designed to simulate common real-world data degradation and represent a domain shift from the training data. The data consists of: (1) Images from people not present in the training set, introducing more variations in hand shape, size, and skin tone. (2) Image were captured in

Figure 4. Example of some classes

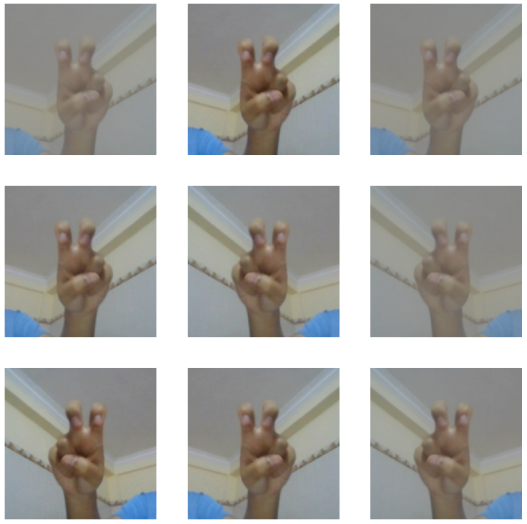


Figure 5. Example of Augmented Image

an environment with different lighting and background (3) Signs performed against busy, real-world backgrounds

By evaluating models on this challenge set as the central focus of our robustness analysis, it will reveal the true generalization capability of each architecture.

3.2 Multi-Modal Data Preprocessing

We implemented a two-stream preprocessing pipeline utilizing Google's MediaPipe HandLandmarker to extract the visual and skeletal features from each raw image.

Image Augmentation

3.2.1 Vision Stream Pipeline

The objective of the vision pipeline is to generate a focused, normalized image of the hand for the Convolutional Neural Network. (1) MediaPipe's HandLandmarker is used to detect the 21 keypoints of the hand in the full resolution image. (2) Bounding box is then calculated from the detected landmark and expanded with a 20-pixel padding to ensure the entire hand is caputred while minimizing the background noise. (3) The original image is then cropped based on the bounding box and resized to a uniform 224x224 pixels. (4) The resized image is processed using the specific preprocess_input function corresponding to its CNN backbone(EfficientNetB0) to scale the pixel values to the expected range. (5) During training, Image augmentation will be applied to the dataset to increase the variation in the Dataset. The image augmentation techniques used for this current model are Horizontal Flipping, Random Contrast, and Random Brightness as shown in figure 5.

3.2.2 Skeleton Stream Pipeline

The skeleton pipeline extracts a quantitativem geometric representation of the hand's pose. (1) The normalized 3D world coordinate (x, y, z) of the 21 keypoints are extracted from MediaPipe output. (2) The landmarks are structured into a tensor with the shape of (21, 3) in order to make it suitable for models designed to process sequential data.

3.3 Model Architectures

To systematically evaluate the different learning strategies for Khmer Consonant Hand Sign Recognition, We compared a collection of unimodal and multi-modal architectures. All multimodal models share a common two-stream structure consisting of a Vision Branch to process image data and a Skeleton Branch to process landmark data, which are then combined by a fusion mechanism.

3.3.1 Baseline Multi-Modal Architecture (CNN + MLP)

Our baseline fusion architecture, illustrated in Figure 6, serves as the foundation for our com-

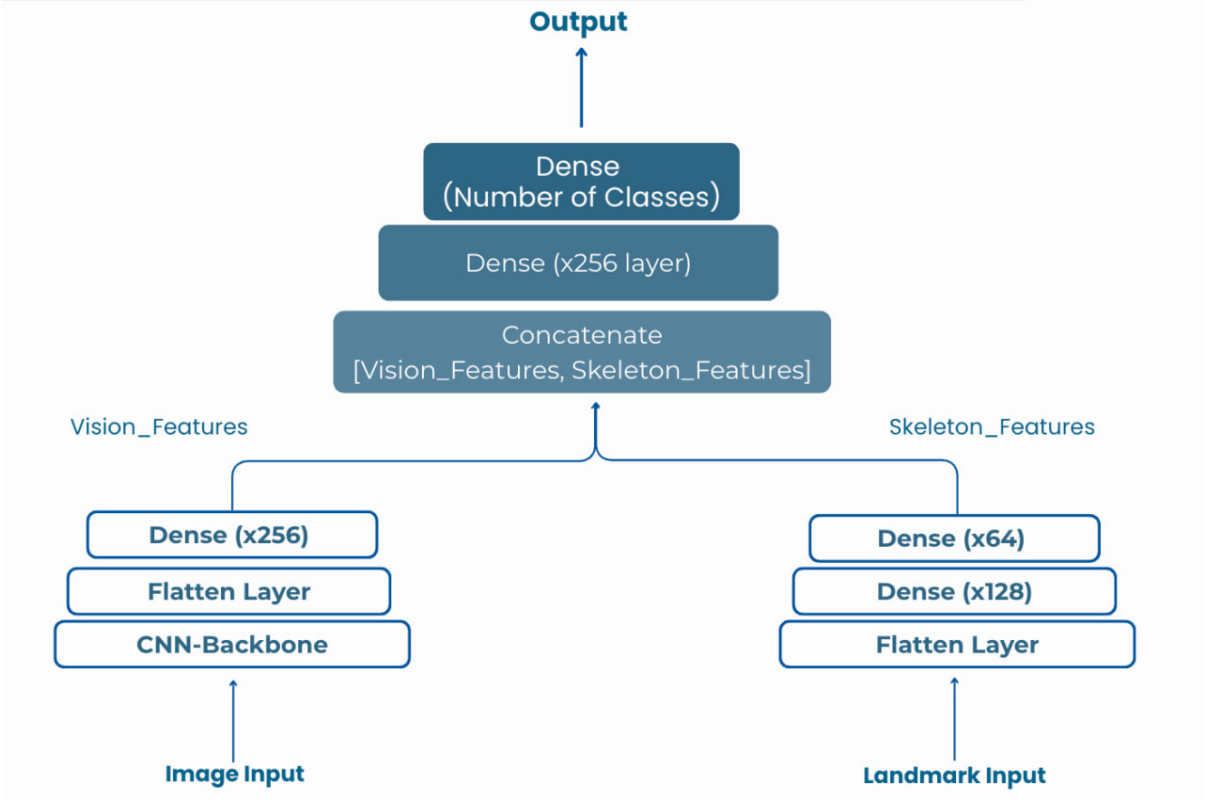


Figure 6. The baseline two-stream multi-modal fusion architecture.

parative analysis.

- **Vision Branch**: The visual stream take a preprocessed 224 x 224 pixel image as input. The image is fed into a pretrained CNN Backbone(EfficientNetB0), with its layers frozen to act as a powerful feature extractor. The resulting feature map is flattened into a vector and passed through a Dense layer with 256 units to produce the final Vision_Features
- **Skeleton Branch**: The skeletal stream takes a (21, 3) tensor of landmark coordinates as input. This tensor is flattened into a 63-dimensional vector and processed by a Multi-Layer Perceptron, consisting of a Dense layer with 128 units followed by Dense layer with 64 units. This produces the final Skeleton_Features
- **Fusion and Classification Head**: The Vision_Features and Skeleton_Features are then combined via a Concatenate layer. This unified feature vector is passed through a final classification head, consisting of a Dense layer with 256 units (ReLU) and a Dropout layer, before the final softmax output layer predicts one of the 33 classes.

3.4 Architectural Variations for Comparative Analysis

To investigate the effectiveness and robustness of the system, we evaluated several key architectural variations.

Unimodal Baselines:

- **Vision-Only Model**: Consists solely of the EfficientNetB0 Vision Branch and the final classification head.
- **Skeleton-Only Model**: Consists solely of a skeleton processing branch (LSTM or MLP) and the final classification head. This is crucial for establishing the standalone robustness of the geometric data.

Fusion Model Variations:

- **EfficientNetB0 + LSTM**: Replaces the MLP in the skeleton branch with a Long Short-Term Memory (LSTM) layer to test

if processing landmarks as a sequence improves performance.

- **Attention-Based Model (EfficientNetB0 + 1D-CNN + Attention)**: Our most sophisticated architecture. It replaces the MLP with a 1D-CNN to detect local geometric motifs in the landmarks. Crucially, it replaces simple concatenation with a gating-based Attention mechanism, designed to allow the model to learn the relative importance of the vision and skeleton streams during fusion.

This suite of models allows for a direct comparison of unimodal vs. multi-modal performance, as well as an analysis of the efficacy of different skeleton processing techniques and fusion strategies.

3.5 Experiments

Our experimental setup was designed to rigorously test our core hypotheses regarding model robustness under realistic conditions. All models were trained and evaluated using the TensorFlow/Keras framework.

- **Training Protocol**: All models were trained on the "clean" training set for a maximum of 50 epochs, using a BATCH_SIZE of 32 and the Adam optimizer. A memory-safe Data Generator was implemented to process images on-the-fly, preventing RAM exhaustion. An EarlyStopping callback with a patience of 7 epochs, monitoring val_loss, was used to ensure each model was trained to its optimal point before overfitting.

- **Evaluation Protocol**: Each optimally trained model was evaluated on two distinct datasets: The held-out Clean Test Set, to measure performance under ideal conditions. The manually curated Challenge Set, to measure true real-world robustness against data degradation.

4 Result

The experiments yielded a clear, consistent, and highly informative set of results. While all models performed exceptionally well on the clean, academic dataset, their performance diverged dramatically on the difficult real-world Challenge Set. The quantitative findings are summarized in table 1.

As shown in table 1, all evaluated models achieved near-perfect accuracy (92-100%) on the clean test set. However, on the Challenge Set, a stark performance hierarchy emerged. The Skeleton Model (LSTM) established itself as the most robust architecture, achieving the highest accuracy of 81.00%. In contrast, all multi-modal fusion models underperformed this unimodal baseline. The EfficientNetB0 + MLP model achieved 60.00%, while the EfficientNetB0 + LSTM and Attention-Based Model experienced a catastrophic performance collapse, dropping to 22.00% and 11.00% respectively.

5 Discussion and Findings

Our comprehensive evaluation led to a decisive and counter-intuitive primary finding: for the task of robustly recognizing Khmer Consonant Hand Signs under real-world conditions, the unimodal Skeleton-Only model is definitively the superior architecture. The failure of all multi-modal fusion attempts, particularly the catastrophic collapse of the advanced Attention-Based Model, provides powerful evidence for the "Confident Garbage" phenomenon. We diagnose this as follows:

- **Vision Model**: The EfficientNetB0 vision model, while powerful, demonstrates extreme brittleness when faced with the domain shift of the Challenge Set. Its standalone accuracy of 56% shows it struggles significantly with visual degradation.

- **Skeleton Model**: The Skeleton-Only model, processing pure geometric data, is largely immune to visual noise. It maintains a high accuracy of 81%, establishing itself as a highly robust and reliable signal source.

- **The Failure of Fusion**:When these two signals are fused, the vision branch produces high-confidence but erroneous feature vectors on degraded images. This "confident garbage" poisons the fusion process. The final classifier, attempting to reconcile a strong, correct signal with a loud, incorrect one, makes a poor compromise. This is why

Table 1. Comprehensive Comparison of Model Performance Across Key Metrics

Model Category	Model Configuration	Inference Time	Test Set (%)	Challenge Set (%)
Baseline Models	Vision Model (EfficientNetB0)	6.3 ms	100	56.00
	Skeleton Model (LSTM)	3.0 ms	100	81.00
Fusion Models	EfficientNetB0 + MLP	7.0 ms	100	60.00
	EfficientNetB0 + LSTM	8.8 ms	92.00	22.00
Attention-Based Model	EfficientNetB0 + 1D CNN + Attention	11.0 ms	100	11.00

the EfficientNetB0 + MLP fusion (60%) is worse than the skeleton model alone (81%).

- **The Attention Mechanism**: The Attention model's collapse to 11% is the most critical piece of evidence. This indicates that during training on the clean dataset, the model learned a fatal policy: to heavily trust the vision branch. When faced with the Challenge Set, it continued to apply this policy, actively ignoring the correct skeleton data and amplifying the "confident garbage" from the vision branch, leading to a near-total failure.

6 Future Work

Our findings clearly indicate that the primary bottleneck for building a state-of-the-art fusion model is not the fusion architecture itself, but the brittleness of the vision branch. Therefore, future work should prioritize improving the vision model's robustness.

- **Data-Centric Approach**: The most promising path is to curate a larger and more diverse visual training dataset. Actively collecting and augmenting the training set with examples of poor lighting, motion blur, multiple signers, and complex backgrounds is essential to teach the vision model to generalize.

- **Self-Supervised Pre-training**: Before fine-tuning on sign language data, the vision backbone could be pre-trained on a large, unlabeled dataset of diverse hand images. This would help it learn more general and robust features specific to hands, rather than relying solely on ImageNet.

- **Uncertainty-Gated Fusion**: Future research could explore fusion mechanisms that explicitly model the uncertainty of each branch's prediction, allowing the model to learn to completely discard the vision input when its confidence is low.

7 Conclusion

This research set out to identify the most robust architecture for Khmer Consonant Hand Sign Recognition. Through a systematic comparison of six distinct unimodal and multi-modal models on both a clean dataset and a real-world "Challenge Set," we arrived at an unambiguous conclusion. A simple, unimodal Skeleton-Only (LSTM) model, which processes geometric hand landmarks, decisively outperformed all complex multi-modal fusion architectures in robust, real-world conditions. Our analysis revealed a critical "Confident Garbage" phenomenon, where a powerful but brittle pretrained vision model actively degrades the performance of the more robust skeleton stream during fusion. This work serves as a vital case study, demonstrating that for applications with a significant domain shift, a simpler, more robust unimodal model can be the superior choice for deployment, challenging the prevailing assumption that multi-modal systems are inherently better. However, despite these impressive results, several limitations still remains. Confusion still persists among groups of similar signs, and the present research focus only on static alphabet gesture rather than continuous gesture. Additionally, the current research does not contains vowel signs or common phrases signs which is widely used by deaf people. Lastly, Real-world robustness which includes varying lighting, different backgrounds, skin tones, varying hand shapes, still have not been thoroughly tested and evaluated yet.

Based on our research and evaluation, future work will focus on expanding Khmer Sign Language Recognition (KSLR) for educational purposes. This paper had proposed the initial stage that concentrate on consonant recognition. After

this, we will extended to include vowels, numbers, and additional sign. This step-by-step expansion will provide a structured learning tool for children with hearing impairments. Alongside this, more experimentation with attention-based architectures will be conducted to enhance class separation, and robustness will be evaluated using real-world datasets under diverse conditions.

Acknowledgment

We would like to express our sincere gratitude to our advisor and professor, Dr. Ly Rottana, for his constant encouragement, insightful guidance, and dedicated support throughout this research. We also thanks our two former teammates who contributed to the initial writing of this paper. Finally, we extend our appreciation to the scholar and researchers whose prior studies provided the foundation upon which this research was built.

References

[1] World Health Organization. Deafness and hearing loss. https://www.who.int/health-topics/hearing-loss, 2025. [Online; accessed 14-September-2025].

[2] United Nations. Sign languages day. https://www.un.org/en/observances/sign-languages-day/, 2025. [Online; accessed 14-September-2025].

[3] EAC News. Mosvy secretary of state encourages persons with disabilities not to despair as the government seeks to help. https://eacnews.asia/home/details/15414, September 2022. [Online; accessed 14-September-2025].

[4] World Health Organization. Deafness and hearing loss: Fact sheet. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss, February 2025. [Online; accessed 14-September-2025].

[5] A. Aldalur et al. Acculturative stress, mental health, and well-being among deaf individuals. *International Journal of Environmental Research and Public Health*, 20(9), 2023. [Online; accessed 14-September-2025].

[6] Tal Lambez, Maayan Nagar, Anat Shoshani, and Ora Nakash. The association between deaf identity and emotional distress among adolescents. *School for Social Work: Faculty Publications*, May 2020. [Online; accessed 14-September-2025].

[7] World Health Organization. World report on hearing, March 2021. [Online; accessed 14-September-2025].

[8] World Health Organization. Deafness and hearing loss. https://www.who.int/health-topics/hearing-loss, February 2025. [Online; accessed 14-September-2025].

[9] Rhoda Olkin. Could you hold the door for me? including disability in diversity. *Cultural Diversity & Ethnic Minority Psychology*, 8(2):130–137, 2002. [Online; accessed 14-September-2025].

[10] Dr. Naa Adzoa Adzeley Boi-Dsane. Being understood: How to expand sign language access for the deaf community, September 2024. [Online; accessed 14-September-2025].

[11] Athulya Sundaresan Geetha. What is yolov6? a deep insight into the object detection model, 2024.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[13] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines. *CoRR*, abs/1906.08172, 2019.

[14] M. Buttar, A. Kumar, and K. Singh. Deep learning in sign language recognition: A hybrid approach for the recognition of static and dynamic signs. *Mathematics*, 11(17):3729, 2023.

[15] Z. Zhang, S. Liu, X. Wang, W. Wang, Z.-J. Zha, and J. Sun. Meta-detr: Few-shot object detection via unified image-level meta-learning, 2022.

[16] H. Zhang, Y. Wang, and Y. Gong. Accurate few-shot object detection with support-

query mutual guidance and hybrid loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12890–12899, 2021.

[17] P. Shenoy, R. Ganesan, and R. Varma. Real-time indian sign language recognition using deep learning and smartphone camera, 2021.

[18] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *CoRR*, abs/2102.12122, 2021.

[19] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[20] T. Kondo, Y. Tan, and K. Matsumoto. A performance comparison of japanese sign language recognition with vit and cnn using angular features. *Electronics*, 14(8):3228, 2024.

[21] R. D. A. Daniels, A. Mulyana, and A. Widodo. Real-time bisindo sign language recognition system using yolov3. In *IOP Conference Series: Materials Science and Engineering*, volume 1077, page 012029, 2021.

[22] Qiang Chen, Xiangbo Su, Xinyu Zhang, Jian Wang, Jiahui Chen, Yunpeng Shen, Chuchu Han, et al. Lw-detr: A transformer replacement to yolo for real-time detection. *arXiv preprint arXiv:2406.03459*, 2024.

[23] X. Dong, Y. Zhang, J. He, and W. Sun. Incremental-detr: Incremental few-shot object detection via self-supervised learning and fine-tuning, 2022. arXiv preprint.

[24] Sharvani Srivastava, Sudhakar Singh, Pooja, and Shiv Prakash. Continuous sign language recognition system using deep learning with mediapipe holistic. *Wireless Personal Communications*, 137:1455–1468, 2024.

[25] Khmer Post Asia. Khmer sign language learning made available online, March 2020. [Online; accessed 14-September-2025].

[26] Peak Team, Ministry of Education Youth, and Sport Cambodia. Ksl – khmer sign language. `https://play.google.com/store/apps/details?id=com.peak.moeys.rti.ksl&hl=en`, 2025. Accessed: 2025-09-14.

Modelling insect interactions in rice crops on the GAMA platform as a basis for educational VR games

Rattanak Seth[1,2] **Lucile Delatouche**[4] **Mathilde Sester**[3]
Alexis Drogoul[2] **Sovuthy Cheab**[1]
[1]Cambodia Academy of Digital Technology, Phnom Penh, Cambodia
[2]UMI 209 UMMISCO, ACROSS Lab, Thuyloi University/IRD, Hanoi, Vietnam
[3]CIRAD, UPR AIDA, Institut of Technology of Cambodia, Phnom Penh, Cambodia
[4]CIRAD, UPRA AIDA, Royal University of Agriculture, Phnom Penh, Cambodia
rattanak.seth@ird.fr

Abstract

Digital Technology and agriculture are crucial in helping farmers and students understand insect interactions in rice fields and how to apply treatments effectively. An agent-based framework is appropriate for studying the ordinary population of insects interacting on a rice crop, but it is still difficult to adapt for non-computer scientists in more specific application contexts (e.g., various treatment scenarios, insect reproduction, insect resistance, and crop yields), which require integrating particular behaviours for agents. In this paper, we present a built-in model integrated into the GAMA open-source modelling and simulation platform, allowing modellers to easily define the interaction of insects with a detailed presentation of rice field, insect population and treatment decision. In particular, it enables modelling the application of insecticide on paddy fields while understanding its effects. This agent-based model serves as a foundation for creating a comprehensive virtual reality (VR) game that presents scenarios involving pesticide application decisions, interactions between insects, and their impact on crop yields.

Keywords: *Agent-based Modelling, GAMA Platform, Rice Paddies, Insect Interaction, Pesticide Application, Simulation, VR Game.*

1 Introduction

The foundation of Cambodia's economy and culture is rice. Producing about 11.6 million tons of paddy by 2022, it is the primary staple food that sustains the livelihoods of about 3 million farmers and takes up about 75% of the country's agricultural land. [1]. In addition to being essential for food security and exports, rice in Cambodia is also extremely susceptible to outbreaks of pests and diseases, which jeopardise yields and farmer profits. While blast and bacterial leaf blight are among the main diseases, common pests include brown planthoppers, rice bugs, leaf folders, and stem borers. Chemical pesticides are frequently used by farmers for control, but excessive use has resulted in ecological imbalance, resistance, and health issues. In order to guarantee sustainable rice production, integrated pest management (IPM) techniques—such as crop rotation, ecological engineering, biological control, and the use of resistant varieties—are being promoted more and more. Therefore, improving productivity and increasing climate change resilience in Cambodia's rice industry requires more sustainable pest management.[2].

Pesticides are utilised globally to improve crop yields by minimising losses due to all kinds of insect pests (weeds, diseases, insects, etc). However, incorrect application of these chemicals can lead to pesticide resistance and a resurgence of pest populations, non-target organisms that are often beneficial [3]. Therefore, improving the knowledge of farmers on pesticides and their risks remains a condition for better pest management. A simulation on insect interaction in rice crop and application of pesticide is implemented to provide knowledge about the behaviour of insects and the negative and positive impact of pesticides. Insecticides have a negative impact on human health and the environment, which is a concern for the

government and consumers. If farmers understand the role of beneficial insects, such as parasitoids and predators, they are likely to reduce pesticide applications. By leveraging these natural enemies to control pest populations, they can enhance rice yields while minimising chemical use.

This research studies a sample of rice field area, where a grid represents one part of the rice crop. The number of grids can change based on the size of the plot. This application is used to build calibrated scientific agent-based models validated in well-documented case studies, ensuring the realism for the development scenarios that participants explore and allowing them to collaborate virtually to explore solutions to sustainability issues. It is a bridge to develop the virtual universes for short modules for workshops with young students or as integral components of the curriculum for high school students.

The aim of this research is to provide a comprehensive understanding of pesticide application in rice fields, exploring both the methods used and the potential consequences. It examines the pesticides commonly employed in rice cultivation, their effectiveness in managing pests, and the practices farmers adopt to maximise their benefits. Additionally, it will address the environmental effects on non-target organisms, soil health, and water quality. It will also discuss the socio-economic implications for farmers and communities, highlighting the balance between pest control and sustainable agricultural practices. Through this exploration, this research seeks to inform farmers about responsible pesticide use and promote strategies that minimise negative outcomes while maintaining crop productivity. All of these are built into an agent-based model for simulation by using the GAMA platform, which is an open-source modelling and simulation platform that allows modellers to easily define the interaction of insects with a detailed presentation of the rice field, insect population and treatment decision.

Initially, formulas and research were developed by the researcher working on the subject and used as a board game using Microsoft Excel, and subsequently, they were transformed into an agent-based model utilising the Gama platform. This agent-based framework is appropriate for studying the ordinary population of insects interacting on rice crop, but still difficult to adapt for non-computer scientists, to a more specific application context, for example, various treatment scenarios, insect reproduction, insect resistance, and crop yields require integrating particular behaviours for agents.

2 Related Work

A dynamic model along with its discretised system to increase the agricultural crop production using some external efforts in the presence of insects and insecticides. This research model in this paper is based on local governments, farmers, and consumers, and a more detailed evolutionary game analysis model is constructed to provide a reference for subsequent policy optimisation. Mathlab is utilised to simulate the behaviour of stakeholders' participation in pesticide reduction and analyse the impact of changes in the behaviour of different subjects on the pesticide reduction evolutionary game system [4]. This paper reviews recent literature on how pesticides harmfully affect beneficial organisms, such as parasitoid wasps, with the goal of enhancing pest control strategies that integrate both chemical and biological methods for sustainable integrated pest management (IPM) [5].

This study develops a novel geospatial agent-based EAB-BioCon model for the interactions of the emerald ash borer (EAB) with the parasitoid Tetrastichus planipennisi (TP) wasp in order to evaluate the spread of forest infestations. The model is implemented on geospatial data from the City of Oakville, Canada and is composed of EAB Baseline model, representing EAB geospatial dynamics and the EAB-TP model that employs scenarios to measure EAB response to variations in TP-based biological control strategies [6]. Similarly, sugarcane production areas in Brazil have experienced a slower evolution in productivity, and one of the reasons for this is related to the increase in phytosanitary problems, such as the presence of the pests. Therefore, an agent-based model has been developed to simulate the

pest population and its dispersal in a one-hectare sugarcane crop field in Pederneiras, São Paulo, Brazil, delimited with the aid of satellite imagery, considering two scenarios: the first without biological control and the second with biological control using the parasites Trichogramma galloi and Cotesia flavipes. This model was developed using the NetLogo 6.3.0 software [7].

3 Implementation of the model

The model has been implemented on the GAMA [1] platform (version 2025-06), dedicated to agent-based modelling (ABM) and simulation. ABM consists of actors, resources, dynamics and interactions (ARDI). The resource consists of rice paddies and rice crops. A rice field is defined as a 10×10 grid in our model, representing the whole configuration. Each cell contains three layers of rice, representing rice crops, which can be damaged or destroyed by pest populations. These pests can devastate the rice crops, leading to a loss in crop yield. The model includes two main actors: farmers and insects. The farmer is responsible for applying treatments, while insects are divided into three main groups: parasitoids, predators, and pests. Parasitoids and predators were introduced to control a rice insect pest. A few introductions of parasitoids from different species of pests related to the target insect have been successful; examples are not common, and none are known in rice [8]. Dynamic models are used to describe objects and their relationships as they change over time. In our model, there are three dynamics included: first, the consumption of pests by parasitoids and predators; second, the movement of insects, such as hunting and fleeing; and third, the farmer's decision on whether to spray or not. There are four interactions in this agent-based model. First, in pest-plant interactions, the model simulates how insects interact with rice plants, including the impact on plant health. Second, insect-insect interactions are actions in which a group of insects, such as parasitoids and predators, eat pests. Third, farmer-insect/plant interactions, farmers' actions, like applying pesticides or adjusting irrigation, can be incorporated as interactions affecting both insects and plants.

[1] https://gama-platform.org/

3.1 Representation of simulation data

Each rice grid consists of three layers, represented by three different colours, and it also has another colour that represents all the layers being lost, which is listed in Table 1. Insects have three groups; the correspondence is shown in Table 2:

Table 1. List of colours used to represent rice health

Layer	Colour	Meaning
3	rgb(0, 112, 48)	Healthy rice
2	rgb(198, 239, 206)	Slightly damage
1	rgb(255, 255, 0)	Heavy damage
0	rgb(88, 57, 39)	Loss all layers

Table 2. List of colours used to represent each group of insects

Type of insect	Colour	Meaning
Pest	red	Harmful insects
Parasitoid	black	Beneficial insects
predator	violet	Beneficial insects

3.2 Data collection

To validate the model and thus the ability of GAMA to simulate insects in rice paddies, we recorded 13 weeks of rice cycles to validate the result. This is important when parameters have been changed so that most of the values in each week will be modified. This data is crucial for analysing the impact of pesticides on insects, both non-target insects and target insects such as pests. Participants are aware of the importance of reducing pesticides in each cycle of rice and the final yield. Furthermore, data from batch simulation was also recorded to analyse the exploration of four parameters which affect the crop yield.

3.3 Description of the model and parameters used

In this simulation, the parameters have been divided into three different groups such as

pesticide, insect, and rice paddy. The pesticide category is used to define user interaction. If it is true, they will interact with this simulation, deciding whether to spray or not; otherwise, it has several scenarios that participants can select the one they prefer, and it will simulate automatically. First, 'None' is an option that no treatment was applied. Second, 'All weeks' is an option that sprays every week except the first and last weeks; the first week marks the start of the rice crop, and spraying in the last week is prohibited to protect the health of the harvesters and consumers. Third, the 'Only 2nd Week' option involves spraying exclusively during the second week, while skipping the first and last weeks, continuing this pattern through week 11.

The insect category is used to define the number of parasitoids and predators. The initial value of the parasitoid will start with 25 as the default, and the initial value of the predator will be 7 as the default. This change will update the population of insects in the rice paddies for the initial simulation.

The first model was built based on a game formula, which we calculated using Microsoft Excel. This initial model serves as a foundation for developing a second, more complex agent-based model. A key aspect of this first implementation is ensuring that the results of the game align with those calculated in Excel. This approach paves the way for the next model, as the complexity of the computations requires a step-by-step process to simplify the calculations.

To calculate each week of this simulation, we first compute the independent variable, followed by the dependent variable that depends on it. Initially, we calculate the dynamic yield, which is first calculated and can be applied in the last computation. Next, according to the Figure 1, the calculation of predators is computed, and then we calculate parasitoids. These calculations increase the population according to the population the week before and the reproduction rate, which depends on whether they eat enough or not. Subsequently, pest and its related variables are calculated. After updating the insect population, the predation rate should also be calculated. The predation rate is the amount of total pests eaten by both predators and parasitoids divided by the total number of pests. It refers to the frequency with which an organism captures and consumes its prey in an ecosystem. Finally, the result is computed to get the remaining field case, potential yield, real yield, and max dynamic yield.

On the other hand, more dynamics have been implemented. It is updated following the first model, which only sticks to the game that is built in MS Excel. The most important parts are the insect's hunting, fleeing and eating, as well as the farmer's decision in many scenarios. Similarly, several parameters have been added for user interaction in the simulation, which you can see in the Table 3. Two types of insect groups, parasitoids and predators, have been implemented using algorithm 1.

Furthermore, four scenarios of this simulation are described in the algorithm 3. It features five functions **sequentialVariableCalculation**, **interactFromParticipant**, **sprayAllTheTime**, **sprayOnceWithinRiceCycle(week_num)**, and **doNotSpray**. The **sequentialVariableCalculation()** function calculates the each week of insect population, and the result of rice yield is mentioned in the Figure 1.

The **interactFromParticipant()** function allows the participant to interact with the simulation by deciding whether to spray or not. This still keeps the first and the last week of the rice cycle, which is our target mentioned above. The **sprayAllTheTime()** function works only *is_interacted* is false, and it will spray every week except for important weeks. The **sprayOnceWithinRiceCycle(week_num)** function runs automatically without interaction from the participant by spraying once within the rice cycle, starting from week 1 to 11. The **doNotSpray()** function indicates that no spray is applied throughout the entire rice cycle, relying on natural enemies such as predators and parasitoids to control the pest population.

4 Experimental results

The experiment was divided into two main categories derived from an 11 factorial combinations scenario. First, user interaction in

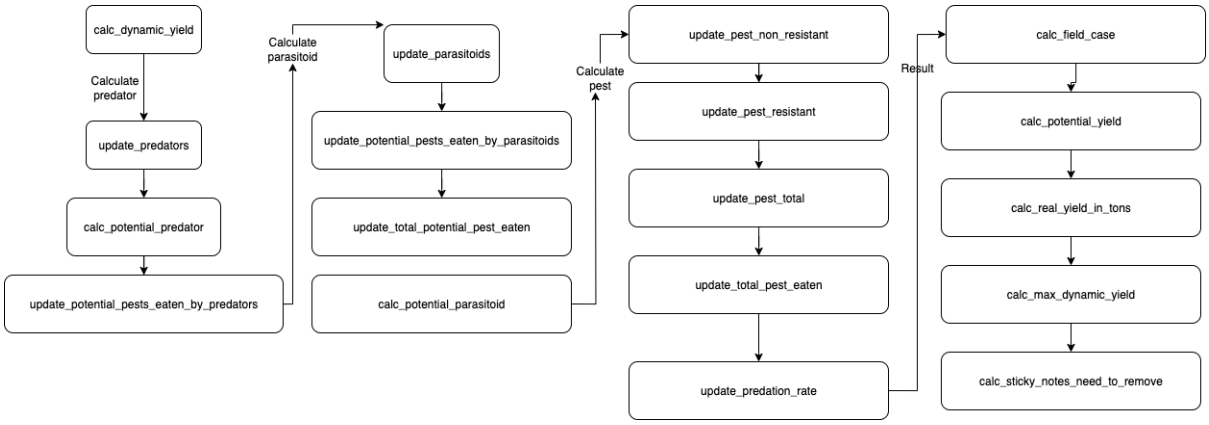


Figure 1. Step to calculate each week of rice paddies

Algorithm 1 Hunting

function HUNTING
  if $goal = nil$ **then**
    $lstPest \leftarrow Pest\ where(!dead(each)\ and\ each.shape\ distance\_to\ self.shape < perception\_radius\ )$
    if $length(lstPest) > 0$ **then**
      $agent\ a \leftarrow first(lstPest\ short\ (each.shape\ distance\_to\ self.shape))$
      if $a = nil$ **then**
        $a \leftarrow any(Pest\ where(!dead(each)))$
      else
        $speed \leftarrow 2.5\#km/\#day$
        $goal \leftarrow a.location$
      end if
    end if
  else if $self.location\ distance\_to\ goal < 0.5$ **then**
    $pestToDie \leftarrow Pests\ where\ (!dead(each)\ and\ each.location\ distance\_to\ goal < 0.5)$
    $remainToEat \leftarrow length(Pest\ where(!dead(each)) - pests\_total)$
    if $remainToEat > 0$ **then**
      $pestToDie \leftarrow remainToEat\ among\ PestToDie$
      ask pestToDie do die
    end if
    $goal \leftarrow nil$
    $speed \leftarrow 1.6\#km/\#day$
  end if
end function

Algorithm 2 Fleeing

```
function FLEEING
    if length(Predators) where(each distance_to self < perception_radius) > 0 or
length(Parasitoids) where(each distance_to self < perception_radius) > 0 then
        speed ← 2.0#km/#day
        is_chased ← true
        color ← #lime
        if goal = nil then
            agent a ← any(Pests !dead(each) and !Insect(each).is_chased)
            if a ≠ nil & !dead(a) then
                if flip(0.5) then
                    goal ← a.location
                else
                    goal ← any_location_in(cell.shape)
                end if
            else
                goal ← any_location_in(cell.shape)
            end if
        end if
    end if
    if goal ≠ nil & self.location distance_to goal < 0.5 then
        goal ← nil
    end if
    if length(Predators) where (each distance_to self ≤ perception_radius) = 0 and
length(Parasitoids) where (each distance_to self ≤ perception_radius) = 0 then
        is_chased ← false
        color ← #red
        speed ← 1.6#km/#day
    end if
end function
```

Algorithm 3 Farmer decision function

```
function FARMER_DECISION
    do sequentialVariableCalculation()
    if is_interacted then
        do interactFromParticipant()
    else if ALL_WEEK then
        do sprayAllTheTime()
    else if ONCE_PER_CYCLE then
        do sprayOnceWithinRiceCycle(week_num)
    else if NONE then
        do doNotSpray()
    end if
end function
```

Table 3. List of parameters used in experiment

Parameter	Category	Value (default)	Note
rice_field_width	Rice Paddy	10	width of rice paddy
rice_field_height	Rice Paddy	10	height of rice paddy
number_of_parasitoid	Insect	25	This will affect and update both the number of parasitoid and potential parasitoid
number_of_predator	Insect	7	This will affect and update both the number of predator and potential predator
interactive	Pesticide	True	
spray_decision	Pesticide	None	Enable only interactive is false

Table 4. Apply pesticide once during the rice cycle (tons/ha)

Week	1	2	3	4	5	6	7	8	9	10	11
Result	0.0	4.02	3.99	3.93	3.83	3.73	3.65	3.65	3.65	3.65	3.65

the simulation. This depends on the participant's decision that they can input like the Figure 2. Second, without interaction from participants, we chose the important scenarios such as no treatment, all-week treatment, and only the second week. Moreover, the result will be shown in a graph and the rice grid user interface (UI). For various scenario experiments, a batch experiment has been implemented with four experiments into one UI for comparing the results, for example, spray only the 2nd week, only the 1st week, only the 3rd week and no spray.

4.1 Interaction from player

The interaction from the player, the result relies on the behaviour of the player who decides to spray or not for each week of rice cycles. Therefore, various scenarios will be presented in this section. For example, the participant decides to spray in the sixth week of the rice cycle, and the result is illustrated as a graph in Figure 3. It is seen that in the second week the pest population stood at 200 and dramatically decreased to just under 60. In the sixth week, pesticides have been applied while the population hit to bottom, indicating that natural enemies can control pest populations. Inversely, this treatment will affect the next week of the rice cycle, as natural enemy such as parasitoids and predator drop their population. It has been concluded that the treatment in the sixth week does not affect rice yield. As a result, rice yield is only just above 3.5 tons with a maximum of 5 tons per hectare.

This is not a good spray based on what was mentioned, so if we spray in the previous week, which is the fifth week what will happen?. According to Figure 4, it is seen that crop yield is just under 4, with the exact amount 3.83, which was recorded. This amount is slightly different from the amount chosen to spray on the sixth week. The reason is that even though we do not spray, it will not affect to the pest population and crop yield because natural enemies can control the pest population by eating them. Reducing pesticide use this week is crucial, as it has no impact on crop yield, lowers pesticide costs, and helps prevent health issues. Taking into account this scenario, the application of the pesticide in the fourth week was experimented.

Regarding the pest population, it consistently remained at 200 in the second week and rose to approximately 220 in week 4. Subsequently, after insecticide was applied, the pest population plummeted to 80 in week 5 and continued to decline through the end of the week. As a result, the crop yield reached 3.93 tons/ha. What happens if a pesticide is applied in the third week?

This scenario is similar to the above; after

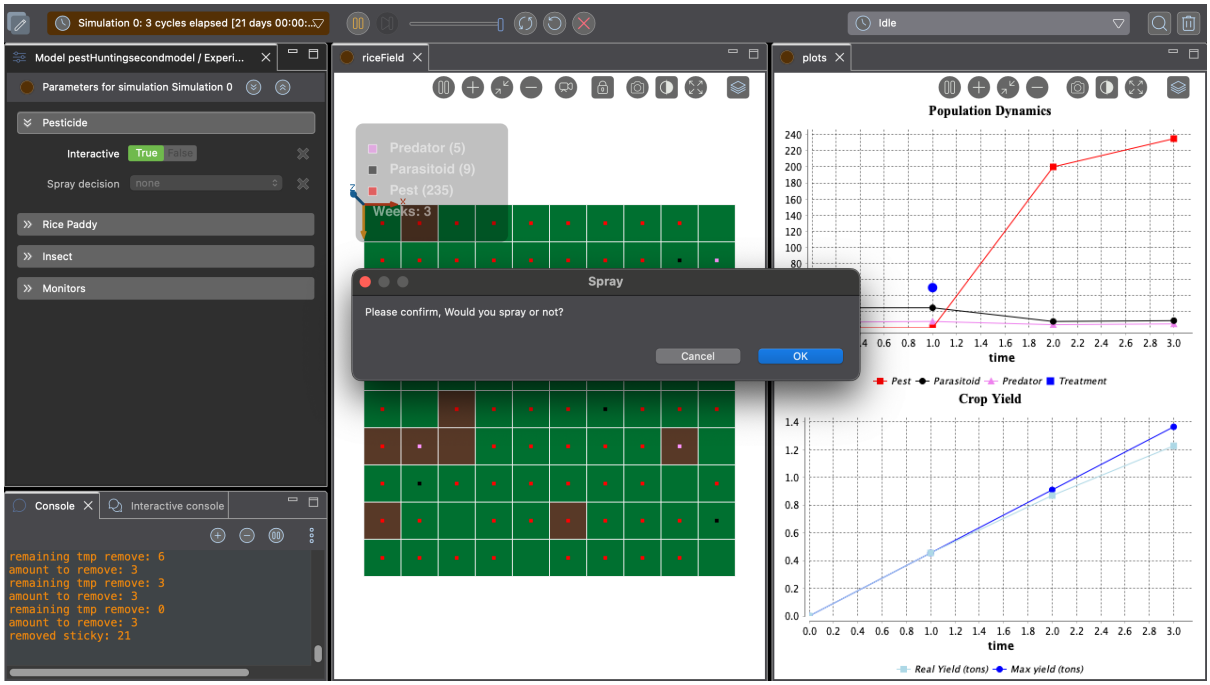
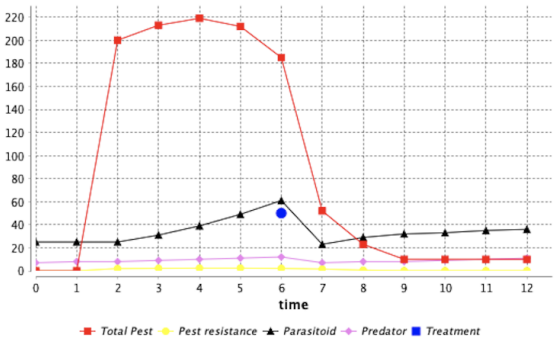
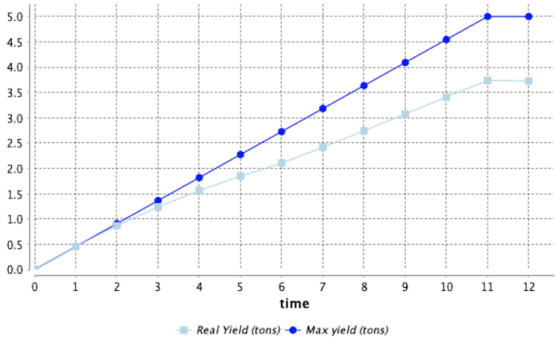


Figure 2. Pop-up dialogue to ask for spraying decision

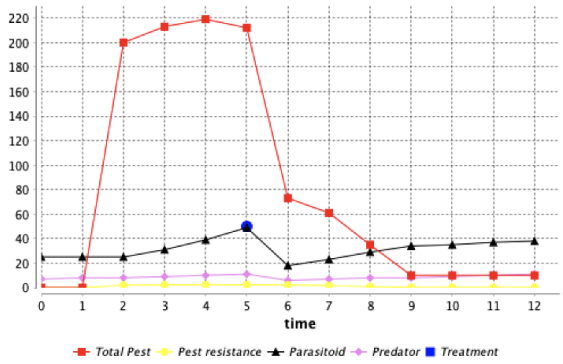


(a) Population dynamics

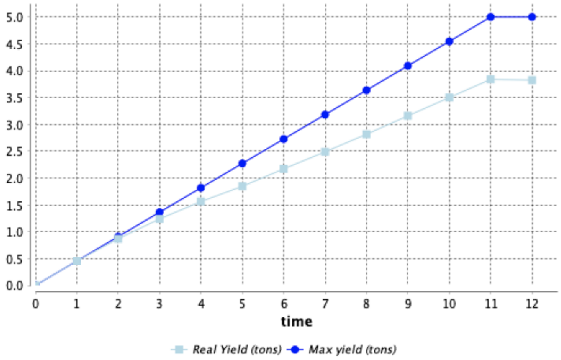


(b) Crop yield

Figure 3. Result spray on the sixth week

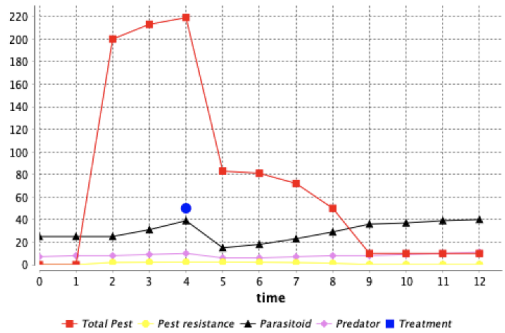


(a) Population dynamics

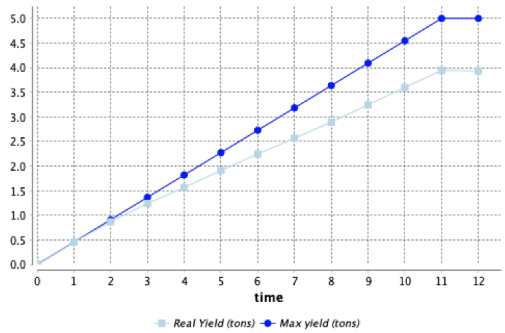


(b) Crop yield

Figure 4. Result spray on the fifth week

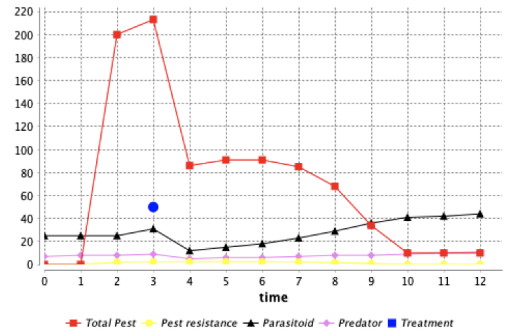


(a) Population dynamics

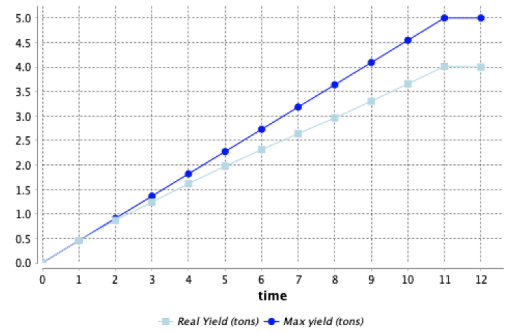


(b) Crop yield

Figure 5. Result spray on the fourth week

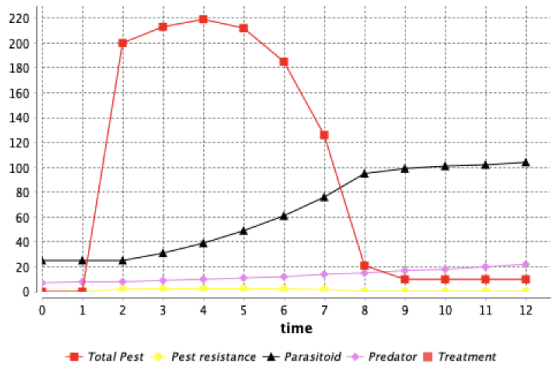


(a) Population dynamics

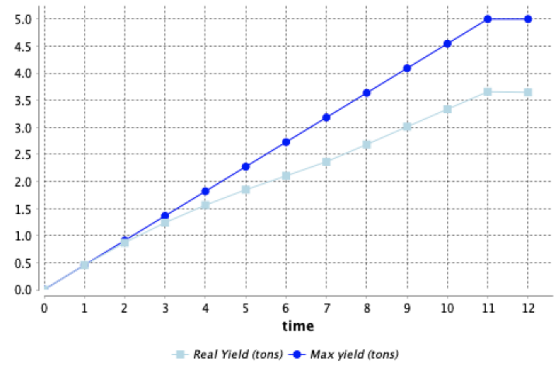


(b) Crop yield

Figure 6. Spray only third week



(a) Population dynamics



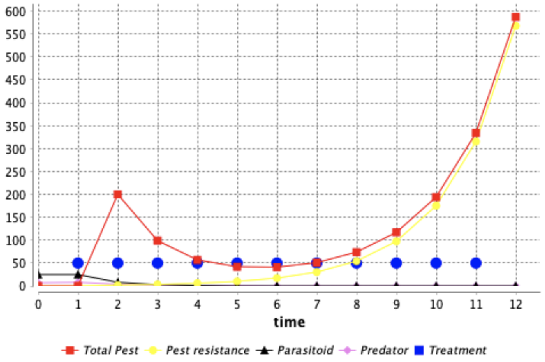
(b) Crop yield

Figure 7. No treatment

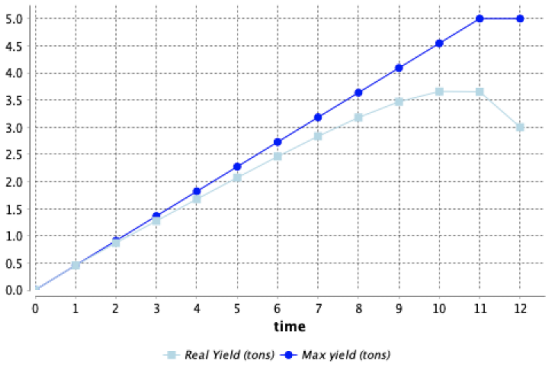
spraying at week 3, the pest population declines to the bottom from week 4 until the crop yield. As a result, rice yield obtained 3.99 tons/ha. In conclusion, early-season spraying (Week 3) significantly reduces pest population growth compared to mid- or late-season spraying. Delayed spraying interventions (Week 6 and beyond) result in greater yield losses due to established pest populations. For more results, refer to the section on spraying once per rice cycle.

4.2 No treatment scenario

No treatment for all weeks is a good scenario, because natural enemies can control the pest population. According to Figure 7, it is observed that as the number of natural enemies slightly increases, the pest population rises to just under 220 in week 4 before decreasing significantly. This indicates that traditional farming practices that utilise beneficial insects for pest control are effective. Consequently, the rice yield is 3.65 tonnes per hectare.



(a) Population dynamics



(b) Crop yield

Figure 8. Spray every week

4.3 Spray every week

Apply treatment every week; Figure 8; will be harmful to all insects, including good and bad insects. The pests develop resistance to pesticides and reproduce rapidly, damaging the crop, which results in a yield of only 3 tons per hectare. Not only is it unhelpful, but it is also harmful to human health.

4.4 Spray once per rice cycle

Applying pesticide once during the rice cycle is a notice scenario that we considered. As shown in Table 4, applying the spray in the second week yields the best result of 4.02 tons per hectare, as this coincides with the week pests enter the rice paddy. For more details about this scenario, refer to Interaction from Participants.

5 Conclusion

This model is designed to provide various scenarios for a VR game aimed at educating high school students and farmers. Initially, a game was defined in Microsoft Excel to lay the groundwork for developing an agent-based model. Subsequently, an agent-based model was constructed to expand on this idea and experiment with different scenarios. After different scenarios were implemented, we found that a farmer has their own rice field, so 4 different rice fields with different spray actions were implemented to compare the results. This will provide four different players to interact in game play for the next implementation in VR. The objective is to help farmers understand the underlying mechanisms and key messages, such as the fact that a high level of pesticides is not the optimal scenario. In the near future, the shapefile will include or be loaded into our agent-based model to simulate a real environment and get more accurate results.

Acknowledgment

I express my gratitude to the authors and researchers whose work provided the insights needed to complete my study. I also extend my thanks to the Cambodia Academy of Digital Technology (CADT) for providing a scholarship for my master's degree. Lastly, I would like to thank IRD Cambodia for its financial support for this research.

References

[1] H. Sarun, T. Pang, H. Linan, I. Sokra, R. Chanra, H. Meta, H. Hiek, and L. Sotheara, "Challenges and opportunities in rice cultivation and marketing of cambodia: A review article," International Journal of Innovative Science and Research Technology (IJISRT), vol. 10, no. 1, Feb. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.14862890

[2] S. Savary, L. Willocquet, S. J. Pethybridge, P. Esker, N. McRoberts, and A. Nelson, "The global burden of pathogens and pests on major food crops," Nature Ecology & Evolution, vol. 3, no. 3, pp. 430–439, 2019. [Online]. Available: https://doi.org/10.1038/s41559-018-0793-y

[3] R. N. C. Guedes and G. C. Cutler, "Insecticide-induced hormesis and arthropod pest management,"Pest Man-

agement Science, vol. 70, no. 5, pp. 690–697, 2014.

[4] Q. He, Y. Sun, and M. Yi, "Evolutionary game of pesticide reduction management for sustainable agriculture: An analysis based on local governments, farmers, and consumers," Sustainability, vol. 15, no. 12, 2023. [Online]. Available: https://www.mdpi.com/2071-1050/15/12/9173

[5] R. Theenoor, A. Ghosh, and R. Venkatesan, "Harmonising control: understanding the complex impact of pesticides on parasitoid wasps for enhanced pest management," Current Opinion in Insect Science, vol. 65, p. 101236, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214574524000786

[6] T. M. Anderson and S. Dragićević, "Geospatial pest-parasitoid agent-based model for optimizing biological control of forest insect infestation," *Ecological Modelling*, vol. 337, pp. 310–329, 2016. [Online]. Available: https://doi.org/10.1016/j.ecolmodel.2016.07.017

[7] R. B. d. O. Alves, D. B. Tomasiello, C. M. d. Almeida, D. L. Rosalen, L. H. Pereira, H. P. d. Silva, and C. L. Rodrigues, "Agent-based spatial dynamic modeling of diatraea saccharalis and the natural parasites cotesia flavipes and trichogramma galloi in sugarcane crops," Remote Sensing, vol. 16, no. 15, 2024. [Online]. Available: https://www.mdpi.com/2072-4292/16/15/2693

[8] P. Ooi and B. Shepard, "Predators and parasitoids of rice insect pests," Biology and management of rice insects, pp. 585–612, 1994.

Student Performance Prediction Based on Final Grades: A Comparative Study of Machine Learning Models

Sokchovy Monirath **Dynil Duch**
Institute of Digital Research and Innovation
Cambodia Academy of Digital Technology, Phnom Penh, Cambodia
sokchovy.monirath@cadt.edu.kh

Abstract

Student performance prediction is increasingly important in higher education as Learning Management Systems (LMSs) capture detailed academic and behavioral data. This paper provided a systematic comparative review of machine learning models for final grade prediction that mainly used benchmark datasets such as the Open University Learning Analytics Dataset (OULAD). Classical models like Logistic Regression, Decision Trees, and Support Vector Machines achieve 75–82% accuracy, ensemble methods such as Random Forest and XGBoost reach 85–91%, and deep learning approaches like Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) up to 93%. Behavioral features submission timeliness, login frequency, and resource engagement consistently as key predictors. However, most studies focus on Western contexts and overlook model interpretability and practical applicability in developing regions rather than optimizing for accuracy. The main goal is to identify at-risk students to enable early interventions. This review synthesizes technical and contextual insights to inform effective deployment in Southeast Asian settings with emphasis on the Cambodia Academy of Digital Technology (CADT).

Keywords: *Student Performance Prediction, Machine Learning, Educational Data Mining, Learning Analytics, Final Grade, Comparative Study, Ensemble Models, Deep Learning, Long-Short Term Memory, Convolutional Neural Network*

1 Introduction

Learning Management Systems have transformed higher education by automatically capturing detailed behavioral data, including log-in patterns, assignment submissions, quiz attempts, and resource engagement [1]. This digital footprint enables early identification of at-risk students, facilitating timely interventions that improve learning outcomes [2]. Educational Data Mining (EDM) and Learning Analytics (LA) leverage machine learning to extract actionable insights from these data, supporting evidence-based decision making in educational institutions [3].

The development of predictive modeling in education mirrors the larger progress in machine learning. Initial methods employed traditional algorithms such as Logistic Regression and Decision Trees, reaching 75-82% accuracy while maintaining high interpretability [4]. Ensemble techniques like Random Forest and XGBoost enhanced performance to 85-91% via aggregation strategies that minimize variance and capture intricate feature interactions [5], [6]. Recent deep learning techniques, especially LSTM networks and attention mechanisms, extend capabilities to 92-93% accuracy by capturing temporal dynamics and acquiring hierarchical representations [7], [8].

Feature engineering is vital for achieving successful predictions. Studies consistently highlight submission timeliness, login frequency, assessment scores, and resource engagement as key indicators in various educational settings [5], [9]. These behavioral indicators measure student effort and engagement more accurately than demographic characteristics by themselves.

Despite technical progress, three remaining challenges limited the practical impact. First, most research uses Western institutional data, particularly OULAD from the UK, raising questions about generalizability to Southeast Asia, where technology access, learning behaviors, and educational norms differ [10]. Second, high-performing models often function as ”black boxes” which lack interpretability that educators require for trust and actionable insights [11]. Third, few studies examine how predic-

tions translate to effective interventions or measure actual impact on student outcomes [12].

This paper provides a comprehensive comparative review of student performance prediction approaches. We systematically analyze datasets, preprocessing methods, feature engineering strategies, modeling techniques, and evaluation frameworks. We synthesize performance across classical machine learning, deep learning that identify the accuracy-complexity. Based on this analysis, we articulate research gaps that could be addressed with the ongoing project at CADT, developing interpretable prediction systems tailored to Southeast Asian contexts.

2 Related Work

2.1 Educational Data Mining and Learning Analytics

EDM and LA emerged as distinct research disciplines leveraging digital education data to improve learning outcomes [3]. Modern LMSs like Moodle, Blackboard, and Canvas record interaction logs including page views, assignment submissions, forum posts, and video engagement [1]. This data enables analyses impossible through traditional observation, revealing behavioral patterns that correlate with academic success.

Machine learning dominates EDM due to its capacity for handling complex, nonlinear educational data [13]. Classical methods like Logistic Regression and Decision Trees established baseline capabilities, demonstrating that performance prediction was feasible [4]. Ensemble methods, including Random Forest and XGBoost became standard tools, consistently outperforming individual classifiers [5], [6]. Deep learning architectures now represent the frontier, with LSTM networks modeling temporal learning trajectories and attention mechanisms providing interpretability [7],[8].

2.2 Benchmark Datasets

The Open University Learning Analytics Dataset (OULAD) serves as the primary benchmark for student performance prediction research [1]. Released by the UK Open University, OULAD contains data for 32,593 students across 22 course modules, including demographic information (age, gender, region, prior education), assessment scores, and approximately 10 million interaction records aggregated daily. This combination of static background, progressive assessments, and behavioral trajectories enables diverse predictive tasks from early dropout detection to final grade classification.

Beyond OULAD, researchers use institution-specific LMS logs from Moodle and Blackboard deployments [14]. These datasets offer richer contextual information but a limited size (typically hundreds to thousands of students) and lack public availability, hindering replication. Some studies incorporate MOOC data from platforms like Coursera for large-scale dropout prediction, though extreme class imbalance and different learner populations limit generalizability [15]. Multimodal datasets augmenting LMS logs with forum text, survey responses, and social network data show promise but introduce collection and integration complexity [16].

2.3 Machine Learning and Deep Learning Approaches

Classical algorithms including Logistic Regression, Decision Trees, and Support Vector Machines provide interpretable baselines, typically achieving 75-82% accuracy on OULAD [4]. Their transparency enables educators to understand which features drive predictions, though limited capacity for nonlinear patterns constrains performance. Ensemble methods offer significant improvements. Random Forest improves prediction accuracy by combining multiple decision trees trained on random samples and offers insight into which features matter most in the prediction process, achieving 85-92% accuracy [5], [17]. XGBoost builds sequential ensembles where each tree corrects predecessor errors, reaching 91% accuracy with careful tuning [6]. These methods effectively balance accuracy and interpretability, making them practical choices for many applications.

Deep learning approach capture complex patterns and temporal dynamics. LSTM process sequential behavioral data, modeling learning trajectories over time and achieving 93% accuracy with particular strength in early prediction scenarios [7]. Convolutional Neural Networks adapted to educational data reach 92% accuracy by learning hierarchical feature representations [18].

3 Methodology

3.1 Dataset and Preprocessing

OULAD encompasses 32,593 student registrations across seven course presentations and 22 modules [1]. Each record includes demographics (age, gender, region, prior education, disability status), assessment results (quiz and assignment scores, final grades), and daily aggregated clickstream data capturing interactions with various resource types. The dataset supports multiple prediction tasks, including dropout forecasting and final grade classification.

Preprocessing addresses noise and inconsistencies in raw LMS logs. Standard practices include removing system-generated records and early withdrawals, aggregating clickstream data to weekly summaries for dimensionality reduction, encoding categorical variables through one-hot or ordinal methods, addressing class imbalance via SMOTE or class weighting, and normalizing continuous features for scale-invariant algorithms [1], [7]. Temporal aggregation choices critically impact model architecture selection, with weekly summaries suitable for tree-based methods and sequential representations enabling LSTM modeling.To ensure consistency across modeling approaches, preprocessing typically involves several key steps that have shown in Table 1:

3.2 Feature Engineering

Feature engineering transforms raw data into predictive representations. Research converges on 4 main categories that are typically used:

- **Demographic Feature:** Age, gender, region, and prior education level.
- **Behavioral Feature:** Number of logins, total clicks, time spent on resources, and frequency of engagement.
- **Assessment Features:** Quiz and assignment scores, submission timeliness, and number of attempts.
- **Temporal Features:** Aggregated weekly activity and learning trajectory measures used in LSTM and CNN models.

3.3 Modeling Workflow

Studies follow consistent end-to-end pipelines linking data preparation, feature design, model training, and interpretation [19].The overall process is summarized in Figure 1, which illustrates the typical workflow for student performance prediction using learning analytics data. The pipeline begins with data collection from Learning Management Systems (LMSs), followed by data preprocessing to clean, encode, normalize, and balance the dataset. The next stage involves feature engineering, where demographic, behavioral, assessment, and temporal features are extracted to capture key learning characteristics. Designed features spanning demographics, participation, assessments, and temporal patterns serve as input to the model. Researchers typically compare multiple approaches in parallel. Simple baselines (Logistic Regression, Decision Trees) establish interpretable performance floors [4]. Ensemble methods (Random Forest, XGBoost) provide accuracy with moderate complexity [5], [6]. Deep learning architectures (LSTM, CNN) are applied when dataset size and computational resources permit [7], [8].

Clear patterns link feature design that could optimize model choice. Rich aggregate engagement features pair naturally with tree-based ensembles that discover feature interactions [5], [17]. Sequential representations suit recurrent architectures that learn temporal dependencies directly [7]. Hybrid approaches combining aggregate and sequential features enable models like CNN-LSTM to leverage both information types [8].

Evaluation uses holdout test sets or cross-validation with multiple metrics including accuracy, precision, recall, F1-scores, and AUC-ROC [20]. Reporting multiple metrics provides complete performance, especially important given the class imbalance that leads to misclassification in educational applications.

Interpretability analysis extracts actionable insights. Tree-based models provide built-in feature importance scores revealing which behavioral and assessment features contribute most [5], [17]. Attention mechanisms visualize learned feature weights for individual predictions [21]. SHAP-based approaches compute feature importance for any model type through game-theoretic frameworks [22]. These explanations build educator trust and generate insights informing pedagogical improvements.

Table 1. Data Preprocessing Steps on the OULAD Dataset

Preprocessing Step	Before Processing (Raw Data)	After Processing (Cleaned Data)
Data Cleaning	Incomplete or duplicate records; early withdrawals included.	Removed missing entries and filtered out early withdrawals.
Feature Encoding	Categorical variables (e.g., Region = East Midlands).	Converted to one-hot encoded binary vectors (e.g., [0, 1, 0, 0, ...]).
Data Aggregation	Daily clickstream logs for each resource type.	Aggregated into weekly summaries to reduce dimensionality.
Normalization	Features with different scales (e.g., scores 0–100).	Scaled to 0–1 range using Min–Max normalization.
Class Balancing	Unequal "Pass/Fail" class distribution.	Applied SMOTE oversampling or class-weight adjustment.

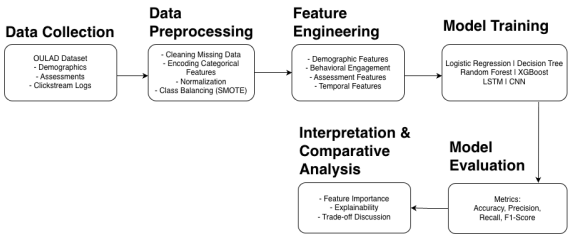


Figure 1. End-to-End Modeling Workflow for Student Performance Prediction

4 Comparative Performance Analysis

As shown in Table 2, synthesizing performance across modeling approaches on OULAD reveals clear accuracy-complexity trade-offs. Classical methods provide transparency at 75-80% accuracy. Ensemble methods achieve 85-91% while maintaining moderate interpretability through feature importance. Deep learning reaches 92-93% by modeling complex patterns and temporal dynamics.

4.1 Key Insights

Several critical patterns emerge from comparative analysis. First, clear trade-offs exist between complexity and interpretability. Simple models provide transparent logic educators can understand and trust, while complex models achieve higher accuracy as opaque black boxes [11], [19]. This matters significantly, requires explainability, and incorrect predictions may trigger inappropriate interventions.

Second, marginal accuracy gains must be weighed against marginal costs in data requirements, computational resources, and interpretability loss. Moving from 80% (Classical Machine Learning) to 90% (Ensemble Method) represents a substantial improvement, which increases complexity. Pushing from 90% to 93% in the Deep Learning may not justify added complexity for many practical applications, especially given real-world deployment uncertainties not captured in test metrics. Although deep learning models such as LSTM and CNN demonstrate superior predictive accuracy, they are often criticized for functioning as "black boxes," offering limited insight into how predictions are derived. Recent studies have addressed this by integrating Explainable Artificial Intelligence (XAI) methods, notably attention mechanisms within LSTM/CNN architectures that highlight influential time steps or features contributing to a prediction. Similarly, SHAP (Shapley Additive exPlanations) values have emerged as a model-agnostic technique to quantify the contribution of each feature to the final prediction. These methods provide interpretive transparency without compromising performance, enabling educators to understand not only what the model predicts, but also why.

Third, feature engineering proves more impactful than algorithm selection. Well-constructed behavioral and temporal features enable simple models to achieve respectable performance, while their absence limits sophisticated algorithms [9], [17]. This suggests institutional investments in data infrastructure and feature design expertise may yield greater returns than pursuing cutting-edge algorithms.

Fourth, important features remain remarkably consistent across modeling approaches. Engage-

ment volume, submission timeliness, and assessment performance consistently emerge as top predictors, whether measured through Logistic Regression coefficients, Random Forest scores [5], [17], [22]. This consistency provides confidence these factors genuinely matter for student success. Finally, optimal model choice are depends on dataset size. Classical Machine learning methods suit small institutional datasets ($<$ 1000 students) where limited data would cause deep learning overfitting [4]. Ensemble methods are suitable for medium datasets (1000-5000 students), providing strong performance without excessive complexity [5], [6]. Deep learning advantages emerge only for large datasets ($>$ 5000 students) where high capacity can be properly utilized [7], [8].

4.2 Ethical and Contextual Adaptation

Predictive systems must balance performance with fairness and transparency. Privacy protection, consent, and algorithmic bias mitigation are essential considerations. For Cambodia and similar contexts, local data characteristics such as mobile-first access and varying internet stability must be integrated into model adaptation. Future work will involve applying these models to EMIS data under the Ministry of Education, Youth and Sport (MoEYS) to explore early dropout detection systems, following approaches like Frontiers in Education

5 Discussion

The comparative analysis highlights both the potential and the limitations of current machine learning approaches for predicting student performance. While deep learning models such as LSTMs and CNNs achieve the highest reported accuracies, their complexity and lack of interpretability limit their adoption in real-world educational settings. Ensemble methods, especially Random Forest and XGBoost, offer a strong balance between predictive accuracy and transparency, making them more practical for institutional deployment. A recurring insight across studies is that feature engineering contributes more to prediction quality than algorithm choice. Behavioral indicators as login frequency, engagement duration, and submission timeliness con,sistently outperform static demographic features [23]. This emphasizes the importance of well-designed LMS data pipelines and careful preprocessing to ensure reliable and actionable predictions.

However, most existing studies rely heavily on the OULAD dataset, which reflects learning behaviors in Western onlinelearning contexts. Applying these models to developing regions like Southeast Asia requires context-specific adaptation. For instance, Cambodian students often access LMSs through mobile devices under inconsistent connectivity, which may influence engagement patterns. These contextual differences must be reflected in both feature design and model training to ensure fair and valid predictions [24]. Ethical and pedagogical considerations also arise from predictive modeling in education. Models trained on historical data risk reproducing systemic inequities if fairness and transparency are not explicitly addressed. Thus, incorporating explainable AI (XAI) techniques, such as SHAP or attention visualizations, is essential to support educator trust and informed decision-making. Lastly, translating predictions into effective interventions remains a research gap. High accuracy alone is insufficient unless insights guide concrete teaching actions such as early alerts or personalized feedback. Collaboration between data scientists and educators is therefore crucial for closing the loop between predictive analytics and improved student outcomes.

6 Future Work

6.1 Methodological Advances

Future research should prioritize interpretability alongside accuracy, developing education-specific explainability methods that maintain high performance while providing transparent insights educators can understand and act upon [19]. Local explanation techniques analyzing individual predictions could enable personalized intervention planning. Multimodal learning combining LMS logs with text, video, and survey data through cross-modal attention mechanisms may improve both performance and interpretability by highlighting important modalities. Temporal modeling deserves greater attention, given that learning is inherently dynamic. LSTM and CNN adapted for sequential educational data could capture complex temporal dependencies better than current approaches.

Table 2. Comparative Performance Analysis Selected Models on OULAD Dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	References
Logistic Regression	78.5	76.9	77.3	77.1	[1], [4]
Decision Tree	80.2	79.5	80.1	79.8	[5], [6]
Random Forest	85.7	84.3	85.7	84.9	[6], [7]
XGBoost	91.2	90.8	91.0	90.9	[7], [13]
LSTM	93.4	92.5	93.1	92.8	[2], [9]
CNN	92.2	91.7	91.9	91.8	[9], [17]

6.2 Dataset and Evaluation Improvements

The field would benefit from standardized datasets and evaluation protocols enabling meaningful cross-study comparison. Multi-institutional collaborative datasets would provide larger, more diverse samples. Standardized feature definitions and formats would facilitate reproducibility. Common evaluation metrics and procedures would eliminate confusion from varying assessment approaches. Reproducibility standards with code sharing would accelerate progress. Longitudinal research tracking students over extended periods would provide insights into the temporal dynamics that current cross-sectional studies miss. Long-term performance tracking would reveal stability and change patterns in the education. A model stability investigation would identify reliability factors. Intervention effectiveness analysis would validate practical utility. Seasonal and cyclical pattern identification could improve prediction timing and accuracy.

7 Conclusion

This paper synthesizes current approaches to student performance prediction, analyzing datasets, preprocessing methods, feature engineering, modeling techniques, and evaluation frameworks that highlight the importance of focusing more on the quality of the data than the complexity of each algorithm. While deep learning will provide the highest accuracy, the assembly methods offer the best balance of performance and interpretability for small to medium datasets. Current research is still limited by a focus on the Western institution that emphasizes accuracy over the interpretability. Furthermore, the future implementation in the CADT will be address the gaps by developing a contextualized, explainable, and intervention prediction system to support the data-driven decision making in Southeast Asian education.

Acknowledgment

We thank the Institute of Digital Research and Innovation at the Cambodia Academy of Digital Technology for supporting this research, the initial steps toward developing localized student performance prediction systems for Southeast Asian educational contexts..

References

[1] J. Kuzilek, M. Hlosta, and Z. Zdráhal, "Open university learning analytics dataset," *Scientific Data*, vol. 4, p. 170171, 11 2017.

[2] M. Hlosta, Z. Zdrahal, and J. Zendulka, "Ouroboros: Early identification of at-risk students without models based on legacy data," in *Proceedings of the Seventh International Learning Analytics & Knowledge Conference (LAK'17)*. Vancouver, BC, Canada: Association for Computing Machinery, 2017, pp. 6–15. [Online]. Available: http://dl.acm.org/citation.cfm?id=3027449

[3] R. S. J. d. Baker and P. S. Inventado, "Educational data mining and learning analytics," *Learning Analytics*, 2014.

[4] E. Amrieh, T. Hamtini, and I. Aljarah, "Mining educational data to predict student's academic performance using ensemble methods," *International Journal of Database Theory and Application*, vol. 9, pp. 119–136, 09 2016.

[5] C. Romero and S. Ventura, "Educational data mining and learning analytics: An updated survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 01 2020.

[6] T. Xie, Q. Zheng, W. Zhang, and H. Qu, "Modeling and predicting the active video-viewing time in a large-scale e-learning system," *IEEE Access*, vol. 5, pp. 11 490–11 504, 2017, publisher Copyright: © 2013 IEEE.

[7] F. Marbouti, H. Diefes-Dux, and K. Madhavan, "Models for early prediction of at-risk students in a course using standards-based grading," *Computers Education*, vol. 103, 09 2016.

[8] Y. Wang, L. Wang, Y. Li, D. He, T.-Y. Liu, and W. Chen, "A theoretical analysis of ndcg type ranking measures," *Journal of Machine Learning Research*, vol. 30, 04 2013.

[9] J. P. Gardner, C. Brooks, and R. S. J. d. Baker, "Evaluating the fairness of predictive student models through slicing analysis," *Proceedings of the 9th International Learning Analytics and Knowledge Conference (LAK '19)*, pp. 225–234, 2019.

[10] A. M. Shahiri, W. Husain, and N. A. Rashid, "A review on predicting student's performance using data mining techniques," *Procedia Computer Science*, vol. 72, pp. 414–422, 2015.

[11] Z. J. Zacharis, "A multivariate approach to predicting student outcomes in web-enabled blended learning courses," *The Internet and Higher Education*, vol. 27, pp. 44–53, 2015.

[12] A. Slim, G. L. Heileman, J. Kozlick, and C. T. Abdallah, "Predicting student success based on prior performance," in *Proceedings of the 2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. Orlando, FL, USA: IEEE, 2014, pp. 410–415. [Online]. Available: https://ieeexplore.ieee.org/document/7008697

[13] S. K. Yadav and S. Pal, "Data mining: A prediction for performance improvement of engineering students using classification," *World of Computer Science and Information Technology Journal*, vol. 2, no. 2, pp. 51–56, 2012. [Online]. Available: https://arxiv.org/abs/1203.3832

[14] M. Sweeney, J. Lester, and H. Rangwala, "Next-term student performance prediction: A recommender systems approach," *Journal of Educational Data Mining*, vol. 8, no. 1, pp. 22–51, 2016. [Online]. Available: https://arxiv.org/abs/1604.01840

[15] J. Whitehill, K. Mohan, D. T. Seaton, Y. Rosen, and D. Tingley, "Delving deeper into mooc student dropout prediction," *arXiv preprint arXiv:1702.06404*, 2017. [Online]. Available: http://arxiv.org/abs/1702.06404

[16] S. Moreno-Marcos, C. Alario-Hoyos, P. J. Muñoz-Merino, and C. D. Kloos, "Prediction in moocs: A review and future research directions," *IEEE Transactions on Learning Technologies*, vol. 12, no. 3, pp. 384–401, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8412110/

[17] B. Guo, R. Zhang, G. Xu, C. Shi, and L. Yang, "Predicting students' performance in educational data mining," in *Proceedings of the 2015 International Symposium on Educational Technology (ISET)*. IEEE, 2015, pp. 125–128. [Online]. Available: https://doi.org/10.1109/ISET.2015.33

[18] L. P. Macfadyen and S. Dawson, "Mining lms data to develop an 'early warning system' for educators: A proof of concept," *Computers & Education*, vol. 54, no. 2, pp. 588–599, 2010. [Online]. Available: https://doi.org/10.1016/j.compedu.2009.09.004

[19] W. Jokhan, V. Sharma, and P. K. Singh, "Early warning system as a predictor for student performance in higher education blended courses," *Studies in Higher Education*, vol. 44, no. 11, pp. 1900–1911, 2019. [Online]. Available: https://doi.org/10.1080/03075079.2019.1597030

[20] S. Gray and D. Perkins, "Utilizing early engagement and machine learning to predict student outcomes," *Computers & Education*, vol. 131, pp. 22–32, 2019. [Online]. Available: https://doi.org/10.1016/j.compedu.2018.11.004

[21] S. Hussain, Z. F. Muhsin, Z. A. Salal, P. Theodorou, F. Kurtoglu, and G. A. Hazarika, "Prediction model on student performance based on internal assessment using deep learning," *International Journal of Emerging Technologies in Learning*, vol. 14, no. 8, pp. 4–22, 2019. [Online]. Available: https://online-journals.

org/index.php/i-jet/article/view/10354

[22] F. Mi and D. Yeung, "Temporal models for predicting student dropout in massive open online courses," in *Proceedings of the 2015 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2015, pp. 256–263. [Online]. Available: https://ieeexplore.ieee.org/document/7417698

[23] B. Rienties and L. Toetenel, "The impact of learning design on student behaviour, satisfaction and performance," *Computers in Human Behavior*, vol. 60, pp. 333–341, 2016. [Online]. Available: https://doi.org/10.1016/j.chb.2016.02.071

[24] K. E. Arnold and M. D. Pistilli, "Course signals at purdue: Using learning analytics to increase student success," in *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge (LAK'12)*. Vancouver, BC, Canada: Association for Computing Machinery, 2012, pp. 267–270. [Online]. Available: https://dl.acm.org/doi/10.1145/2330601.2330666

An Efficient OCR Pipeline for Semi-Structured Khmer Documents Using Layout Analysis and Text-Type Classification

Sopanha Lay **Vichet Kao** **Seavchhing Kong** **Mengchhuong Ang** **Hongly Va**
Cambodia Academy of Digital Technology (CADT)
Phnom Penh, Cambodia
hongly.va@cadt.edu.kh

Abstract

The digitization of documents presents a significant challenge, particularly for complex, low-resource scripts like Khmer language. Standard Optical Character Recognition (OCR) systems often fail when faced with this mixed-media content, hindered by the complexity and scarcity of Khmer training data. This paper introduces a modular, resource-efficient pipeline designed to address these limitations through a classification-first approach that identifies text type prior to recognition, enabling the application of specialized OCR engines accordingly. At the core of our framework is a lightweight text-type classifier based on the MobileNetV3 architecture, integrated within a pipeline that leverages pre-trained models for layout analysis. Trained on a custom Khmer dataset, our experiments establish a performance benchmark for this difficult task: the classifier achieves an accuracy of **74.6%** on a held-out validation set. Crucially, its performance remains stable at **74.5%** within the full end-to-end pipeline, indicating the model is robust to noise from automated segmentation. Our work validates the classification-first strategy as essential for mixed-media OCR and provides a realistic benchmark for this low-resource task.

Keywords: *Optical Character Recognition (OCR), Khmer Language, Deep Learning, Text Type Classification, Document Analysis*

1 Introduction

As Cambodia accelerates its national digital transformation, digitizing its vast archive of official documents is a critical priority. Key documents, such as birth certificates, serve as foundational records for civil registration but exist primarily in paper format. This creates significant bottlenecks in data management, retrieval, and analysis, processes that remain manual, slow, and prone to error.

The primary challenge in automating the digitization of these documents is their **mixed-media content structure**: a combination of standardized, machine-printed template text and highly variable handwritten entries. Standard Optical Character Recognition (OCR) engines, which are typically optimized for only one text type, struggle to process this mixed content accurately [1].

This core problem is exacerbated by two further constraints inherent to the Khmer language context. First, the **Khmer script is inherently complex**, featuring a large character set and diacritics that challenge many OCR architectures. Second, there is a significant **scarcity of large-scale, annotated datasets** for Khmer document OCR [2],[3]. This data scarcity, in particular, makes it computationally infeasible to train a single, large, state-of-the-art model to handle all challenges simultaneously. Therefore, our work focuses specifically on solving the mixed-media problem through an efficient, modular approach practical within these low-resource constraints.

Initial research for this project considered a monolithic, end-to-end pipeline leveraging state-of-the-art Transformer-based models, such as TrOCR [4], which builds upon prior CNN-RNN architectures [5]. However, iterative analysis revealed that such an approach was impractical due to the aforementioned data and resource constraints. Consequently, a strategic pivot was made toward a more modular and efficient pipeline. This paper presents a novel hybrid approach that intelligently separates document analysis from targeted text recognition. The primary contribution is the development of a lightweight, highly accurate text-type classifier, built on a MobileNetV3 architecture [6]. This component enables the system to first identify the document's structure using robust tools like PaddleOCR [7], classify each text segment as ei-

ther printed or handwritten, and then apply the most suitable recognition engine for each type. The objectives of this work are to:

- **Establish a robust baseline for document analysis** by leveraging a pre-trained model (PP-DocLayout & RT-DETR-L) to perform layout segmentation on complex, semi-structured Khmer documents.
- **Design, train, and evaluate a lightweight text-type classifier** (MobileNetV3) to accurately distinguish between printed and handwritten Khmer text snippets, quantifying its performance in both ideal and real-world conditions.
- **Demonstrate the value of a classification-first OCR pipeline** by integrating the classifier and using quantitative metrics (CER) to prove its superiority over a naive, single-engine approach for mixed-media documents.

2 Related Work

Research into Khmer OCR has evolved significantly, moving from traditional machine learning methods targeting isolated characters to end-to-end deep learning systems capable of handling complex documents.

2.1 Evolution from Traditional Methods

Early research focused on recognizing isolated, printed Khmer characters using traditional machine learning. Techniques such as Support Vector Machines (SVMs) were successfully applied, achieving high accuracy on clean, well-segmented characters [8]. However, these methods were often highly dependent on the quality of pre-processing and character segmentation, making them less robust for noisy, real-world documents with varied layouts.

2.2 Modern Deep Learning Approaches

The advent of deep learning brought more powerful and flexible solutions [9]. Key advancements in Khmer OCR include:

- **Efficient Architectures:** Annanurov and Noor demonstrated that compact CNN models could effectively recognize handwritten consonants, proving the feasibility of efficient models for resource-constrained environments [10]. This insight is central to our work's focus on lightweight models.
- **Historical Documents:** For the unique challenges of historical texts, researchers developed specialized datasets like SleukRith for palm-leaf manuscripts [11] and applied sophisticated encoder-decoder models to handle stylistic variations and document degradation [12].
- **State-of-the-Art Models:** Recent work by Buoy et al. has advanced the field with Transformer-based models specifically designed to handle the long, unbroken text lines common in Khmer [3]. This work, along with the introduction of the KhmerST dataset by Nom et al. [2], has highlighted that even state-of-the-art architectures still face significant challenges with the Khmer script, underscoring the need for domain-specific solutions.

2.3 Research Gap

Despite these advances, a specific research gap remains in the processing of semi-structured official documents that contain a **mixture of printed and handwritten text**. Prior work has largely focused on either purely printed or purely handwritten text, or on different document types (e.g., historical manuscripts, scene text). The challenge of efficiently and accurately handling mixed-media content in a single pipeline for official documents like birth certificates is largely unaddressed. This project directly targets this gap by proposing a modular pipeline designed to classify and process this mixed content.

3 Methodology

To address the challenges of digitizing mixed-media Khmer documents, we designed a modular, multi-stage pipeline that prioritizes resource efficiency and accuracy. Our approach combines powerful pre-trained models for general tasks (like layout analysis) with a lightweight, custom-trained classifier for the specialized sub-task of differentiating text types. This section details the overall architecture and each of its core components, from initial image preprocessing to the final targeted text recognition.

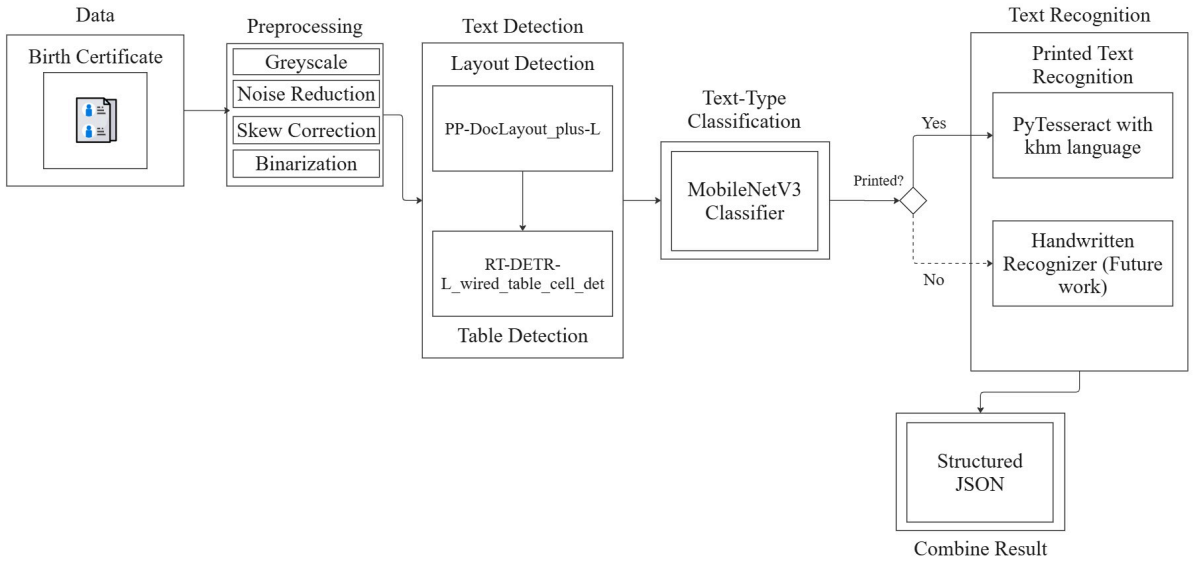


Figure 1: The high-level architecture of the proposed modular OCR pipeline. The path for handwritten text recognition is shown as a placeholder for future integration, for which a model like TrOCR could be used. This component was not implemented in the current work.

3.1 Pipeline Architecture Overview

The system ingests a scanned document image and processes it through a sequential workflow, as illustrated in **Figure 1**. The pipeline consists of four main stages: (1) automated image preprocessing to normalize input; (2) layout analysis to detect all text regions; (3) text-type classification to label each region as printed or handwritten; and (4) targeted text recognition where the appropriate OCR engine is applied based on the classification.

3.2 Document Preprocessing

To standardize input and improve the performance of subsequent modules, a series of automated image processing steps are applied using the OpenCV library. This crucial stage transforms raw, often-degraded scans into clean, machine-readable images. The visual progression of this workflow is detailed in **Figure 2**. The algorithmic process is as follows:

1. **Grayscale Conversion:** The input 3-channel RGB image is first converted to a single-channel grayscale format.

2. **Noise Reduction:** A Gaussian blur with a 5x5 kernel is applied to mitigate high-frequency noise from the scanning process without overly blurring character edges.

3. **Skew Correction:** The dominant angle of the text is detected and corrected. This is achieved by first applying adaptive thresholding to a temporary copy of the image, finding text contours, and calculating the orientation of the minimum area rectangle enclosing these contours. The original grayscale image is then rotated by this calculated angle to ensure all text lines are horizontal.

4. **Final Binarization:** A second adaptive thresholding is applied to the deskewed grayscale image. Unlike global thresholding, this method calculates a localized threshold for different regions of the image, making it highly robust to variations in lighting and background noise, resulting in a clean final output.

3.3 Layout and Text-Type Analysis

This phase performs a two-step analysis of the document's content. First, a pre-trained layout detector identifies all logical text regions. Each detected snippet is then passed to our novel, custom-trained text classifier, which labels it as either 'printed' or 'handwritten'. This critical step prepares each snippet for the appropriate recognition engine in the final stage. The models used are summarized in **Table 1**.

(a) Original Image **(b)** Grayscale Conversion **(c)** Skew Correction **(d)** Final Binarized Output

Figure 2: The automated preprocessing workflow from (a) Raw input, to (b) Grayscale, to (c) Deskewed, and finally (d) the Final binarized output.

Table 1: Core Models and Their Roles in the Pipeline

Component	Model Used	Role
Layout Analysis	PP-DocLayout	Detects logical regions (text, tables).
Table Cell Detection	RT-DETR-L	Detects cells in table.
Text type Classifier	MobileNetV3	Classifies regions as 'printed' or 'handwritten'.
Printed OCR	Tesseract v5	Transcribes text from 'printed' regions.

3.3.1 Layout Analysis with PaddleOCR

Our pipeline employs a two-stage approach for document structure detection, leveraging pre-trained models from the PaddleOCR framework [7]. This strategy allows us to analyze both the high-level document layout and the fine-grained table structures without needing to train custom detectors.

First, for overall document structure, we use the **PP-DocLayout_plus-L** model. This identifies high-level layout elements, such as text paragraphs and full table regions.

once a table region is identified, we use a specialized **RT-DETR-L_wired_table_cell_det** model trained for table cell detection. This model processes the cropped table area to precisely localize individual cells, preparing them for subsequent analysis.

3.3.2 Text-Type Classifier: Architecture, Training, and Inference

A key contribution of this work is the development of a lightweight and accurate classifier capable of distinguishing printed from handwritten Khmer text snippets. This section details its architecture, the dataset it was trained on, and its role during the pipeline's inference process.

Architecture and Dataset The classifier uses a **MobileNetV3-Large** architecture [6], pre-trained on ImageNet. Input snippets (224×224) pass through Inverted Residual Blocks and a Global Average Pooling layer, producing a feature vector fed to a fully-connected layer that outputs logits for printed and handwritten classes (Figure 3). Training used a custom dataset of 1,677 text regions from Cambodian birth certificates (956 printed, 721 handwritten), manually cropped and labeled with PPOCRLabelV3, a tool from the PaddleOCR suite. The dataset was stratified into 1,342 training and 335 validation samples (**Table 2**), with this fixed split used for all training and evaluation instead of k-fold cross-validation.

Table 2: Dataset Split for Text-Type Classifier

Class	Training	Validation	Total
Printed	765	191	956
Handwritten	577	144	721
Total	1,342	335	1,677

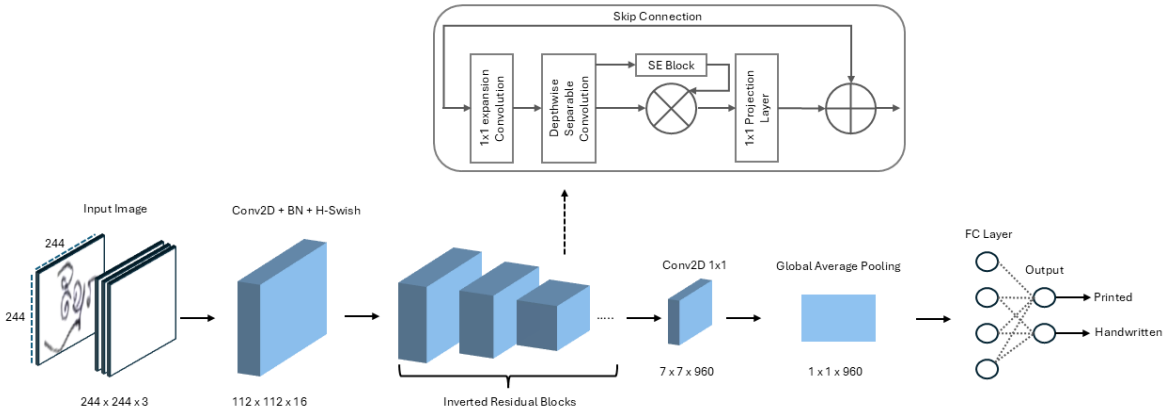


Figure 3: The proposed MobileNetV3-Large architecture for Khmer text-type classification. The main data flow passes through a convolutional backbone of Inverted Residual Blocks. A detailed view of the Inverted Residual Block, with its internal expansion, depthwise convolution, SE module, projection layer, and residual (skip) connection, is shown in the callout box.

Table 3: Text-Type Classifier Hyperparameters

Parameter	**Value**
Architecture	
Model Architecture	MobileNetV3-Large (x1.0)
Pre-trained Weights	ImageNet
Input Size	224 x 224 x 3
Training Hyperparameters	
Optimizer	Adam
Learning Rate	Piecewise (0.001 - 0.0001)
LR Scheduler	Step Decay (at epoch 15)
Batch Size	32
Epochs	25
Loss Function	Cross-Entropy Loss (CELoss)
Data Augmentation	RandCrop, RandFlip, AutoAugment

Training Training is performed using the Adam optimizer to minimize categorical cross-entropy loss. For this binary task, the model's final layer outputs two logits corresponding to each class, which are passed through a Softmax activation function. The loss is then calculated using one-hot encoded labels with the following formula:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=0}^{1} y_{i,c}\log(p_{i,c}) \qquad (1)$$

where $y_{i,c}$ denotes the one-hot encoded true label and $p_{i,c}$ the predicted probability for sample i and class c. Key training hyperparameters are detailed in Table 3.

Inference Workflow During inference, each text snippet detected by the layout analysis module is resized to 224x224 and passed through the trained MobileNetV3 classifier. Based on the predicted class, snippets are routed as follows:

- **Printed text:** Sent to Tesseract OCR, configured with the Khmer language pack, for transcription.
- **Handwritten text:** Bounding box and class label are preserved for future processing by a specialized handwritten recognizer.

This workflow allows the pipeline to efficiently direct each snippet to the most suitable recognition engine. The modular design also enables straightforward future integration of a dedicated handwritten OCR module, maintaining the pipeline's efficiency and adaptability.

3.3.3 Targeted Text Recognition

After the classifier sorts the text snippets, they are routed to specialized recognition engines. This targeted approach is the final and crucial stage of the pipeline.

Printed Text Recognition Snippets classified as 'printed' are sent to the Tesseract OCR engine for transcription. We use the standard Tesseract model configured with the official Khmer language pack, as detailed in **Table 4**. This engine forms the baseline for printed text recognition within our pipeline.

Table 4: Configuration for the Printed OCR Engine

Parameter	Value
OCR Engine	Tesseract v5.3.3
Language Pack	Khmer ('khm')
OCR Engine Mode (OEM)	3 (Default)
Page Seg. Mode (PSM)	3 (Default)

Handwritten Text Recognition Snippets classified as 'handwritten' are preserved with their bounding boxes and class labels but are not transcribed. Although developing a handwritten recognition model was beyond the current scope, the pipeline's modular design allows for the seamless integration of a specialized model such as a fine-tuned TrOCR in future work.

3.4 Evaluation Metrics

The performance of our custom text-type classifier is evaluated using four standard metrics derived from the confusion matrix components: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The formulas for Accuracy, Precision, Recall, and F1-Score are standard and defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)$$

These metrics provide a comprehensive assessment of the classifier's performance, which is critical for the overall success of the pipeline.

4 Experiments and Results

To evaluate our proposed pipeline, we conducted experiments on both the isolated text-type classifier and the end-to-end system. The results highlight both the potential of our approach and the challenges of bridging the gap between controlled validation and real-world application.

4.1 Experimental Setup

The classifier was trained in a cloud-based environment on a single NVIDIA T4 GPU, using PyTorch Lightning and Weights & Biases for experiment tracking. The pipeline was implemented in Python 3.9, leveraging PaddlePaddle, OpenCV, and Pytesseract.

4.2 Classifier Performance on Curated Snippets

The classifier's baseline performance was first established by evaluating it on the held-out validation set, which consists of 335 manually cropped text snippets. These snippets were sourced from real documents and intentionally represent a range of visual qualities, thereby providing a realistic benchmark for the classification task itself, isolated from any pipeline-induced artifacts.

The overall results of this evaluation are presented in **Table 5**, with a detailed breakdown of the predictions shown in the confusion matrix in **Figure 4**. The model achieved an overall accuracy of **74.6%**, a figure that highlights the inherent difficulty of reliably distinguishing between printed and handwritten Khmer text even under these controlled conditions.

A closer analysis of the errors in **Figure 4** reveals a key characteristic of the baseline model. The primary source of error is the misclassification of 74 handwritten snippets as printed, which is also reflected in the low Recall for the handwritten class (48.6%) in **Table 5**. This indicates a baseline bias in the model towards the 'printed' class. This behavior is likely because many instances of neat or simple handwriting in the dataset are visually similar to printed fonts, making them difficult for the model to differentiate without further contextual features. This baseline performance provides a crucial point of comparison for the end-to-end pipeline evaluation.

Table 5: Performance Metrics on Curated Validation Set

Class	Precision	Recall	F1-Score
Printed	70.9%	94.2%	80.9%
Handwritten	86.4%	48.6%	62.2%
Macro Avg.	78.6%	71.4%	71.6%
Accuracy		**74.6%**	

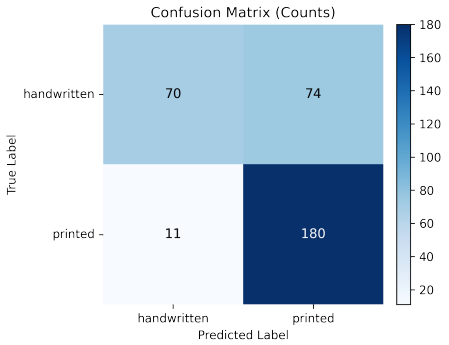


Figure 4: Confusion Matrix on the Curated Validation Set (335 Samples).

4.3 End-to-End Pipeline Performance and Challenges

While the classifier's performance on curated snippets is a strong proof-of-concept, its true effectiveness was evaluated by processing 20 complete, unprocessed birth certificates through the end-to-end pipeline. In this scenario, the layout analysis module first segments the document, and these automatically generated "wild" snippets are then fed to the text-type classifier.

The performance of the classifier within this live pipeline dropped significantly. The detailed confusion matrix from this real-world test is presented in **Table 6**, and the corresponding performance metrics are in **Table 7**.

Table 6: Confusion Matrix for Classifier in End-to-End Test

	Predicted Printed	Predicted Handwritten	Total
Actual Printed	241	166	407
Actual Handwritten	11	275	286

The results reveal a surprising and important finding. The classifier's end-to-end accuracy of **74.5%** is nearly identical to its baseline performance on the manually cropped validation set (74.6%). This consistency suggests that our model, having been trained on a realistic dataset of varied quality, is **robust to the additional noise and segmentation artifacts** introduced by the live pipeline. The analysis confirms that the primary bottleneck is not the pipeline's segmentation stage, but the fundamental difficulty of the classification task itself, which our 74.6% baseline accurately reflects.

Table 7: Performance Metrics for Text-type Classifier in End-to-End Pipeline Test

Class	Precision	Recall	F1-Score
Printed	95.6%	59.2%	73.1%
Handwritten	62.4%	96.2%	75.7%
Macro Avg.	79.0%	77.7%	74.4%
Accuracy		**74.5%**	

4.4 End-to-End Pipeline Performance and Error Analysis

When the classifier is integrated into the end-to-end pipeline, its overall accuracy remains stable at **74.5%** (**Table 7**). However, a comparison of the confusion matrices reveals a fascinating and critical shift in the model's error pattern.

While the baseline model's primary error was misclassifying 'handwritten' text as 'printed' (**Figure 4**), this pattern completely reverses in the live pipeline. As shown in **Table 6**, the predominant error becomes the misclassification of 166 'printed' snippets as 'handwritten'. This reversal strongly suggests that artifacts from the automated segmentation stage are the primary cause. When clean printed text is corrupted with noise or table lines, its visual features begin to resemble handwriting, causing the classifier to error. This demonstrates that while the overall accuracy is similar, the nature of the challenge changes significantly between curated and real-world pipeline conditions, providing a deeper insight into the model's behavior.

4.5 Downstream OCR Performance

To quantify the impact of our classification-first approach, we evaluated the performance of a standard OCR engine (Tesseract v5) on both printed and handwritten text snippets from our curated validation set. The primary metric used was the Character Error Rate (CER), a standard

for evaluating OCR accuracy. CER is calculated as the Levenshtein distance between the predicted text and the ground truth, normalized by the length of the ground truth text:

$$\text{CER} = \frac{S + D + I}{N} \tag{6}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the total number of characters in the ground truth. A lower CER indicates higher accuracy.

Table 8: Tesseract CER on Curated Text Snippets

Text Type	Snippets	CER (%)
Printed	191	43.71%
Handwritten	144	96.35%

The results clearly illustrate the necessity of a hybrid pipeline. While Tesseract is designed for printed text, it still yields a high CER of **43.71%**, underscoring the inherent difficulty of processing scanned, low-quality Khmer documents. As expected, its performance on handwritten text is extremely poor, with a CER of **96.35%**, rendering the output unusable. This confirms that a single OCR engine is insufficient for mixed-media documents and validates our approach of separating text types before recognition.

4.6 Comparative Analysis

Our classification-first pipeline represents a pragmatic compromise compared to other common strategies for mixed-media OCR. Unlike a monolithic **End-to-End Unified Model** (e.g., TrOCR) [4], which requires massive datasets and computational resources often unavailable for low-resource scripts like Khmer, our hybrid method is lightweight and modular. It is also more efficient than **Engine Blending**, which is computationally expensive as it processes each snippet with multiple engines and fails to solve the core problem if no specialized engine for a given text type exists. While the performance of our approach is dependent on the classifier's accuracy, it provides a practical and adaptable solution.

To provide a quantitative analysis of our approach's value, we compare its performance against a naive, single-engine strategy using the CER results from our validation set. A naive approach would apply Tesseract to all snippets, whereas our hybrid approach uses the classifier to route them first, as summarized in **Table 9**.

The data proves the necessity of our classification-first method. The naive approach is severely degraded by Tesseract's 96.35% CER on handwritten text, leading to an unusable overall CER of 65.42%. Our classifier acts as a critical filter, demonstrating that an intelligent, pre-recognition classification step is a mandatory component for achieving viable OCR on mixed-media documents.

4.7 Pipeline Output on Full Documents

To assess the system's real-world capabilities and provide a qualitative example of its operation, a complete birth certificate was processed through the entire pipeline. **Figure 5** provides a visual narrative of this process, illustrating the output at each critical stage and demonstrating a successful end-to-end run.

The process begins with the foundational layout analysis stages. First, the `PP-DocLayout_plus-L` model accurately identifies the document's high-level structure, separating text regions from the main table (**Fig. 5a**). Next, the specialized `RT-DETR-L` model processes the detected table region to precisely localize individual cells (**Fig. 5b**). The final classified output (**Fig. 5c**) is the culmination of the entire workflow and serves as a powerful proof-of-concept for our hybrid approach. This image visually confirms that the custom classifier has successfully labeled the machine-printed template text (blue) and the variable handwritten entries (red) on this document. This successful differentiation is the critical step that enables targeted and accurate text recognition in the final stage.

5 Conclusion and Future Work

In this paper, we presented a modular, resource-efficient OCR pipeline for semi-structured Khmer documents, successfully demonstrating the value of a classification-first approach. Our core contribution is the development and rigorous analysis of a lightweight text-type classifier.

Our key findings are threefold. First, we quantitatively proved the necessity of our approach: applying a standard OCR engine to

Table 9: Quantitative Comparison of Naive vs. Hybrid OCR Approach on the Curated Validation Set

Approach	Description	Resulting CER (%)
Naive Approach	Tesseract is applied to all 335 snippets (printed and handwritten). Performance is degraded by the engine's failure on handwritten text.	**65.42%**
Our Hybrid Approach	The classifier first separates text types. Tesseract is only applied to the 191 printed snippets. The 144 handwritten snippets are routed away, avoiding catastrophic errors.	**43.71%**

(a) Layout Detection **(b)** Table Cell Detection **(c)** Final Classified Output

Figure 5: Visual results from the end-to-end pipeline. (a) The layout analysis module correctly identifies high-level text and table regions. (b) A specialized model detects individual cells within the table. (c) The final output demonstrates successful classification, with printed text (blue) and handwritten entries (red) clearly distinguished.

the wrong text type results in a near-total failure (96.35% CER on handwritten text), making a pre-recognition classification step mandatory. Second, we established a realistic performance benchmark for this difficult task, showing our classifier achieves 74.6% accuracy on a validation set of varied-quality snippets. Third, we demonstrated the model's robustness, as its performance remains stable at 74.5% within the noisy, end-to-end pipeline. This indicates the primary challenge is the inherent visual complexity of the text, not segmentation artifacts.

Our work establishes a robust foundation for future development. The clear next steps are:

- **Robustness Enhancement:** While the model showed robustness, future work should still focus on improving the classifier by training it on an even more diverse dataset, including automatically segmented ("wild") snippets.
- **Handwritten Recognition Integration:** The pipeline's modular design allows for the seamless integration of a dedicated handwritten Khmer recognition model (e.g., a fine-tuned TrOCR) to achieve full end-to-end transcription.
- **Post-processing NLP:** The structured JSON output enables further natural language processing, such as developing modules for key-value pairing to link labels (like "Name") to their corresponding

transcribed text.

In conclusion, this project provides a significant step towards a practical solution for low-resource, mixed-media OCR, presenting not only a functional proof-of-concept but also a data-driven validation of the classification-first strategy and a clear, realistic roadmap for future research.

This paper was drafted by the authors. Language editing tools (e.g., AI-assisted writing support) were used only for grammar and phrasing; all content, results, and analyses are original and authored by the listed contributors.

References

[1] S. Patil and et al., "Enhancing optical character recognition on images with mixed text using semantic segmentation," *Journal of Sensor and Actuator Networks*, vol. 11, no. 4, p. 63, 2022.

[2] V. Nom, S. Bakkali, M. M. Luqman, M. Coustaty, and J.-M. Ogier, "Khmerst: A low-resource khmer scene text detection and recognition benchmark," 2024. Accepted at ACCV 2024, arXiv:2410.18277 [cs.CV], https://doi.org/10.48550/arXiv.2410.18277.

[3] R. Buoy, M. Iwamura, S. Srun, and K. Kise, "Toward a low-resource non-latin-complete baseline: An exploration of khmer optical character recognition," *IEEE Access*, vol. 11, pp. 128044–128060, 2023.

[4] M. Li, T. Lv, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "Trocr: Transformer-based optical character recognition with pre-trained models," *arXiv preprint arXiv:2109.10282*, 2021.

[5] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," 2015. arXiv:1507.05717 [cs.CV], https://doi.org/10.48550/arXiv.1507.05717.

[6] A. Howard, M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.

[7] C. Cui, T. Sun, M. Lin, T. Gao, Y. Zhang, and et al., "Paddleocr 3.0 technical report," 2025. arXiv:2507.05595 [cs.CV].

[8] P. Sok and N. Taing, "Support vector machine (svm) based classifier for khmer printed character-set recognition," in *Proceedings of the 2014 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pp. 1–9, 2014.

[9] S. Long, X. He, and C. Yao, "Scene text detection and recognition: The deep learning era," 2018. arXiv:1811.04256 [cs.CV], https://doi.org/10.48550/arXiv.1811.04256.

[10] B. Annanurov and N. Noor, "A compact deep learning model for khmer handwritten text recognition," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 584–591, 2021.

[11] D. Valy, M. Verleysen, S. Chhun, and J.-C. Burie, "A new khmer palm leaf manuscript dataset for document analysis and recognition: Sleukrith set," in *Proceedings of the 9th International Conference on Advances in Information Technology (IAIT)*, pp. 1–6, Nov. 2017.

[12] S. Born, D. Valy, and P. Kong, "Encoder-decoder language model for khmer handwritten text recognition in historical documents," in *Proceedings of the 14th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pp. 234–238, 2022.

Effective Resource Allocations Based on Network Slicing in O-RAN Networks

Soriya Prum[1] **Hongcheng Ngouch**[1] **Tha Khieng**[2] **Bondeth Hun**[2] **Sa Math**[2] **Tharoeun Thap**[2]
[1]Dept. of Telecommunication and Electronics Engineering
Royal University of Phnom Penh (RUPP), Cambodia
[2]Telecommunication Regulator of Cambodia (TRC),
Ministry of Post and Telecommunications, Cambodia
thaptharoeun@trc.gov.kh

Abstract

Recent research has increasingly focused on resource allocation strategies tailored for effective use of resources to meet the diverse Quality of Service (QoS) requirements of 5G networks, especially within the Open RAN (O-RAN) architecture. This paper presents an evaluation of the E2E QoS of millimeter wave communications in 5G O-RAN networks by investigating key QoS metrics, including throughput, latency, and signal-to-interference-plus-noise ratio (SINR). The result provides critical insights for the design and optimization of resource allocations in the O-RAN networks. Accordingly, we proposed an extensible Application (xApp) that resides in the near-real-time RAN intelligent controller (near-RT RIC), tackling the allocation of resource blocks based on QoS requirements of different services and varied traffic conditions in the near real-time scale.

Keywords: *Resource allocations, Network slicing, O-RAN, RIC*

1 Introduction

The ITU-R M.2083 defined three usage scenarios for IMT-2020, such as enhanced mobile broadband (eMBB), massive machine type communications (mMTC), and ultra-reliable and low latency communications (uRLLC) [1]. Specifically, eMBB requires substantial bandwidth and throughput, uRLLC demands ultra-low latency and high reliability, while mMTC emphasizes efficient resource allocations for massive device connectivity. These diverse requirements significantly make resource allocations in 5G networks, particularly with the O-RAN architecture, become even more complex that involving optimizing network resources to meet various service requirements. The rapid growth of diverse applications demands more resources in the 5G networks, including compute, storage, network, and radio resources (e.g., frequency, bandwidth, and physical resource blocks) that require both flexibility and precision in the resource management process.

To address this need, network slicing has emerged as the 5G's new features, enabling technology to meet the requirements of each service with abstracted of the physical resources. A network slice acts as a virtualized network that is isolated between different services to meet its requirements. Network slicing is a logical self-contained network across the network infrastructure, including RAN, transport, and core. However, this technology also introduces challenges in terms of isolation, customization, elasticity, and end-to-end lifecycle management.

Building on the foundations of C-RAN and V-RAN, O-RAN architecture advances the RAN through four guided key principles, including disaggregation, virtualization, open interfaces, and intelligent closed-loop control, enabling openness and interoperability, eliminating vendor lock-in [2]. Central to the O-RAN architecture are two types of RAN intelligent controller (RIC), which are categorized into near-real-time (near-RT) RIC and non-real-time (non-RT) RIC. Each RIC is designed to control and optimize RAN performance on different time sensitivities based on the decision-making process. The non-RT RIC, operating on time scales above 10 ms, is responsible for implementing the policy enforcement and training of AI/ML models. The near-RT RIC operates in the 1 ms to 10 ms time range and executes near-real-time control functions guided by the policies in the non-RT RIC through the A1 interface. Together, these components enable intelligent and flexible resource allocations

tailored to the QoS requirements of diverse network slices in O-RAN environments.

This paper presents a comprehensive E2E QoS analysis of mmWave communications within a 5G O-RAN network. Key performance metrics, including throughput, latency, and SINR, are evaluated to illustrate the important resource allocations tackling in the networks. We proposed a method that enables RAN to allocate the physical resource blocks (PRB) to each service with respect to their QoS, priority, and traffic conditions in near-real time. Additionally, a detailed discussion of the simulation configurations is provided to enhance the accuracy and reliability of experimental evaluations, providing deeper insights into the network performance in 5G O-RAN environments.

2 Literature Review

2.1 Resource Allocations

Recent studies in 5G network slicing have explored various resource allocation mechanisms to meet the diverse requirements across slices. Two approaches have emerged in the literature are share-based and reservation-based allocation schemes [3]. Share-based approaches assign a predetermined network share to each slice, which can be dynamically redistributed across nodes according to traffic variations. This model supports high efficiency with statistical multiplexing and is particularly suitable for elastic services, such as eMBB. However, share-based models do provide only statistical guarantees and poor isolation, thus making them less appropriate for mission-critical services requiring strict QoS compliance. Reservation-based approaches contrast and allow slices to request and pre-reserve guaranteed resources, with high isolation and hard performance guarantees. Though good for latency-critical services like URLLC, such methods have lower efficiency and higher complexity due to traffic prediction and admission control overhead.

The Xu et al. [4] survey details the 5G networks' distribution of power and bandwidth resources between various base stations. Methods like OFDMA and NOMA allow for the best possible speed with reduced interference. Systems use relays or cloud control to have more control over the resources. The mechanism is designed to cater to many users with varying

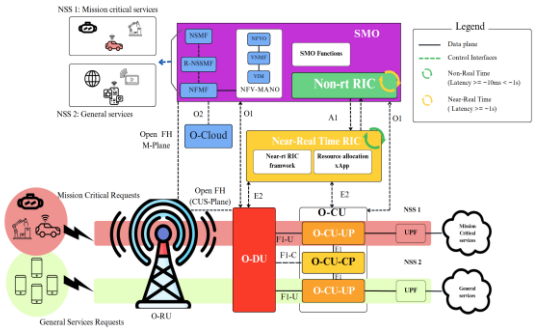


Figure 1. Slicing Architecture in O-RAN

demands. Most of the techniques are, however, elaborate, need accurate channel information, and are hard to make real-time modifications. They are also prone to being overcome by network slicing and ensuring fair service to all classes of users. These weaknesses limit their use in dynamic environments like O-RAN.

2.2 Network Slicing

Key enabling technologies of network slicing are Network function virtualization (NFV) and Software-defined networking (SDN). Their combined implementation offers greater scalability, flexibility, availability, and programming requirements while lowering both capital expenditures (CAPEX) and operational expenditures (OPEX). The decoupling of control and data plane in SDN enables the independent deployment of the traffic forwarding and decision-making. A centralized SDN controller maintains a global, logical view of the network infrastructure, optimizing traffic management, service provisioning, and resource allocation. The NFV-MANO manages the resource allocations for VNFs and the life cycle of the network slices, including the creation, updating, and termination of network functions by VM or containers [5].

In accordance to O-RAN Alliance slicing architecture [6], the near-RT RIC, and the O-CU Control Plane (O-CU-CP) are shared across multiple network slices, while each slice is assigned a dedicated O-CU User Plane (O-CU-UP) instance to ensure isolation at the user-data level as illustrated in Figure 1. As defined by ETSI NFV-MANO [7], NFV Orchestrator (NFVO) can reside in the Service Management Orchestrations (SMO) responsible for providing FCOM functionality, termination of O2 IMS, and

Table 1. Proposed Algorithm Scheme

1:	**SET** NSS 1 = mission-critical services
2:	**SET** NSS 2 = general services
3:	**Phase 1:** Resource Allocation
4:	Preconfigure resources for slices, RB_{total}
5:	**If** Slice = NSS 1 **then**
6:	**SET** RB_{crit} && Latency < 100 ms && Priority level = "high"
7:	**else**
8:	**SET** $RB_{general}$ && Latency < 300 ms && Priority level = "low"
9:	**end if**
10:	**Phase 2:** Operation
11:	Clustering U = {U_{crit}, U_{gen}}
12:	**If** NSS 1's traffic > threshold, **then**
13:	$RB_{crit\_congest} = (U_{crit} \times RB_{ms}) + RB_{crit}$
14:	**else** NSS 2's traffic > threshold
15:	$RB_{gen\_congest} = (N_{gen} \times RB_{gen}) + RB_{general}$
16:	**end if**

potentially NFO and O2 dms support, while also offering some element management capabilities. The Virtual Network Function Manager (VNFM) also maps to SMO, handling NFO-related operations and potentially terminating parts of the O2 DMS interface. The Virtualized Infrastructure Manager (VIM) maps to O-Cloud and provides IMS and DMS functionalities for managing VM-based virtualized resources. In O-RAN, network slice components are followed by 3GPP [8], including the Network Slice Management Function (NSMF), Network Slice Subnet Management Function (NSSMF), and Network Function Management Function (NFMF). NSMF interprets network slice requirements into network slice subnet requirements and distributes those requirements to the NSSMF. NSSMF is responsible for the management and orchestration of the network slice subnet instances (NSSIs). Connected to the NFV-MANO, they can dynamically allocate resources to network functions. Finally, NFMF is responsible for managing the lifecycle and configuration of individual network functions (such as VNFs and PNFs) involved in the network slice.

3 Proposed Scheme

This paper proposed a resource allocation based on network slicing, which is controlled by the xApp residing in the near-rt RIC of the O-RAN architecture. Two types of services are being sliced, one for mission-critical services (e.g., MCPTT voice, MCPTT video, and MCPTT data) and the other for general applications (e.g., internet traffic, streaming services). The purpose of this work is to allocate dedicated PRBs to each type of application in isolation to meet its QoS requirements, including throughput and latency. The network slice for critical applications is resource prioritized, providing lower latency, while the general services slice is given with lower resource priority with best-effort in terms of latency.

The resource management is divided into two critical phases: (i) Resource allocations and (ii) Operations, where the resource allocations xApp dynamically adapts resource blocks distribution in response to near real-time network conditions and traffic demands. The system process is illustrated in Table 1.

3.1 Resource Allocations

Both slices operate on a shared the same frequency spectrum and are provided with a pre-configured number of resource blocks. At the O-DU level, each slice is assigned a MAC scheduler for resource blocks allocations. The resource blocks are being distributed in terms of bandwidth in a normalized traffic condition, where 20 MHz bandwidth is preconfigured for the mission-critical services slice and 30 MHz is pre-allocated for the general services.

At a higher level, which is the near-rt RIC, the xApp is a container designed to perform resource allocation per slice to meet the QoS of each service in the network. The xApp monitors and collects the data from E2 nodes (e.g., O-CU and O-DU) and O-RU through E2, which is a logical interface in O-RAN. The A1 policy, which defined the optimization objectives such as latency and throughput target, guided the execution of the xApp through O1 interface.

3.2 Operations

During runtime, UEs are clustered into two different types based on their service requested characteristics: mission critical service users and general service users. UEs with similar service types are mapped to its VM, hosting a dedicated service, schedule with pre-allocated resource blocks. In case of a congestion event in both slices, the xApp is triggered and modifies the

update of the E2 performance through near-rt RIC guided by the A1 policy from the non-rt RIC. In addition, the number of resource blocks required to be reallocated is calculated in the accounts of the total number of users and the resource blocks needed for each user, in addition to the allocated resources to each service respectively. For instance, if the critical slice experiences overload, it temporarily borrows the reserve resource blocks to uphold its QoS guarantees. Once the network conditions are normalized, it releases those borrowed resources and restores the original resource allocation scheme.

Let RB_{total} denotes the total of resource blocks, $RB_{reserved}$ represent the reserved resource block, RB_{crit} as the resource block allocated to mission-critical services, and $RB_{general}$ for the resource block of general service. The RB_{total} can be expressed as:

$$RB_{total} = RB_{reserved} + RB_{crit} + RB_{general} \quad (1)$$

Let RB_{avail_cri} denote 80% of the total available resource block reserved for mission-critical services, and RB_{avial_gen} represents 20% of the total available resource block reserved for general services. The equation for resource allocation based on traffic conditions can be expressed as:

$$RB_{reserved} = RB_{avail\_cri} + RB_{avial\_gen} \quad (2)$$

$$RB_{avail\_cri} = RB_{reserved} \times \frac{80}{100\%} \quad (3)$$

$$RB_{avail\_gen} = RB_{reserved} \times \frac{20}{100\%} \quad (4)$$

Using (3) and (4), the $RB_{reserved}$ can be calculated as:

$$RB_{reserved} = (RB_{reserved} \times \frac{80}{100\%}) + (RB_{reserved} \times \frac{20}{100\%}) \quad (5)$$

Let $RB_{crit\_congest}$ be the total number of resource blocks required for U mission-critical users, and $RB_{gen\_congest}$ denotes the total number of resource blocks required for N general service users. The equations can be expressed as:

$$RB_{crit\_congest} = (U_{crit} \times RB_{ms}) + RB_{crit} \quad (6)$$

Table 2. Simulation Parameters

Parameters	Value
Center frequency	30 GHz
Bandwidth	80 MHz
Numerology	2
BS Tx power	40 dBm
Antenna height	BS = 25 m, UE = 1.5 m
MIMO antenna	BS = 8×8, UE = 4× 4
Antenna gain	BS = 8 dBi, UE = 5 dBi
Noise Fig.	BS = 7 dB, UE = 10 dB
Pathloss model	Umi_Streetcanyon
Channel condition	Line of sight (LoS)
Peak data rate	20 Gbps
Max UDP packet	10, 000
UDP packet size	1024 bytes

$$RB_{gen\_congest} = (N_{gen} \times RB_{gen}) + RB_{general} \quad (7)$$

where U_{crit} is the number of mission-critical users, RB_{ms} is the number of resource blocks required for each mission-critical user, N_{gen} is the number of mission-critical users, RB_{gen} is the number of resource blocks required for each general user.

4 Simulation and Results

The simulation is conducted using the open-source ns-O-RAN Open RAN simulation platform, which enables large-scale 5G network simulations by incorporating 3GPP-compliant channel models and a detailed implementation of the full 3GPP RAN protocol stack within the ns-3 framework**,** augmented with an O-RAN-compliant E2 interface [9].

4.1 Simulation Setup

Figure 2 illustrates a simulation scenario of an urban environment with two gNBs positioned at the height of 25 meters and six user equipment's (UEs), each at the height of 1.5 meters, moving randomly within a 1000 square meters Line-of-sight (LOS) region at the speed ranging from 1 m/s to 9 m/s. The scenario is designed by the Dense Urban-eMBB test environment as specified in ITU-R M.2412 [10] and is illustrated in Table 2, which defines configuration

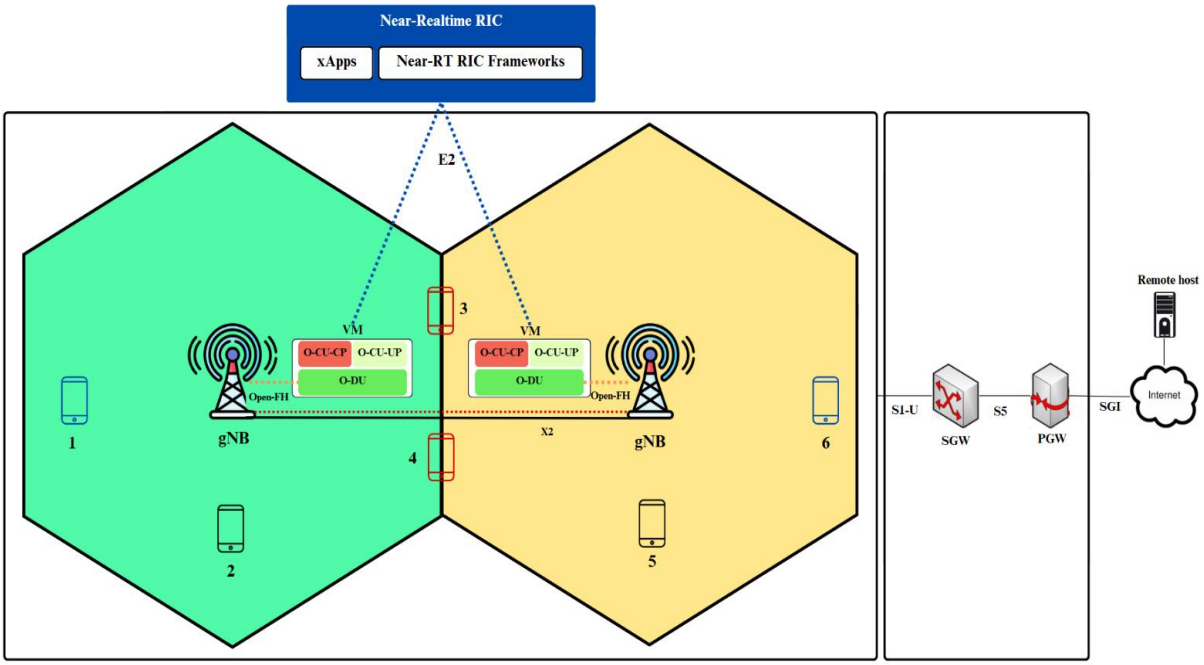


Figure 2. Simulation Scenario

parameters for this simulation. Additionally, the base station dynamically detects and manages HO based on an adaptive signal-to-noise ratio (SNR) threshold, initiating the process when the SNR drops below -5 dB. Finally, the experiment is conducted through a simulation spanning a total duration of 120 seconds.

4.2 Performance Metrics

To evaluate the impact of mmWave communications within the O-RAN framework, this study investigates key QoS metrics, including throughput, latency, and signal-to-interference-plus-noise ratio (SINR).

- Throughput refers to the rate at which information is correctly delivered and received via a connection in a period. It can usually be measured in megabits per second (Mbps), depending on the manner and extent to which the network is utilized. Throughput is the performance seen by end users and applications, and hence is one of the most important metrics of end-to-end network quality of service and efficiency. Throughput can be affected by numerous factors along the transmission path. They are channel conditions, i.e., signal strength, noise levels, and interference from other users or proximate cells, which will degrade signal quality and affect successful data transfer. Such other factors as packet loss, i.e., packet loss due to congestion or transmission, and retransmissions, bandwidth-consuming, and delay, thus reducing the effective throughput.
- Latency also known as delay, is the time it takes for a packet of data to travel from the source (e.g., remote host) to reach the destination (e.g., mobile users), typically measured in milliseconds. Mean delay per user device is an indicative measure and can be used to quantify network performance.
- Signal-to-interference-plus-noise-ratio (SINR) is a performance metric used to evaluate the quality of a wireless communication link. The higher the SINR, the clearer, better, and more dependable the signal will be, and therefore, the increased data rate and decreased error rates are obtained. In contrast, unfavorable channel conditions signaled by low SINR can cause packet losses, retransmissions, low data rates, or even trigger handovers.

4.3 Results

Figure 3 presents the throughput performance of all six UEs under the simulated mmWave communications in a 5G O-RAN scenario. The mean throughput, measured in megabits per second (Mbps), is plotted against the individual UEs. The results show that UE 4 and UE 6 achieved the highest throughput, receiving

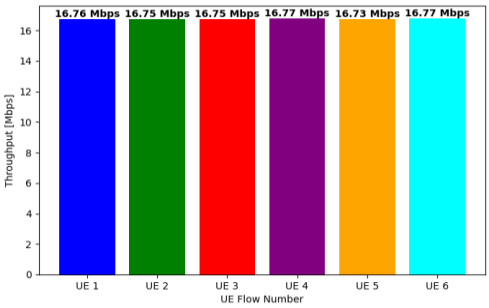


Figure 3. Comparison of Average Throughput

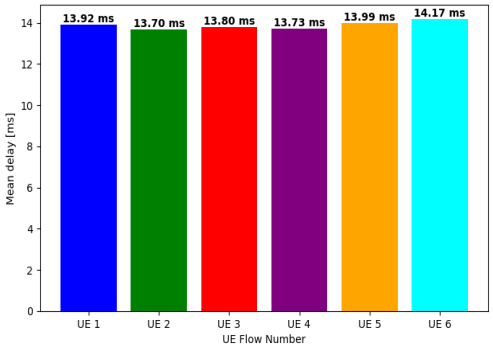


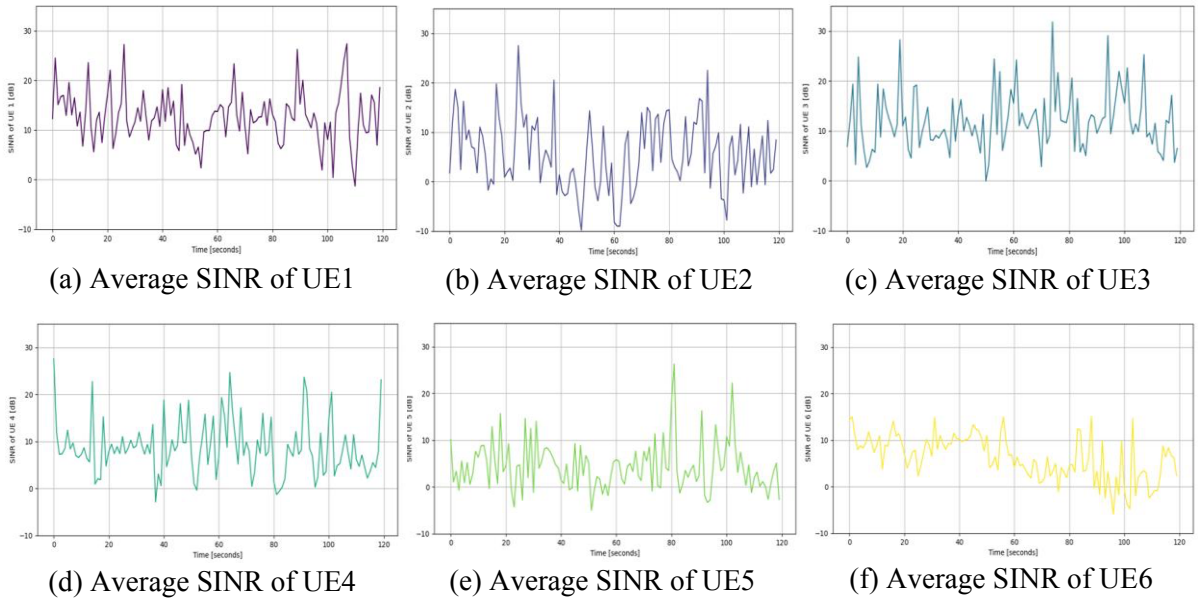
Figure 4. Comparison of Average Delay

packets from the remote host at 16.77 Mbps. UE 1 followed closely with a throughput of 16.76 Mbps. UE 2 and UE 3 recorded a slightly reduced throughput of 16.75 Mbps, while UE 5 experienced the lowest throughput at 16.73 Mbps, potentially attributable to link instability, network congestion, or channel coding.

Figure 4 illustrates the end-to-end delay comparison across all UEs. The mean delay, expressed in milliseconds (ms), reveals that UE 2 experienced the lowest delay at 13.70 ms, with UE 3 and UE 4 closely trailing at 13.80 ms and 13.73 ms, respectively. UE 1 displayed a marginally higher delay of 13.92 ms, while UE 5 recorded a delay of 13.99 ms. UE 6 exhibited the highest mean delay at 14.17 ms, suggesting momentary congestion or less favourable channel conditions during the transmission interval.

Figure 5 indicates the SINR for all UEs over a 120-second simulation period, where values are sampled at 1-second intervals. UE 1 recorded the highest average SINR at 12.61 dB, but its SINR pattern revealed pronounced fluctuations between 1 dB and 27 dB, particularly during the final 40 seconds, suggesting transient degradation likely driven by mobility-induced channel variation or degraded propagation conditions. UE 3 followed closely with an average SINR of 11.99 dB with values oscillating between 0 dB to 31 dB and showed a relatively smoother signal trajectory, with intermittent enhancements possibly reflecting the near distance between the base station and the UE, favorable link reestablishment or beam alignment following mobility events. UE 4, with an average of 8.66 dB, exhibited dense oscillations throughout the simulation from -2 dB to 27 dB, maintaining a consistent but moderately turbulent channel quality, which may indicate frequent transitions between coverage zones due to user mobility. In contrast, UE 2 displayed a lower average SINR of 5.55 dB and experienced persistent dips and variations from -9 dB to 27 dB, suggesting it encountered degraded coverage areas, unfavorable positioning, ineffective beamforming, or suboptimal handover performance. UE 6 averaged 6.27 dB and demonstrated a more gradual decline in SINR over time, punctuated by brief recoveries with the highest SINR value of 15 dB, indicative of partial adaptation to the dynamic environment. Radio resource utilization and favorable signal conditions. UE 5 registered the lowest average SINR performance at 4.40 dB, characterized by severe and frequent signal degradation from -4 dB to 26 dB, which maybe attributed to unfavorable positioning, ineffective beamforming, or repeated handover failures.

The overall performance of the simulated scenario reveals the measurements of QoS metrics mmWave communications within the 5G O-RAN, including throughput, latency, and signal quality. UEs located closer to their serving gNBs or in the beamforming path demonstrated minimal delay, underscoring the advantages of stable LOS conditions and reduced handover frequency. In contrast, UEs that experience lower SINR profiles experienced noticeable performance degradation. Notably, some UEs with moderate SINR values performed better in latency metrics than those with higher SINR, indicating that radio link quality alone does not dictate end-to-end performance. Instead, transient factors such as handover interruptions, resource scheduling delays, and varying channel conditions also play a crucial role and



(a) Average SINR of UE1 (b) Average SINR of UE2 (c) Average SINR of UE3

(d) Average SINR of UE4 (e) Average SINR of UE5 (f) Average SINR of UE6

Figure 5. Average SINR across UEs Over Time

significantly affect the perceived user experience. These conditions suggest that ensuring QoS in O-RAN environments requires not only managing radio conditions but also implementing adaptive resource allocation strategies that respond dynamically to real-time network behavior.

5 Conclusion

In this paper, a comprehensive analysis of end-to-end QoS for mmWave communications in a 5G O-RAN system was conducted, with multiple performance metrics under a dense urban deployment scenario. The simulation results reveal that factors such as handover frequency, link stability, and scheduling delays also have a substantial impact on overall performance, reinforcing the need for real-time channel adaptation and dynamic resource allocation to maintain service continuity. Thus, the future work will focus on developing a slice-aware xApp operating within the near-real-time RAN Intelligent Controller (Near-RT RIC), designed to dynamically allocate resource blocks based on the QoS requirements of different service types and real-time traffic conditions. This approach aims to enhance adaptability and resource efficiency in the O-RAN networks.

Ackowledgement

This research was supported by the Ministry of Post and Telecommunications, Kingdom of Cambodia.

References

[1] M Series, "IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond," Recommendation ITU-R M.2083-0, ITU Recommendation Sector, Geneva, Sep. 2015.

[2] M. Polese, L. Bonati, S. D'Oro, S. Basagni and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research Challenges," in IEEE Communications Surveys & Tutorials, vol. 25, no. 2, pp. 1376-1411, Secondquarter 2023, doi: 10.1109/COMST.2023.3239220.

[3] A. Banchs, G. de Veciana, V. Sciancalepore and X. Costa-Perez, "Resource allocation for network slicing in mobile networks," in IEEE Access, vol. 8, pp. 214696-214706, 2020, doi: 10.1109/ACCESS.2020.3040949.

[4] Y. Xu, G. Gui, H. Gacanin and F. Adachi, "A survey on resource allocation for 5G heterogeneous networks: Current research, future trends, and challenges," in IEEE Communications Surveys & Tutorials, vol.

23, no. 2, pp. 668-695, Secondquarter 2021, doi: 10.1109/COMST.2021.3059896.
[5] R. Su et al., "Resource allocation for network slicing in 5G telecommunication networks: A survey of principles and models," in IEEE Network, vol. 33, no. 6, pp. 172-179, Nov.-Dec. 2019, doi: 10.1109/MNET.2019.1900024.
[6] O-RAN Working Group 1, “Slicing architecture 14.00,” O-RAN, Alfter, Germany, document O-RAN.WG1. Slicing-Architecture-v14.00 Technical Specification, 2025.
[7] ETSI GR NFV-IFA 046. Network Functions Virtualisation (NFV) Release 5; architectural framework; report on NFV support for virtualisation of RAN, May 2023. Available: https://encr.pw/gv4bf.
[8] 3GPP, Management and Orchestration, Architecture framwork, Standard TS 28.533, Version 19.1.0, 3rd Generation Partnership Project (3GPP), March 2025.
[9] A. Lacava, M. Polese, R. Sivaraj, R. Soundrarajan, B.S. Bhati, T. Singh, T. Zugno, F. Cuomo, and T. Melodia, "Programmable and customized intelligence for traffic steering in 5G networks using open RAN architectures," arXiv:2209.14171 [cs.NI], pp. 1-15, October 2022. Available: https://arxiv.org/pdf/2209.14171v3.
[10] M Series, "Guidelines for evaluation of radio interface technologies for IMT-2020," Recommendation ITU-R M.2412-0, ITU Recommendation Sector, Geneva, Oct. 2017

Building a Khmer NER Benchmark from Health News Data Towards Event Extraction

Cheaminh Chiep, Natt Korat, Vathna Lay, Rottana Ly
Cambodia Academy of Digital Technology
cheaminh.chiep@student.cadt.edu.kh

Abstract

Khmer Named Entity Recognition (NER) is a sub-task in Khmer Natural Language Processing (NLP) that extracts information to locate and classify named entities in Khmer text into predefined categories, such as names of persons, organizations, and locations. As a low-resource language, there is a lack of high-quality datasets for Khmer NER. This study addresses the lack of an NER dataset for the Khmer language, particularly in the public health domain. The Khmer Health Event Extraction Dataset (KHEED) was introduced. It comprises 525 annotated articles (5,980 sentences) from Khmer health news, covering eight entity types: Disease, Pathogen, Location, Human Count, Organization, Symptom, Medication, and Date. To evaluate the performance of Khmer NER on the proposed dataset, five Khmer-compatible NLP models were selected, and from the experimental results, XLM-RoBERTa Base achieved the best performance with a moderate F1-score of 0.7646. The KHEED Dataset will be publicly available and serve as the foundation and benchmark for future Event Extraction (EE) Datasets.

Keywords: *Khmer NLP, Khmer NER, Event Extraction, Health News Data, KHEED*

1 Introduction

Cambodia faces two major health issues: non-communicable diseases (NCDs), which cause 64% of deaths [1], and ongoing communicable diseases, especially in rural areas [1]. Khmer-language health news, provides important real-time information on outbreaks and health policies. However, this information is often unstructured and hard to access due to challenges in processing the Khmer language [2].

Named Entity Recognition (NER) is the process of highlighting text spans of important information, which are known as entities [3]. In the context of public health in Cambodia, these entities are utilized to represent diseases, pathogens, locations, human counts, organizations, dates, and medications, which are critical in monitoring health trends such as disease outbreaks. Event Extraction (EE), on the other hand, is a task of identifying event types, triggers, and arguments. NER and EE automation help accelerate the analysis of health trends and summarize Khmer health news, eliminating the need for time-consuming work by human annotators [4]. Developing NER and EE systems for the Khmer language presents numerous challenges due to the language's complexity and the scarcity of resources, including well-annotated corpora and pre-trained models [5]. In addition, Khmer writing does not use spaces like other languages, which means that an accurate word tokenization tool is required, as this is crucial in Natural Language Processing (NLP) [6].

In this paper, the contributions are:

1. Introduction of a novel domain-specific dataset: Creation of the Khmer Health Event Extraction Dataset (KHEED), comprising a 525-annotated corpus of named entities from Khmer news articles in public health.
2. Extension of general NER capabilities: Provision of a domain-specific dataset extending the general NER dataset for familiar entities.
3. Baseline performance evaluation: Evaluation of five existing Khmer-compatible NLP models on the NER task using the KHEED dataset.
4. Foundation for event extraction (EE): A fundamental step toward supporting the future creation of the EE dataset.

The paper is organized as follows. In Section 2, related work is presented, followed by the preparation of the KHEED Dataset in Section 3. Section 4 outlines the experiments of the selected Khmer-compatible models on the KHEED. Last but not least, a Discussion is provided in Section 5 before the conclusion and future work discussed in Section 6.

2 Related Work

This section reviews the previous work of NER and EE in the health domain, including existing datasets and models, as well as common methodologies employed.

2.1 Existing NER and EE for English and other Languages

Numerous datasets are published for English and other widely spoken languages. They are crucial for researchers to develop NLP models for NER and EE tasks. *CoNLL-2003* is a common gold standard NER dataset, a shared task of annotations of news wire articles with four familiar entities [7]. *OntoNotes 5.0* is another well-known annotation of a large multilingual corpus with 18 named entities [8]. More recently, *MultiNERD* was introduced as a large silver standard NER dataset. This dataset consists of a wide range of 10 languages, including English, Chinese, and Dutch, covering an extensive 15 NER categories, such as familiar entities (e.g., Person, Location, Organization) and less common entities (e.g., Biological entities, Mythological entities) [9]. MultiNERD addresses the work of manual annotation by automatically detecting entities using a combination of *mBERT + BiLSTM + CRF* and benchmarking against gold standard datasets, achieving high scores. For the EE task, the *2005 Automated Content Extraction (ACE)* is a standard collection of annotated entities, events, and relations at the sentence level for English, Chinese, and Arabic [10]. *MAVEN*, another EE dataset, is a general domain event detection dataset designed to expand upon previous EE datasets, covering a wide range of 168 event types and 118,732 event mentions, which is significantly larger than the ACE dataset [11]. However, these general domain datasets are not specialized in the health domain and lack support for the Khmer language.

2.2 Previous Work in Health Event Extraction

There are many ongoing efforts to support health EE due to its applications in disease monitoring. One of the key contributions in 2013 is the *GENIA* dataset, which is composed of biomedical articles with annotations of biomolecular entities and events [12]. In recent times, the *BAND* dataset was introduced to address the lack of publicly available surveillance on health news in the English Language [13]. This dataset comprises 1,508 samples sourced from news and emails. In addition, it offered several evaluation tasks, including NER, EE, and Question Answering (QA), developed by epidemiology experts. Similarly, the *SPEED++* dataset, a multilingual EE dataset comprising four languages collected from social media, features seven event types and 20 argument roles, aiming to provide early warnings of health hazards [14]. While these datasets are valuable for a standard health EE dataset, they may not be suitable for the Khmer news reporting style.

2.3 Existing Khmer NER and EE Resources

Publicly available resources for Khmer NER and EE tasks are minimal [15]. Currently, Khmer researchers are developing models and datasets [16]. However, an elementary open-source dataset was released to foster the development of NER for the Khmer Language [17]. This dataset provided labeling of the persons and location entities in Khmer text for over 47,700 sentences. In terms of Khmer-compatible models, Transformer-based models were often fine-tuned for downstream tasks. Models like mBART50 [18] were experimented with in Khmer NLP research and have achieved moderate results [16]. Despite these advances, datasets for various domains with complex annotation depths are still necessary.

2.4 Standard Annotation Schema and Tools

Effective NER and EE rely on well-defined annotation schemas and tools. A standard scheme is the BIO (Beginning, Inside, Outside) tagging scheme, the entities are broken down into tokens or sub-words, the first token is labeled as ’B-’ and the consequence tokens are labeled as ’I-’ while tokens that are not part of an entity are labeled as ’O [19]. An extension to this scheme

is the BIOES (Beginning, Inside, Outside, End, Single) tagging scheme, where similar principles are applied, but the ending token is labeled as 'E-' and single-token entities are labeled as 'S-' [19]. For event extraction, the scheme includes an event type supplement with event triggers and event arguments.

Various tools are available to support annotation tasks. A popular open-source labeling tool is *Label Studio*, a customizable interface for manual annotation workflow [20]. Another well-known annotation tool is *Docanno*, which provides support for text classification tasks and other tasks, such as sequence labeling and sequence-to-sequence tasks [21]. In addition, *Prodigy* is a modern annotation tool that enables the efficient development of AI systems [22].

2.5 Standard Benchmark and Evaluation Metrics

Evaluation of NER and EE can be done at three different levels. A standard evaluation for NER is the entity-level evaluation, which involves the correct and exact matching of predicted entities against their proper tags. The metrics used at this level are Precision, Recall, and F1-score [23]. Precision measures the percentage of correct predictions among all predicted entities. Recall measures the proportion of correct predictions among all actual entities. F1-score the the harmonic mean of the two metrics. A more precise evaluation is the event-level or tuple-level evaluation, which assesses the entire event extraction structure, including the event type, triggers, and argument roles, against a manually annotated gold standard dataset [24]. The metrics used at this level are more complex and can vary depending on their real-world application. The most advanced level is the document-level event evaluation, where models at this level learn the context within an entire document rather than just sentences [25].

3 KHEED Dataset

This section describes in detail the creation of the KHEED dataset.

3.1 Data Collection

We collected 28,057 health news articles from 2020 to 2025 from nine Khmer news outlets: VOA Khmer, RFA Khmer, Fresh News, Camboja, Khmer Times, Kampuchea Thmey Daily, DAP News, Cambodia Express News, and Khmer Breaking News (Table 1). Articles were scraped and thoroughly cleaned by removing HTML tags, after which they were segmented into sentences.

Table 1. Data Collection Summary

Source	Articles
VOA Khmer	1,120
RFA Khmer	1,907
Fresh News	15,018
Camboja	59
Khmer Times	106
Kampuchea Thmey Daily	1,657
DAP News	3,803
Cambodia Express News	1,775
Khmer Breaking News	2,612
Total	**28,057**

3.2 Annotation Schema

We annotate eight entity types: Disease, Pathogen, Location, HumanCount, Organization, Symptom, Medication, and Date. Table 2 provides names and examples for each type. We use these canonical labels throughout the paper and dataset.

These entities capture the essential details in a health news report. The distribution shows that Organization, Disease, and Location are the most frequent entities in the dataset, accounting for 73.4% of all entities. Date, HumanCount, Pathogen, Symptom, and Medication are less common; this creates an imbalance in the dataset, which could potentially reduce model performance. Figure 1 shows the distributions of each entity type.

3.3 Annotation Process

To improve the efficiency of manual annotation as a single annotator, a pre-annotation Python script was developed to assist in capturing entities in the articles. First, word segmentation was applied using the Khmer-NLTK library [26] in Python. Next, the script utilizes a predefined dictionary lookup, prefixes and suffixes with subsequent tokens, and regular expressions to automatically annotate entities. These pre-annotations are then imported into Label Studio for manual verification and correction, while also

Table 2. Named Entity Types and Examples

Entity	Examples
DIS	ជំងឺអេដស៍ (AIDS)
PAT	វីរុសកូរ៉ូណា (Coronavirus)
SYM	ក្អក (Cough)
MED	ថ្នាំអង់ទីប្យូទិច (Antibiotic)
HUM	៣០នាក់ (30 people)
DAT	ថ្ងៃទី១៥ ខែមករា ឆ្នាំ២០២៥ (January 15, 2025)
LOC	ខេត្តកំពង់ចាម (Kampong Cham Province)
ORG	មន្ទីរពេទ្យកាល់ម៉ែត (Calmette Hospital)

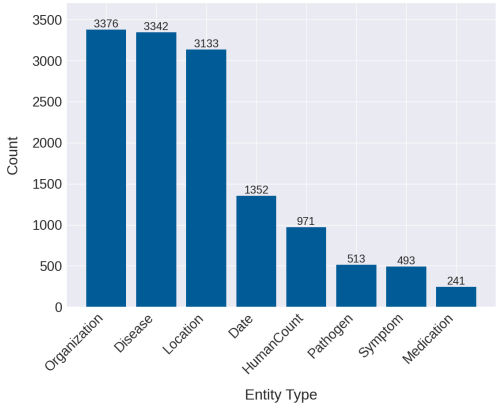


Figure 1. Distribution of Entity Types

annotating additional entities. Finally, verified annotations are exported as JSON, structured with text, entity spans, and labels.

Several challenges were encountered during annotation, including ambiguous entities, where organization names could be either specific names or general categories (e.g., ”royal government”). Another challenge was the dual-category entities, which are entities that could mean different things depending on the context (e.g., ”school” as an institution vs. physical location).

3.4 Data Processing

During processing, sentences containing fewer than five tokens were removed to focus on informative sentences, resulting in a final dataset of 5,980 sentences. The annotated data were then BIO tagged using KhmerNLTK [26] for tokenization and stored in JSON format. The data was then split into a ratio of 80:10:10 for training, validation, and testing, using random sampling to reduce bias and improve generalization.

4 Experiments

This section outlines the fine-tuning and evaluation of five Khmer-compatible NLP models on the KHEED dataset.

4.1 Model Selection

The models for evaluation were chosen based on their multilingual capabilities and pre-training in the Khmer language: XLM-RoBERTa-Khmer-Small, BERT-Khmer-Small-Uncased, PrahokBART-Base, XLM-RoBERTa-Base, and BiLSTM-CRF. Table 3 summarizes the specifications in each model.

4.2 Experimental Setup

The models were trained on a server equipped with an NVIDIA GeForce RTX 4070 Ti SUPER GPU and 16GB of memory. Hyperparameters were tuned on the validation set, with a fixed seed (42) for reproducibility. Early stopping was implemented to prevent the model from overfitting, halting training when the validation loss stopped improving. The best model was selected based on validation loss. Table 4 presents the hyperparameter settings for each model.

4.3 Model Performance

The evaluation was performed at the entity level, following standard NER evaluation metrics, as calculated using the formulas below.

- **Precision**: The ratio of correctly predicted positive tokens to the total predicted

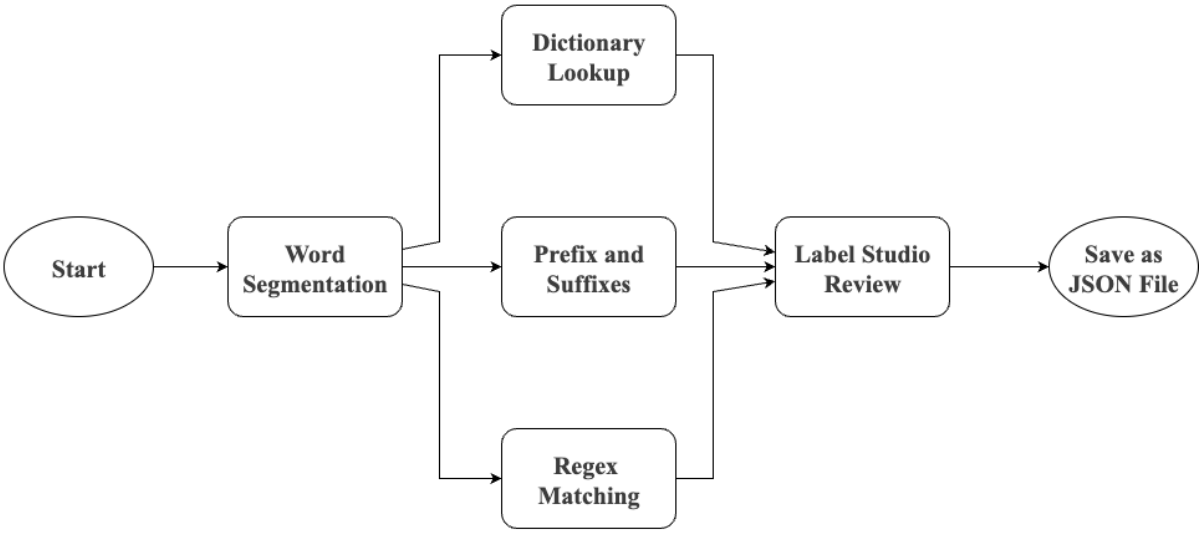


Figure 2. Annotation Process Flowchart

Table 3. Model Architecture Specifications

Model	Type	Architecture	Parameters
XLM-RoBERTa-Khmer-Small	Encoder	Transformer	49M
BERT-Khmer-Small-Uncased	Encoder	Transformer	29.1M
PrahokBART-Base	Enc-Dec	Transformer	62M
XLM-RoBERTa-Base	Encoder	Transformer	125M
BiLSTM-CRF	Sequential	LSTM+CRF	N/A

positive tokens, calculated as:

$$P = \frac{TP}{TP + FP} \tag{1}$$

- **Recall**: The ratio of correctly predicted positive tokens to all actual positive tokens, calculated as:

$$R = \frac{TP}{TP + FN} \tag{2}$$

- **F1-Score**: The harmonic mean of Precision and Recall, providing a balanced measure of performance:

$$F1 = \frac{2 \times (P \times R)}{P + R} \tag{3}$$

Table 5 present the overall performance of each model.

XLM-RoBERTa Base achieves the highest F1 Score of 0.7646, demonstrating its effectiveness on Khmer text. BERT-Khmer Small follows closely with a 0.7048 F1-score, showing that small pre-trained models are also effective. The BiLSTM-CRF model significantly underperformed compared to transformer-based models. Table 6 shows the scores of each model by entity type.

The results reveal a moderate performance in Date, HumanCount, Location, Pathogen, and Disease, benefiting from the patterns of these entities in news articles. A slightly lower performance for the Organization is due to excessively long text spans, which are challenging for the models to capture accurately. For medication entities, the presence of pathogen and disease names in the medication entities confused the models. Regarding symptoms, the models recognized them but failed to capture the whole entities.

Table 4. Model Hyperparameter Configurations

Model	Epochs	Batch Size	Learning Rate
XLM-RoBERTa-Khmer-Small	5	16	0.00002
BERT-Khmer-Small-Uncased	7	16	0.00002
XLM-RoBERTa-Base	8	16	0.00002
PrahokBART-Base	15	4	0.00002
BiLSTM-CRF	23	32	0.001

Table 5. Entity-Level Performance Results

Model	Precision	Recall	F1-Score
XLM-RoBERTa Base	**0.7006**	**0.8414**	**0.7646**
BERT-Khmer Small	0.6575	0.7595	0.7048
XLM-RoBERTa Khmer-Small	0.5943	0.7793	0.6744
PrahokBART Base	0.5732	0.6641	0.6153
BiLSTM-CRF	0.5147	0.5609	0.5368

5 Discussion

In this section, we will discuss about the current state of Khmer compatible NLP models, their limitation and capabilities.

5.1 Error Analysis

We defined four errors types, which include false positives, where the model mistakenly over predict entities, missed annotations could also cause false positives, boundary errors arising from segmentation issues, and type confusions, including cases where hospitals were tagged as Location instead of Organization due to contextual cues like នៅ (*at*). While tagging hospitals as Location in phrases like នៅមន្ទីរពេទ្យ (*at the hospital*) is contextually valid, consistently tagging them as Organization could improve model learning by reducing ambiguity. Figure 3 compares the errors of the predictions made by the models on the test set.

These errors reflect our challenges in the Khmer NER task, such as overfitting, inconsistent annotations, and opportunities for schema improvement. The following examples illustrate these issues:

- **False Positive**: The word ដង់ស៊ីតេ (*density*) in អត្រាកើតមានជំងឺថ្មី (ដង់ស៊ីតេ) (*density of new disease cases*) was incorrectly predicted as a Disease entity, but it is not an entity. This error occurred due to its proximity to the keyword ជំងឺ (*disease*).
- **Missed Annotation**: The word ខែមករា (*January*) in នៅខែមករាឆ្នាំនេះ (*in January this year*) was predicted as a Date entity, but the gold standard incorrectly labeled it as a non-entity, indicating annotation inconsistency.
- **Boundary Error (Under-Segmentation)**: The entity ជំងឺសរសៃឈាមបេះដូង (*heart disease*) was partially tagged as ជំងឺសរសៃឈាម (*vascular disease*) as a Disease entity, missing បេះដូង (*heart*). The model failed to recognize the correct entity boundary.
- **Boundary Error (Over-Segmentation)**: The entity ប្រទេសថៃ (*Thailand*) was correctly predicted as a Location entity, but the gold standard incorrectly annotated only ថៃ (*Thai*) as Location, indicating an annotator error.
- **Missed Entity**: The entity ជំងឺទឹកនោមផ្អែម (*diabetes*) in ក្តីព្រួយបារម្ភអំពីជំងឺទឹកនោមផ្អែម (*concern about diabetes*) was not predicted

Table 6. F1-Score Performance by Entity Type

Entity Type	XLM-RoBERTa Khmer-Small	XLM-RoBERTa Base	BERT-Khmer Small	BiLSTM-CRF	PrahokBART Base
Date	0.85	**0.86**	0.83	0.62	0.65
Disease	0.74	**0.78**	0.77	0.67	0.74
HumanCount	0.74	**0.84**	0.77	0.47	0.49
Location	0.73	**0.83**	0.77	0.56	0.66
Medication	0.30	**0.46**	0.00	0.08	0.12
Organization	0.53	**0.69**	0.59	0.43	0.50
Pathogen	0.63	0.73	0.65	0.55	**0.78**
Symptom	0.51	**0.58**	0.46	0.37	0.55

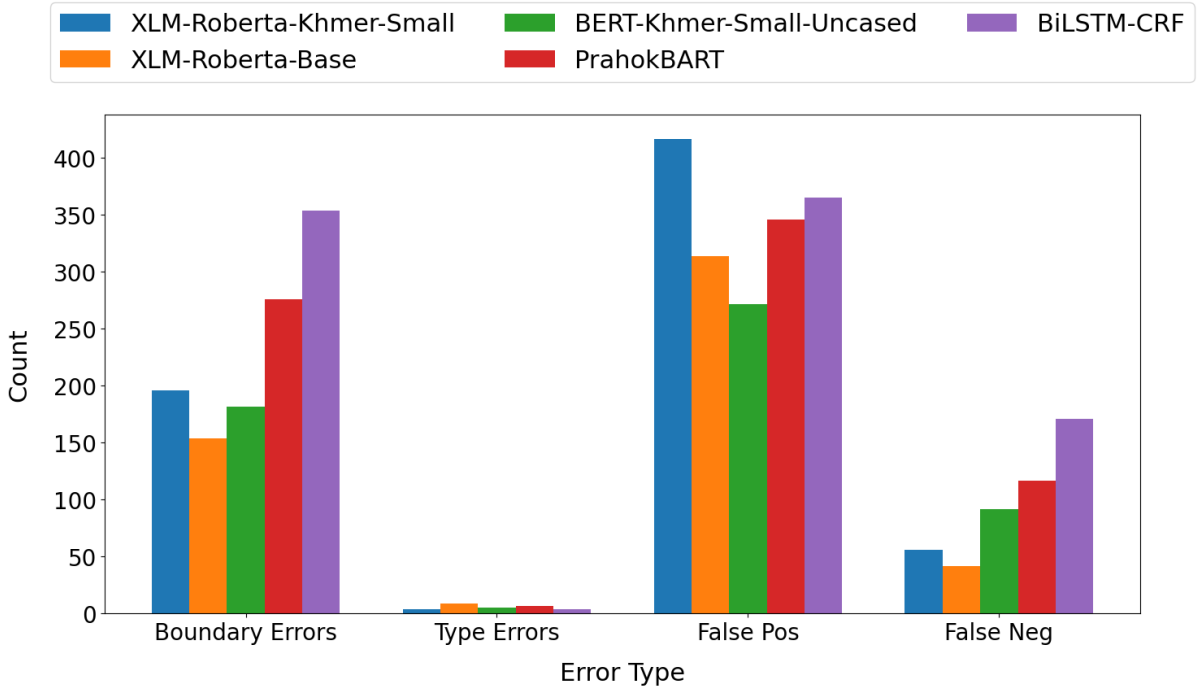


Figure 3. Model Performance by Error Type

as a Disease entity, due to segmentation issues.

- **Type Confusion**: The entity អាជ្ញាធរជាតិប្រយុទ្ធនឹងជំងឺអេដស៍ (*National Authority for Combating AIDS*) was partially tagged as ជំងឺអេដស៍ (*AIDS*) as a Disease entity instead of the correct Organization entity. This reflects confusion caused by disease terms within organizational names.
- **Schema Ambiguity (Type Confusion)**: The entity នៅមន្ទីរពេទ្យបង្អែកខេត្តកំពង់ធំ (*at Kampong Thom Provincial Hospital*) was predicted as an Organization entity, but the gold standard labeled it as Location. This confusion in the model indicates that it is unable to distinguish between organization and location in different contexts.

5.2 Current State

Our results demonstrate that the models are at a practical state, but still require significant advancement to compare with state-of-the-art models for other languages.

6 Conclusion

In closing, we'll summarize the contributions of this study and look closer into the extensibility of the dataset.

6.1 Limitations

Several limitations temper these results: severe entity class imbalance (e.g., Medication and Symptom constitute <5% of annotations) significantly degrades performance on rare types (F1 $\leq$ 0.46 for Medication); single-annotator labeling risks inconsistency, as evidenced by boundary errors and type confusion (e.g., hospitals misclassified as Location); domain restriction to formal news text limits applicability to clinical records or conversational Khmer; pre-annotation biases from dictionary and regex heuristics may introduce systematic errors; and moderate overall performance (best F1 Score $\sim$0.76) remains far below English NER benchmarks ($\sim$0.90+), highlighting the need for larger, higher-quality training resources.

Future work should focus on multi-annotator refinement, balanced sampling or loss reweighting, cross-domain data integration, advanced Khmer tokenization, and larger pre-trained models to improve robustness, generalization, and clinical utility. KHEED lays a critical foundation for advancing health NLP in Khmer.

6.2 Primary Contributions

This work presents two significant contributions to Khmer NLP community. First, we introduced the KHEED dataset, comprising of 525 articles with eight entity type, this dataset provide an automation to monitor health events from Khmer heatlh news. Second, we have evaluated the performance of Khmer compatible NLP models accommodated by the analysis about their limitations and key challenges for future research. We commit to publicly release the dataset, models, and documentation to facilitate similar research. The dataset, splits, tokenization scripts, and training configurations will be released under a CC BY-NC 4.0 license on Github (`https://github.com/CADT-LLM/kheed.git`), fostering reproducibility.

6.3 Future Research

Several promising research avenues emerge from this work. Enhancing KHEED to a large-scale health corpus that incorporates medical literature, clinical notes, and patient forums could improve the model's understanding, potentially eliminating boundary detection and data imbalance. Building upon entity recognition, studying the relationship between entities and the linking of events would allow for a more sophisticated analysis of health news. We plan to develop an EE schema, annotating triggers and arguments for events such as Outbreak (Disease, Location, Date, HumanCount), which will enable more sophisticated health trend analysis. Entity relation extraction and document-level event modeling are also promising areas of research.

References

[1] World Health Organization and Ministry of Health Cambodia. Prevention and control of noncommunicable diseases in cambodia: Progress report 2018. *WHO UniATF*, 2018. Accessed: 2025-11-11.

[2] H. Kaing, S. Deth, S. Uth, S. Sorn, S. Sar, S. Sam, and S. Nop. Towards tokenization and part-of-speech tagging for khmer. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 21(1):1–22, 2021.

[3] K. Pakhale. Comprehensive overview of named entity recognition: Models, domain-specific applications and challenges. *arXiv.org*, 2023.

[4] C. Yu, Z. Cao, P. Zhao, D. D. Zeng, T. Luo, and J. Wang. Named entity recognition for epidemiological investigation in covid-19. In *2023 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6, 2023.

[5] S. Jiang, S. Fu, N. Lin, and Y. Fu. Pretrained models and evaluation data for the khmer language. *Tsinghua Science and Technology*, 27(4):709–718, 2022.

[6] H. Kaing, C. Ding, M. Utiyama, E. Sumita, S. Sam, S. Seng, K. Sudoh, and S. Nakamura. Towards tokenization and part-of-speech tagging for khmer: Data and discussion. *ACM Trans. Asian Low Resour. Lang. Inf. Process.*, 20(6):104:1–104:16, 2021.

[7] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147,

2003.

[8] R. Weischedel, S. Pradhan, M. Palmer, W. Xiong, and H. Xiong. Ontonotes release 5.0. `https://catalog.ldc.upenn.edu/LDC2013T19`, December 2013. LDC2013T19.

[9] S. Tedeschi and R. Navigli. Multinerd: A multilingual, multi-genre and fine-grained dataset for named entity recognition (and disambiguation). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 801–812, Seattle, United States, July 2022. Association for Computational Linguistics.

[10] C. Walker et al. Ace 2005 multilingual training corpus, 2006.

[11] X. Wang, W. Lu, Z. Cao, Z. Yu, Y. Padigela, R. Deng, H. Chen, F. Sun, and K. Li. Maven: A massive general domain event detection dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1652–1671, Online, November 2020. Association for Computational Linguistics.

[12] J.-D. Kim, Y. Wang, K. Takemoto, F. Berard-Dufresne, K. B. Cohen, M. Colosimo, J. Kim, J.-J. Kim, A. Névéol, S. Pyysalo, and J. Tsujii. The genia event extraction shared task, 2013 edition - overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[13] Z. Fu, M. Zhang, Z. Meng, Y. Shen, D. Buckeridge, and N. Collier. Band: Biomedical alert news dataset. *arXiv preprint arXiv:2305.14480*, 2023.

[14] T. Parekh, J. Kwan, J. Yu, S. Johri, H. Ahn, S. Muppalla, K.-W. Chang, W. Wang, and N. Peng. Speed++: A multilingual event extraction framework for epidemic prediction and preparedness. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12936–12965, Miami, Florida, 2024. Association for Computational Linguistics.

[15] S. Jiang, S. Fu, N. Lin, and Y. Fu. Pretrained models and evaluation data for the khmer language. *Tsinghua Science and Technology*, 27(4):709–718, 2021.

[16] R. Buoy, N. Taing, S. Chenda, and S. Kor. Khmer printed character recognition using attention-based seq2seq network. *HO CHI MINH CITY OPEN UNIVERSITY JOURNAL OF SCIENCE-ENGINEERING AND TECHNOLOGY*, 12(1):3–16, 2022.

[17] R. Buoy. Khmer ner dataset. `https://huggingface.co/datasets/rinabuoy/khmer-ner-dataset`, 2023. Accessed: 2023-08-04.

[18] Y. Tang, C. Tran, X. Li, P.-J. Chen, N. Goyal, V. Chaudhary, J. Gu, and A. Fan. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint*, arXiv:2008.00401, Aug. 2020.

[19] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[20] V. Naik, P. Patel, and R. Kannan. Legal entity extraction: An experimental study of ner approach for legal documents. *International Journal of Advanced Computer Science and Applications*, 14(1), 2023.

[21] O. Irrera, S. Marchesin, F. Shami, and G. Silvello. Doctron: A web-based collaborative annotation tool for ground truth creation in ir. *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025.

[22] C. Macri, I. Teoh, S. Bacchi, M. Sun, et al. Automated identification of clinical procedures in free-text electronic clinical records with a low-code named entity recognition workflow. *Journal of Medical Systems*, 46(7):1–10, 2022.

[23] R. Yacouby and D. Axman. Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. *Proceedings of the first workshop on evaluation and comparison of NLP systems*, pages 70–79, 2020.

[24] P. Liu, Y. Guo, J. Lei, and G. Li. Tagging

schemes can do more in named entity recognition: Take chinese as an example. *2022 International Conference on Natural Language Processing and Knowledge Engineering (NLPKE)*, pages 1–6, 2022.

[25] M. Tong, B. Xu, S. Wang, M. Han, Y. Cao, J. Zhu, C. Shi, et al. Docee: a large-scale and fine-grained benchmark for document-level event extraction. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3084–3096, 2022.

[26] P. V. Hoang. Khmer natural language processing tookit. `https://github.com/VietHoang1512/khmer-nltk`, 2020.

This page is intentionally left blank.

ISBN : 978-9924-9918-3-0

ACET 2025

RESEARCH | SHARE | CONNECT

More Information

acet@cadt.edu.kh

https://acet.cadt.edu.kh