# Does Visual Rendering Bypass Tokenization? Investigating Script-Tokenizer Misalignment in Pixel-Based Language Models

Anonymous ACL submission

## Abstract

While pixel-based language modeling aims to bypass the sub-word tokenization bottleneck by rendering text as images, recent multimodal variants such as DualGPT reintroduce text tokenizers to improve autoregressive performance. We investigate a fundamental question, **does visual rendering truly decouple a model from tokenization constraints?** Focusing on four Indonesian low-resource local languages that have their own non-Latin scripts (i.e., Javanese, Balinese, Sundanese, and Lampungnese), we evaluate the impact of script-tokenizer alignment within the DualGPT architecture. Our results show that, despite visual rendering, reintegrating a text tokenizer into the architecture reintroduces the same issue that pixel-based language modeling aims to resolve, which is the tokenizer misalignment problem. Despite having lower OOV and fertility rates, we show that the Llama 2 tokenizer performs significantly worse than a custom tokenizer, with improvements of up to 30.15 chrF++. Our findings serve as a warning for future multimodal variants, as text tokenizers remain a significant barrier to equitable models.

## 1 Introduction

While many Visual Language Models (VLMs) claim broad multilingual support (see Figure 1; Bercovich et al., 2025; Gemma Team, 2025; Wang et al., 2025; Bai et al., 2025; OpenAI, 2025), their performance on low-resource languages with non-Latin scripts remains poor (Han et al., 2025). A key challenge is script-tokenizer misalignment, where tokenizers trained predominantly on English data exhibit orthographic bias, causing excessive fragmentation of non-Latin scripts where single semantic units are split into multiple sub-word pieces.

Recent Pixel-based approaches introduced by Rust et al. (2022) avoid tokenization by processing rendered text as images. PixelGPT (Chai et al., 2024) extends this with autoregressive variants, reintroducing text tokenizers in multimodal variants such as DualGPT. **If the visual modality truly decoupled the model from tokenization constraints, any tokenizer with high vocabulary coverage should yield similar results**. In contrast, our DualGPT results on four Indonesian low-resource local languages and scripts show that script-tokenizer misalignment remains a key bottleneck once tokenizers are reintroduced, even in pixel-based models.

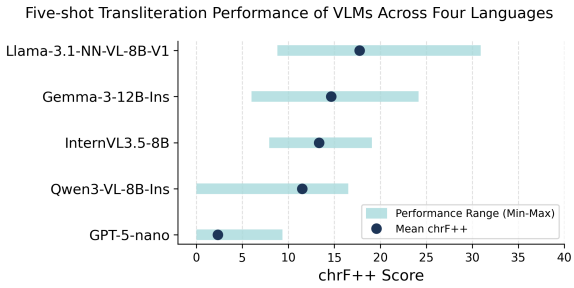


Figure 1: General VLMs image transliteration performance averaged across Javanese, Balinese, Sundanese, and Lampung on the NusaAksara evaluation dataset. Llama-3.1-NN-VL-8B-V1 is the Nemotron-Nano variant.

## 2 Background and Motivation

**Tokenization and Cross-lingual Disparities.** Current LLMs predominantly use BPE-based tokenization (Sennrich et al., 2016; Abagyan et al., 2025), including Llama 2 (Touvron et al., 2023), which PixelGPT adopts (Chai et al., 2024). However, BPE tokenizers trained on English-dominant corpora create inequities for underrepresented languages through excessive subword fragmentation (Foroutan et al., 2025; Petrov et al., 2023). Research on tokenization-free models (Clark et al., 2022; Xue et al., 2022) also offers an alternative approach for handling multilingual data. However, its application on generation tasks remains underexplored (Sun et al., 2023).

**Pixel-based Language Modeling.** PIXEL (Rust et al., 2022) addresses tokenization disparities by rendering text as images, theoretically enabling unlimited multilingual vocabulary. Kesen et al. (2025) demonstrate strong low-resource performance with this approach. However, rendering strategies affect patch diversity and learned representations (Lotz et al., 2023; Tatariya et al., 2024), with Tatariya et al. (2024) noting that structured rendering may reintroduce tokenization-like constraints.

**Multimodal Variants.** Recent work explores combining pixel-based and tokenization-based approaches. PixelGPT (Chai et al., 2024) extends PIXEL to autoregressive modeling, with its multimodal variant DualGPT integrating both visual rendering and text tokenization to leverage complementary strengths of each modality. While this hybrid approach demonstrates improved downstream performance, it raises an important question, **does reintroducing text tokenizers reintroduce the cross-lingual disparities that motivated pixel-based approaches?** We investigate this question by examining tokenizer alignment effects in DualGPT across low-resource Indonesian scripts and languages.

# 3 Methodology

## 3.1 Datasets

In this work, we analyze four Indonesian low-resource local scripts: Javanese (jav), Sundanese (sun), Balinese (ban), and Lampung (ljp), representing a range of resource levels and linguistic diversity.

| Language | # Train | # Eval |
|---|---|---|
| Javanese | 400,726 | 816 |
| Sundanese | 293,933 | 823 |
| Balinese | 54,017 | 450 |
| Lampung | 945 | 84 |

Table 1: Dataset Statistics.

Training data comes from Wikidumps (July 2025) and digitalized Indonesian folklore[1] for Javanese, Sundanese, and Balinese, preprocessed as described in Section 3.3. We use NusaAksara (Adilazuarda et al., 2025) as the evaluation dataset and preprocess it in the same manner as the training data. Moreover, we split the NusaAksara's Lampung data for training and evaluation. Dataset statistics are presented in Table 1[2].

[1]extracted using `pdfplumber`.

[2]Full dataset will be released after the double-blind review.

## 3.2 PixelGPT

We use DualGPT (Chai et al., 2024)[3], pretraining the model on text, images, and paired text-image data, then finetuned for image-to-text transliteration. Each model went through a single training and finetuning run, with all reported metrics based on that execution.

## 3.3 Tokenizer and Renderer

Both a tokenizer and a renderer are required as the DualGPT model relies on both text and image modalities during pre-training. The visualization for the data preprocessing pipeline can be seen in Appendix B.

**Tokenizer.** We train our models using two different tokenizers: the default tokenizer used in the PixelGPT architecture (Llama 2 tokenizer) and a custom-built tokenizer. Our tokenizer is inspired by a grapheme-based approach (Basher et al., 2023), rather than the default BPE-based method. First, we back-transliterate our training dataset from Latin script text into *aksara* (the Indonesian local scripts) using community-built tools [4]. We then tokenize the aksara at the grapheme level and transliterate it back into Latin script text. Finally, we apply a word-level tokenization scheme to map the text into token IDs.

**Renderer.** With the findings of Tatariya et al. (2024) and the nature of Javanese and Balinese, which do not mark word boundaries with whitespace, we adopt a continuous rendering strategy following Lotz et al. (2023). We use a custom Pillow-based renderer instead of the default Pygame renderer on our back-transliterated datasets, as the default renderer is unable to correctly render additional characters and diacritics. The renderer parameters can be seen in Appendix C.

## 3.4 Evaluation

We use image transliteration as the evaluation task, which directly assesses script-tokenizer alignment by measuring surface-form preservation with minimal confounding variables. We report ChrF++ (Popović, 2017) in the main works, alongside BLEU (Papineni et al., 2002) and Word Error Rate in percentage (WER) in the Appendix (see Appendix E).

[3]Weights taken from `ernie-research/DualGPT`

[4]Heuristic transliteration for Javanese, Balinese, and Sundanese; and a custom font for Lampung.

### 3.5 Model Setup

Models are trained on 4×A40 GPUs with a maximum training time of 24 hours and early stopping (`patience = 5`) enabled. Training and finetuning hyperparameters are detailed in Appendix A.

### 3.6 Tokenizer Statistics

| Language | Llama 2 | Custom | Inflation |
|---|---|---|---|
| Javanese† | 45.97 | 65.11 | +41.6% |
| Sundanese‡ | 39.41 | 61.42 | +55.8% |
| Balinese† | 130.04 | 130.86 | +0.6% |
| Lampung‡ | 3.73 | 4.21 | +12.8% |

Table 2: Average text token length on train data. †Processed using the Javanese tokenizer. ‡Processed using the Sundanese tokenizer.

Table 2 and Figure 2 show that the Llama 2 tokenizer produces shorter sequences with lower fertility rates than custom tokenizers. The custom Javanese and Sundanese tokenizers achieve near-0% Out-of-Vocabulary (OOV) rates for their target languages but exhibit 22% and 10% OOV rates on Balinese and Lampung, respectively. The Llama 2 tokenizer maintains 0% OOV across all languages due to its broader vocabulary coverage.

By conventional metrics, sequence length, fertility rate, and OOV rate, the Llama 2 tokenizer appears superior. However, as we demonstrate in Section 4, these efficiency metrics do not translate into real model performance on low-resource scripts.

## 4 Results

### 4.1 VLM Evaluation

Five-shot in-context prompting reveals that current VLMs fail to transliterate rendered Indonesian scripts (Figure 1). While Llama-3.1-Nemotron-Nano-VL-8B-V1 achieves the highest chrF++ among tested models, its performance is negligible, with an average BLEU of 1.64 and a WER of 1,145.55 (many more errors than words in the reference). We report the full generation setup and results in Appendix D.

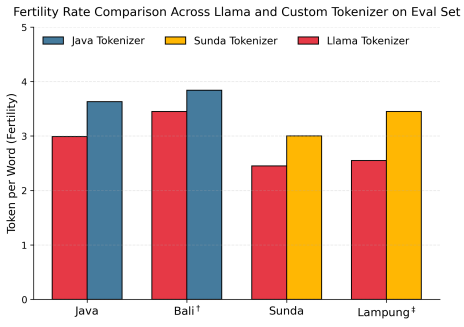


Figure 2: Fertility rate on the evaluation dataset per language. Bali† and Lampung‡ uses Javanese's and Sundanese's tokenizer, respectively.

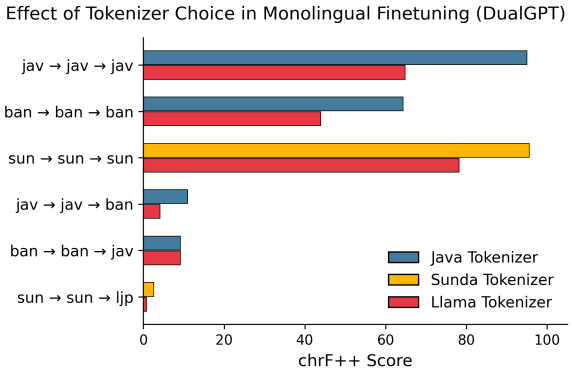


Figure 3: Impact of tokenizer choice on **Monolingual** DualGPT pretraining and finetuning: Pretraining → Finetuning → Evaluation.

### 4.2 DualGPT Architecture

Custom tokenizers significantly outperform Llama 2 in monolingual settings (Figure 3), with chrF++ improvements of +30.15 for Javanese, +20.45 for Balinese, +17.40 for Sundanese, and +1.8 for Lampung (see Appendix E for complete metrics). However, zero-shot cross-lingual transfer fails for both tokenizers (e.g., Java→Bali only achieves 10.94 chrF++ with custom vs. 4.15 with Llama 2), indicating that improved in-language alignment does not enable cross-lingual generalization.

In multilingual training (Table 3), custom tokenizers maintain their advantage. Joint Java+Bali training with the Javanese tokenizer achieves 69.2 chrF++, more than 6 times Java→Bali monolingual training performance, suggesting that multilingual exposure improves cross-lingual transfer, though high WER (108.73%) indicates continued word-level challenges.

Crucially, the Llama 2 tokenizer's 0% OOV rate did not translate to superior performance. Despite having significantly higher OOV rates in cross-lingual settings (Section 3.6), our custom tokenizers maintained a substantial lead, suggesting that visual features do not compensate for poorly aligned text embeddings.

This script-tokenizer alignment problem is also evidenced by the Javanese and Balinese performance mismatch, although to a lesser extent than

| Pretrain Data | Finetune Data | Eval Data | Tokenizer | ChrF++ ↑ | WER ↓ | BLEU ↑ |
|---|---|---|---|---|---|---|
| Javanese | Javanese + Balinese | Javanese | Llama | 62.52 | 125.42 | 16.18 |
| | | | Java | **94.27** | **21.79** | **78.05** |
| | | Balinese | Llama | 42.29 | **95.65** | 6.51 |
| | | | Java | **64.11** | 135.62 | **7.92** |
| Balinese | Javanese + Balinese | Javanese | Llama | 62.48 | **74.84** | 23.88 |
| | | | Java | **87.93** | 90.60 | **42.30** |
| | | Balinese | Llama | 42.64 | **105.35** | **5.98** |
| | | | Java | **62.40** | 285.08 | 4.40 |
| Java + Balinese | Javanese + Balinese | Javanese | Llama | 64.06 | 127.51 | 17.33 |
| | | | Java | **94.96** | **16.85** | **81.30** |
| | | Balinese | Llama | 44.89 | 159.35 | 5.11 |
| | | | Java | **69.20** | **108.73** | **10.18** |
| Sundanese | Sunda + Lampung | Sundanese | Llama | 78.35 | 40.16 | 59.33 |
| | | | Sunda | **96.02** | **5.33** | **92.45** |
| | | Lampung | Llama | 2.95 | 194.12 | 0.05 |
| | | | Sunda | **9.67** | **159.85** | **0.13** |

Table 3: Image transliteration performance comparing Llama and Custom tokenizers under a multilingual training setup. **Bolded** values indicate the best performance across tokenizers for each metric within each evaluation language. We use chrF++ as the main metrics for analysis (see Appendix F).

the Llama 2 tokenizer. Despite their shared Austronesian roots, Romanization conventions, and similar orthographic structures, the Javanese and Balinese models exhibit a significant chrF++ performance mismatch. Analyzing the vocabulary overlap, we find that a grapheme-based tokenizer trained on the Balinese data only has an overlap of 16.4% with the used Javanese tokenizer (4,994 shared tokens out of 30,346 unique tokens). This indicates that the Balinese's performance mismatch stems from embedding space mismatch caused by the script-tokenizer misalignment rather than mere data scarcity.

These results demonstrate that tokenizer alignment critically affects performance, contradicting traditional efficiency metrics (Section 3.6): the Llama 2 tokenizer's superior fertility rate and zero OOV did not translate to better transliteration.

## 5 Discussion and Future Works

**Tokenizer metrics are not indicative of tokenizer fit** While standard metrics (Section 3.6) suggest the Llama 2 tokenizer is superior for all four languages, each model performs significantly better with the custom tokenizer. We thus urge the creation of evaluation frameworks that better align tokenizer quality with model performance for both monolingual and multilingual cases (Chelombitko et al., 2024).

**Text tokenizer remains a bottleneck despite pixel-based modeling** One might expect that pixel-based architectures would not require a robust text tokenizer. However, we show that the text tokenizer remains a crucial bottleneck. While this work focuses on pixel-based models, current LLMs likely face similar issues with low-resource scripts, even when Romanized. We require further analysis on the impact of tokenizer choice on multilingual LLM performance (Limisiewicz et al., 2023).

**Cross-lingual transfers are still limited by the tokenizer** Our analysis indicates that poor performance stems not only from data scarcity but from tokenizer mismatch. Despite Javanese and Balinese being linguistically highly related, cross-lingual transfer remains ineffective. This suggests that as long as the tokenizer issue persists, low-resource languages may benefit more from specialized models than from cross-lingual transfer. Advancements in architectures that balance specialization with generalization are vital for equitable performance.

**Leveraging Rendered Representations** Despite our criticism, we believe pixel-based language modeling is a promising path toward more equitable technology. This work acts as a warning, reintroducing text tokenizers into such architectures may bring unintended consequences and must be done with proper justification and extreme care.

## Limitations

**Scope of Architectural Analysis.** This work focuses specifically on the impact of tokenizer alignment within hybrid vision-text architectures (e.g., DualGPT). While a comparison with pure vision-based models (without text-heads) would provide additional context, our study is designed to isolate the behavior of the text-modality when reintroduced into pixel-based frameworks. Future work will expand this to a wider range of architectural configurations.

**Resource Disparity.** The available training data for Javanese, Balinese, Sundanese, and Lampung varies significantly, which is reflected in the performance gap observed in the Lampung experiments. However, the consistent improvement across all languages when using aligned tokenizers suggests that our findings are robust to variations in data scale.

**Task Specificity.** We utilize image transliteration as a primary probe because it directly measures surface-form preservation and script-alignment with minimal confounding variables. While this task effectively exposes the tokenizer bottleneck, further research is needed to determine how these alignment issues affect higher-level semantic tasks like NLI or abstractive summarization in pixel-based models.

## Acknowledgment

Anonymized due to double blind.

## Ethical Consideration

Despite rendering native scripts as images, we still use the Romanized form to simplify the analysis in this work. This approach is not meant to supplant the use of native scripts with Latin texts, but is out of necessity, as many local Indonesian languages' scripts are not supported by Unicode. Further advancements in this line of research are pivotal to overcoming this digital infrastructure gap alongside improving language and script equity in the field of AI.

## References

Diana Abagyan, Alejandro R. Salamanca, Andres Felipe Cruz-Salinas, Kris Cao, Hangyu Lin, Acyr Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. 2025. One tokenizer to rule them all: Emergent language plasticity via multilingual tokenizers. *Preprint*, arXiv:2506.10766.

Muhammad Farid Adilazuarda, Musa Izzanardi Wijanarko, Lucky Susanto, Khumaisa Nur'aini, Derry Tanti Wijaya, and Alham Fikri Aji. 2025. NusaAksara: A multimodal and multilingual benchmark for preserving Indonesian indigenous scripts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28371–28401, Vienna, Austria. Association for Computational Linguistics.

Shuai Bai and 1 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Mohammad Jahid Ibna Basher, Mohammad Raghib Noor, Sadia Afroze, Ikbal Ahmed, and Mohammed Moshiul Hoque. 2023. Bngraphemizer: A grapheme-based tokenizer for bengali handwritten text recognition. *2023 IEEE 9th International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*, pages 183–188.

Akhiad Bercovich and 1 others. 2025. Llama-nemotron: Efficient reasoning models. *Preprint*, arXiv:2505.00949.

Yekun Chai, Qingyi Liu, Jingwu Xiao, Shuohuan Wang, Yu Sun, and Hua Wu. 2024. Autoregressive pre-training on pixels and texts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3106–3125, Miami, Florida, USA. Association for Computational Linguistics.

Iaroslav Chelombitko, Egor Safronov, and Aleksey Komissarov. 2024. Qtok: A comprehensive framework for evaluating multilingual tokenizer quality in large language models. *Preprint*, arXiv:2410.12989.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Negar Foroutan, Clara Meister, Debjit Paul, Joel Niklaus, Sina Ahmadi, Antoine Bosselut, and Rico Sennrich. 2025. Parity-aware byte-pair encoding: Improving cross-lingual fairness in tokenization. *Preprint*, arXiv:2508.04796.

Gemma Team. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Wenhan Han, Yifan Zhang, Zhixun Chen, Binbin Liu, Haobin Lin, Bingni Zhang, Taifeng Wang, Mykola Pechenizkiy, Meng Fang, and Yin Zheng. 2025. Mubench: Assessment of multilingual capabilities of large language models across 61 languages. *Preprint*, arXiv:2506.19468.

Ilker Kesen, Jonas F. Lotz, Ingo Ziegler, Phillip Rust, and Desmond Elliott. 2025. Multilingual pretraining for pixel language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29582–29599, Suzhou, China. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Tomasz Limisiewicz, Jiří Balhar, and David Mareček. 2023. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5661–5681, Toronto, Canada. Association for Computational Linguistics.

Jonas Lotz, Elizabeth Salesky, Phillip Rust, and Desmond Elliott. 2023. Text rendering strategies for pixel language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10155–10172, Singapore. Association for Computational Linguistics.

OpenAI. 2025. GPT-5 System Card. `https://openai.com/index/gpt-5-system-card`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Aleksandar Petrov, Emanuele La Malfa, Philip H.S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Phillip Rust, Jonas F. Lotz, Emanuele Bugliarello, Elizabeth Salesky, Miryam de Lhoneux, and Desmond Elliott. 2022. Language modelling with pixels. *ArXiv*, abs/2207.06991.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Jimin Sun, Patrick Fernandes, Xinyi Wang, and Graham Neubig. 2023. A multi-dimensional evaluation of tokenizer-free multilingual pretrained models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1725–1735, Dubrovnik, Croatia. Association for Computational Linguistics.

Kushal Tatariya, Vladimir Araujo, Thomas Bauwens, and Miryam de Lhoneux. 2024. Pixology: Probing the linguistic and visual capabilities of pixel-based language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3307–3320, Miami, Florida, USA. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Weiyun Wang and 1 others. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *Preprint*, arXiv:2508.18265.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

## A Hyperparameters

Pretraining hyperparameters and finetuning hyperparameters are available in Table 4 and Table 5, respectively.

| Category | Hyperparameter | Value |
|---|---|---|
| Architecture | Model Backbone | DualGPT |
| | Hidden / Interm. Size | 768 / 3072 |
| | Layers / Attn Heads | 12 / 12 |
| | Image Size | $16 \times 16{,}384$ |
| | Patch Size | 16 |
| Optimization | Optimizer | AdamW |
| | Learning Rate | $5 \times 10^{-4}$ |
| | LR Scheduler | Cosine |
| | Warmup Steps | 1000 |
| | Weight Decay | 0.1 |
| | Numerical Precision | BF16 |
| Training | Per-device Batch | 2 |
| | Grad. Accumulation | 4 |
| | Num. GPUs (A40) | 4 |
| | Effective Batch Size | 32 |
| | Max Training Time | 24 Hours |
| | Early Stop Patience | 5 |
| | Eval / Save Steps | 1000 |

Table 4: Pre-training hyperparameters.

| Category | Hyperparameter | Value |
|---|---|---|
| Model Setup | Pretrained Weight | DualGPT-pt |
| | Dropout | 0.1 |
| | Frozen Layers | `lm_pixel_head` |
| Optimization | Learning Rate | $2 \times 10^{-5}$ |
| | LR Scheduler | Cosine |
| | Warmup Ratio | 0.03 |
| | Weight Decay | 0.1 |
| | Numerical Precision | BF16 |
| Training | Per-device Batch | 2 |
| | Grad. Accumulation | 4 |
| | Num. GPUs (A40) | 4 |
| | Effective Batch Size | 32 |
| | Max Epochs | 10 |
| | Early Stop Patience | 5 |
| | Eval / Save Steps | 1000 |

Table 5: Hyperparameters for image-to-text transliteration fine-tuning. We swap **Warmup steps** to **Warmup ratio** to account for Lampung finetuning data (945 entries). DualGPT-pt indicates the pretrained DualGPT model.

## B Dataset Construction

Figure 4 illustrates the dataset construction pipeline used for DualGPT training. Starting from romanized text, each word is back-transliterated into Indonesian scripts (Aksara), which is then tokenized using a grapheme-based scheme to preserve script-specific character structures. The Aksara text is rendered into an image using a custom renderer, forming the visual modality, while the tokenized Aksara is transliterated back into Latin and converted into token IDs using a custom tokenizer. The resulting paired image–token representation constitutes the final training dataset.

## C Renderer Parameters

We follow the original PixelGPT renderer parameters, modifying only the font and font size to prevent text from overflowing beyond the image boundaries as shown in Table 6. Lampung language used custom font since it is not inherently suppported by Unicode.

## D VLM Evaluation Setups and Results

We use open small-sized to medium-sized VLMs, including `Gemma-3-12B-it` (Gemma Team, 2025), `Qwen3-VL-8B-Instruct` (Bai et al., 2025), `InternVL3.5-8B` (Wang et al., 2025), and `Llama-3.1-Nemotron-Nano-8B-V1` (Bercovich et al., 2025), as well as a closed-source model, GPT-5-nano (OpenAI, 2025). Few-shot examples

| Parameter | jav/sun/ban | ljp |
|---|---|---|
| Background Color | White | White |
| DPI | 120 | 120 |
| Font Color | Black | Black |
| Font Type | Noto Sans | Had Lampung Yuzu Rounded |
| Font Size | 7 | 10 |
| Max sequence length | 1024 | 1024 |
| Padding size | 3 | 3 |
| Pixels per patch | $16 \times 16$ | $16 \times 16$ |

Table 6: Configuration of text rendering for Javanese, Sundanese, Balinese, and Lampung language.

are randomly selected from pretrain data to avoid leakage. For the open models, we deploy them using vLLM (Kwon et al., 2023). For the generation hyperparameters, we constrain `temperature = 0.4` and `max_output_tokens = 2048`. Full VLMs evaluation performance is available at Table 8.

## E Full Monolingual Model Performance

Full model monolingual performance is available in Table 9.

## F chrF++ as the Main Metric

Due to the nature of languages we are working with (i.e., low resource, native scripts), finding a suitable dataset for the evaluation remains a significant challenge. NusaAksara exist as one of the only benchmark available to support our transliteration task. However, NusaAksara has limitations. Shown in Table 7, NusaAksara's Balinese entry consist mainly of short word entries, disadvantaging metrics such as BLEU and WER.

| ID | Reference | Prediction |
|---|---|---|
| 7 | prathiwitala | prathiwitala |
| 9 | mandaaakranta | man**t**aaakran**b**a |
| 11 | 134 , | 13 , |
| 13 | dumala | **mera**lumala |

Table 7: Transliteration sample for Balinese language. ID refers to NusaAksara's dataset entry ID.

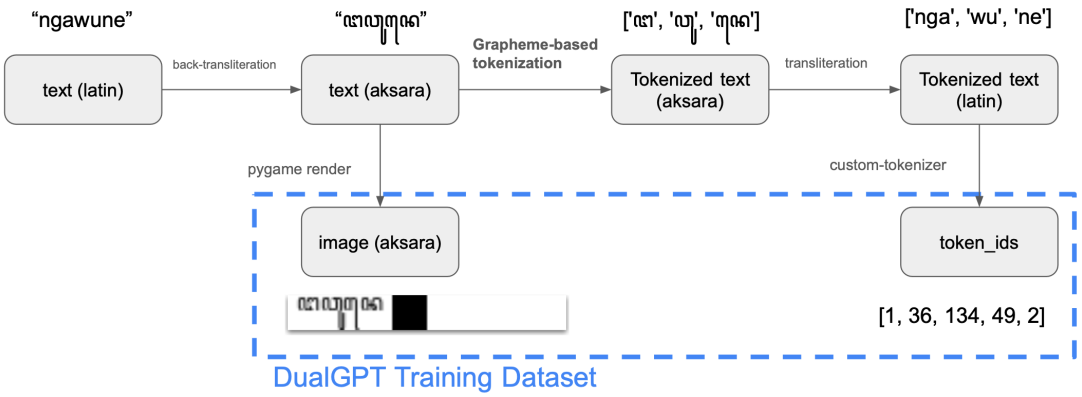


Figure 4: Dataset building process using custom tokenization rules.

| Model | Balinese | | | Javanese | | | Lampung | | | Sundanese | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BLEU ↑ | chrF++ ↑ | WER ↓ | BLEU ↑ | chrF++ ↑ | WER ↓ | BLEU ↑ | chrF++ ↑ | WER ↓ | BLEU ↑ | chrF++ ↑ | WER ↓ |
| InternVL3.5-8B | 0.00 | 9.18 | 564.94 | 0.00 | 17.10 | 1947.98 | 0.00 | 7.92 | 625.11 | 6.57 | 19.08 | 1444.19 |
| Llama-3.1-Nemotron-Nano-VL-8B-V1 | 4.99 | **19.44** | 794.22 | 0.52 | 8.82 | 907.54 | **1.09** | **11.83** | 2086.88 | 7.81 | **30.91** | 1200.58 |
| Qwen3-VL-8B | 1.62 | 14.21 | 238.35 | 2.91 | 16.50 | 316.89 | 0.00 | 0.00 | 1447.06 | **8.12** | 15.34 | 710.77 |
| Gemma-3-12B-it | **6.57** | 15.26 | 295.63 | **3.83** | **24.15** | 625.47 | 0.00 | 6.01 | 225.57 | 5.52 | 13.20 | 519.27 |
| GPT-5-Nano | 0.00 | 0.00 | **205.42** | 0.00 | 0.00 | **161.94** | 0.00 | 9.36 | **101.81** | 0.00 | 0.00 | **123.71** |

Table 8: VLM Zero-shot image transliteration performance across four Indonesian languages. Higher is better for BLEU and chrF++; lower is better for WER. Note that extremely low WER in models with zero BLEU (e.g., GPT-5-Nano) typically indicates empty or extremely short output rather than accuracy.

| Pretrain & Finetune | Eval | Tokenizer | ChrF++ ↑ | WER ↓ | BLEU ↑ |
|---|---|---|---|---|---|
| Javanese | Javanese | Llama | 64.83 | 79.01 | 24.99 |
| | | Java | **94.98** | **19.95** | **80.28** |
| | Balinese | Llama | 4.10 | **506.99** | 0.01 |
| | | Java | **10.94** | 550.00 | 0.01 |
| Balinese | Javanese | Llama | **9.20** | 250.00 | 0.03 |
| | | Java | 9.16 | **102.30** | 0.03 |
| | Balinese | Llama | 43.91 | **86.16** | **7.53** |
| | | Java | **64.36** | 131.91 | 7.00 |
| Sundanese | Sundanese | Llama | 78.19 | 40.54 | 58.93 |
| | | Sunda | **95.59** | **7.87** | **90.55** |
| | Lampung | Llama | 0.77 | **325.79** | 0.00 |
| | | Sunda | **2.57** | 585.64 | 0.00 |

Table 9: Image transliteration performance comparing Llama and Custom tokenizers under a monolingual tranining setup. Bold values indicate which tokenizer performs better.