Tokenizer Choice Matters Again: A Study of Dual-Modality Pixel Language Models on Abugida Scripts

Anonymous ACL submission

Abstract

Pixel-based language models were introduced to sidestep tokenization, but dual-modality variants such as DualGPT reintroduce a text tokenizer alongside the pixel objective. We ask whether this reintroduction also resurfaces classical tokenizer problems, focusing on three low-resource Abugida scripts (Balinese, Javanese, Sundanese) where the absence of whitespace boundaries makes tokenization especially consequential. We propose a two-axis framework decomposing tokenizer quality into image-conditioned tokenization uncertainty and token informativeness, and introduce visual-grapheme boundary alignment as a novel metric for dual-modality settings. Across four tokenizers and three scripts, we find that grapheme-based tokenizers achieve $3.5\times$ lower in-domain bits-per-character than the next-best alternative, but this dominance does not transfer to out-of-domain transliteration, indicating that Grapheme-based tokenizers may overfit to the training data whereas subword tokenizers appear to act as implicit regularization. As such, tokenizer choice remains a meaningful design decision in dual-modality pixel language models, and in-domain metrics alone do not predict downstream performance.

1 Introduction

Although much effort has been poured into the inclusivity of low-resource languages in large language models (Bercovich et al., 2025; Gemma Team, 2025; Wang et al., 2025; Bai et al., 2025), low-resource language scripts remain far behind current SOTA model performance (Figure 1). This lagged performance is caused by many factors specific to the lack of general support (Joshi et al., 2020), such as Unicode availability for low-resource languages' native scripts (Adilazuarda et al., 2025; Chen et al., 2024; Shih et al., 2025). However, one issue that is very close to the field of NLP is the tokenization of these scripts with issues such as over-segmentation, which inflates sequence lengths and potentially degrades model performance (Ahia et al., 2024; Lundin et al., 2026).

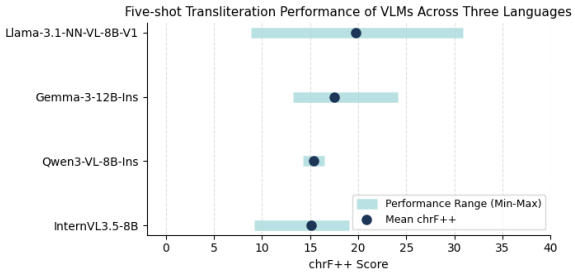


Figure 1: VLMs Performance on rendered text transliteration on the NusaAksara benchmark for Javanese, Sundanese, and Balinese.

Text tokenization is a complex process (Mielke et al., 2021). Even to date, other than through the inclusion of ever more data, there are no conclusive or standard methods to create a text tokenizer. Although BPE has become one of the most popular methods in creating a text tokenizer, due to its ability to nullify the appearance of out-of-vocabulary tokens (Sennrich et al., 2016), it is still with its own set of issues (Alqahtani et al., 2026; Pilana Liyanage and Yvon, 2026).

Recent works have moved towards the utilization of rendered texts as images as another modality to model languages. Known as Pixel-based language modeling (henceforth Pixel-LM) (Rust et al., 2022), the architecture in principle removes the need for a text tokenizer to model a language. This, however, comes at a cost. Pixel-LM reliance on pixel reconstruction objectives over raw pixel values is not inherently aligned with linguistic understanding (Tatariya et al., 2024). Despite prior research showing that Pixel-LM can indeed learn these languages (Tai et al., 2024; Kesen et al., 2025; Hu et al., 2026), we will later show that the pixel reconstruction objective provides a less linguistically-aligned learning signal than discrete token prediction, making it

less sample-efficient in low-resource monolingual settings compared to text-based LMs (henceforth Text-LM) such as BERT (Devlin et al., 2019). To alleviate this issue, works such as Gao et al. (2024), Lotz et al. (2025b), and Chai et al. (2024)'s DualGPT have shown that jointly learning from rendered images and texts can prove beneficial to the language understanding of the model.

In principle, the text head in DualGPT serves only as a language modeling signal, so any reasonable tokenizer should converge to similar performance. We show this is not the case. This paper continues Chai et al. (2024)'s work by empirically testing this hypothesis. Specifically, our contributions are as follow:

- We introduce a two-axis framework for analyzing tokenizer quality in dual-modality pixel language models, decomposing tokenizer impact into image-conditioned tokenization uncertainty and token informativeness.
- We propose visual-grapheme boundary alignment as a novel tokenizer quality metric for dual-modality architectures, measuring how well tokenizer boundaries align with grapheme structure recoverable from rendered images.
- Across three Abugida scripts (Balinese, Javanese, Sundanese) and four tokenizers, we show that grapheme-based tokenizers achieve 3.5× lower BPC in-domain, but this dominance does not transfer to out-of-domain downstream performance, where subword tokenizers' coarser units appear to act as implicit regularization.

2 Related Work

2.1 Pixel-based Language Modeling

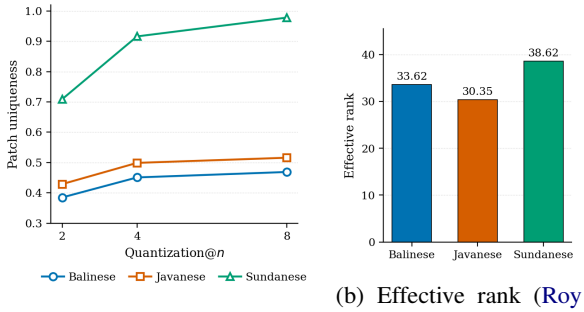
To our knowledge, Pixel-based Language Modeling was first popularized by Rust et al. (2022)'s PIXEL, demonstrating that a model can learn language purely from rendered text images using a masked autoencoding objective. Subsequently, the architecture then continues to be expanded to encoder-decoder architecture (Gao et al., 2024) and the decoder-only architecture such as Tai et al. (2024)'s PIXAR and Chai et al. (2024)'s PixelGPT. Further work has demonstrated its potential to model non-Latin scripts across multiple languages (Kesen et al., 2025; Hu et al., 2026). However, models trained solely on rendered text still fall short of SOTA, scoring slightly below Text-LM, such as BERT and GPT-2 (Devlin et al., 2019; Radford et al., 2019) on most benchmarks. Tatariya et al. (2024) explains that this happens because Pixel-LM has to recover from not having pre-defined tokens, and although rendering the texts using strategies such as word-level and bigram-level rendering helps (Lotz et al., 2023; Tatariya et al., 2024), many languages in the world lack whitespaces.

The weakness of pixel reconstruction objectives, namely the poor readability of generated patches, was alleviated by Tai et al. (2024)'s PIXAR, which introduced a two-phase pretraining strategy incorporating a GAN to improve the visual quality of reconstructed patches. This was further extended by Hu et al. (2026)'s MIXAR, which scaled the approach through increased parameter count and broader language and script coverage. Concurrently, Chai et al. (2024)'s PixelGPT explored a similar direction to PIXAR, achieving comparable results. More notably, through its DualGPT variant, the work introduced a dual-modality training strategy that reintegrates a text tokenizer alongside the pixel objective, demonstrating that joint modeling of rendered images and text can significantly benefit language understanding. However, whether this reintroduction of text tokenizers resurfaces tokenizer-related issues, such as script misalignment and over-segmentation, remains an open question.

2.2 Impact of Text-tokenizers in Text-LM

The choice of text tokenizer has been shown to significantly affect model performance, particularly in multilingual settings (Rust et al., 2021; Han et al., 2025), with comparatively less consistent impact observed in monolingual ones (Ali et al., 2024). Despite this, choosing or creating a text tokenizer remains one of the least standardized components of the NLP pipeline (Mielke et al., 2021), with most Text-LMs utilizing BPE-based tokenizer (Sennrich et al., 2016) or its derivatives (Foroutan et al., 2025).

Despite its widespread adoption, BPE's frequency-based segmentation has been shown to produce splits that are not linguistically aligned, particularly for morphologically complex scripts (Basher et al., 2023; Ahia et al., 2024). Recent efforts have been made to create a more universal tokenizers that perform equitably across languages and scripts, such as Abagyan et al. (2025). At the extreme end, tokenizer-free Text-LMs such as CANINE (Clark et al., 2022) and ByT5 (Xue



(a) Patch uniqueness across quantization levels.

(b) Effective rank (Roy and Vetterli, 2007) of patch embeddings.

Figure 2: Patch distinguishability across the three languages.

et al., 2022) attempts to eliminate the tokenization process by operating directly on characters and byte level, respectively. However, both approaches remain unusable for languages without Unicode support (Joshi et al., 2020).

2.3 Low-resource script NLP

Incorporation of non-Latin scripts in NLP is largely limited to high-resource languages (Joshi et al., 2020). For low-resource language scripts, many exist primarily in non-digitized forms, with ongoing efforts relying on technologies such as OCR to build usable corpora (Agarwal and Anastasopoulos, 2024). In large language models, the incorporation of these scripts often involves transliteration to Latin scripts (J et al., 2024), which sacrifices the language's original orthography and relies on heuristic tools that may not exist for all scripts (Lotz et al., 2025b). Thus, Pixel-LM remains a promising research direction for equitable and inclusive language technologies.

3 Theoretical Groundwork

Reintroducing the text-tokenizer into the Pixel-LM architecture may potentially cause several concerns, with one of the most concerning being the reintroduction of problems inherent to text-tokenizers. For this work, we focus only on `PixelGPT`'s DualGPT architecture, trained on paired (Image, Text) data. In this section, we provide the theoretical foundation on when and how DualGPT helps Pixel-based language modeling through the lens of a two-axis tokenizer-quality framework.

Setup Let $\mathcal{L}$ denote a random variable over linguistic content from a distribution characterized by typological properties such as morphology, syntax, and inventory, as exemplified by (Littell et al., 2017). We define random variables $T = \tau(\mathcal{L})$ and $\mathbf{I} = \phi(\mathcal{L})$, where τ is the tokenization function and ϕ is the rendering function. Note that τ and ϕ are independent: the rendering process does not rely on the tokenizer and vice versa. We assume both functions are deterministic and lossless on $\mathcal{L}$, which holds for the byte-fallback tokenizers and the script-supporting renderers used in our experiments. We compare three training regimes: *Text-LM* optimizes Cross Entropy (CE) loss on tokens; *Pixel-LM* optimizes Binary Cross Entropy (BCE) on rendered pixels; and *DualGPT* optimizes both, such that $l_{\text{Dual}} = \lambda\, l_{\text{CE}} + l_{\text{BCE}}$.

Axis 1: Image-conditioned tokenization uncertainty For any model, CE loss is bounded below by the conditional Shannon entropy (Shannon, 1948) of its prediction target given what it conditions on. Text-LM's floor is the marginal sequence entropy $H(T) = \sum_t H(T_t \mid T_{<t})$. Under DualGPT's image-first regime, each token prediction additionally conditions on the image, lowering the floor to $H(T \mid \mathbf{I}) = \sum_t H(T_t \mid \mathbf{I}, T_{<t}) \leq H(T)$. The gap, defined as the mutual information $\text{MI}(T; \mathbf{I}) = H(T) - H(T \mid \mathbf{I})$, is the information the image provides about the tokens. For our deterministic tokenizers, $H(T \mid \mathbf{I})$ captures the information the tokenization carries *beyond* what the image already exposes (for instance, BPE merge rules learned on a different corpus, or transliteration conventions absent from the rendered glyphs). A tokenizer whose boundaries are predictable from the image (e.g., grapheme on Brahmic) has small $H(T \mid \mathbf{I})$; one whose boundaries depend on extra-visual information has large $H(T \mid \mathbf{I})$. We coin the **visual-token alignment** metric to empirically proxy this term.

Axis 2: Token informativeness Axis 1 alone does not determine tokenizer quality. If it did, byte-level tokenization, which is fully predictable from the image at the character level, would dominate (and in practice, it does not (Xue et al., 2022)). We hypothesize that the missing axis is related to *how much linguistic content each token carries*. We define token informativeness as

$$\text{Inf}(\tau) = \frac{\text{MI}(T; \mathcal{L})}{\mathbb{E}[|T|]} = \frac{H(\mathcal{L})}{\mathbb{E}[|T|]}, \qquad (1)$$

where the second equality follows from our information-preservation assumption. Higher informativeness packs the same linguistic content into fewer tokens, so each CE prediction is over a

Table 1: Training and Downstream dataset statistics. **Patches** refer to image patches consumed by the pixel-based model, **Words** and **Characters** are calculated based on the raw text. Per-entry values are reported as mean $\pm$ std.

			Patches		Words		Characters	
Language	Split	Entries	Total	Per entry	Total	Per entry	Total	Per entry
Balinese	Train	54,017	4,628,291	85.7 ± 40.4	2,294,705	42.5 ± 20.1	16,605,015	307.4 ± 148.7
Balinese	Test	450	8,501	18.9 ± 10.1	4,461	9.9 ± 6.0	28,872	64.2 ± 39.3
Javanese	Train	400,726	15,540,137	38.8 ± 21.2	6,569,400	16.4 ± 9.1	45,666,996	114.0 ± 65.2
Javanese	Test	816	10,026	12.3 ± 4.0	4,456	5.5 ± 2.3	28,515	34.9 ± 14.0
Sundanese	Train	293,933	8,477,294	28.8 ± 18.0	4,131,407	14.1 ± 9.4	28,532,068	97.1 ± 64.6
Sundanese	Test	823	6,286	7.6 ± 5.7	3,130	3.8 ± 3.4	18,938	23.0 ± 23.0

more linguistically meaningful unit. Operationally, this is equal to the inverse of the average sequence length, closely related to compression-based tokenizer quality metrics.

Implication The framework predicts that tokenizers minimizing $H(T \mid \mathbf{I})$ while maximizing $\mathrm{Inf}(\tau)$ should dominate in DualGPT. This also explains why DualGPT outperforms pure Pixel-LM (Chai et al., 2024), as BCE loss for pixel reconstruction is not directly aligned with language's symbolic structure (Tatariya et al., 2024), while DualGPT's CE head provides a complementary symbolic signal whose usefulness is mediated by both axes. The benefit should be most visible in two regimes: (1) under low-resource data, where any reduction in sample complexity matters, and (2) under high-informativeness tokenizers, where each CE update carries a dense signal. The tradeoff between the two axes is acute for Brahmic scripts because they lack the whitespace word-boundary signal that Latin-script languages provide for free. Thus, the tokenizer must instead encode aksara segmentation explicitly to achieve visual alignment. BPE tokenizers trained on Latin-dominant corpora lack this knowledge, potentially disrespecting grapheme boundaries or merging across them. In this work, we empirically show that grapheme-based tokenizers recover alignment at the cost of higher fertility.

4 Methodology

4.1 Language and Data

In this work, we use three local Indonesian language scripts of related origins. These three languages are Balinese, Javanese, and Sundanese. These languages are chosen due to the availability of community-built rendering tools [1]. The pretraining data comes from wikidumps (taken on July 2025) and digitized Indonesian folklore[2]. The breakdown of the data we gathered can be seen in Table 1. To approximate language difficulty, we utilize Grapheme Similarity metrics through quantization and effective rank (Roy and Vetterli, 2007) (Figure 2). Texts are rendered at 120 DPI using the same fonts as Kesen et al. (2025) at font size 6, then divided into 16×16 Grayscale pixel patches with a maximum sequence length of 1024 patches.

4.2 Tokenizer

In this work, we analyze four different types of tokenizers. The first is our Grapheme-based tokenizers, which are unique per language. These grapheme-based tokenizers are made with visual-alignment in mind, as shown by Figure 5. The second and third tokenizers are Llama2 (Touvron et al., 2023) and Komodo (Owen et al., 2024), both being BPE-based tokenizers. Komodo shares the same 32K base vocabulary from Llama2, but adds around 3000 tokens specifically to lower fertility in Indonesian and its regional languages, including Balinese, Sundanese, and Javanese. Lastly, we also used the mT5 tokenizer in this work, a massive multilingual tokenizer with a 250K vocabulary size made using SentencePiece. These tokenizers' statistics are available in Table 2.

4.3 Training Details

All models explored in this work are trained under a monolingual setting. In total, there are 3 PIXAR models (1 model x 3 languages) and 12 DualGPT models (3 languages x 4 tokenizers). PIXAR models were trained using 1 A100 for 24 hours, and DualGPT models were trained using 2 A40 GPUs for 24 hours for pretraining + 2 A100 for 24 hours on finetuning. Note that PIXAR never utilized the full 24 hours, as its eval loss quickly stagnated. Full

[1] Community-built transliteration tools for Balinese, Javanese, and Sundanese.

[2] extracted using `pdfplumber`.

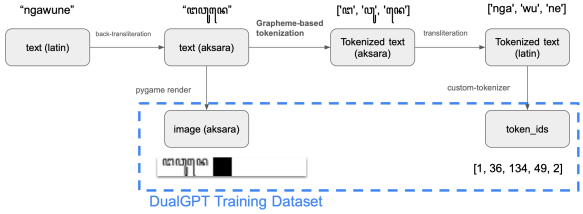


Figure 3: The rendering pipeline. Note that the Grapheme tokenizer is trained on the tokenized text. Padded white space is eventually removed before being fed to the model.

hyperparameter settings available in Appendix A.

4.4 Evaluation

We evaluate PIXAR using the last patch prediction (Tai et al., 2024). For DualGPT, we evaluate its training-phase evaluation loss and its downstream performance on Rendered Text Transliteration utilizing the NusaAksara dataset (Adilazuarda et al., 2025), re-rendered using our renderer. For PIXAR and DualGPT's training-phase evaluation, we hold out 1000 entries per language from the pretraining data as the evaluation set, sampled uniformly at random before training, which is then used for analysis and as an early stopping criterion. Note that NusaAksara's data distribution is out-of-domain from the Wikidumps distribution. For downstream performance evaluation, we report chrF++ (SacreBLEU) and CER utilizing (jiwer).

4.5 Tokenizer Quality Metrics

To assess what makes a tokenizer good in DualGPT, we utilize three groups of metrics, which are:

- **Compression Statistics**: Includes Vocabulary size, Out-of-vocabulary (OOV) rate, and Fertility rate.
- **Distribution Statistics**: Taken from Lotz et al. (2025a), consisting of metrics that capture how the token-frequency distribution deviates from Zipf's law (Powers, 1998). Includes Cardinality, AUC, Slope, and Deviation.
- **Visual-Grapheme Alignment**: A novel tokenizer quality metric for DualGPT. Includes boundary precision and recall of tokenizers compared with a reference.

4.6 Visual-Grapheme Alignment

Precision and Recall Boundary precision and recall are defined as:

$$P(\tau) = \frac{|B_\tau \cap B_{\text{ref}}|}{|B_\tau|}, \quad R(\tau) = \frac{|B_\tau \cap B_{\text{ref}}|}{|B_{\text{ref}}|}.$$

We also report *intra-word* variants, $P_{\text{intra}}(\tau)$ and $R_{\text{intra}}(\tau)$, which restrict both sets to boundaries falling strictly within words (excluding word-end positions identified from whitespace in the source text). Intra-word variants are reported as word-end boundaries are trivially recoverable from whitespace present in nearly all tokenizers.

Reference References to the boundaries are taken from each language's community-built tool's transliteration function. These tools contain within them their own method of grouping related atomic units, such as diacritics, into graphemes. Note that these tools may contain some level of errors. However, it is deterministic and serves as a good proxy for the reference. Thus, we define boundaries as the positions between tokens after transliteration to their native scripts (See Figure 3).

Alignment However, different tokenizers have different rules and assumptions, such as the use of an underscore to represent space, or a dash to represent a continuation. Thus, we first align the tokenizer and the reference using the Needleman-Wunsch (NW) algorithm (Needleman and Wunsch, 1970). The algorithm computes a character-level alignment between two strings, finding the cheapest sequence of matches, substitutions, insertions, and deletions that transforms one into the other[3].

Note, as the Grapheme-based tokenizers are directly trained on these references, their boundary alignment reaches a perfect score of 1.

5 Experiments

5.1 PIXAR Performance

PIXAR's performance on each language, as seen in Figure 4, depicts that although PIXAR has learned some level of language understanding, its performance is not adequate enough to generate human-readable output. This outcome is expected, as the BCE loss for pixel reconstruction requires massive amount of data for the model to learn linguistic features. We skipped Phase 2 of PIXAR's pretraining, as the GAN architecture assumes that the generator is not far from the real distribution (Goodfellow et al., 2014; Arjovsky and Bottou, 2017), which our results indicate is not the case.

Table 2: Tokenizer statistics across the four tokenizers evaluated. Per-entry values are mean $\pm$ std token counts.

		Tokens per entry (mean $\pm$ std)				Tokens per word			
Language	Split	Grapheme	Llama2	Komodo	mT5	Grapheme	Llama2	Komodo	mT5
Balinese	Train	133.7 ± 66.5	132.9 ± 101.1	114.5 ± 96.4	101.8 ± 54.7	3.15	3.13	2.70	2.40
Balinese	Test	31.2 ± 15.9	33.5 ± 18.7	32.3 ± 18.2	27.5 ± 16.2	3.15	3.38	3.26	2.78
Javanese	Train	67.2 ± 38.2	48.0 ± 30.7	39.0 ± 27.0	36.0 ± 20.3	4.10	2.93	2.38	2.20
Javanese	Test	21.9 ± 7.8	18.1 ± 6.2	16.6 ± 5.7	14.2 ± 5.2	4.00	3.31	3.03	2.60
Sundanese	Train	63.6 ± 41.3	41.5 ± 27.9	34.7 ± 24.2	30.9 ± 20.0	4.52	2.95	2.47	2.20
Sundanese	Test	15.4 ± 13.5	12.1 ± 9.7	10.8 ± 8.2	9.1 ± 7.7	4.04	3.19	2.84	2.40

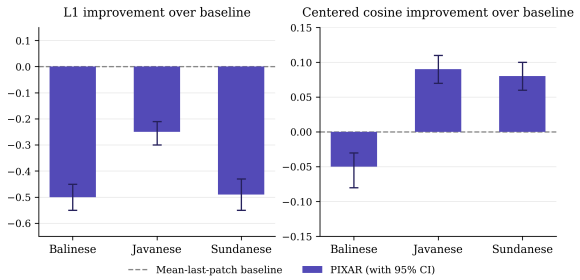


Figure 4: PIXAR phase 1 performance on each language, measured as improvement over a mean-last-patch baseline. Metrics are L1 and centered cosine on reconstructed patches. Error bars show 95% confidence intervals.

Tokenizer	Vocab	Vocab Params	Total
Sunda	4,178	8.6M	293.3M
Bali	15,368	31.5M	316.2M
Java	20,066	41.1M	325.8M
Llama2	32,000	65.5M	350.3M
Komodo	35,009	71.7M	356.4M
mT5	250,100	512.2M	**797.0M**

Table 3: Parameter counts for each DualGPT tokenizer configuration.

5.2 DualGPT In-Domain Performance

We report bits per character (BPC) in Table 4. Though the metric has received criticism and requires caution in multilingual settings (as noted by Cotterell et al. (2018)), this is not a concern for our monolingual, cross-tokenizer setup. Thus, BPC provides us with a tokenizer-agnostic comparison of the model's evaluation loss. The results show that **Grapheme tokenizers achieve the lowest BPC in every language**, which is roughly $3.5\times$ lower than the second-best tokenizer. We note that Llama2 and Komodo perform comparably to each other, which makes sense as they share the same 32K base vocabulary.

Language	Tokenizer	Chars/Tok	Eval Loss	BPC
Balinese	Grapheme	2.29	0.06	**0.038**
	Komodo	2.70	0.24	0.128
	Llama2	2.33	0.31	0.192
	mT5	3.03	0.32	0.153
Javanese	Grapheme	1.71	0.04	**0.034**
	Komodo	2.91	0.17	0.084
	Llama2	2.38	0.14	0.085
	mT5	3.14	0.28	0.129
Sundanese	Grapheme	1.51	0.01	**0.010**
	Komodo	2.79	0.09	0.047
	Llama2	2.36	0.10	0.061
	mT5	3.17	0.25	0.114

Table 4: In-domain language modeling performance under DualGPT, computed on the held-out evaluation set. **Chars/Tok** is characters per token, **Eval Loss** is the cross-entropy loss returned by HuggingFace's Trainer, and **BPC** is bits per character.

Based on these results, we show that Grapheme-based tokenizers do beat off-the-shelf BPE-based tokenizers. As mentioned in our Theoretical Groundwork section, the $H(T \mid I)$ term is bounded below by conditional entropy, thus lower BPC under DualGPT directly reflects how predictable the tokens are given the image. Grapheme-based tokenizers reflect the act of minimizing this term through achieving perfect visual alignment (by construction), albeit with a higher fertility rate compared to the other tokenizers ($1.5\times$ higher in Sundanese). Meanwhile, other tokenizers carry information beyond what the image exposes, but this additional information may be irrelevant to help the model actually perform efficient training. We note that mT5's lagging performance is not explained by our framework. We attribute this to the vocabulary size affecting the model's parameter count (see Table 3), leaving it in a sub-optimal state compared to models trained with other tokenizers, as predicted by scaling laws (Hoffmann et al., 2022).

[3] We use unit edit costs (match: 0, substitution: 1, indel: 1) with a deterministic tie-breaking rule favoring diagonal moves.

When viewed in the lens of language difficulty, as proxied by Figure 2, the results provide further evidence that when the image is informative and distinguishable, tokens are highly predictable, leading to low $H(T \mid I)$ term.

5.3 DualGPT Out-of-Domain Performance

We highlight that NusaAksara has an out-of-domain distribution from the Wikipedia text data, as NusaAksara was built utilizing mostly through digitalization of previously undigitalized scripts. This, however, ensure that there's no data leak during pretraining. Furthermore, this also means that tokenizers must generalize beyond the lexical and stylistic distribution they were exposed to.

Our results, shown in Table 5, demonstrate that **no single tokenizer wins across languages**. While Grapheme-based tokenizers lead on Sundanese (chrF++ 86.72), Llama 2 wins on Javanese (66.11), and Komodo wins on Balinese (33.26). We reiterate that mT5's results are confounded by its inflated parameter count, and we exclude it from cross-tokenizer claims. More notably, this OOD picture contrasts sharply with the in-domain results in Section 5.2, where Grapheme achieved the lowest BPC across *every* language. The gap between in-domain dominance and OOD inconsistency suggests that Grapheme's finer-grained tokenization may overfit to the pretraining distribution, with subword tokenizers' coarser units acting as a form of implicit regularization that aids OOD generalization.

These results are particularly interesting when viewed from the lens of data availability, as Javanese has around $1.3\times$ the data count of Sundanese, yet its performance is significantly lower. We attribute this to the language difficulty of each language, as Sundanese seems to be very easy to distinguish compared to the other two languages, as seen in Figure 2.

Lastly, utilizing the framework in this paper, the results paint a more nuanced scenario compared to the In-domain BPC results suggested. On Javanese and Sundanese, Grapheme-based tokenizers minimize $H(T \mid I)$ by construction but pay for this with reduced $\text{Inf}(\tau)$, as the tokenizers have the highest Fertility on these languages. Meanwhile, Llama2 and Komodo make the opposite tradeoff, violating visual-token grapheme boundaries ($R_{intra} \leq 0.44$ across all language) in exchange for better compression. However, for Balinese, Komodo's extended vocabulary seems to contribute significantly to its downstream performance, as the added vocabulary provides some level of lexical prior. This highlights that Grapheme-based tokenizers may be too granular.

6 Discussion

This work is an exploratory study of how text tokenizers interact within the DualGPT architecture for low-resource Abugida scripts, with its framework extendable to other dual-modality Pixel-based language models. Using the framework as an analysis tool, we find that:

Pixel-based architectures are lower-bounded by text-based architectures. Results reported by Chai et al. (2024), Gao et al. (2024), and Tai et al. (2024) all show that even dual-modality language models struggle to beat text-based language models. We provide the mathematical foundation, suggesting that this may not be something fixable. However, Pixel-based LMs' advantage has always been their flexibility and inclusivity, as they do not require any text tokenizers. We believe this is not a problem, especially as the gap has narrowed with more compute and scaling, as shown by Hu et al. (2026)'s MIXAR.

In-domain metrics may not predict downstream OOD performance. Our framework, paired with the BPC results in our Experiment Section, would suggest that Grapheme-based tokenizers should dominate downstream performance. However, Grapheme only wins on Sundanese. This divergence between in-domain language modeling loss and OOD downstream performance is, to us, an interesting observation worth flagging. We suspect that Grapheme-based tokenizers' finely-grained tokenization makes the model more prone to overfitting, or perhaps doesn't provide them room to learn proper linguistic priors. Verifying this pattern beyond our three-language scope remains an open question.

Lexical priors could be a potentially important axis our framework does not capture. On Balinese, Komodo wins downstream performance despite Grapheme scoring better on every axis our framework measures. We expect this is due to Komodo's vocabulary extension (Owen et al., 2024), which includes regional language tokens and encodes lexical priors that compensate for the other two axes.

		Model Performance		Compression	Distribution			Visual Alignment	
Language	**Tokenizer**	**chrF++** ↑	**CER** ↓	**Fertility**	**AUC**	**Slope**	**Dev** ↓	P_{intra} ↑	R_{intra} ↑
Balinese	Grapheme	30.93	46.18	3.14	64.54	-1.29	0.24	1.00	1.00
	Llama 2	29.02	65.69	3.38	64.03	-1.02	0.20	0.46	0.44
	Komodo	33.26	63.84	3.25	62.87	-0.97	0.15	0.47	0.41
	mT5	23.31	68.78	2.77	61.70	-1.03	0.13	0.55	0.32
Javanese	Grapheme	53.94	22.70	4.00	71.92	-1.35	0.22	1.00	1.00
	Llama 2	66.11	18.15	3.30	69.50	-0.95	0.14	0.18	0.25
	Komodo	60.72	18.04	3.03	67.59	-0.89	0.11	0.20	0.24
	mT5	59.22	29.12	2.59	66.36	-0.89	0.03	0.40	0.31
Sundanese	Grapheme	86.72	3.98	4.03	70.13	-1.57	0.32	1.00	1.00
	Llama 2	78.23	9.02	3.19	66.90	-1.00	0.17	0.25	0.24
	Komodo	69.09	12.75	2.83	65.38	-0.94	0.16	0.29	0.23
	mT5	66.79	23.38	2.39	64.09	-0.97	0.06	0.59	0.30

Table 5: Consolidated results across three Abugida scripts. Downstream performance (chrF++, CER) is compared against tokenizer-specific metrics: **Compression** (Fertility), Zipfian **Distribution** statistics on the train set (evaluation set lacks the raw data count), and **Visual**-grapheme boundary **Alignment** (P_{intra}, R_{intra}).

7 Future Work

Towards eventual tokenizer-free Pixel-LM. We should remember that Pixel-LM is meant to deal with the tokenization issue by removing it entirely. However, previous works and our framework strongly suggest that BCE on pixel reconstruction is a hard task, especially in low-resource monolingual settings. That said, there seems to be room to explore scheduler-based training, specifically such that the lambda attached to l_{CE} in l_{Dual} could eventually approach 0, thus relegating the use of text tokenizers to a plug-in role, like works such as Lotz et al. (2025b) did. However, this requires further analysis of the impact of the text tokenizer on architectures such as DualGPT and its other variants, such as ScreenshotLM.

Towards better Grapheme-based tokenizers. We previously mentioned that Grapheme-based tokenizers may have issues akin to byte-based or character-based tokenizers in Text-LM. This is based on the fact that graphemes are, in most cases, the minimum unit of language for these scripts. Thus, future work expanding directly on this work should explore running BPE on top of Grapheme-based tokenizers, with the hope of increasing compression rate and creating a tokenizer that supports the learning of linguistic priors.

Limitations

Single seed runs. We acknowledge that the single-seed runs of our models are not ideal. Thus, this work cannot claim statistical significance over any of its observations.

Single dual-modality architecture. Although our framework can potentially be expanded to other dual-modality architecture, this work only explored tokenizer choice impact in DualGPT.

Limited data points. Our work is comparatively small when compared to most intensive experimental work, utilizing only three languages and four tokenizers. Broader cross-language studies are needed before any of the observations can be claimed as general patterns. For now, we can only claim our observations for Abugida scripts.

Unicode-supported languages only. Although we have results for a language unsupported by Unicode, namely Lampung, its performance is untrustworthy due to the minimal volume of data the language has. This, sadly, remains one of the biggest bottlenecks for low-resource script progress in NLP. Extension to truly unsupported scripts remains as future work, contingent on data collection efforts.

Acknowledgments

Our work utilizes community-built tools for rendering and obtaining reference grapheme boundaries. We are grateful for these communities, whose work makes the computational study of these scripts possible. Other acknowledgements are anonymized due to double blind.

Ethical Considerations

Despite rendering native scripts as images, we still use the Romanized form of these scripts. While romanization is a practical necessity for our analysis, we emphasize that it should not replace native

scripts in downstream applications, as rendered images preserve the original orthography, and that the eventual return of DualGPT to become tokenizer-free is a real possibility.

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.

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

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah A. Smith. 2024. Magnet: Improving the multilingual fairness of language models with adaptive gradient-based tokenization. *Preprint*, arXiv:2407.08818.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, and 2 others. 2024. Tokenizer choice for LLM training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico. Association for Computational Linguistics.

Sawsan Alqahtani, Mir Tafseer Nayeem, Md Tahmid Rahman Laskar, Tasnim Mohiuddin, and M Saiful Bari. 2026. Stop taking tokenizers for granted: They are core design decisions in large language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8432, Rabat, Morocco. Association for Computational Linguistics.

Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. *Preprint*, arXiv:1701.04862.

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.

Danlu Chen, Freda Shi, Aditi Agarwal, Jacobo Myerston, and Taylor Berg-Kirkpatrick. 2024. LogogramNLP: Comparing visual and textual representations of ancient logographic writing systems for NLP. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14238–14254, Bangkok, Thailand. Association for Computational Linguistics.

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.

Ryan Cotterell, Sabrina J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 536–541, New Orleans, Louisiana. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Tianyu Gao, Zirui Wang, Adithya Bhaskar, and Danqi Chen. 2024. Improving language understanding from screenshots. *Preprint*, arXiv:2402.14073.

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

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *Preprint*, arXiv:1406.2661.

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.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Chen Hu, Yintao Tai, Antonio Vergari, Frank Keller, and Alessandro Suglia. 2026. Mixar: Scaling autoregressive pixel-based language models to multiple languages and scripts. *Preprint*, arXiv:2604.11575.

Jaavid J, Raj Dabre, Aswanth M, Jay Gala, Thanmay Jayakumar, Ratish Puduppully, and Anoop Kunchukuttan. 2024. RomanSetu: Efficiently unlocking multilingual capabilities of large language models via Romanization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15593–15615, Bangkok, Thailand. Association for Computational Linguistics.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

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*.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain. 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.

Jonas F. Lotz, António V. Lopes, Stephan Peitz, Hendra Setiawan, and Leonardo Emili. 2025a. Beyond text compression: Evaluating tokenizers across scales. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32155–32173, Vienna, Austria. Association for Computational Linguistics.

Jonas F. Lotz, Hendra Setiawan, Stephan Peitz, and Yova Kementchedjhieva. 2025b. Overcoming vocabulary constraints with pixel-level fallback. *Preprint*, arXiv:2504.02122.

Jessica M. Lundin, Ada Zhang, Nihal Karim, Hamza Louzan, Guohao Wei, David Ifeoluwa Adelani, and Cody Carroll. 2026. The token tax: Systematic bias in multilingual tokenization. In *Proceedings of the 7th Workshop on African Natural Language Processing (AfricaNLP 2026)*, pages 103–112, Rabat, Morocco. Association for Computational Linguistics.

Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *Preprint*, arXiv:2112.10508.

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453.

Louis Owen, Vishesh Tripathi, Abhay Kumar, and Biddwan Ahmed. 2024. Komodo: A linguistic expedition into indonesia's regional languages. *Preprint*, arXiv:2403.09362.

Vijini Pilana Liyanage and François Yvon. 2026. AdaptBPE: From general purpose to specialized tokenizers. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2607–2620, Rabat, Morocco. Association for Computational Linguistics.

David M. W. Powers. 1998. Applications and explanations of zipf's law. *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Olivier Roy and Martin Vetterli. 2007. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pages 606–610.

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

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

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.

Claude E. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.

Yu-Fei Shih, Zheng-Lin Lin, and Shu-Kai Hsieh. 2025. Reasoning over the glyphs: Evaluation of llm's decipherment of rare scripts. *Preprint*, arXiv:2501.17785.

Yintao Tai, Xiyang Liao, Alessandro Suglia, and Antonio Vergari. 2024. PIXAR: Auto-regressive language modeling in pixel space. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14673–14695, Bangkok, Thailand. 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 6 and Table 7, 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	5×10^{-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 6: Pre-training hyperparameters.

B 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 8.

C 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). Few-shot examples are randomly selected from pre-training 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

Category	Hyperparameter	Value
Model Setup	Pretrained Weight	DualGPT-pt
	Dropout	0.1
	Frozen Layers	`lm_pixel_head`
Optimization	Learning Rate	2×10^{-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 7: 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.

Parameter	Value
Background Color	White
DPI	120
Font Color	Black
Font Type	Noto Sans
Font Size	6
Max sequence length	1024
Padding size	0
Pixels per patch	16×16

Table 8: Configuration of text rendering for Javanese, Sundanese, and Balinese.

VLMs evaluation performance is available at Table 10.

D 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 9, 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 9: Transliteration sample for Balinese language. ID refers to NusaAksara's dataset entry ID.

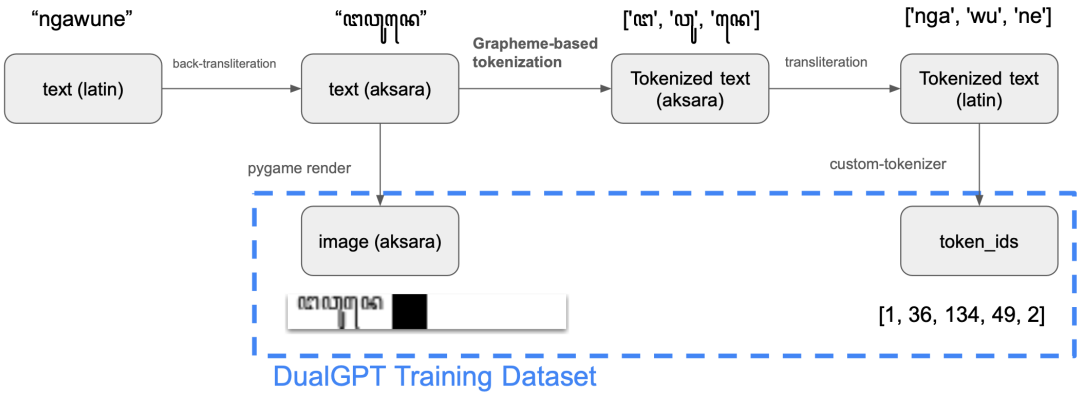


Figure 5: Dataset building process using custom tokenization rules.

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 10: 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.