Korean Text Rendering in Text-to-Image Models: A Reproducible Character-Error-Rate Benchmark

Han Kim

IOV Labs (아이오브연구소) · hankim@iovstudio.kr · ORCID 0009-0000-5998-1358

Open benchmark · snapshot 2026-05-29 · compiled 2026-05-30

Abstract. Benchmarks for text inside generated images are overwhelmingly English, which conceals the writing systems where models actually fail. We measure one of them directly. Nine text-capable text-to-image models are each asked to draw fourteen Korean (Hangul) phrases on an identical plain poster; the rendered text is transcribed by a vision-language model (GPT-4o) and scored by **character error rate** (CER). The result is a sharp ranking and one blunt failure. Three models — recraft-v4-pro, seedream-5, and nano-banana-pro — render every prompt perfectly (CER 0.000, 14/14 exact), and a clear quality gradient follows. At the bottom, **imagen-4 cannot write Hangul at all**: it produces plausible-looking Korean-shaped gibberish on every prompt (0/14, mean CER 1.33), turning 커피 한 잔 ("a cup of coffee") into 소동석 고려아는 아라해안. The central finding is that **strong English text rendering does not transfer to Korean**, and is invisible to an English-only benchmark. The harness is open, runs with a single command, resumes from saved results, and is trivially extensible to new prompts and models.

Keywords: text-to-image generation · visual text rendering · Hangul · Korean · OCR · character error rate · evaluation · benchmark · reproducibility

Contributions.

1. A reproducible, single-command benchmark of **Korean** text rendering across nine current text-to-image models, with a character-error-rate metric and exact-match rate.
2. Evidence that English text-rendering quality **does not transfer** to Hangul: a top English renderer (imagen-4) scores 0/14 on Korean.
3. A qualitative error taxonomy over Hangul-specific failure modes — complex jamo clusters, tense consonants, digit drift, and long or uncommon strings.
4. An open harness (prompts, models, OCR scoring) that resumes from saved results and is easy to extend, archived with a DOI.

Contents

1 Introduction

Modern text-to-image models can place legible words inside an image — a storefront sign, a poster headline, a product label. This capability has improved quickly for English [1], [2], and recent systems render short English strings nearly flawlessly. But "the model can render text" is a claim almost always evaluated in English, and that choice quietly hides the writing systems where the same models break. A model that writes perfect English signage may be unable to draw a single correct Korean word, and an English-only benchmark will never reveal it.

This paper measures Korean (Hangul) text rendering directly, reproducibly, and across a current field of nine models. The task is deliberately narrow and unambiguous: draw a given Korean phrase as the only text on a plain white poster, in black sans-serif lettering. Because the target string is known, rendering quality can be scored objectively — we transcribe what each model actually drew with a vision-language model and compute the character error rate against the intended text. The narrowness is the point: by stripping away style, composition, and semantics, the benchmark isolates the one capability of interest.

The headline result is stark. Three of the nine models render all fourteen prompts perfectly. A clear gradient of partial competence follows. And one model widely regarded as a strong English renderer, imagen-4, fails **completely** on Korean — not with occasional typos, but by emitting plausible-looking Hangul-shaped nonsense on every prompt. The finding generalizes beyond a single model: **visual text-rendering skill is script-specific and does not transfer from English to Korean**, and the only way to know whether a model can write a given script is to measure it in that script.

1.1 Why Korean is a good stress test

Hangul is featural and compositional: each syllable block is assembled from an initial consonant (초성), a medial vowel (중성), and an optional final consonant (받침), drawn from a set of jamo that includes doubled "tense" consonants (ㄲ, ㄸ, ㅃ, ㅆ, ㅉ) and complex final clusters (ㄳ, ㄵ, ㄼ, ㄽ, …). A renderer must place several sub-glyphs in the correct spatial arrangement **within** a single square block, then sequence the blocks. This is a more structured target than a Latin letter string, and it exposes failure modes — dropped or swapped jamo, collapsed clusters, mis-stacked blocks — that have no English analogue. Korean is therefore an informative probe of whether a model has learned glyph composition or merely memorized common English word-images.

1.2 Contributions and roadmap

Our contributions are listed above. Section 2 reviews the technical background — diffusion models, text encoders, tokenization, visual text rendering, and evaluation. Section 3 describes the Korean writing system and why its structure and encoding stress a tokenized model. Section 4 specifies the benchmark. Sections 5 and 6 present the leaderboard and a Hangul-specific error taxonomy. Section 7 weighs three mechanistic hypotheses for the failures, and Section 8 examines the validity of the OCR judge. Sections 9 and 10 discuss the practical implications and the epistemics of measurement. Sections 11 and 12 cover limitations and reproducibility, Section 13 sketches future work, and Section 14 concludes. Appendices give the full prompt set, the per-prompt CER matrix, the metric definition, a worked scoring example, and a glossary of Hangul and encoding terms.

2 Background and related work

2.1 Text-to-image generation

Contemporary text-to-image systems are largely denoising diffusion models [3], [4] conditioned on a learned text representation [5], [6]. A diffusion model learns to invert a gradual noising process, generating an image by iteratively denoising from Gaussian noise; the prompt enters as a conditioning signal, typically through cross-attention to the embeddings produced by a frozen text encoder. The image decoder never sees the prompt as characters — it sees a sequence of continuous vectors. This indirection is central to the present study: whether a model can draw a given string depends not only on the pixel decoder but on whether the **text encoder's**

representation preserves the identity of the requested characters in the first place. Early systems were notoriously poor at legible in-image text, a limitation widely attributed to the encoder's tokenization and to the scarcity of glyph-level supervision [1]. The models evaluated here are recent commercial and open systems served through a common API; we treat them as black boxes and measure only their output, but the mechanisms in Section 7 turn on this encoder–decoder structure.

2.2 Text encoders and tokenization

The two text encoders dominant in image generation are CLIP [7], a contrastively trained vision-language encoder, and T5 [8], a large text-to-text transformer used by Imagen-family models [5]. Both consume **sub-word tokens** produced by a learned tokenizer — byte-pair encoding [9] or the unigram/SentencePiece scheme [10] — rather than raw characters. Tokenization is the hinge on which script-specific rendering turns. A tokenizer trained on a predominantly English-and-code corpus allocates most of its vocabulary to frequent Latin sub-words; lower-resource scripts [11] are represented by rarer, coarser, or byte-level tokens, so a Korean syllable that a human reads as one composed block may reach the model as an opaque or fragmented unit. Liu et al. [1] make the causal link explicit: replacing the sub-word encoder with a **character-aware** one substantially improves visual text rendering, which directly implicates tokenization as a cause of failure. Korean is an especially sharp test of this hinge, for reasons that are best stated by describing the writing system itself (Section 3).

2.3 Visual text rendering

A focused line of work has improved in-image text specifically. Liu et al. [1] show that **character-aware** text encoders — which see characters rather than only sub-word tokens — substantially improve visual text rendering, directly implicating tokenization as a cause of failure. Glyph- and layout-conditioned methods such as GlyphControl [12] and TextDiffuser [2] add explicit glyph or position control to place text more reliably. This literature is overwhelmingly evaluated on English (and occasionally Chinese); Korean is rarely measured, despite Hangul's compositional structure making it an especially clean test of glyph-level competence. Our benchmark fills that specific gap with a direct, reproducible measurement.

2.4 Evaluation: OCR, edit distance, and model-as-judge

Rendering quality is naturally scored by reading the produced text and comparing it to the target. The comparison is an edit-distance problem: the Levenshtein distance [13] counts the minimum single-character insertions, deletions, and substitutions to transform one string into another, and normalizing it by target length yields the **character error rate** (CER), a standard measure in OCR and speech recognition. The reading step itself uses a model — here a vision-language model transcribes the image — which connects to the broader "model-as-judge" paradigm [14] and inherits its central caveat: the judge can be wrong, so the metric is a calibrated proxy, not ground truth (Section 6.2). Reference-free image-text metrics such as CLIPScore [15] measure semantic alignment, not character-exact rendering, and are therefore unsuitable for this task; exact transcription plus CER is the appropriate instrument.

3 The Korean writing system, technically

Korean is written in Hangul, an alphabet whose letters are grouped into syllabic blocks. Understanding why this is hard for a tokenized generative model requires three layers: the script's compositional structure, its Unicode encoding, and how that encoding meets a sub-word tokenizer.

3.1 Compositional structure

Each Hangul syllable block is composed of two or three **jamo** (letters): an initial consonant (초성, one of 19), a medial vowel (중성, one of 21), and an optional final consonant or consonant cluster (종성, one of 27 plus the empty final). The combinatorics give $19 \times 21 \times 28 = 11{,}172$ possible modern syllable blocks. Crucially, the jamo are not written left to right like Latin letters; they are **arranged within a square cell** — initial at top-left,

vowel to the right or below depending on its orientation, final consonant along the bottom. A renderer must therefore solve a small two-dimensional layout problem **inside every character**, and then sequence the cells. Two structural features make this unforgiving: **tense (doubled) consonants** (ㄲ, ㄸ, ㅃ, ㅆ, ㅉ), which differ from their plain counterparts only by a repeated stroke, and **complex final clusters** (ㄳ, ㄵ, ㄶ, ㄺ, ㄻ, ㄼ, …), which stack two consonants into the already-crowded bottom slot. These are exactly the cells that the weaker models drop or merge (Section 6).

3.2 Unicode encoding and normalization

Unicode encodes Hangul two ways [16]. The **precomposed** form places each of the 11,172 syllables at its own code point in the Hangul Syllables block (U+AC00–U+D7A3), so 맑 is the single code point U+B9D1. The **decomposed** form spells the same syllable as a sequence of conjoining jamo from the Hangul Jamo block (U+1100–U+11FF): 맑 becomes ㅁ + ㅏ + ㄹ ㄱ. The two are related by Unicode normalization — NFC composes jamo into precomposed syllables, NFD decomposes them — and are visually identical when rendered but are **different byte sequences**. A model's behaviour can depend on which form its training text and tokenizer use: a precomposed corpus presents each syllable as one (often rare) code point, while a decomposed corpus exposes the jamo structure but lengthens every string. Neither is uniformly better, and the mismatch is a documented source of subtle text-processing bugs.

3.3 Where it meets the tokenizer

Put the two together. Under precomposed encoding, a syllable such as 맑 is one code point that a sub-word tokenizer trained mostly on English will likely see as a single rare token or a multi-byte fallback — an atom with little learned internal structure, so the model has no compositional handle on the ㄻ cluster it contains. Under decomposed encoding the structure is visible but the sequence is long and unusual. Either way, unless the encoder is character- or jamo-aware [1], the path from "the user asked for 맑음" to "place these specific jamo in these specific cells" is lossy. This is the technical reason to expect that a model can learn the **statistics of Hangul's appearance** — what well-formed syllable blocks look like — while failing to render **specific requested** strings, which is precisely the imagen-4 signature documented in Section 4. We stress that this is a hypothesis about a mechanism; the models are black boxes, and Section 7 treats the evidence for and against it.

4 Benchmark design

4.1 Task

For each (model, prompt) pair the model receives an identical instruction: produce a white poster whose only text is the target Hangul string, set in black sans-serif lettering. Fixing the surface — white background, single black string, no decoration — isolates **text rendering** from style, layout, and semantics, so that any error is attributable to the model's ability to draw the requested glyphs rather than to compositional choices.

4.2 Prompts

The fourteen prompts span easy-to-hard Hangul, chosen to exercise the script's structure rather than its vocabulary (Appendix A). They range from a short greeting (안녕하세요) and word-spacing (커피 한 잔) through deliberately hard cases: rare final clusters (값을 매기다 with ㅄ, 맑음 with ㄻ), tense consonants (떡볶이), a complex cluster plus tense consonant (닭갈비 맛집, ㄺ), a longer brand string (주식회사 아이오브), a full sentence (안녕하세요 반갑습니다), place names (서울특별시 강남구), and digit–Hangul mixes (9월 14일 토요일, 커피 2잔 주세요). The set is small by design — the goal is a sharp, reproducible probe, not exhaustive coverage — and is trivially extensible via the repository's `prompts.json`.

4.3 Models

Nine text-capable models served through the fal.ai API were evaluated in the 2026-05-29 snapshot: recraft-v4-pro, seedream-5, nano-banana-pro, gpt-image-2, recraft-v4, gpt-image-1.5, flux-2-flash, ideogram-v3, and imagen-4. One image was generated per (model, prompt) cell, for 126 generations in total. Model endpoints and versions move over time; the numbers are therefore explicitly dated.

4.4 Scoring

Let g be the string the model actually rendered and t the target. A vision-language model (GPT-4o) transcribes the largest visible text in the image to obtain g. The character error rate is the length-normalized Levenshtein distance with whitespace removed:

$$\mathrm{CER}(g, t) = \frac{\mathrm{lev}(g', t')}{|t'|}, \tag{1}$$

where x' denotes x with whitespace stripped and lev is the Levenshtein edit distance [13]. $\mathrm{CER} = 0$ is a perfect render; values near or above 1 indicate text bearing little or no relation to the target. For each model we report the mean CER over the fourteen prompts and the **exact-match rate**, the fraction of prompts rendered with $\mathrm{CER} = 0$.

CER is the character-level analogue of the word error rate standard in speech recognition, and inherits its properties. It is a **quasi-metric** on strings: non-negative, zero iff the strings are identical, and symmetric, but unbounded above — and there lies its most informative feature for this task. Because the numerator is an edit **count** and the denominator is the target length, $\mathrm{CER} > 1$ whenever the rendered string requires more single-character edits than the target has characters, which happens when the output is both wrong and not shorter than the target. A model that **omits** text caps its CER near 1 (delete everything); a model that **confabulates** wrong text of similar or greater length can exceed 1. The distinction matters here: imagen-4′s mean of 1.33 is not a scaling artefact but a substantive signal that it emits confident wrong characters rather than blanks (Section 7). We strip whitespace before scoring because poster line-breaking is a layout decision orthogonal to whether the correct glyphs were drawn; we do **not** apply Unicode normalization, so a precomposed and a decomposed rendering of the same syllable would score identically only after the OCR step returns one canonical form, a caveat we revisit in Section 8.

Figure 1: The same prompts drawn by the nine models (one frame of the cycling comparison); each label shows the model and its CER on that prompt.

5 Results

5.1 Leaderboard

Table Table 1 gives the full ranking. The field separates into three regimes: a perfect tier, a competent gradient, and a single complete failure.

Rank	Model		Mean CER	Exact
1	recraft-v4-pro		**0.000**	14 / 14 (100%)
2	seedream-5		**0.000**	14 / 14 (100%)
3	nano-banana-pro		**0.000**	14 / 14 (100%)
4	gpt-image-2		0.038	12 / 13 (92%)
5	recraft-v4		0.071	13 / 14 (93%)
6	gpt-image-1.5		0.083	12 / 14 (86%)
7	flux-2-flash		0.145	9 / 14 (64%)
8	ideogram-v3		0.302	5 / 14 (36%)
9	imagen-4		1.332	0 / 14 (0%)

Table 1: Korean text-rendering leaderboard, nine models × fourteen Hangul prompts (2026-05-29 snapshot via fal.ai). Lower mean CER and higher exact-match rate are better.

Figure 2: The harness prints the full leaderboard and a per-model exact-match bar chart in the terminal, reproducible with one command.

5.2 Three perfect renderers

recraft-v4-pro, seedream-5, and nano-banana-pro rendered all fourteen prompts with zero character error, including the hard cluster and tense-consonant cases. Their existence is the most important positive result: **correct Hangul rendering is achievable today**, so the failures below are not an inherent limitation of the medium but a property of specific models.

5.3 The competent gradient

Between the perfect tier and the failure lies a smooth gradient. gpt-image-2 (0.038), recraft-v4 (0.071), and gpt-image-1.5 (0.083) miss only on the hardest one or two strings. flux-2-flash (0.145, 9/14) and ideogram-v3 (0.302, 5/14) degrade on complex jamo and longer or less common strings, in the systematic ways catalogued in Section 5. The gradient is informative: it shows that Hangul competence is partial and string-dependent, not a binary the model either has or lacks — except at the extremes.

5.4 The blunt failure

imagen-4 is the outlier and the paper's central cautionary example. It scored 0 of 14 with a mean CER of 1.33 — above one, meaning its output is on average **further** from the target than an empty string would be, because it confidently emits wrong characters rather than omitting them. It does not produce Korean with occasional errors; it produces **Hangul-shaped nonsense** on every prompt: 커피 한 잔 → 소동석 고려아는 아라해안, and 맑음 → 옹반재다 (Figure Figure 3). The glyphs are individually well-formed Korean syllable blocks, which is what makes the failure striking — the model has learned what Hangul **looks like** without learning to render **specific** requested words.

Figure 3: imagen-4 on Korean prompts. Each panel's label is the intended text; the image is what the model actually drew — well-formed but unrelated Hangul.

6 Error analysis

Beyond the aggregate scores, the **kinds** of errors are diagnostic of where glyph composition breaks. We group the observed failures (excluding imagen-4, whose output is unrelated to the target) into four Hangul-specific modes.

Complex final clusters (겹받침). Syllables ending in a two-consonant cluster are a frequent failure point. The cluster ㄺ in 맑음 collapses to a single consonant — flux-2-flash renders 맘음 — losing one jamo from the block. These clusters require stacking two final consonants in a position that is itself already crowded, and weaker renderers drop or merge them.

Tense (doubled) consonants. Doubled consonants are mis-rendered as their single counterparts: 떡볶이 → 덕볶이 (flux-2-flash), where the tense ㄸ is read back as a plain ㄷ. The doubling is a fine visual distinction that lower-fidelity renderers smear.

Digit–Hangul mixing. Strings that mix Arabic numerals with Hangul are a soft spot for the weaker renderers: on the date prompt 9월 14일 토요일, flux-2-flash drifts (CER 0.375) while the perfect tier and even ideogram-v3 handle it. Switching scripts mid-string appears to destabilize the character count.

Long or uncommon strings. Less frequent or longer strings degrade most: the brand string 주식회사 아이오브 collapses in flux-2-flash (CER 0.625), dropping several syllables. Rarer word-images give the model less to memorize and more to compose, exposing weak composition.

The hardest single prompt, 닭갈비 맛집 — combining a complex final cluster (ㄺ in 닭) with a tense consonant — is shown across all nine models in Figure Figure 4. The spread on this one prompt mirrors the leaderboard: the perfect tier draws it cleanly, the gradient stumbles on the cluster, and imagen-4 produces unrelated glyphs.

Figure 4: The hardest prompt, 닭갈비 맛집, rendered by all nine models (label = model + CER). The complex cluster ㄺ and the tense consonant separate the field.

7 Mechanisms: why Hangul rendering fails

The black-box nature of the models forbids a definitive causal account, but the **shape** of the errors constrains the space of explanations. We state three hypotheses and weigh each against the observed pattern.

7.1 Hypothesis 1: tokenization starvation

If a model's text encoder tokenizes Korean into rare or byte-level units (Section 3.3), the conditioning signal carries the **presence** of "some Korean" more reliably than the **identity** of each requested syllable. The prediction is a gradient that tracks string frequency and length: common short strings, whose token sequences the model has seen often, render well; rare clusters, brand names, and long strings degrade. This matches the data closely — the failures concentrate on 값을 매기다 (ㅄ), 맑음 (ㄻ), 주식회사 아이오브 (long, uncommon), while greetings and common phrases are universally perfect — and it is consistent with the causal evidence that character-aware encoders fix exactly this [1]. Tokenization starvation is the best-supported hypothesis.

7.2 Hypothesis 2: glyph manifold without conditioning

imagen-4 is the critical case. It does not blur or omit; it draws **well-formed but unrequested** syllable blocks — 커피 한 잔 becomes 소동석 고려아는 아라해안, glyphs that are individually valid Hangul. This is the signature of a decoder that has learned the **manifold of Hangul-shaped images** — what the script looks like in aggregate — decoupled from any faithful mapping from the conditioning vector to specific characters. The model has fluency in glyph-space and no faithfulness to the prompt: it knows how to make Korean-looking marks but not which marks the user asked for. A CER above one is the quantitative fingerprint of this regime (Section 4.4). Whether the break lies in the encoder (the character identity never survives tokenization) or in the cross-attention (the identity is present but not used) cannot be resolved from outputs alone, but the **form-without-identity** character of the failure is unambiguous in the images.

7.3 Hypothesis 3: memorization versus composition

The two hypotheses above share a deeper axis: does a model **compose** a requested syllable from its jamo, or **retrieve** a remembered word-image? A composer should generalize to rare-but-regular strings; a retriever should excel on frequent strings and fail on novel ones. The observed gradient — perfect on common, degrading

precisely on the rare and the long — is the signature of retrieval-dominant behaviour with weak composition, the visual-text analogue of the compositional-generalization gap documented for sequence models [17]. The three perfect models demonstrate that compositional Hangul rendering is **achievable**; the gradient and the imagen-4 failure show that several deployed systems have not achieved it, and instead lean on memorized appearance.

8 The OCR judge: validity and bounds

Our metric reads each image with a vision-language model, so the judge is itself a fallible model [14], and a benchmark is only as trustworthy as its instrument. We treat the judge's validity explicitly rather than asserting it.

Failure modes of the judge. A judge error is a false negative (it misreads a correct render, inflating CER) or a false positive (it "reads" the intended text from an incorrect image, deflating CER). The task is deliberately constructed to suppress false positives: targets are short, the surface is a clean high-contrast poster, and the prompt asks for one string, so a perfect transcription is unlikely to be awarded to a wrong image. False negatives are more plausible and would, if present, **understate** the top models — yet three models still score exactly zero across fourteen prompts, bounding the judge's false-negative rate on clean renders at near zero in this sample.

The headline is robust to judge noise. The result that anchors the paper — imagen-4 at mean CER 1.33, 0/14 — sits far outside any plausible OCR error band. No transcription mistake turns a faithful 커피 한 잔 into 소동석 고려아는 아라해안; the judge is reading, accurately, glyphs the model truly drew. The conclusion that imagen-4 cannot render Hangul therefore does not depend on the judge being perfect, only on its being approximately able to read clear Korean, which contemporary vision-language models demonstrably are.

Where judge noise does matter. The fine ordering in the middle of the table — gpt-image-2 at 0.038 versus recraft-v4 at 0.071 — is within the range where a single misread shifts a rank. Those distinctions should be read as approximate, and a stronger design would ensemble multiple OCR judges or add human adjudication on disagreements; we flag this as the metric's principal soft spot (Sections 11 and 13) rather than papering over it.

9 Discussion

9.1 English skill does not transfer to Korean

The single most important takeaway is a non-transfer result. imagen-4 is, by reputation and on English benchmarks, a strong text renderer; on Korean it is the worst model tested, unable to draw a single correct word. Visual text rendering is therefore **script-specific**: competence in one writing system says little about another, especially across a script boundary as large as Latin-to-Hangul. The most plausible mechanism is consistent with the character-aware-encoder finding of Liu et al. [1] — if a model's text conditioning is tokenized in a way that fragments or under-represents Hangul jamo, the model can learn the **appearance** of Korean (well-formed syllable blocks) without learning to compose **specific** targets, exactly the imagen-4 signature. Whatever the cause, the practical consequence is unambiguous: a model's English text score must not be assumed to hold for Korean, and the only reliable knowledge is a measurement in the target script.

9.2 Implications

For practitioners, the result is directly actionable: if an application renders Korean, the model must be chosen on a Korean measurement, and three models are shown to be safe choices today. More broadly, the benchmark is a concrete instance of a cheap automatic checker (OCR + CER) used as a **quality gate** and a **routing signal** — the same pattern the lab has argued for in generative-media verification: a fast, objective check that can both reject bad outputs and route a request to a model known to handle its script.

10 Epistemics and the philosophy of measurement

A benchmark is a small epistemological instrument, and this one — by isolating a single, objectively checkable capability across a script boundary — sharpens several questions that are usually left blurry. We take them up directly, because the lab's position is that the value of a measurement lies as much in what it teaches about knowing as in the leaderboard it produces.

10.1 Form without identity

The imagen-4 failure is philosophically peculiar in a way worth naming. The model does not produce noise or blanks; it produces **well-formed Hangul that is not the requested Hangul**. It has captured the **form** of the script — the statistical look of valid syllable blocks — while missing the **identity** of the specific characters asked for. This is a visual echo of the distinction Bender and Koller [18] draw between form and meaning in language models: a system trained to reproduce surface form can be fluent without being faithful. Here the inversion is instructive — usually "form" is syntax and "meaning" is semantics, but in glyph rendering the **form** is the script's appearance and the **identity** is the referential link to the requested string. imagen-4 is a system with the syntax of Hangul and none of its reference, and it is confidently so. One is reminded of Searle's Chinese Room [19]: symbol shapes manipulated into plausible arrangements with no grip on what they denote. We invoke the analogy not to settle the old debate about understanding but to label a concrete, measurable phenomenon — **plausible glyph-shaped output decoupled from the requested content** — that a renderer can exhibit and a benchmark can catch.

10.2 Memorization, composition, and what "can write Korean" means

Behind the leaderboard is a question of competence versus performance. To say a model "can write Korean" could mean it has **memorized** the appearance of many Korean strings, or that it can **compose** any well-formed string from the script's parts. These are different capabilities that a coarse benchmark would conflate and that the easy-to-hard gradient pulls apart: a memorizer aces frequent strings and fails novel ones; a composer generalizes [17]. The fourteen prompts are chosen so that the rare clusters and uncommon strings act as a probe for composition, and the observed pattern says most of the field is memorization-dominant, with three genuine composers at the top. The philosophical payoff is a more honest predicate: not "model X can write Korean," but "model X renders frequent Korean and fails on rare structure," which is what a user actually needs to know.

10.3 Confident error and the calibration of generation

A model that omits text when unsure would be, in a sense, well-calibrated: its silence signals its ignorance. imagen-4 does the opposite — it emits wrong characters with full visual confidence, the generative-image analogue of a language model's confabulation [20]. The CER-above-one regime is the quantitative trace of **miscalibrated generation**: the model's fluency is uncorrelated with its faithfulness, so its output carries no internal signal of its own failure. This is the deeper danger for downstream use. A blank or a blur is visibly broken; well-formed Korean nonsense on a poster looks correct to anyone who cannot read it, and ships. Measurement is the only defense, which is the whole motivation for a cheap automatic check.

10.4 What you do not measure, you do not see

The most consequential choice in this paper is upstream of any model: the decision to measure Korean at all. Text-in-image rendering is overwhelmingly benchmarked in English, and that convention is not neutral — it renders the failure documented here **invisible**. An English-only evaluation would have ranked imagen-4 among the strong renderers and never surfaced that it cannot write a single Korean word. This is a small instance of a general epistemic hazard: a field's metrics quietly define what it can perceive, and scripts that no one measures become scripts whose failures no one fixes [11]. The corrective is not heroic, only deliberate — measure the writing systems your users actually use — and it has an equity dimension, since the unmeasured scripts are disproportionately those of non-English-speaking communities.

10.5 Reproducibility as honesty

Finally, the instrument is built to be doubted. The metric is computed by a fallible judge, the sample is small, and the model endpoints drift; rather than hide these, the design dates every number, ships the raw transcriptions, resumes from saved results so anyone can re-run a cell, and states the judge's soft spots (Section 8). A benchmark one cannot reproduce or interrogate is an assertion, not a measurement. The point is not that this snapshot is the final word — it is explicitly not — but that it is **checkable**, and a checkable claim that may be wrong is worth more than an unfalsifiable one that sounds right.

11 Limitations and threats to validity

We are explicit about what this benchmark does and does not establish.

The OCR judge is fallible. CER is a proxy computed from a model's transcription, not ground truth; fine differences in the middle of the table are approximate (Section 6.2).

Small n. One image per (model, prompt) cell, with no sweep over seeds, aspect ratios, or prompt phrasings. This is a **snapshot**, not a verdict: a model near a boundary could move with a different seed. The fourteen prompts probe structure, not coverage of the language.

Endpoints move. Model versions and API paths change; the numbers are dated 2026-05-29 and should be re-run before being cited as current.

Surface specificity. The clean-poster surface isolates rendering but is not representative of in-the-wild prompts (busy scenes, stylized type); a model could render Hangul well in isolation yet poorly in a complex composition, or vice versa.

Single language. Korean is one informative script; the non-transfer claim is demonstrated from English to Korean and is not a general statement about every script pair, though it strongly motivates per-script measurement.

12 Reproducibility

The benchmark is open and runs with a single command given a fal.ai key (generation) and an OpenAI key (OCR):

```
export FAL_API_KEY=...      # image generation
export OPENAI_API_KEY=...   # OCR transcription (GPT-4o)
node run.mjs                # writes results.json + REPORT.md
node summary.mjs            # pretty-prints the saved leaderboard
```

A full run is roughly 126 generations (a few dollars of API). The harness **resumes from saved results**: re-running retries only failed cells, so a partial or interrupted run is cheap to complete. The prompt list (`prompts.json`) and the model list (in `run.mjs`) are the two extension points. The repository ships a `CITATION.cff`, is archived on Zenodo with a DOI, and licenses code under MIT and the writeup and results under CC BY 4.0.

13 Future work

Several extensions would deepen the measurement without changing its character. The most direct is **scale**: more prompts spanning a controlled difficulty grid (systematically varying cluster type, string length, and frequency), multiple images per cell with seed sweeps to turn the snapshot into an estimate with confidence intervals, and a larger model roster refreshed on a schedule so the leaderboard tracks a moving field. Second, **judge robustness**: an ensemble of OCR systems with disagreement-triggered human adjudication would tighten the fine ordering in the middle of the table and quantify the judge's own error rate, converting the soft spot of Section 8 into a measured quantity. Third, **mechanism**: pairing each render with the model's tokenization of the prompt — where the API exposes it — would let the tokenization-starvation hypothesis

(Section 7.1) be tested directly rather than inferred from the error gradient, and comparing precomposed against decomposed prompt encodings would isolate the normalization effect of Section 3.2. Fourth, **breadth across scripts**: the same harness applied to other non-Latin, compositional, or low-resource writing systems [11] would test whether the form-without-identity failure is specific to Hangul or a general property of under-tokenized scripts, and would extend the equity argument of Section 10.4 with data. Finally, **positional error analysis**: scoring at the jamo level (초성/중성/종성) rather than the syllable level would reveal whether failures concentrate in the crowded final-consonant slot, as the qualitative taxonomy suggests.

14 Conclusion

We measured how well nine current text-to-image models draw Korean, with a narrow, objective, one-command benchmark: render a Hangul phrase on a plain poster, transcribe it, score it by character error rate. Three models are perfect, a clear gradient of partial competence follows, and one model widely held to be a strong English renderer — imagen-4 — cannot write Hangul at all, producing well-formed nonsense on every prompt. The lesson is general and easy to state: **visual text-rendering skill does not transfer across scripts, and you only learn whether a model can write Korean by measuring it in Korean.** The harness is open, dated, and reproducible, so the measurement can be repeated, extended to other scripts, and kept current as the models move.

Data and code availability. The harness, prompts, raw results (`results.json`, including image URLs), per-prompt report (`REPORT.md`), and this paper's source are public in the project repository, archived on Zenodo with a DOI. Code is MIT-licensed; the writeup and results are CC BY 4.0.

Acknowledgements. Internal research of IOV Labs (아이오브연구소). Image generation via fal.ai endpoints; OCR via GPT-4o. Typeset with Typst; terminal figures recorded with `vhs`.

References

[1] R. Liu *et al.*, "Character-Aware Models Improve Visual Text Rendering," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023, pp. 16270–16297.

[2] J. Chen, Y. Huang, T. Lv, L. Cui, Q. Chen, and F. Wei, "TextDiffuser: Diffusion Models as Text Painters," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 10684–10695.

[5] C. Saharia, W. Chan, S. Saxena, and others, "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[6] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," *arXiv preprint arXiv:2204.06125*, 2022.

[7] A. Radford, J. W. Kim, C. Hallacy, and others, "Learning Transferable Visual Models From Natural Language Supervision," in *International Conference on Machine Learning (ICML)*, 2021, pp. 8748–8763.

[8] C. Raffel, N. Shazeer, A. Roberts, and others, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[9] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016, pp. 1715–1725.

[10] T. Kudo and J. Richardson, "SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing," in *Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, 2018, pp. 66–71.

[11] P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury, "The State and Fate of Linguistic Diversity and Inclusion in the NLP World," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 6282–6293.

[12] Y. Yang *et al.*, "GlyphControl: Glyph Conditional Control for Visual Text Generation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[13] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[14] L. Zheng, W.-L. Chiang, Y. Sheng, and others, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[15] J. Hessel, A. Holtzman, M. Forbes, R. Le Bras, and Y. Choi, "CLIPScore: A Reference-free Evaluation Metric for Image Captioning," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021, pp. 7514–7528.

[16] The Unicode Consortium, "The Unicode Standard: Hangul Syllables (U+AC00–U+D7A3) and Hangul Jamo (U+1100–U+11FF)." 2024.

[17] B. M. Lake and M. Baroni, "Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks," in *International Conference on Machine Learning (ICML)*, 2018, pp. 2873–2882.

[18] E. M. Bender and A. Koller, "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 5185–5198.

[19] J. R. Searle, "Minds, Brains, and Programs," *Behavioral and Brain Sciences*, vol. 3, no. 3, pp. 417–424, 1980.

[20] Z. Ji, N. Lee, R. Frieske, and others, "Survey of Hallucination in Natural Language Generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

Appendix A — Prompt set

The complete fourteen-prompt set (from the repository's `prompts.json`), ordered as in the harness, with the Hangul structure each one targets:

id	target	what it tests
greeting	안녕하세요	basic greeting (baseline)
spacing	커피 한 잔	word spacing
batchim	책갈피	final consonants (받침)
date	9월 14일 토요일	digits + Korean
phrase	행복을 드립니다	full phrase with 받침
doublecons	닭갈비 맛집	complex jamo cluster ㄹㄱ (hardest)
brand	주식회사 아이오브	longer brand string
short	사랑해	short, simple
value	값을 매기다	rare cluster ㅄ
clear	맑음	rare cluster ㄺ
tteok	떡볶이	tense consonants ㄸ / ㄲ
sentence	안녕하세요 반갑습니다	longer sentence
place	서울특별시 강남구	place names
order	커피 2잔 주세요	number + honorific

The set is intentionally small — a sharp probe of structure rather than broad lexical coverage — and is designed to be extended by users for their own targets.

Appendix B — Per-prompt character error rate

Full CER matrix (2026-05-29 snapshot). Columns are the nine models in leaderboard order; "—" marks a generation that failed to return an image. A perfect render is 0.000; values above 1 (imagen-4) indicate output further from the target than an empty string.

prompt	r4-pro	sd-5	nbp	gi-2	r4	gi-1.5	flux	ideo	im-4
greeting	0	0	0	0	0	0	0	0	1.00
spacing	0	0	0	0	0	0	0	0	2.75
batchim	0	0	0	0	0	0	0	0.67	1.33
date	0	0	0	0	0	0	0.38	0	0.50
phrase	0	0	0	0	0	0	0	0.29	1.00
doublecons	0	0	0	0	0	0	0	0.40	1.80
brand	0	0	0	0	0	0	0.63	0	1.00
short	0	0	0	0	0	0	0	0.33	1.33
value	0	0	0	—	0	0	0.20	0.20	1.60
clear	0	0	0	0.50	1.00	0.50	0.50	1.00	2.00
tteok	0	0	0	0	0	0.67	0.33	0.67	1.33
sentence	0	0	0	0	0	0	0	0.30	1.00
place	0	0	0	0	0	0	0	0.38	1.00
order	0	0	0	0	0	0	0	0	1.00

Columns: recraft-v4-pro, seedream-5, nano-banana-pro, gpt-image-2, recraft-v4, gpt-image-1.5, flux-2-flash, ideogram-v3, imagen-4. The single hardest prompt is `clear` (맑음, cluster ㄺ), the only one to trip a member of the otherwise-strong middle tier; imagen-4 fails uniformly.

Appendix C — Metric definition

For target t and rendered (transcribed) string g, let t', g' be the strings with all whitespace removed. The Levenshtein distance $\mathrm{lev}(g', t')$ is the minimum number of single-character insertions, deletions, and substitutions transforming g' into t' [13]. The character error rate is

$$\mathrm{CER} = \frac{\mathrm{lev}(g', t')}{|t'|}, \tag{2}$$

which is 0 for a perfect render and can exceed 1 when the rendered string is both wrong and not shorter than the target (the imagen-4 regime, mean 1.33). The **exact-match rate** for a model is the fraction of its prompts with $\mathrm{CER} = 0$. Mean CER averages Equation 1 over the fourteen prompts. Whitespace is ignored because poster line-breaking is a layout choice orthogonal to whether the correct characters were drawn.

Appendix D — Worked scoring example

Take the target $t =$ 맑음 (two syllables) and a rendering transcribed as $g =$ 맘음. After whitespace removal both have length 2, so $|t'| = 2$. Transforming 맘음 into 맑음 requires substituting the first syllable 맘 → 맑 (one edit; the final cluster ᆰ is restored), and the second syllable 음 already matches. Thus $\mathrm{lev}(g', t') = 1$ and

$$\mathrm{CER} = 1/2 = 0.500, \tag{3}$$

the value recorded for flux-2-flash on the `clear` prompt (Appendix B). For contrast, imagen-4 on the `spacing` prompt rendered 커피 한 잔 (length 4 after stripping spaces) as a string sharing essentially no characters with the target, requiring on the order of eleven edits against the four-character target, for $\mathrm{CER} = 2.75$ — a value far above 1 that records confident, wholly wrong output rather than a near miss. The two examples bracket the metric's range: a single restorable jamo costs 0.5 on a short word, while unrelated output costs more than the target's own length.

Appendix E — Glossary of Hangul and encoding terms

Term	Meaning
Hangul (한글)	the Korean alphabet, whose letters group into syllabic blocks
jamo (자모)	the individual Hangul letters (consonants and vowels) that compose a block
초성 / 중성 / 종성	initial consonant / medial vowel / final consonant(s) of a syllable block
받침 (batchim)	the final consonant slot at the bottom of a block; empty, single, or a cluster
tense consonant	a doubled consonant (ㄲ, ㄸ, ㅃ, ㅆ, ㅉ), distinct from its plain form
complex cluster (겹받침)	a two-consonant final, e.g. ㄺ in 닭, ㄻ in 맑, ㅄ in 값
precomposed (NFC)	Unicode form with one code point per syllable (U+AC00–U+D7A3)
decomposed (NFD)	Unicode form spelling a syllable as conjoining jamo (U+1100–U+11FF)
normalization	NFC/NFD conversion between the two forms; visually identical, different bytes
tokenizer	the sub-word segmenter (BPE / SentencePiece) that maps text to model tokens
CER	character error rate: edit distance to the target, normalized by target length
exact-match rate	fraction of prompts a model renders with CER 0