Jamo Is All You Need: Enhancing Korean OCR with Style Tags

Younguk Kim Seokhoon Kang Ye Ji Chun Juhee Chae
Seoul National University, Korea
{yu5.kim, getar98, yejichun, chaejuhee}@snu.ac.kr

Abstract

We present an efficient and generalizable OCR framework for Hangeul that leverages sub-character (jamo-level) prediction instead of treating each syllable as a distinct class. By decomposing each character into its constituent initial, medial, and final components, our model reduces the output class space from over 11,000 full syllables to just 144 jamo units. This leads to a 40% reduction in model size and improved generalization to rare or unseen syllables. When fine-tuned on multilingual datasets, our jamo-based OCR achieves higher word accuracy (97.84%) than a conventional full-syllable baseline (97.31%), while preserving compatibility with Latin scripts. In addition, we extend the model to recognize formatting styles such as superscripts and subscripts by introducing a parallel classification head and training on a custom synthetic dataset. Despite class imbalance, the model learns to identify visual position cues and produce layout-aware outputs. These results demonstrate the benefit of structurally aligned modeling for complex writing systems like Hangeul and open up avenues for style-aware OCR in multilingual scenarios.

1. Introduction

Optical character recognition (OCR) is a technology that extracts text from images, increasingly powered by machine learning and deep learning. With the advancement of deep learning, OCR performance has improved significantly. Typically, an OCR system converts the text in an image into a machine-encoded format (such as Unicode), enabling digital usage. A wide range of approaches have been explored, from classical pattern recognition methods to modern high-precision deep learning models. Recently, a model called PARSeq [1] introduced a permutation-based autoregressive sequence training strategy for text recognition, achieving state-of-the-art results.

These OCR models perform well on many benchmarks. However, most of the research has focused on the English alphabet, and there is a lack of studies targeting scripts such as Hangeul (the Korean writing system). Hangeul characters are composed of smaller units (called *jamo*), and we hypothesize that recognizing these subcomponents instead of whole syllables could be more effective. Therefore, in this work, our aim is to contribute by proposing a model that recognizes Hangeul at the jamo level. This approach differentiates our work from previous Korean OCR studies as we attempt to recognize subunits using compositional components, which we believe bring notable benefits. After we developed a model to detect Hangeul successfully, we also attempted to upgrade the model by adding the function to distinguish normal, superscript, or subscript letters. We realized that few previous studies have thoroughly explored the recognition of superscripts and subscripts using recent OCR models, let alone integrated this with Hangeul recognition. Therefore, we also further tested a model variant that included a function to distinguish whether a character is a superscript, subscript, or a normal letter. In this aspect, we believe that our work could contribute to future superscript and subscript notation recognition, not only for Hangeul but also for Latin-based languages.

Hangeul syllabic blocks consist of individual consonants and vowels. Existing OCR systems generally treat each syllable as an atomic character, thereby ignoring Hangeul's compositional nature. By contrast, our method explicitly leverages Hangeul's structure: we extend the PARSeq framework to recognize the constituent jamo of each character and assemble them into syllables. This design aligns the OCR process with the intrinsic makeup of Hangeul, potentially improving accuracy and generalization for Hangeul text.

In many real-world OCR applications, practitioners have adapted open-source or commercial OCR engines

to support multiple languages by expanding the character set (e.g., including Hangeul, Chinese characters (Hanja), Latin letters, special symbols, etc.). However, existing approaches—including open-source systems like EasyOCR [2] and PaddleOCR [3], as well as recent research work [1, 4]—share certain limitations because they predict each character in an image directly as a Unicode code point. We highlight two major issues with this paradigm:

1. **Explosion of character classes for compositional scripts.** Languages such as Korean and Chinese have extremely large sets of possible characters. For example, English text can be handled with 52 classes (26 letters in two cases), whereas modern Hangeul requires 11,172 distinct syllable classes. Similarly, Latin-based languages introduce additional letters with diacritics (Á, Ç, Ü, etc.), dramatically increasing the number of symbols to recognize. An OCR model that treats each possible character as a separate class faces increased complexity and data sparsity.

2. **Difficulty in recognizing text style information.** Many visual text elements encode stylistic or formatting information (such as subscripts, superscripts, underlines, strikethroughs, or italics) that cannot be captured by a single Unicode character. For instance, the number **0** versus the superscript $^{\mathbf{0}}$, or the presence of underlining or strikethrough, represent distinctions that are not preserved if the model outputs only plain Unicode text. A conventional OCR model might recognize styled text such as $\mathbf{CO_2}$ or footnote markers (e.g., $^{\mathbf{1}}$) simply as **CO2** or **1**, thus losing formatting information or potentially causing misinterpretation.

OCR is a crucial technology for digitizing documents and images, finding applications in search, augmented reality, and large language models, among others. The need for high accuracy, especially for a complex script like Hangeul, motivates new approaches that can effectively handle compositional character systems. By developing a model that recognizes text via subcharacter components, we aim not only to improve Hangeul recognition but also to lay groundwork for better handling of other languages. For instance, the same idea could help recognize Latin characters with diacritics or Chinese characters composed of multiple radicals. Of course, naively expanding the character set to cover all such cases would greatly increase the computational burden and reduce training efficiency. This underscores the need for a more efficient approach, which our model attempts to provide. In addition to this core model we have developed, we recognized that previous OCR models lacked the ability to recognize superscripts and subscripts, which are essential for accurately capturing a wide range of textual content. To address this limitation, we tested a model that incorporates a classifier to distinguish between normal, subscript, and superscript characters, enabling a more comprehensive OCR recognition.

2. Related Work

2.1. Scene Text Recognition Methods

Scene text recognition (STR) has evolved along two primary model paradigms. According to a recent survey by Wang *et al.* [5], early STR models were often based on Connectionist Temporal Classification (CTC) decoders, while later models adopted attention-based encoder-decoder architectures. CTC-based models are effective for recognizing text with regular patterns, as they align predictions with positions in the image. However, they tend to underperform on images with irregular text layouts or complex backgrounds. In contrast, attention-based models can dynamically focus on relevant parts of the image during decoding, yielding higher performance on complex scene text images.

Publicly available OCR engines (e.g., EasyOCR, PaddleOCR) are generally designed to recognize Hangeul as pre-composed syllable characters, and in practice they can only handle a subset of all Hangeul syllables (about 1,471 or 1,758 out of the 11,172 possible) due to model and data limitations. This means they cannot represent many rare syllables. Moreover, for the same reason, such models cannot capture style annotations: for example, they might recognize $\mathbf{CO_2}$ simply as **CO2** or interpret a footnote marker like $^{\mathbf{1}}$ as the digit **1**. Misreading of these styled characters is possible since style information is disregarded.

A historical example that illustrates the problem of class explosion is the early standard for Hangeul character encoding. The legacy character set KS X 1001 [9], commonly referred to as the Wansung code, defined only 2,350 frequently used Hangeul syllables out of over 11,000 possible combinations, primarily due to memory constraints at the time. In contrast, representing Hangeul by decomposing syllables into their constituent consonants and vowels (as adopted by modern IME keyboards) is significantly more memory-efficient and computationally practical. This comparison highlights the potential benefits of a compositional approach for OCR systems, especially when dealing with complex writing systems like Korean.

Over the years, OCR models have integrated convolutional networks (for visual feature extraction) with

sequence models (for text sequence decoding). A variety of architectures—such as CRNN, LSTM, attention mechanisms, Transformer, and Vision Transformer —have been applied to text recognition [1, 4], yielding continual performance improvements. Nonetheless, the majority of these studies report results on English-centric benchmarks. Research focusing on languages with non-Latin scripts (such as Korean) is still relatively scarce.

2.2. Hangeul OCR Research

The unique composition of Hangeul has spurred some research into specialized OCR methods. Unlike Latin characters where each symbol is atomic, a single Hangeul character is a combination of multiple jamo (letters), and an OCR system must analyze a clustered arrangement of these parts. A Hangeul syllable can be decomposed into an initial consonant, a medial vowel, and an optional final consonant. Many OCR models for Hangeul text, however, have typically not taken this structure into account, instead training on complete syllable characters as if they were indivisible units. [6–8]

In recent work, Kim *et al.* [10] and Lee *et al.* [11] developed OCR models recognizing handwritten and printed Korean text at the whole-syllable level without leveraging jamo decomposition. Later, Ko *et al.* [12] applied a CNN-based model to Korean scene text recognition, again treating Hangeul as pre-composed characters. There is an expectation that recognizing Hangeul text by decomposing it into jamo could improve performance, especially for less frequent syllables. If the model learns to recognize sub-character units, it could correctly predict rare or unseen syllables by recombining known components, even when trained on relatively limited real data. This suggests that a compositional approach might enable more robust OCR for Hangeul with fewer data, by handling characters that seldom appear in fully composed form.

Beyond OCR, the idea of leveraging jamo units has appeared in other language processing tasks. For example, some prior works explored jamo-level tokenization for Korean machine translation [13, 14] and language modeling [15]. These demonstrate the feasibility and utility of breaking Hangeul text into sub-character units in NLP contexts. However, there has been little work applying such ideas to OCR, which is the gap our research addresses.

2.3. Jamo Decomposition for Hangeul

A major challenge in extending OCR to Hangeul is the large number of possible syllables. Modern Hangeul comprises 11,172 valid syllable combinations

Figure 1. Illustration of Hangeul syllable decomposition and output class reduction. The example syllable 강 is split into its jamo components: ㄱ (initial), ㅏ (medial), ㅇ (final). Our model predicts each of these components with separate outputs, which are then combined to form the recognized character. This approach reduces the Hangeul output space from thousands of syllables (11,172 possible combinations) to a few dozen sub-character classes. The right side illustrates a similar composition approach for Latin script with diacritics (e.g., Á can be represented as A with an accent mark).

(or 11,265 Unicode code points when including historical or obsolete forms). A naïve approach that treats each syllable as an independent class results in an output softmax with over 11,000 classes, in addition to those for other scripts. This massive output space significantly increases model complexity and leads to severe data sparsity, as many syllables appear only rarely.

Although there has been prior research that has adopted jamo-level decomposition into deep learning methods in OCR detection for Hangeul, or Korean letters, there is a lack of studies that integrate this linguistic knowledge into recent state-of-the-art OCR models as far as we have observed. The most recent study that attempted to address the structural decomposition of Hangeul was proposed in 2019 by Chang et al. [21], who took into account the decompositional characteristics of Hangeul and Chinese characters into a CNN-TDNN model for OCR tasks. This study reports that utilizing the decompositional structure of Hangeul and Chinese characters significantly reduced the parameters used in the model. However, there is still a lack of research that focusing on Hangeul OCR detection using SOTA models.

To address this, we adopt a *jamo decomposition* strategy for Hangeul. Instead of predicting whole syllable characters, our model predicts the constituent components of each Hangeul syllable. Specifically, each Hangeul character is represented as a combination of three subunits: an initial consonant (choseong), a medial vowel (jungseong), and a final consonant (jongseong, which may be absent). We modify the decoder so that at each time step it outputs *three* tokens: one for each type of jamo. These three predicted tokens collectively form one syllable in the recognized text.

Using this approach, any Hangeul syllable can be generated from an appropriate triplet of jamo classes. Importantly, this decomposition also means that the model can generate syllables it never saw during training, as long as each jamo in the syllable was seen in some context. This offers a way to generalize to unseen character combinations, addressing the long-tail character distribution problem in Hangeul.

2.4. Radical-Based Recognition in Other Languages

Our approach is conceptually related to techniques in other languages that use sub-character components. In Chinese text recognition, for instance, researchers have studied recognizing radicals or character components to improve performance. Deng *et al.* [16] and Zhang *et al.* [17] proposed models that leverage radical information—parts of Chinese characters—to assist the recognition process. These studies suggest that a similar strategy (segmenting characters into smaller constituent parts like radicals, or in our case jamo or diacritics) can enhance an OCR model's capability. Inspired by this, our work can be seen as an extension of the compositional recognition concept to Hangeul and potentially to Latin scripts with diacritics.

2.5. Superscript and Subscript Detection in OCR

Previous research has aimed to detect superscripts and subscripts in OCR, particularly in domains such as scientific documentation, mathematics, and chemical notation.

Garain [22] presented one of the earliest studies in this field, focusing on the structural identification of superscripts and subscripts in mathematical documents. This rule-based method relied on analyzing spatial relationships between characters, such as vertical or horizontal alignments. More recent work includes that of Orji et al. [23] who developed a method to recognize subscripts in the context of image-to-LaTex conversion. The paper increased accuracy in OCR detection by augmenting of training data with LaTex syntax constraints, active learning strategies, and the employment of active learning feedback loops. Another related study by Hsu et al. [24] introduced a generative fusion decoding algorithm that integrates Large Language Models (LLMs) with traditional OCR pipelines. Although this work did not focus on superscript or subscript recognition, this model showed improved performance in processing structured and instruction-sensitive tokens using generative models.

In this research, we have attempted to incorporate Hangeul detection with image-to-LaTeX conversion to handle superscripts and subscripts. We have also developed a model to detect superscripts and subscripts in both Hangeul and English text, as well as Arabic numerals.

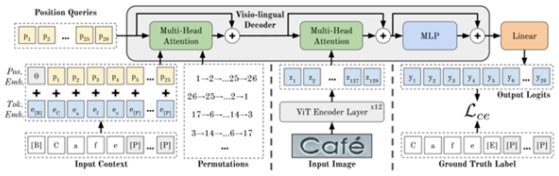


Figure 2. Overview of our PARSeq-based OCR model architecture. The model uses a Vision Transformer encoder for feature extraction and an autoregressive Transformer decoder for sequence prediction. We modify PARSeq's output layer by introducing separate or extended heads for Hangeul jamo components to support multiple scripts.

3. Method

3.1. Model Architecture

We base our text recognition model on the recently proposed PARSeq architecture [1], a permutation-based autoregressive sequence model designed for scene text recognition. PARSeq utilizes a Vision Transformer (ViT) backbone [18] to extract visual features from input text images, and a Transformer decoder that generates the output sequence token by token in an autoregressive manner. This design enables the model to leverage both visual and linguistic context during decoding.

Starting from a pretrained English PARSeq model, we fine-tune it for multilingual OCR by extending the output layer to support multiple scripts, including Hangeul. The core architecture—the ViT encoder and Transformer decoder—is preserved to retain PARSeq's strong performance and robustness. Only the output embedding and prediction layers are modified to accommodate the expanded character set required for our task. The overall model structure is illustrated in Figure 2.

3.2. Jamo Decomposition for Hangeul

Hangeul characters are composed of one *choseong* (initial consonant), followed by a *jungseong* (medial vowel), and may or may not include a *jongseong* (final consonant). According to the rules of Hangeul composition, there are 19, 21, and 27 Hangeul letters that can occupy the positions of choseong, jungseong, and jongseong, respectively. Among the 27 possible jongseongs, some are combinations of two Hangeul characters that function as a single final consonant in a syllable, such as ㄴㅎ, ㄴㅈ, etc. Each of these combinations is also counted as a distinct type of possible

jongseong. Figure 1 illustrates this idea. For example, the Hangeul syllable 강 consists of three jamo components: ㄱ (initial consonant), ㅏ (medial vowel), and ㅇ (final consonant). In theory, there are 11,172 possible Hangeul syllables, considering all combinations of choseong, jungseong, and jongseong. However, only a subset of these syllables actually appears in Korean texts, making it impractical to train a model with each syllable as a distinct label class.

To incorporate this structure into our model, we designed it to detect Korean letters in the following order: choseong, jungseong, and jongseong. If a jongseong is not found in a sequence, the model proceeds to detect the next choseong. For example, when detecting the syllable 강, the model predicts one label for each component — ㄱ, ㅏ, and ㅇ —in a fixed initial–medial–final sequence, repeating this pattern for subsequent syllables. Although Hangeul syllables generally follow consistent structural patterns, it is difficult to encode these patterns into a simple rule. For instance, the positional behavior of three different jungseongs — ㅣ, ㅡ, and ㅚ —varies depending on the syllable. We expected the model to form attention patterns corresponding to the type of Hangeul letter. For characters without a final consonant, the model omits the final position. By leveraging the fact that there are only 19 initial consonants, 21 medial vowels, and 27 final consonants, we reduce the output space from 11,172 full-syllable classes to a more compact set of sub-character classes. In future work, we plan to explore more advanced architectures that predict the initial, medial, and final components in parallel rather than sequentially.

3.3. Superscript and Subscript Recognition

To enable fine-grained character formatting recognition, we extend our model to jointly predict the Unicode character and its positional attribute—specifically, whether it is rendered as a normal character, superscript, or subscript. We achieve this by attaching an auxiliary classification head to the decoder, operating in parallel with the main character recognition head. This additional head performs position classification with three discrete labels: 0 for normal, 1 for superscript, and 2 for subscript.

Given the absence of public datasets with positional annotations at the character level, we constructed a custom synthetic dataset by modifying the SynthTiger [25] engine. The extended generator supports rendering of characters in superscript and subscript positions across a wide variety of fonts, character compositions, and background conditions. Representative examples are shown in Fig. 3, illustrating both the diversity and positional richness of the generated data. For each

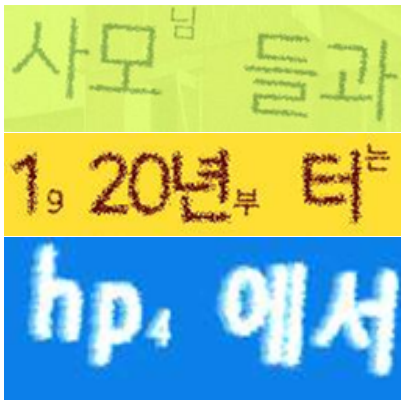


Figure 3. Examples of synthesized text images with positional annotations. Each sample contains characters rendered in normal, superscript, or subscript positions. The first image shows a superscripted 님, and is represented by the position label sequence `00100` (0: normal, 1: superscript, 2: subscript). The second includes subscripts in **9** and 부, a superscript in 는, and is labeled as `02000201`. The third contains a subscripted **4** and is labeled as `00200`.

image, we provide a corresponding position label sequence (e.g., `00100`, `02000201`, `00200`) that aligns with the character-level annotations. To capture these positional variations, we define the end-of-sequence (EOS) token as "3", leading to the formation of a head with a total of four classes.

3.4. Multi-Task Learning for Character and Position Prediction

To train the model to simultaneously recognize the character identity and its positional attribute, we adopt a multi-task learning framework. Specifically, two separate output heads are used: one for character classification and the other for position classification. Each head outputs a probability distribution over their respective label spaces, and the model is supervised using two cross-entropy loss functions.

Let $\mathcal{L}_{\text{char}}$ denote the cross-entropy loss for character prediction and $\mathcal{L}_{\text{pos}}$ for position classification (normal, superscript, subscript). The total loss is computed as a weighted sum of the two:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{char}} + \lambda \cdot \mathcal{L}_{\text{pos}}, \tag{1}$$

where λ is a balancing hyperparameter that controls the contribution of the position loss. In our experiments, λ is selected empirically based on validation performance, with a lower value assigned to the position loss to reflect the higher priority of accurate character identity recognition. This formulation encourages the model to learn rich representations that are primarily optimized for character classification, while still

capturing meaningful positional semantics. To ensure that the position loss retains its significance despite being much smaller in magnitude, we experimented with different values of λ and ultimately selected a value of 20, which was found to strike the right balance between character accuracy and positional accuracy.

3.5. Dynamic Loss Weighting for Robust Position Classification

To further stabilize training and address class imbalance, particularly the relative sparsity of superscript and subscript labels, we adopt a dynamically scaled loss weight. This weight modulates the contribution of the position classification loss $\mathcal{L}_{\text{pos}}$ relative to the primary character loss $\mathcal{L}_{\text{char}}$, and is computed at each epoch using a warm-up strategy followed by ratio-based scaling.

During the initial phase of training, the weight is set to zero for early epochs and then gradually increases toward a maximum value $\lambda_{\max}$ using a quadratic schedule. This allows the model to focus on stable character recognition before incorporating the auxiliary supervision for positional classification.

To further improve robustness to class imbalance, we replace the standard cross-entropy loss for $\mathcal{L}_{\text{pos}}$ with a *Focal Loss* [26]. This formulation reduces the impact of well-classified (i.e., high-confidence) samples and emphasizes learning from harder examples such as rare superscript or subscript characters. The focal loss is defined as:

$$\mathcal{L}_{\text{pos}} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t), \tag{2}$$

where p_t is the predicted probability for the true class, γ is the focusing parameter (set to 1.5 in our experiments), and α_t is an optional class-balancing weight. The values of α_t are computed by inversely weighting the class distribution of the training data, which helps balance the loss function under highly imbalanced conditions. Specifically, α_t values are calculated based on the inverse frequency of each class in the training set, ensuring that underrepresented classes receive higher weights. In our experiments, we set $\alpha_t = [0.035, 0.4825, 0.4825, 0.01]$, where the values correspond to the weights for each class based on their relative frequencies: the lower values correspond to more frequent classes, and the higher values are assigned to less frequent classes, helping to mitigate class imbalance. We ignore padded positions via a mask and apply the loss only to valid targets.

4. Experiments

4.1. Dataset Construction

Our training data consists of both real and synthetic sources. For real-world scene text, we used two public multilingual datasets: the AIHub Multilingual OCR dataset [19] and the ICDAR 2019 MLT (Multi-Lingual Text) dataset [20]. These datasets contain a broad range of scripts, including Hangeul, English, and other Latin-based alphabets—making them well-suited for our multilingual OCR setting. We used their official training splits, while evaluation was performed on a curated held-out test set consisting of half of the AI-Hub validation set and the full MLT-2019 test split. All data underwent consistent preprocessing, including resizing, normalization, and sequence tokenization.

For position-aware supervision, we incorporated the synthetic dataset described above. A total of 1 million images were generated and split into training, validation, and test subsets with an 8:1:1 ratio. This synthetic dataset supplements the real-world data by explicitly modeling superscript and subscript phenomena, which are rare or unannotated in natural scene datasets.

4.2. Data Preparation

Given the annotated scene text images in AIHub and MLT, we preprocessed the data by cropping each individual text instance to create focused word images. Each text annotation in the source datasets is provided as a polygon (often a quadrilateral) that tightly bounds a word or text line. We extracted an axis-aligned rectangular patch that tightly encloses the polygon. This approach simplifies the preprocessing pipeline while still preserving the majority of the visual information needed for recognition.

Cropping was automated using the annotated polygon coordinates. For each polygon, we computed the minimum enclosing bounding box and extracted that region from the image. The corresponding text label was also stored for each cropped patch. As a result, our training set consists of tens of thousands of word-level image patches paired with ground-truth text labels. We applied the same procedure to prepare the validation and test sets from the reserved data.

4.3. Character Set and Labeling

We define a unified character set that covers all symbols appearing in our training data. For the baseline model (which predicts complete characters), the vocabulary includes: (1) English uppercase and lowercase letters (52 classes), (2) digits 0–9 (10 classes), (3) common punctuation marks, and (4) the full set of 11,265

modern Hangeul syllables. This results in a total of over 11,200 classes, with Hangeul syllables comprising the majority.

In contrast, the jamo-based model recognizes 51 unique Hangeul jamo symbols, excluding duplicates across the initial, medial, and final positions. These symbols represent the sub-character components—initial consonants, medial vowels, and final consonants—that are composed to form complete Hangeul syllables. English letters and digits are handled identically to the baseline model, each assigned to its own class and emitted in a consistent position within the jamo decoding sequence.

4.4. Experimental Setup

We trained both the full-character baseline and our jamo-based model under the same settings for a fair comparison. Training was performed using the Adam optimizer. We started with a learning rate of 7×10^{-4} and applied learning rate decay on plateau of validation loss. Early stopping was used based on validation accuracy to prevent overfitting. Both models were trained for up to 200 epochs. Due to storage constraints on the system, the current experiments were conducted using a subset of the data, specifically 918,464 image-text pairs.

For input preprocessing, all word images were resized to 128 $\times$ 32 pixels. We applied basic data augmentation techniques, including rotation, shearing, translation, and noise injection. These settings follow the same preprocessing configuration used in the original PARSeq implementation.

5. Results

5.1. Jamo Decomposition Results

To establish a rigorous baseline, we surveyed prior work on Hangeul OCR. The most recent effort that explicitly targets Hangeul characters as classification labels is Bakrie et al. [27] they created a small custom dataset and trained an SVM on zoning and GLCM features, but did not employ a modern sequence recogniser. The latest study to explore jamo decomposition is Chang et al. [21], which used a CNN–TDNN pipeline on printed-line images—an approach that predates today's Transformer-based scene-text models and lacks publicly available code. Because no up-to-date recogniser jointly handles Korean text and sub-character decoding, we fine-tune the strongest open-source STR backbone, PARSeq, on Korean data with its original full-syllable vocabulary and use this syllable-level model as the baseline against which we compare our proposed jamo decomposed variant.

Model	Classes	Word Accuracy
Full Syllable Model (Unicode 11,265)	11,265	97.31%
Jamo Model (Sub-units 144)	144	97.84%

Table 1. Comparison of word-level accuracy on the Hangeul test set (178,966 words) for the baseline full-syllable model vs. our jamo-decomposed model. The jamo-based model achieves higher accuracy despite operating with a vastly smaller output class space.

We evaluated the models on the Hangeul portion of the test set using word-level accuracy. A prediction was considered correct only if the entire predicted word exactly matched the ground truth. Both models were evaluated on the same set of word images, consisting of a total of 178,966 words. We conducted experiments with two different character encoding schemes: (1) the **Full Syllable** model, which uses the Unicode character set comprising 11,265 precomposed Hangeul syllables, and (2) the **Jamo Decomposed** model, which predicts the initial, medial, and final jamo components separately, with a total of 144 possible sub-character classes for Hangeul.

Table 1 summarizes the results. The jamo-based model achieved a word accuracy of 97.84%, which is 0.53% higher than the baseline model's 97.31%. This indicates that the compositional approach yields a measurable improvement in recognition performance. We also observed that during training, the full-syllable model showed signs of overfitting (its training accuracy continued to improve while validation accuracy plateaued and eventually declined), whereas the jamo-based model converged more stably. The latter exhibited lower validation loss and higher validation accuracy, as shown in Figure 4, suggesting better generalization.

We analyzed the frequency distribution of Hangeul syllables in our dataset to better understand the challenges of Hangeul OCR. The results reveal a pronounced long-tail pattern: the vast majority of possible Hangeul syllables are either extremely rare or entirely absent in the training data. Specifically, out of 11,265 possible syllables, 9,302 (82.6%) do not appear at all in our training set, and an additional 513 syllable types appear fewer than 10 times. Even in the test set —which follows a similar distribution —9,815 syllables are unseen during training, making them effectively out-of-vocabulary for a model that memorizes only full syllables. Table 2 provides a breakdown of syllable frequency categories in the training and test sets. These findings highlight that Hangeul OCR "in the wild"is inherently a long-tail (and often zero-shot) recognition problem.

Frequency Category	Training Set	Test Set
Never appeared	9,302	9,815
1–10 occurrences	513	524
11–100 occurrences	446	482
Over 100 occurrences	1,004	444
Total distinct Hangeul	11,265	11,265

Table 2. Frequency of Hangeul syllable classes in the training and test sets. A large majority of possible syllables are not present or are very rare in training data, underscoring the importance of handling unseen or rare characters.

Input Image	Full Syllable Model	Jamo Model
	접껍대다	쩝쩝대다
	잖다	짢다
	앉다	앉다
	앉다	핞다

Table 3. Comparison of OCR outputs on words containing rare or unseen Hangeul syllables. Each row shows (left) the input image, (middle) the output of the full-syllable baseline model, and (right) the output of our jamo-based model. The baseline model tends to misread unseen or infrequent syllables (e.g., predicting **잖다** instead of **짢다**, or **앉다** instead of **핞다**), while the jamo-based model accurately reconstructs the target word by composing valid jamo sequences.

Table 3 illustrates several such cases. In the first example, the input word is **쩝쩝대다**, which contains the rare syllable **쩝**. The baseline model misreads it as **접껍대다**, substituting more frequent syllables, whereas our jamo-based model correctly outputs **쩝쩝대다**. In the second example, the input word is **짢다**, which includes the unseen syllable **짢**. The baseline predicts **잖다**, replacing it with a more familiar syllable, while the jamo-based model accurately reads **짢다**. In the third example, the input **핞다** contains the unseen syllable **핞**. The baseline outputs **앉다**—a visually similar but incorrect substitution—whereas our model again produces the correct **핞다**. These results demonstrate that the jamo-level model generalizes significantly better to unseen or rare characters by virtue of its compositional decoding. In contrast, the baseline model, lacking representations for novel syllables, tends to fall back on the closest known alternatives.

We also conducted a qualitative experiment to verify how each model handles completely unseen character combinations. We selected several Korean words containing syllables that never appeared in the training set (for example, syllables like **짢** or **핞**). The baseline model (trained on complete Unicode syllables) failed to recognize these unseen characters correctly, often substituting them with visually or phonetically similar known characters. In contrast, our jamo-based model correctly recognized these syllables by composing them from the learned sub-character features.

Module	Full Syllable Model	Jamo Model
Encoder (ViT backbone)	5.4M	5.4M
Decoder	594K	594K
Output Head	2.2M	28.0K
Token Embedding	2.2M	28.2K
Total Parameters	**10.3M**	**6.0M**

Table 4. Model size comparison. The jamo-based model uses far fewer output classes for Hangeul, resulting in much smaller output-head and embedding layers. Overall, the jamo model has about 40% fewer parameters than the full-syllable model (6.0M vs 10.3M total).

Beyond accuracy, our approach offers a significant advantage in model size. We compared the number of parameters required by each model, focusing especially on the output-related components. The jamo-based model substantially reduces the parameter count by eliminating the need for large embedding and classification layers to handle thousands of full-character classes. Specifically, our jamo model uses approximately 4.3 million fewer parameters than the baseline model —about a 40% reduction in total model size. In the baseline, both the token embedding and output head modules comprise roughly 2.2 million parameters each, whereas in the jamo model, these components are only 28,000 parameters each. Table 4 provides a breakdown of parameter counts across key components of both models.

5.2. Superscript and Subscript Recognition Results

To assess the effectiveness of our model in recognizing character position (normal, superscript, subscript), we evaluate its performance on the synthetic test set described in Section 3. This test set includes character-level annotations for both Unicode identity and positional labels, enabling fine-grained analysis of both textual and positional prediction capabilities.

Table 5 summarizes the model's performance on both word and positional accuracy. Although the overall accuracy is relatively modest, the results highlight the model's ability to learn layout-sensitive features. Despite the class imbalance, where normal

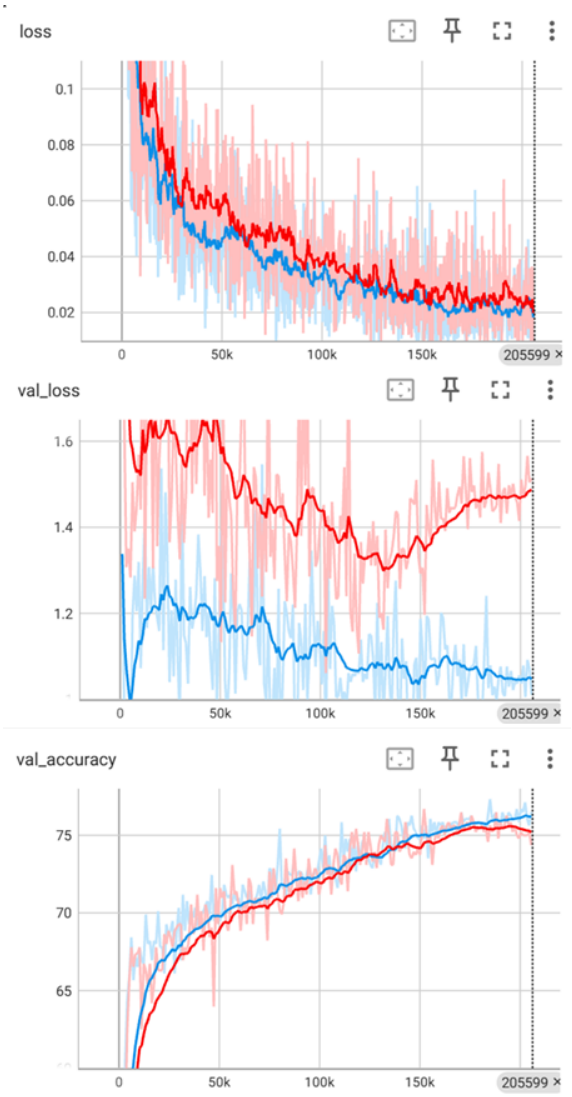


Figure 4. Training and validation performance of the baseline full-syllable model (red) versus the jamo-based model (blue). **Top:** Training loss curves. **Middle:** Validation loss curves. **Bottom:** Validation accuracy curves. While the full-syllable model achieves lower training loss, it suffers from higher validation loss, indicating overfitting. In contrast, the jamo-based model maintains lower validation loss and achieves higher final accuracy.

Head Configurations	Word Accuracy	Pos Accuracy
Simple Linear	81.47 %	92.79 %
Linear+LayerNorm+ReLU +Dropout+Linear	82.94 %	95.11 %

Table 5. Comparison of Word and Positional Accuracy for Superscript and Subscript Recognition Using Different Model Configurations.

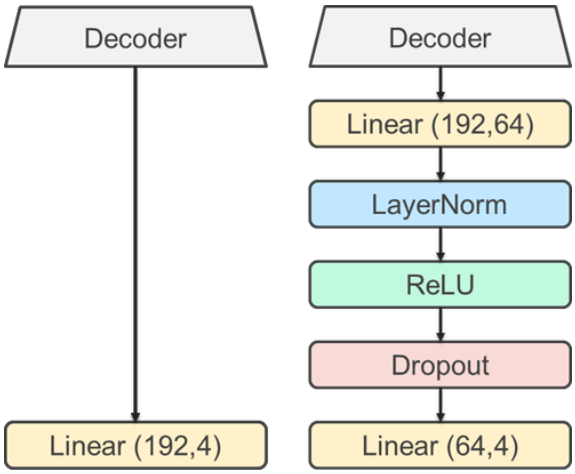


Figure 5. Comparison of head configurations. The figure illustrates two different head architectures: the "Simple Linear" head and the "Linear+LayerNorm+ReLU+Dropout+Linear" head. The addition of LayerNorm and ReLU layers in the second configuration improves both word and positional accuracy, enhancing the model's ability to capture positional layout information.

characters significantly outnumber positional variants, the model demonstrates competitive accuracy in both tasks: character recognition and position classification.

Comparing the two head configurations, the "Simple Linear" head achieves a word accuracy of 81.47% and positional accuracy of 92.79%. On the other hand, the "Linear+LayerNorm+ReLU+Dropout+Linear" head shows a clear improvement, with a word accuracy of 82.94% and a positional accuracy of 95.11%. This indicates that incorporating additional layers such as LayerNorm and ReLU in the model enhances its ability to capture the positional layout of characters more effectively.

In this experiment, we investigated the feasibility of recognizing superscript and subscript characters using only synthetically generated data, without any real annotated examples. As expected, absolute performance remains modest compared to standard character recognition. Nevertheless, the model successfully learned to distinguish positional variants such as superscripts and subscripts under highly imbalanced conditions. Qualitative results further support this, showing that the model can visually identify and separate these characters in layout-sensitive contexts.

Despite limitations in available resources (training time and GPU capacity), our findings suggest promising potential for further improvement. We anticipate that incorporating real-world annotated data and

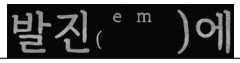
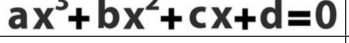
Input Image	Text Prediction	Position Prediction	Final LaTeX-Formatted Output
	DASH 가	00202	DA_{S}H_{가}
	발진 (em) 에	0021100	발진 _{(}x^{em}) 에
	Point	02000	P_{o}int
	ax3+bx+cx+d=0	0010000000000	ax^{3}+bx+cx+d=0
	C2H5OH	000000	C2H5OH

Table 6. Examples of superscript and subscript recognition outputs. The Position Tags column shows the model's predicted tag sequence for each character (0 = normal, 1 = superscript, 2 = subscript). The Output as LaTeX column shows the corresponding text with formatting, using LaTeX notation for subscripts/superscripts (e.g., "_{}" denotes subscript content).

optimizing the loss function and head architecture could significantly enhance the style recognition performance. These results validate the viability of explicit positional modeling and highlight the value of continued exploration in this direction. For illustration, Table 6 provides examples of the model's output on the test set, including the predicted position tags and the corresponding LaTeX expressions for those outputs.

However, as shown in the two samples below, there are still cases of incomplete recognition where the model fails to capture superscripts or subscripts. These errors likely stem from insufficient learning or limitations of the synthetic data, suggesting that more training or refined data is needed for better accuracy. For example, in one case, the model fails to recognize the superscript in a word, while in another, it misses a subscript notation in an equation. These specific errors highlight areas for further improvement in the model learning process and data representation.

6. Conclusion

In this work, we presented a novel OCR framework for Korean Hangeul that leverages a jamo-level decomposition strategy. By exploiting Hangeul's inherent compositional structure, our method drastically reduces the output character space compared to conventional approaches that treat each syllable as an indivisible class. This design enabled the model to generalize better and achieved higher word-level accuracy than a full-syllable baseline. Notably, the jamo-based approach also proved effective at recognizing rare and previously unseen characters—an important capability for real-world OCR applications.

Additionally, we extended our model to detect superscript and subscript characters, addressing the under-explored challenge of style-aware OCR. By training on a synthetically generated positional dataset and employing techniques like dynamic loss weighting with focal loss, the model learned to capture layout-sensitive attributes of text. However, the style recognition component remains imperfect: in some cases the model failed to detect certain superscripts or subscripts, likely due to the limited variability of the synthetic training data. This highlights a current limitation of our study, indicating that the model's style awareness is not yet fully robust and could benefit from further data and refinement.

Our findings underscore the value of structurally aligned modeling in OCR, particularly for writing systems with compositional properties like Hangeul. Moreover, explicitly modeling text formatting through style tags can enhance the expressive power of OCR systems beyond plain Unicode output, preserving information (such as superscript/subscript notation) that would otherwise be lost in a standard text recognition pipeline.

For future work, we plan to integrate real annotated data for positional attributes (instead of relying solely on synthetic data), utilize stronger vision backbones to improve overall recognition accuracy, and explore parallelizing the decoding of jamo components to increase efficiency. Ultimately, our goal is to develop a generalizable, style-aware OCR system capable of handling complex, multilingual text in the wild, thereby broadening the scope and applicability of OCR technology.

References

[1] Bautista, D., and Atienza, R. 2022. Scene text recognition with permuted autoregressive sequence models.

In ECCV, pages 178–196, 2022.

[2] JaidedAI. 2020. EasyOCR. GitHub: https://github.com/JaidedAI/EasyOCR.

[3] PaddlePaddle. 2021. PaddleOCR. GitHub: https://github.com/PaddlePaddle/PaddleOCR.

[4] Baek, J., *et al.* 2019. What is wrong with scene text recognition model comparisons? Dataset and model analysis. In ICCV, pages 4715–4723, 2019.

[5] Wang, X.-F., *et al.* 2023. A survey of text detection and recognition algorithms based on deep learning technology. Neurocomputing, 556:126702, 2023.

[6] Kang, Ga-Hyeon, et al. *A study on improvement of Korean OCR accuracy using deep learning.* Proceedings of the Korean Institute of Information and Communication Sciences Conference. The Korea Institute of Information and Communication Engineering, 2018.

[7] Park, Sun-Woo. *A Study on the OCR of Korean Sentence Using DeepLearning.* Annual Conference on Human and Language Technology. Human and Language Technology, 2019.

[8] Park, Youngki, and Youhyun Shin. *Gradual OCR: An effective OCR approach based on gradual detection of texts.* Mathematics, vol. 11, no. 22, 2023, p. 4585.

[9] Korean Agency for Technology and Standards (KATS), *KS X 1001: Code for Information Interchange (Hangeul and Hanja)*, National Standard of the Republic of Korea, 1997. Formerly KS C 5601.
https://standard.go.kr/KSCI/standardIntro/getStandardSearchView.do?ksNo=KSX1001

[10] Kim, M. S., *et al.* 2004. Segmentation of handwritten characters for digitizing Korean historical documents. In Proc. Int. Workshop on Document Analysis Systems (DAS), 2004.

[11] Lee, J.-S., Kwon, O.-J., and Bang, S.-Y. 1999. Highly accurate recognition of printed Korean characters through an improved two-stage classification method. Pattern Recognition, 32(12):1935–1945, 1999.

[12] Ko, D.-G., *et al.* 2017. Convolutional neural networks for character-level classification. IEIE Trans. Smart Processing and Computing, 6(1):53–59, 2017.

[13] Lee, J., *et al.* 2025. Jamo-level subword tokenization in low-resource Korean machine translation. In Proc. 8th Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT), 2025.

[14] Park, J., and Zhao, H. 2020. Korean neural machine translation using hierarchical word structure. In Proc. International Conference on Asian Language Processing (IALP), 2020.

[15] Kim, S., *et al.* 2024. KOMBO: Korean character representations based on the combination rules of subcharacters. In Findings of ACL, 2024.

[16] Deng, X., *et al.* 2023. RRecT: Chinese text recognition with radical-enhanced recognition transformer. In Proc. International Conference on Artificial Neural Networks (ICANN), 2023.

[17] Zhang, J., Du, J., and Dai, L. 2020. Radical analysis network for learning hierarchies of Chinese characters. Pattern Recognition, 103:107305, 2020.

[18] Dosovitskiy, A., *et al.* 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929, 2020.

[19] National Information Society Agency (NIA), "Multilingual OCR Dataset," AIHub, 2023.
Available: https://www.aihub.or.kr/aihubdata/data/view.do?dataSetSn=71730

[20] N. Nayef, Y. Patel, M. Busta, P. N. Chowdhury, D. Karatzas, W. Khlif, J. Matas, U. Pal, J.-C. Burie, C.-L. Liu, *et al.*,
"ICDAR2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition—RRC-MLT-2019,"
in *Proc. 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1582–1587, IEEE, 2019.

[21] Chun-Chieh Chang, Ashish Arora, Leibny Paola Garcia Perera, David Etter, Daniel Povey, and Sanjeev Khudanpur. Optical character recognition with Chinese and Korean character decomposition. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 5, pages 134–139. IEEE, 2019.

[22] U. Garain. Identifying subscripts and superscripts in mathematical documents. *International Journal of Document Analysis and Recognition*, 2005. Structural analysis to differentiate baseline, sub- and superscript.

[23] Everistus Z. Orji, Ali H. Haydar, Ibrahim Erşan, and Othmar Mwambe. Advancing OCR accuracy in image-to-LaTeX conversion—A critical and creative exploration. *Applied Sciences*, 2023. 10.3390/app132212503. Focus on recognition of subscripts in chemical formulas.

[24] Chan-Jan Hsu, Yi-Chang Chen, Feng-Ting Liao, Pei-Chen Ho, Yu-Hsiang Wang, Po-Chun Hsu, and Da-shan Shiu. Let's fuse step by step: A generative fusion decoding algorithm with LLMs for robust and instruction-aware ASR and OCR. *arXiv preprint arXiv:2405.14259*, 2024. Cross-modal fusion improves OCR of structured tokens.

[25] Yim, M., Kim, Y., Cho, H.-C., and Park, S. SynthTIGER: Synthetic Text Image GEneratoR Towards Better Text Recognition Models. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 109–124, 2021. Springer.

[26] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. 2017. Focal Loss for Dense Object Detection. In ICCV, pages 2980–2988.

[27] Dinda Nadila, *Pengenalan Karakter Huruf Korea (Hangul) Menggunakan Algoritma Support Vector Machine (SVM)*, Ph.D. thesis, Universitas Bakrie, 2023.