Adaptation of CRNNs for Low-Resource Armenian Handwriting Recognition via Fine-Tuning and Static Detection

DS299 Capstone, Spring 2025

Ani Aloyan
BS in Data Science
American University of Armenia

Supervisor Elen Vardanyan
American University of Armenia

***Abstract*—Handwritten Text Recognition (HTR) is the task of automatically transcribing handwritten content into digital text. While substantial progress has been made in resource-rich languages such as English, applying HTR to under-represented scripts like Armenian remains significantly less explored. The development of effective recognition systems in such contexts is complicated by the limited availability of annotated data, natural variation in handwriting styles, and the absence of dedicated tools or pipelines adapted to the specific characteristics of the script. This work presents the development of an end-to-end word-level HTR system for Armenian, combining a detection module with dual recognition models. A CRAFT-based detector is used to identify word regions, which are then passed to one of two recognition pipelines: either a CRNN adapted from the SimpleHTR framework or a TPS-ResNet-BiLSTM-CTC architecture adapted from Clova AI's Deep Text Recognition Benchmark [5]. Both models are fine-tuned using a combination of manually collected handwritten text and synthetically composed samples. The approach was designed to reduce reliance on large-scale annotated datasets while maintaining a realistic script representation.**

***Index Terms*—Handwritten Text Recognition, HTR, OCR, Armenian script, synthetic data generation, data augmentation, CRNN, transfer learning**

I. Introduction

Handwritten Text Recognition (HTR) remains an ongoing area of research focused on the automatic transcription of handwritten content into digital form. While substantial advancements have been made in languages with abundant training data and well-established benchmarks, applying HTR to low-resource scripts remains relatively unexplored. The effectiveness of current systems often relies heavily on the availability of large, diverse, and consistently labeled datasets, which are typically absent for underrepresented languages.

Armenian, despite being a historically rich and actively used script, falls into this under-resourced category. It is structurally distinct from Latin-based scripts and lacks publicly available datasets for training modern recognition models. As a result, developing HTR systems for Armenian requires addressing both the scarcity of data and the absence of tools that take into account the unique visual and structural properties of the script.

To approach this task, the project first focuses on constructing a high-quality Armenian word-level dataset through manual collection, careful annotation, and synthetic generation. The synthetic samples are composed by stitching isolated character images based on visual contour extraction and typographic structure, aiming to reflect natural handwriting flow. Augmentation techniques further enhance variability across the dataset to simulate realistic writing conditions.

With the dataset prepared, the next step is to investigate how different recognition architectures perform when adapted to a low-resource script. For word-level segmentation, this work integrates the CRAFT text detector proposed by Baek et al. [4], also developed at Clova AI, initially designed for character-level detection in scene text. CRAFT produces affine-adjusted bounding boxes by predicting character centers and affinities, making it suitable for extracting coherent word regions even from complex layouts. While the model was trained on printed text in natural scenes, its ability to detect irregularly shaped text makes it a viable solution for locating handwritten words in raw input images. In this implementation, the pretrained model is used without modification, and minimal postprocessing is applied to merge nearby boxes based on geometric proximity.

Once word regions are isolated, two distinct recognition models are evaluated. The first is a Convolutional Recurrent Neural Network (CRNN) adapted from the SimpleHTR framework by Harald Scheidl, originally proposed for English handwriting recognition. The second is a modular OCR architecture introduced by Baek et al. in Clova AI's Deep Text Recognition Benchmark, which incorporates Thin-Plate Spline (TPS) rectification, ResNet-based feature extraction, BiLSTM sequence modeling, and CTC decoding. While the latter was designed for scene text

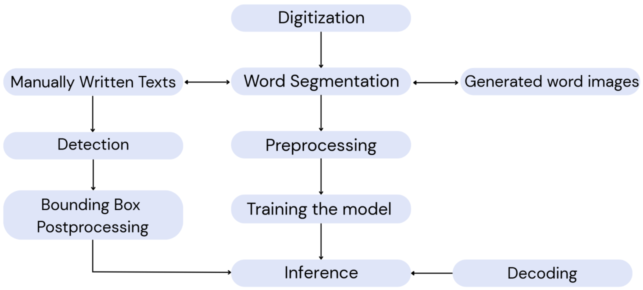

Figure 1. Project Pipeline

recognition and pretrained on synthetic printed datasets such as MJSynth (MJ) [6] and SynthText (ST) [7], its flexible design allows it to generalize across domains. Both models are fine-tuned on the same Armenian dataset, enabling a direct comparison of their performance on handwriting recognition tasks in a low-resource setting.

II. Literature Review

HTR has evolved significantly over the past decades, transitioning from statistical models to deep learning architectures [20]. Early systems relied on segmentation-based pipelines where each character was extracted and classified independently. While conceptually simple, this method performs poorly on cursive or naturally variable scripts, where characters often lack clear separation. Segmentation in such settings tends to introduce fragmentation and ambiguity.

This motivated the shift toward alignment-free approaches, notably CRNNs trained with CTC loss [20]. These models operate on word or line-level inputs and combine convolutional layers for spatial encoding with recurrent layers for sequential modeling. Puigcerver's CRNN [8] introduced a deep architecture with stacked convolutional and BLSTM layers, achieving high accuracy but at the cost of increased parameter count. Bluche proposed a more lightweight variant [9] that replaces some convolutional blocks with gated convolutions to reduce complexity while maintaining performance.

Further work extended the use of gating mechanisms. Gated CNNs and GRUs regulate information flow through the network, helping capture long-term dependencies, adapt to contextual variations, and reduce error propagation [21]. Gated-CNN-BLSTM and Gated-CNN-BGRU architectures have shown strong performance in noisy and low-resource conditions, offering a balance between accuracy and compactness, becoming ideal for scripts with limited training data.

In parallel, Optical Character Recognition (OCR) tools have matured significantly, yet their applicability to handwriting remains limited. Tools like Tesseract [14], EasyOCR [13], and Google Cloud Vision API [15] can recognize printed Armenian text but generally fail on handwritten inputs due to their assumptions about spacing and stroke consistency. Portmind's Armenian OCR [16], while script-specific, is also limited to printed material. In principle, OCR models can be fine-tuned on handwriting, but the underlying architecture often lacks the flexibility to deal with handwritten variability.

Transformer-based models such as TrOCR [11] offer a powerful alternative, processing input at a broader scale using self-attention mechanisms. These models excel at learning global context and can handle paragraph-level or even full-page recognition tasks. However, they typically require large-scale training data and computing resources, which limits their accessibility for niche or low-resource languages.

Several advanced HTR systems, such as PyLaia [10] and Flor's [19] gated models, offer competitive results. However, they are often difficult to reproduce due to missing pretrained weights, or they depend heavily on large annotated datasets. In practice, the lack of accessible, well-documented implementations makes their adaptation non-trivial.

Given these challenges, this work considers the SimpleHTR framework by Scheidl [3]. Though modest in size, it captures the core components of modern HTR pipelines: CNN-based feature extraction, gated recurrent modeling, and CTC decoding. Its public availability, pretrained weights on IAM, and word-level input format make it a practical and flexible choice for adaptation to Armenian handwriting.

III. Data

A. Reference Datasets

This project draws on several benchmark datasets as references for model initialization and data design. The IAM dataset [1], a widely used English handwritten text corpus containing segmented word and line images, served as both a benchmark and a structural reference throughout the data preparation process. The formatting and annotation pipeline for the Armenian dataset was partially inspired by IAM to ensure compatibility with existing HTR architectures such as SimpleHTR.

Additionally, two large-scale synthetic datasets, MJSynth (MJ) and SynthText (ST), were used indirectly through the pretraining of the Clova AI Deep Text Recognition Benchmark model. MJ and ST contain millions of rendered word images with varying fonts, colors, and background conditions, and are widely used to pretrain OCR systems that operate in visually diverse environments. While these datasets do not include handwritten text or Armenian script, they contribute useful low-level visual representations that are later adapted to Armenian handwriting through fine-tuning.

B. Data Collection and Annotation

The first stage of this work involved the manual collection and labeling of Armenian handwritten *word-level*

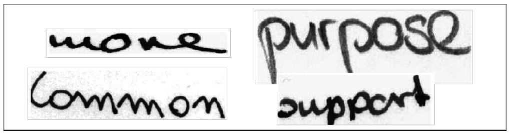

Figure 2. Example images from the IAM dataset.

Figure 3. Example images from the MJSynth and SynthText datasets.

data. Participants were asked to write a list of Armenian words by hand, which were then scanned. These scans were segmented into individual word images and labeled accordingly. This process resulted in a dataset of over 10,000 word images.

To improve the size and robustness of this real handwritten data, data augmentation techniques were applied. These transformations helped introduce variability without distorting the semantic structure of the images. After augmentation, the manually written data portion of the dataset was increased to approximately 30,000 images.

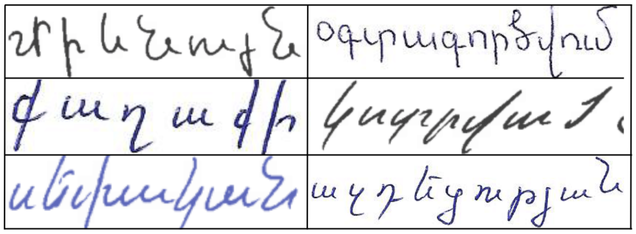

Figure 4. Examples of real handwritten Armenian word images.

C. Data Augmentation

To enhance variability in the manually collected handwritten Armenian word dataset, a custom data augmentation pipeline was implemented. This augmentation routine applied a sequence of randomized transformations with controlled probabilities to simulate real-world variations in handwriting style, ink flow, and scanning conditions.

Each word image was passed through a transformation loop that guaranteed at least one form of augmentation was applied per sample. The following operations were used:

- **Stroke thickness adjustment:** With a probability of 66%, stroke morphology was altered. One-third of cases underwent thinning through morphological erosion, while another third were thickened using dilation. The image was first inverted (turning black strokes into white), the operation was applied, and then it was inverted back. This mimicked variations in writing pressure or pen nib width.
- **Gaussian noise injection:** With a 60% chance, zero-mean Gaussian noise was added to the image with a randomly sampled standard deviation between 1 and 3. This simulated scanning artifacts, ink spread, and general noise introduced by hardware imperfections.
- **Affine scaling:** In 70% of cases, the image was either horizontally or vertically squeezed or stretched. Horizontal scaling occurred with 80% probability and vertical with 50%. The scaling factors ranged from 0.70 to 1.20 horizontally, and 0.75 to 1.25 vertically. The transformation matrix was centered on the image midpoint, preserving structural balance. Border pixels introduced by transformation were filled with a white background to maintain consistency.

The augmentation pipeline was wrapped in a retry loop, such that images were only accepted after at least one transformation was successfully applied in order to ensure meaningful variability across the dataset.

The output images were clipped and normalized to ensure all pixel values fell within the 8-bit grayscale range. Through this procedure, the dataset of real handwritten word images was expanded from approximately 10,000 to 20,000 samples, providing a broader spectrum of handwriting styles and deformation patterns for training.

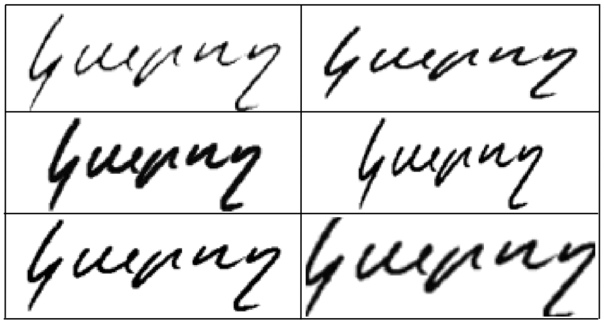

Figure 5. Examples of augmented Armenian handwritten word images.

D. Synthetic Dataset Generation

To further increase dataset diversity, a synthetic dataset was created using the Mashtots dataset [2], which contains approximately 70,000 isolated images of Armenian letters in both uppercase and lowercase forms. The dataset is organized into subfolders for each letter, and each image is a 64x64 pixel grayscale depiction of a single character.

The synthetic word list was generated by scraping and cleaning an Armenian Wikipedia word dump. A filtering function ensured that only words composed entirely of Armenian letters (including normalization for composite letters like ու and ՈՒ) were kept.

Direct placement of the character images would have led to unrealistic compositions due to fixed-size bounding boxes and inconsistent whitespace. Instead, each character image was processed by extracting contours and cropping the active region tightly. These were then resized based on their typographic structure to simulate natural alignment.

Characters were grouped into four visual categories:

- **Tall ascenders**: characters that extend high above the baseline.
- **Short descenders**: characters that extend below the baseline.
- **Short letters**: visually compact characters.
- **Long on both sides**: characters with both ascender and descender components.

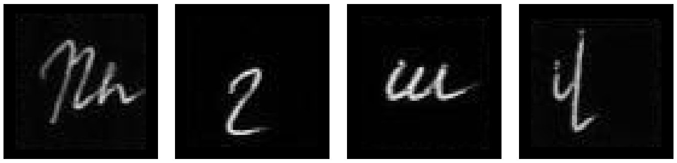

Figure 6. Examples from each letter category respectively.

Each category was assigned a height range, allowing random variation in letter sizes while maintaining structural consistency. Characters were placed onto a blank canvas, horizontally aligned with a small fixed padding and vertically positioned based on their category-relative baseline alignment. The final canvas was trimmed to the bounding box of the composed word.

During deskewing, skew angles were estimated using PCA. However, if the detected angle exceeded 20 degrees, it was reset to zero to avoid introducing unrealistic distortions. This adjustment was based on empirical observations showing that some characters could yield unreliable angle estimates due to their independent and symmetric shapes.

The resulting word images were visually diverse due to variation in letter samples, resizing, and spacing, while remaining structurally realistic. After manual inspection to remove overly thick, faint, or misaligned samples, the final synthetic dataset included approximately 45,000 word images.

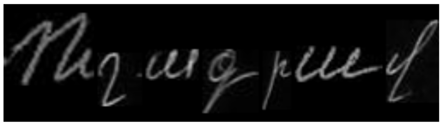

Figure 7. Example of a raw synthetic handwritten word image.

E. Preprocessing for Model Training

After model selection, synthetic word images were preprocessed to match the expected input format. Each image was inverted such that the characters appear in black on a white background. This inversion ensured compatibility with typical model architectures and improved contrast during training.

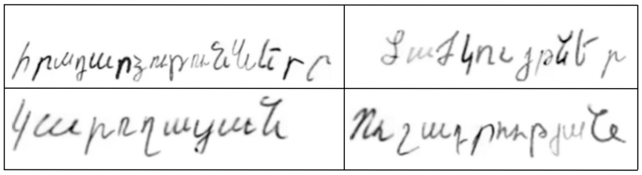

Figure 8. Examples of synthetically generated Armenian word images.

To further refine the visual quality of synthetic images, a multi-step preprocessing pipeline was introduced:

- **Gamma correction:** Adjusted pixel intensity nonlinearly to improve visibility of faded strokes.
- **Gaussian blurring and sharpening:** Smoothed noise and emphasized edges.
- **Local contrast enhancement:** Applied OpenCV's detail enhancement to suppress uneven background.
- **S-curve transformation:** Applied a sigmoid function to improve mid-tone contrast.
- **Shadow lifting:** Slightly brightened very dark strokes to prevent overcompression.
- **Final sharpening and clipping:** Enhanced edge visibility while ensuring pixel values stayed within bounds.

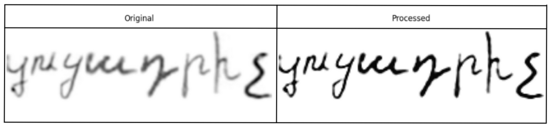

Figure 9. Word image before and after inversion and visual enhancement.

IV. Text Detection

To extract word-level regions from raw input images, this project integrates the CRAFT (Character Region Awareness for Text detection) model proposed by Baek et al. CRAFT is a fully convolutional network originally designed for detecting character-level regions in printed and scene text. It produces heatmaps for both character locations and inter-character affinities, enabling the grouping of individual characters into flexible, well-aligned word-level bounding boxes. Although primarily trained on synthetic scene text, CRAFT has demonstrated strong generalization to handwritten and multilingual text, and is widely used as the default detection module in libraries such as EasyOCR.

In this work, the publicly available pretrained CRAFT model is used without modification. The output heatmaps are parsed to produce bounding boxes around potential words. However, because the model occasionally outputs fragmented or overly dense bounding boxes, especially in handwritten inputs where inter-character spacing varies, custom postprocessing was applied.

To refine the detections, an overlap-based merging strategy was implemented. For each predicted bounding box, the algorithm iteratively checks whether it significantly overlaps with any unprocessed boxes. If an overlap is detected, the two boxes are merged into a larger one that encapsulates both. This process continues until no further merges are possible, resulting in a cleaner and more coherent set of word-level regions. The refined crops are then passed to the recognition stage.

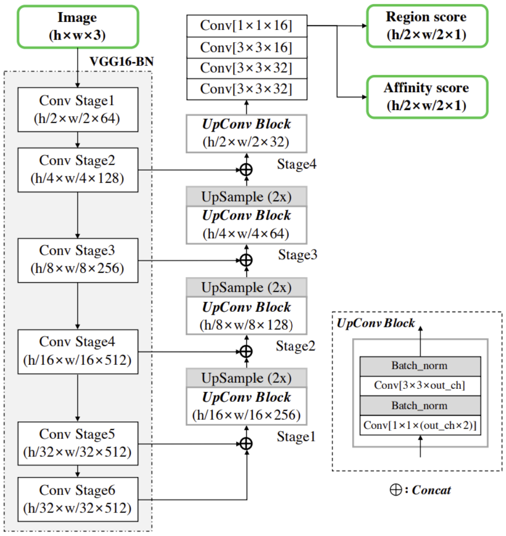

Figure 10. CRAFT Model Architecture

V. Training

A. SimpleHTR Framework

1) Model Architecture Overview: SimpleHTR is based on a CNN-RNN-CTC pipeline. The architecture consists of:

- **5 Convolutional Neural Network (CNN)** layers to extract spatial features from grayscale images.
- **2 Bidirectional LSTM layers** that capture temporal dependencies in both directions.
- A **projection layer** that maps the output to character probabilities.
- The **CTC loss function** to align predicted sequences with target text without requiring frame-level annotations.

Fine-tuning support has been added through two additional parameters:

- `specified_model_path`: Directory path to restore weights from a pre-trained model.
- `fine_tune`: Boolean flag to determine whether to restore all weights or exclude the projection layer.

When `fine_tune=True`, the final projection layer (i.e., `Variable_5`) is reinitialized to match the new character set (switching from English to Armenian in particular), while other layers are restored from a saved checkpoint.

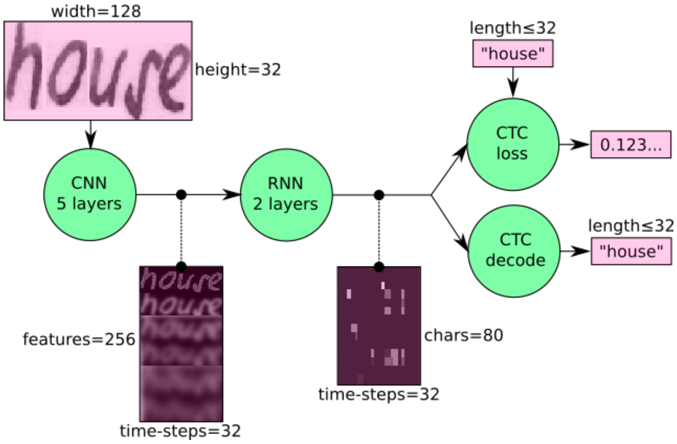

Figure 11. Model Architecture

2) Character Set and Corpus:

- The `char_list` used for training and decoding is dynamically specified depending on the language (Armenian or English).
- For Word Beam Search decoding, 2 more files are additionally created:
  - A list of word-forming characters saved in `wordCharList.txt`
  - An Armenian corpus derived from Wikipedia scrapes and saved in `corpus.txt`

These files are passed to the decoder at runtime and ensure compatibility across languages.

3) Decoding Strategies: The model supports three decoding types:

1) **Best Path Decoding (Greedy)**: selects the highest-probability character at each time step.
2) **Beam Search Decoding**: maintains a beam of likely candidates, exploring multiple paths.
3) **Word Beam Search Decoding** [18]: integrates language modeling via:
   - Dictionary constraints
   - Word-character separation
   - Word likelihoods derived from a corpus

By default, the model is initialized with Best Path Decoding. To specify a decoder type, the model should be initialized with the `decoder_type` parameter.

B. Clova AI's OCR Framework

1) Model Architecture Overview: The OCR model used in this project follows the modular architecture proposed by Baek et al. as part of Clova AI's Deep Text Recognition Benchmark. It is composed of four distinct stages: transformation, feature extraction, sequence modeling, and prediction. The architecture supports both scene text and handwritten recognition and is designed to evaluate the contribution of each module independently. While it was originally trained on large-scale synthetic datasets (MJSynth and SynthText) for printed text, its flexibility makes it suitable for transfer learning in low-resource handwritten scripts like Armenian.

The first stage applies a Thin-Plate Spline (TPS) transformation to spatially normalize the input image, addressing distortions caused by slanted or curved text. TPS uses a localization network composed of convolutional layers followed by fully connected layers that learn control points for warping.

Feature extraction is done by a ResNet-based backbone [17], which produces a sequence of high-level visual features from the normalized image. This is followed by adaptive average pooling to adjust the output shape for downstream processing.

For sequence modeling, two stacked Bidirectional LSTM layers are used to capture contextual dependencies in both directions along the text line.

Finally, a linear projection layer maps the LSTM outputs to a character probability distribution, and CTC loss is used to enable alignment-free sequence training.

Unlike the TensorFlow-based SimpleHTR implementation, this model is built in PyTorch and follows a clean structure that allows for relatively straightforward fine-tuning and modification.

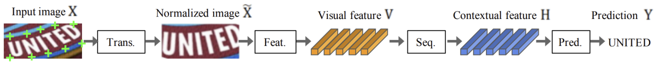

Figure 12. Pipeline for the TPS-ResNet-BiLSTM model

C. Connectionist Temporal Classification (CTC) Loss

CTC loss [12] is used for training sequence models when alignment between input and output is unknown. For input x and target sequence y:

$$\mathcal{L}_{CTC} = -\log \sum_{\pi \in \mathcal{B}^{-1}(y)} P(\pi|x)$$

Where:

- $\mathcal{B}$ is the collapsing function removing duplicates and blanks.
- π are valid alignment paths.
- $P(\pi|x)$ is calculated as the product of the model's output probabilities at each time step.

D. Training Procedure

Training proceeds using batches of word images with corresponding ground truth labels. The loss is backpropagated using the Adam optimizer. Model weights are saved whenever an improvement in CER in the validation set is being observed.

During fine-tuning, the projection layer is reinitialized and trained from scratch, while all other weights are restored from the original model.

E. Evaluation Metrics

Character Error Rate (CER) is used as the primary evaluation metric:

$$\text{CER} = \frac{S + D + I}{N}$$

Where:

- S: Number of substitutions
- D: Number of deletions
- I: Number of insertions
- N: Total number of characters in the ground truth

Word Error Rate (WER) is given by the following formula:

$$\text{WER} = \frac{S + D + I}{N}$$

Where:

- S: Number of substitutions
- D: Number of deletions
- I: Number of insertions
- N: Total number of words in the ground truth

Since the metric is being evaluated with respect to a single-word ground truth, each sample yields a WER of either 1 if the prediction is incorrect (either the prediction is blank, meaning $D = N = 1$ and $S = I = 0$, or the prediction is wrong, meaning $S = N = 1$ and $D = I = 0$) or 0 if it is correct ($S = I = D = 0$, $N = 1$) . Therefore, the average WER across the dataset becomes equivalent to $1-$ Word Accuracy Rate (WAR) of the model's predictions.

F. Model Checkpointing

Both models were saved using the checkpointing utilities provided by their respective frameworks: TensorFlow for the SimpleHTR model and PyTorch for the Clova AI OCR model. In both cases, checkpoints were written to the specified output directory whenever the validation set's Character Error Rate (CER) showed an improvement.

G. Data Split and Batch Size

For both models, the combined Armenian dataset was randomly split into training, validation, and test sets using an 80-10-10 ratio. This consistent partitioning ensured fair evaluation across both recognition pipelines.

H. Training Environment and Runtime

All model training and fine-tuning procedures were conducted on a local machine equipped with an NVIDIA GeForce GTX 1650 GPU. Despite its relatively modest computational capacity, each model took approximately 10 to 11 hours to train.

VI. Results

Performance was evaluated using Character Error Rate (CER) and Word Error Rate (WER) across all dataset splits. For the SimpleHTR model, three decoding strategies were tested: Greedy (Best Path), Beam Search, and Word Beam Search.

Greedy and Beam Search decoders performed similarly in terms of CER, both achieving 0.7% on the training set and increasing to 3.7% and 3.6% on the validation and test sets, respectively. Their WER values also followed a similar trend: Greedy decoding reached 5.2% on the training set, increasing to 19.4% and 19.2% on the validation and test sets, while Beam Search resulted in 5.1% WER on training and 19.3% and 19.0% on the validation and test sets, respectively.

In contrast, the Word Beam Search decoder demonstrated a significant improvement, achieving 0.48% CER and 2.7% WER on the training data, while maintaining strong generalization with 1.8% CER and 6.5% WER on the validation set, and 1.7% CER and 6.3% WER on the test set. The language-aware decoder proved particularly effective in modeling Armenian word structures, offering robust performance in a low-resource setup.

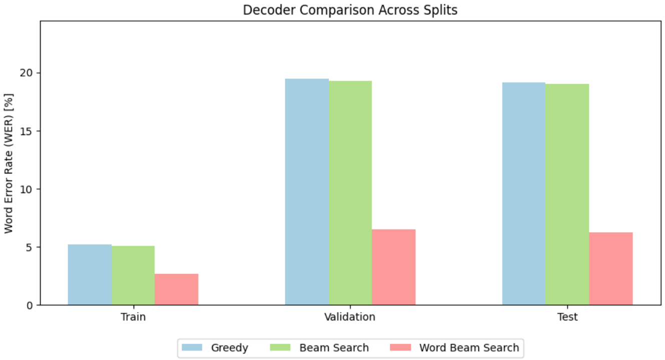

Figure 13. Comparison of SimpleHTR performance across data splits using different decoding strategies

The OCR-based model from Clova AI, which uses only CTC decoding without an integrated language model, achieved 0.2% CER and 1.9% WER on the training set and generalized well with 1.2% CER and 8.6% WER on the validation set and 1.2% CER and 9.5% WER on the test set. Despite lacking explicit language constraints, the model showed strong recognition capability, suggesting that its deeper and more expressive architecture may compensate for the absence of decoding priors.

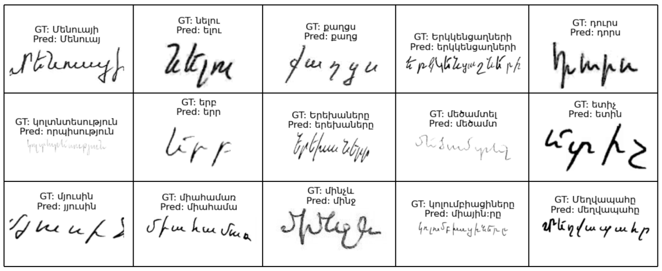

Figure 14. Examples of wrong predictions done by SimpleHTR using the WBS Decoding

When comparing the two systems, SimpleHTR with Word Beam Search slightly outperforms the Clova AI's OCR model on all evaluation splits. However, the gap narrows significantly when decoding strategies are controlled, and the Clova AI model shows a higher baseline performance with simple CTC decoding.

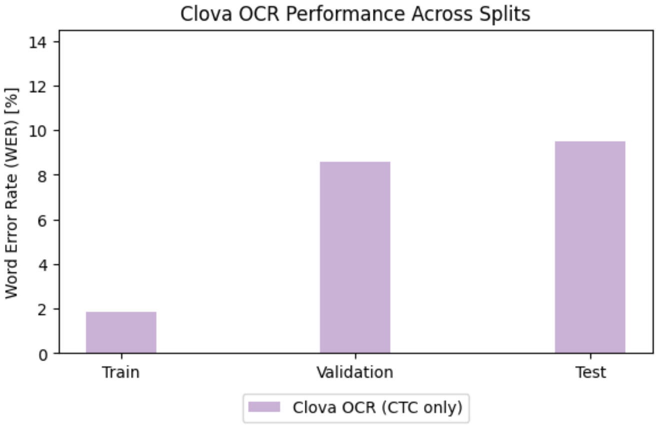

Figure 15. Comparison of Clova's OCR performance across data splits

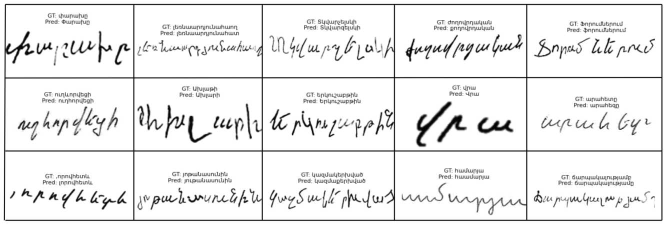

Figure 16. Examples of wrong predictions done by Clova's OCR model

VII. Pipeline

The pipeline accepts a full image that may contain multiple rows of handwritten words. The CRAFT-based detector first identifies and extracts word-level bounding boxes from the input image. These boxes are sorted from top to bottom and left to right to preserve reading order. It is done by computing the midpoint of each detected bounding box and iteratively assigning boxes to lines by comparing each box's vertical position to the current average y-coordinate of an existing line. As boxes are added to a line, the average y of that line is updated, and if a box deviates significantly from the current global line average, it starts a new line. After grouping, each line is sorted left-to-right based on horizontal midpoints. Each word image is then passed individually to a recognition model. The user can choose between two available models: fine-tuned SimpleHTR or Clova. After inference, the recognized words are reassembled in the correct order to produce the final output.

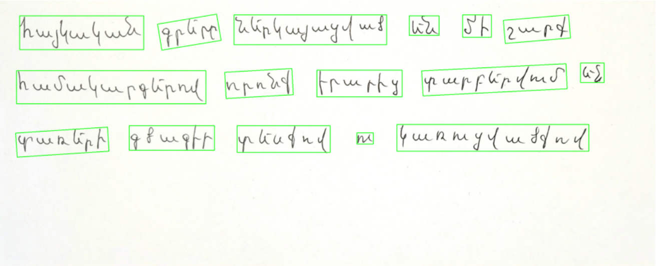
հայկական գրերը ներկայացված են մի շարք
համակարգերով որոնք իրարից տարբերվում են
տառերի գծագիր տեսքով ու կառուցվածքով

Figure 17. Visual output of the inference pipeline.
Recognized text by both SimpleHTR and Clova AI OCR Models: Հայկական գրերը ներկայացված են մի շարք համակարգերով որոնք իրարից տարբերվում են տառերի գծագիր տեսքով ու կառուցվածքով

VIII. Conclusion

This project presents a complete word-level handwritten text recognition system for the Armenian language. The two-stage approach combines CRAFT-based word detection with recognition models fine-tuned specifically for Armenian script. Both SimpleHTR and Clova were trained on a dataset that includes real and synthetic word images, addressing the lack of large-scale annotated Armenian handwriting. Evaluation on the training, validation, and test sets shows reliable performance, with Word Accuracy Rates exceeding 90% in optimal decoding configurations. Although some errors persist in cases of ambiguous or low-quality input, the overall results demonstrate the effectiveness of this approach for digitizing Armenian handwritten content.

IX. Future Work

One promising direction for future work is to fine-tune the text detection component on Armenian handwritten page-level data. The current CRAFT detector was not trained on handwritten inputs, and its performance occasionally degrades when dealing with dense or uneven layouts. Access to annotated page-format data would enable more robust bounding box predictions and better overall recognition. Additionally, incorporating a language model or spellchecker could improve final predictions, especially for visually similar Armenian characters. Further extensions could also explore end-to-end paragraph recognition or mobile deployment of the system.

Acknowledgments

This project builds upon the open-source implementations of Harald Scheidl's SimpleHTR and Word Beam Search Decoding system as well as Clova AI's deep-text-recognition-benchmark and CRAFT detection system. The author acknowledges the developers of these tools for making their work publicly available and enabling reproducible research in text recognition.

References

[1] U.-V. Marti and H. Bunke, "The IAM-database: An English sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition*, vol. 5, pp. 39–46, 2002. doi: 10.1007/s100320200071

[2] D. Ghazaryan and N. Potikyan, "Mashtots Dataset v2," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/competitions/mashtots-dataset-v2

[3] T. Scheidl, "Handwritten text recognition with TensorFlow," GitHub, 2018. [Online]. Available: https://github.com/githubharald/SimpleHTR

[4] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character Region Awareness for Text Detection," *arXiv preprint arXiv:1904.01941*, 2019. [Online]. Available: https://arxiv.org/abs/1904.01941

[5] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, "What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis," *arXiv preprint arXiv:1904.01906*, 2019. [Online]. Available: https://arxiv.org/abs/1904.01906

[6] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition," *arXiv preprint arXiv:1406.2227*, 2014. [Online]. Available: https://arxiv.org/abs/1406.2227

[7] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic Data for Text Localisation in Natural Images," *arXiv preprint arXiv:1604.06646*, 2016. [Online]. Available: https://arxiv.org/abs/1604.06646

[8] J. Puigcerver, "Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?," in *Proc. 14th Int. Conf. on Document Analysis and Recognition (ICDAR)*, vol. 1, 2017, pp. 67–72. doi: 10.1109/ICDAR.2017.20

[9] T. Bluche and R. Messina, "Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition," in *Proc. 14th Int. Conf. on Document Analysis and Recognition (ICDAR)*, vol. 1, 2017, pp. 646–651. doi: 10.1109/ICDAR.2017.111

[10] J. Puigcerver and C. Mocholí, "PyLaia," GitHub repository, 2018. [Online]. Available: https://github.com/jpuigcerver/PyLaia

[11] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models," *arXiv preprint arXiv:2109.10282*, 2022. [Online]. Available: https://arxiv.org/abs/2109.10282

[12] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proc. 23rd Int. Conf. on Machine Learning (ICML)*, 2006, pp. 369–376. doi: 10.1145/1143844.1143891

[13] J. Jaided and S. Sornlertlamvanich, "EasyOCR," GitHub Repository, 2020. [Online]. Available: https://github.com/JaidedAI/EasyOCR

[14] R. Smith, "An Overview of the Tesseract OCR Engine," in *Proc. 9th Int. Conf. on Document Analysis and Recognition (ICDAR)*, vol. 2, 2007, pp. 629–633. doi: 10.1109/ICDAR.2007.4376991.

[15] Google Cloud, "Cloud Vision API," [Online]. Available: https://cloud.google.com/vision

[16] Portmind, "Armenian OCR," GitHub, 2022. [Online]. Available: https://github.com/portmind/armenian-ocr

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

[18] H. Scheidl, S. Fiel, and R. Sablatnig, “Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm,” in *Proc. 16th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, 2018, pp. 253–258.

[19] A. F. de Sousa Neto, B. L. D. Bezerra, A. H. Toselli, and E. B. Lima, ”HTR-Flor++: A handwritten text recognition system based on a pipeline of optical and language models,” in *Proc. 23rd Int. Conf. on Intelligent User Interfaces (IUI)*, [Online]. Available: https://doi.org/10.1145/3395027.3419603

[20] C. Garrido-Muñoz, A. Ríos-Vila, and J. Calvo-Zaragoza, “Handwritten Text Recognition: A Survey,” *arXiv preprint arXiv:2502.08417*, 2025. [Online]. Available: https://arxiv.org/abs/2502.08417

[21] W. AlKendi, F. Gechter, L. Heyberger, and C. Guyeux, “Advancements and Challenges in Handwritten Text Recognition: A Comprehensive Survey,” *Journal of Imaging*, vol. 10, no. 1, p. 18, 2024. doi: https://doi.org/10.3390/jimaging10010018