# Extraction of Handwritten and Printed Cyrillic Text from Documents: A Resource-Efficient Pipeline

Daniel Halachev
Sofia University "St. Kliment Ohridski"
Sofia, Bulgaria
danihalachev@gmail.com

Ivan Koychev
Sofia University "St. Kliment Ohridski"
Sofia, Bulgaria
koychev@fmi.uni-sofia.bg

*The digitisation of historical, administrative, and personal documents in Bulgarian faces considerable challenges due to the lack of robust Optical Character Recognition (OCR) and Handwritten Text Recognition (HTR) systems tailored for the Cyrillic alphabet. While modern Vision-Language Models (VLMs) and large transformer-based architectures achieve state-of-the-art results, their performance and resource efficiency on low-resource languages remain prohibitive for decentralised, privacy-preserving applications. In this paper, we present a comprehensive, resource-efficient pipeline for extracting printed and handwritten Cyrillic text. Our system integrates advanced image preprocessing, YOLO-based document structure and table recognition, and a Permuted Autoregressive Sequence (PARSeq) model trained specifically for Bulgarian. We generated custom synthetic Bulgarian cursive datasets to mitigate the severe lack of real-world training data. Our evaluation indicates that the specialised PARSeq model outperforms traditional OCR tools such as Tesseract and EasyOCR on our custom degraded printed test set, and provides a practical, resource-efficient baseline for handwriting recognition compared to a modern local VLM (Qwen3-VL-4B). Finally, we discuss the discrepancy between synthetic and real handwritten data, highlighting the urgent need for a standardised, annotated Bulgarian HTR dataset.*

**Keywords:** Optical Character Recognition, Handwritten Text Recognition, Bulgarian Language, Cyrillic Alphabet, PARSeq, Document Layout Analysis

## 1. Introduction

The modern world relies heavily on digital information, yet a vast amount of historical, administrative, and cultural data in Bulgarian remains locked in physical paper documents. The automatic extraction of text from these documents—Optical Character Recognition (OCR) and Handwritten Text Recognition (HTR)—is essential for preserving, searching, and analysing this data. Documents such as exams, homework, essays, forms, books, magazines, and archival records contain valuable information that is yet to be digitised and analysed in its entirety. Such a system would improve accessibility and inspire future Bulgarian OCR research.

Despite substantial advancements in OCR technologies, most existing libraries and commercial systems are heavily optimised for the Latin alphabet. Tools that support Cyrillic

https://doi.org/10.47810/JCLIB.2.2026.06

often exhibit unsatisfactory accuracy, particularly when processing highly degraded printed text or cursive handwriting. Furthermore, while recent Large Language Models (LLMs) and Vision-Language Models (VLMs) have revolutionised natural language processing, their application to document recognition introduces severe drawbacks regarding high computational costs and privacy concerns when sending sensitive documents to third parties.

This research investigates the feasibility of using lightweight, highly efficient architectures for Bulgarian text recognition. Rather than relying on massive, resource-intensive models, we focus on optimising the quality-speed-cost trilemma through targeted experiments and architectural choices. A driving factor for our methodology was the strict requirement that the system must run locally on consumer-grade hardware (e.g. a standard laptop with 16 GB of RAM and an 8-core CPU). This constraint is essential to ensure data privacy, decentralisation, and accessibility. The primary goal was to design and evaluate a complete pipeline for recognising printed and handwritten Cyrillic text that is accurate, usable, and resource-efficient. To support practical digitisation workflows, the system was designed to expose a REST API and output results in standardised formats such as plain text, ALTO XML, and HOCR. Table 1 summarises the practical constraints that shape the full system.

| Aspect | Constraint or Objective |
|---|---|
| Hardware | Local execution on 16 GB RAM and 8 CPU cores |
| Deployment | Privacy-preserving, open-source, locally runnable |
| Interface | Browser GUI and REST API |
| Outputs | Plain text, ALTO XML, HOCR |

**Table 1**
Practical constraints shaping the system architecture.

To achieve these goals, we developed a complete document processing pipeline. We investigated the feasibility of using lightweight, highly efficient architectures for Bulgarian text recognition. We trained and evaluated a Permuted Autoregressive Sequence (PARSeq) (Bautista and Atienza 2022) model and benchmarked it against widely used open-source tools, Tesseract (Smith 2007) and EasyOCR (Kittinaradorn et al. 2020), as well as a local VLM, Qwen3-VL-4B (Qwen Team 2025).

In summary, our main contributions are:

- A practical, resource-efficient OCR/HTR pipeline prototype tailored for Bulgarian documents.
- A Bulgarian-adapted PARSeq training recipe utilising custom synthetic datasets to overcome the low-resource barrier.
- An empirical analysis of the discrepancy between synthetic and real-world handwriting, establishing a useful pilot baseline for future research.

To support reproducibility, the source code of the system is provided as anonymised supplementary material for review and will be made publicly available upon publication.

## 2. Related Work

The field of text recognition has evolved substantially from early mechanical devices to modern deep learning architectures. Early OCR systems relied on explicit character segmentation followed by classification using matrix matching or feature extraction combined with classifiers like Support Vector Machines (SVMs) or k-Nearest Neighbours (k-NN). Systems like Tesseract and EasyOCR remain standard open-source baselines due to their accessibility. However, they struggle considerably with cursive handwriting and heavily degraded documents due to their reliance on brittle segmentation algorithms.

The paradigm shifted with the introduction of Convolutional Neural Networks (CNNs) for feature extraction, which replaced manual engineering of shape descriptors with automatic representation learning. The subsequent introduction of Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks, and the Connectionist Temporal Classification (CTC) loss function (Graves et al. 2006) allowed for end-to-end sequence recognition without explicit segmentation. This led to the popular Convolutional Recurrent Neural Network (CRNN) architecture (Shi, Bai, and Yao 2017).

More recently, transformer-based architectures have dominated academic research in text recognition. Models such as ABINet (Fang et al. 2021), ViTSTR (Atienza 2021), and TrOCR (Li et al. 2022) leverage self-attention mechanisms to capture complex spatial and linguistic dependencies in parallel. While models like TrOCR achieve state-of-the-art results by combining powerful visual encoders with language-aware decoders, they often demand substantial computational resources and large labelled corpora, making them difficult to deploy in decentralised, low-resource environments.

To balance accuracy and efficiency, the PARSeq (Permuted Autoregressive Sequence) architecture was introduced. It utilises a Vision Transformer (ViT) (Dosovitskiy et al. 2021) encoder and a permutation-based autoregressive decoder. By training on permuted reading orders, it learns ensembles of internal language models, making it highly robust to noise and occlusion while remaining lightweight and fast during inference.

Modern multimodal large language models and document-understanding systems, such as Donut or GPT-family models, can perform OCR-like tasks as part of broader image understanding. However, these systems are not optimised specifically for text recognition, consume disproportionate resources, and create privacy concerns when documents must be sent to third-party infrastructure.

**Cyrillic and Bulgarian text recognition.** High-accuracy Cyrillic OCR systems are rare, and most public resources target Russian and Kazakh. For Bulgarian, existing work focuses on printed and historical text: using Bulgarian language resources to post-correct OCR output (Kratchanov, Laskova, and Simov 2020), and a benchmark with methods for post-OCR correction of historical documents in pre-1945 orthography (Beshirov et al. 2025). Tesseract and EasyOCR handle printed Bulgarian reasonably well, but no system reliably recognises Bulgarian cursive, which is the gap we address.

## 3. Proposed Methodology

To handle the high variance of real-world documents, the system is designed as a modular pipeline: Image Input → Preprocessing → Text Detection → Structure Recognition → Text Recognition → Output. Full-document recognition was rejected due to the lack of suitable annotated datasets and the insufficiency of synthetic data alone for reliable real-world performance.

### 3.1 System Architecture and Preprocessing

Document images often suffer from noise, poor lighting, stains, and geometric distortions. The system implements a suite of selectable preprocessing techniques to normalise the input. Crucially, preprocessing is treated as a configurable search space rather than a fixed recipe. Because the effect of preprocessing cannot be predicted reliably in advance, the system allows arbitrary user-selected preprocessing sequences. These dependencies form a Directed Acyclic Graph (DAG) of valid combinations.

The image representation is another critical design choice. Images are stored in the LAB colour space, where the L channel acts as a luminance-based black-and-white representation, while the A and B channels preserve colour information. This allows grayscale-style algorithms to operate only on the L channel, while spatial transformations (such as deskewing or dewarping) are propagated consistently to all channels. This enables in-place processing with good cache locality and minimal copying.

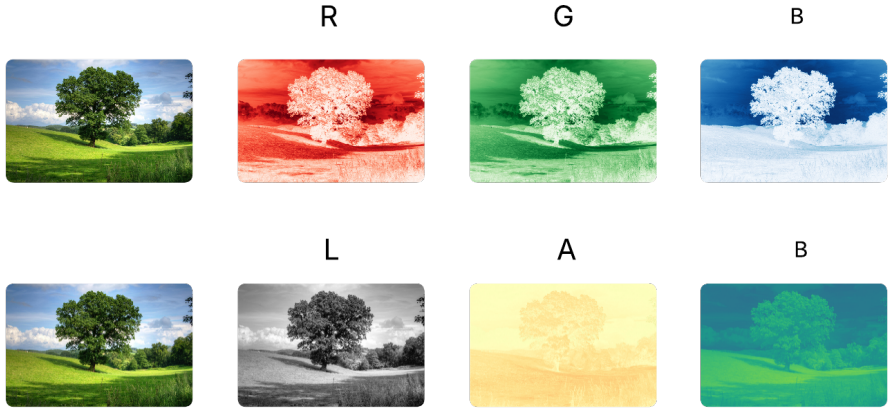


**Figure 1**
Comparison of processing in RGB vs. LAB colour space. Operating on the L channel preserves colour information while allowing grayscale algorithms to function correctly.

The implemented preprocessing techniques are categorised into four main areas to handle the wide variety of degradations found in historical and administrative documents:

- **Noise Reduction:** Documents often suffer from sensor noise, paper degradation, and ink bleed-through. The system implements linear filters (Average, Gaussian) for high-frequency noise, non-linear filters (Median) for salt-and-pepper noise, and advanced edge-preserving filters (Wiener, Bilateral, Non-Local Means) that smooth homogeneous regions while keeping text boundaries sharp.
- **Contrast Enhancement:** Faded ink and uneven illumination are addressed using Histogram Equalisation for global contrast and CLAHE (Contrast Limited Adaptive Histogram Equalisation) to prevent noise amplification in homogeneous areas while locally enhancing contrast.
- **Binarisation:** Converting images to black and white is essential for legacy OCR engines. The system includes global methods (Otsu's, Kittler-Illingworth) and local adaptive methods (Niblack's, Sauvola's) that calculate thresholds for each pixel based on its local neighbourhood, effectively separating text from varying backgrounds.
- **Geometric Correction:** Skewed and warped pages severely degrade recognition accuracy. The system uses the Hough Transform and Projection Profiles to detect and correct global page skew, and attempts Piecewise Vertical dewarping and Thin Plate Spline transformations to flatten curved text lines.

### 3.2 Text Detection and Structure Recognition

Accurate text recognition requires precise localisation. To locate text bounding boxes, the system supports either CRAFT (Baek et al. 2019) or YOLO-family text detectors (Jocher, Chaurasia, and Qiu 2024), exposed as interchangeable backends so users can choose whichever works best on their material without tying the write-up to a specific model version or training recipe.

For document structure recognition, we use an off-the-shelf layout detector from the YOLO family (Ultralytics 2023) (release/version not fixed in our pipeline) whose published weights were trained on *DocLayNet* (Pfitzmann et al. 2022); we did not train this component ourselves. It exposes high-level region classes (paragraphs, titles, lists, figures). However, most layout detection algorithms only recognise these broad semantic elements and do not support individual text line extraction. Furthermore, bounding-box regression models can be highly sensitive to document defects, often fragmenting paragraphs incorrectly when faced with skewed or degraded scans.

To address this, we cluster word-level detections into lines (and optionally group lines into blocks) with DBSCAN (Ester et al. 1996): the algorithm does not require a fixed number of clusters in advance and drops isolated noise. Radius $\epsilon$ is set proportional to row height to track font size, with boxes sorted by reading order. This is a lightweight geometric glue layer between detection and recognition, not a full document-understanding model.

Additionally, we integrate a YOLO-based table detector using weights trained on *PubTables-1M* (Smock, Pesala, and Abraham 2022) (released checkpoints; we did not train this submodule from scratch), extending the prototype toward broader document understanding.

### 3.3 Text Recognition Model Selection

Our approach relies on word-level recognition, offering the optimal compromise between visual difficulty, available data, latency, and the ability to leverage implicit language modelling.

Before selecting the final architecture, we conducted controlled experiments comparing three distinct paradigms: a custom CNN + Transformer Decoder, a Deformable CNN (DCNN) (Zhu et al. 2018) + Transformer Decoder, and PARSeq.

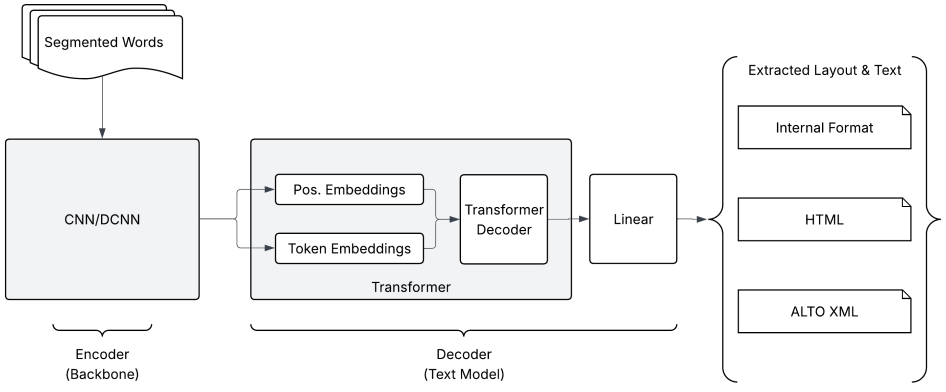


**Figure 2**
The baseline architecture evaluated during preliminary experiments, consisting of a ResNet-50 feature extractor coupled with a Transformer Decoder.

The CNN + Transformer Decoder served as our baseline. Our experiments revealed that a full encoder-decoder transformer is unnecessary; a lightweight transformer decoder paired with a strong CNN feature extractor is sufficient for the language-modelling component. We then hypothesised that Deformable Convolutions (DCNN), which adapt their receptive fields to the geometric variations of the input, would excel at capturing the high variance of cursive handwriting.

Surprisingly, replacing standard convolutions with DCNNs resulted in slower convergence and inferior accuracy (Figure 3). The deformable filters struggled to find consistent patterns in the highly irregular handwritten strokes.

Ultimately, PARSeq demonstrated better efficiency, faster convergence, and lower Character Error Rate (CER). The PARSeq model represents a substantial departure from traditional CRNN architectures by addressing the limitations of both purely autoregressive (AR) models (which suffer from error accumulation) and non-autoregressive (NAR) models (which lack language modelling capabilities).

PARSeq employs a Vision Transformer (ViT) as its visual encoder. The ViT processes the input image as a sequence of non-overlapping patches, utilising self-attention to capture global spatial dependencies across the entire image. This is particularly advantageous for

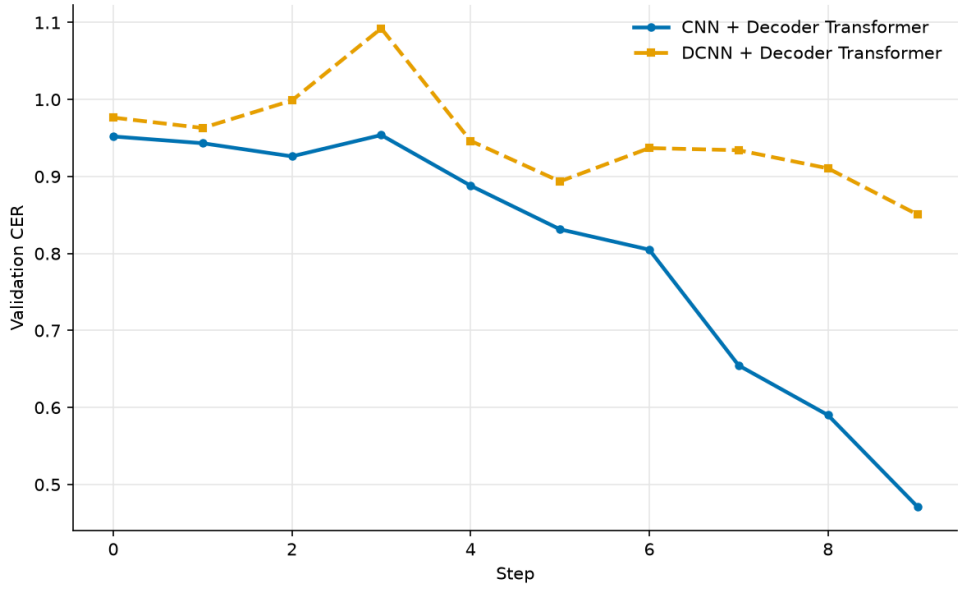


**Figure 3**
Validation *CER* for *CNN + DecoderTransformer* vs. *DCNN + DecoderTransformer* (same decoder). The DCNN encoder minimises error more slowly than the standard CNN backbone.

highly degraded text, where local features might be obscured, but global context can aid recognition.

The decoder is a lightweight, permutation-based autoregressive transformer. During training, PARSeq generates the target text sequence using multiple permuted reading orders. Predicting characters from arbitrary context subsets enables the model to learn an internal language model ensemble. During inference, it can operate in a standard left-to-right AR mode for maximum accuracy, or in an iterative refinement mode where it uses bidirectional context to correct initial predictions. This flexibility makes PARSeq robust to the high variance and occlusions typical of handwritten Cyrillic text.

Handling both printed and handwritten styles with a single model presents challenges like catastrophic forgetting. To mitigate the issue, we considered explicit style tokens (e.g. `[HAND]` and `[PRINT]`) as a lightweight way to condition a *single* recogniser so print and handwriting share weights instead of separate heads. This approach is applicable to all architectural variants, including PARSeq, but we did not pursue it, for three reasons. First, the architecture comparison had already consumed much of our experimental budget, and evaluating style conditioning properly would have required a separate set of experiments. Second, we expected little benefit: PARSeq is a strong recogniser and should handle both styles without an explicit switch, whereas a style token adds a single point of failure, since a misclassified style propagates errors through the rest of the pipeline. Third, the ViT encoder

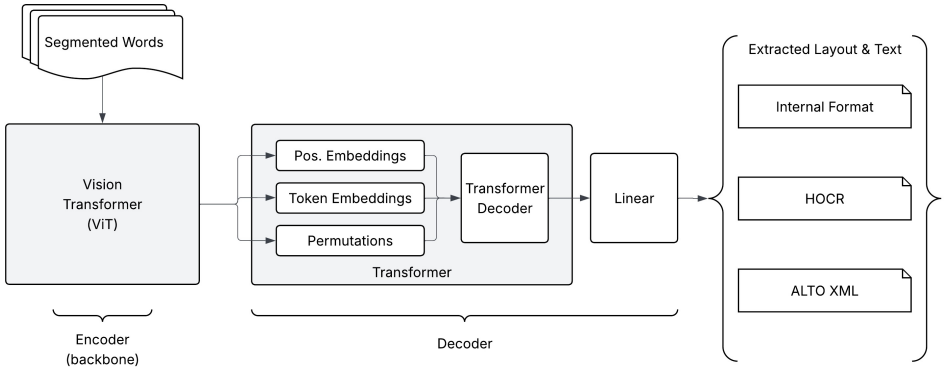


**Figure 4**
The PARSeq architecture, utilising a ViT encoder and a permutation-based autoregressive decoder.

already handles most of the recognition, so conditioning the smaller decoder would have limited effect.

### 3.4 Datasets and Training Curriculum

A major challenge in developing HTR systems for Bulgarian is the severe lack of annotated real-world datasets. To overcome this, we utilised a combination of synthetic and real-world Cyrillic datasets. For printed text, we utilised the MJSynth dataset (Jaderberg et al. 2014) (a massive collection of synthetic Latin scene text) to provide a strong baseline for visual feature extraction and regularisation. For printed Cyrillic, we aggregated several synthetic datasets (OCR Cyrillic Printed 1, 9, and 10), which contain millions of rendered words in various standard Cyrillic fonts.

For handwriting, due to the severe lack of real-world Bulgarian data, we generated four custom synthetic corpora based on free fonts that mimic different handwriting and typewritten styles: Pacifico (connected cursive), MarckScript (loose cursive), Comforter (informal handwriting), and GNU Typewriter (typewritten). The generation process involved rendering text with random variations in font size, spacing, and stroke width. During training, rendered line images were further augmented (Gaussian and motion blur, elastic warping, random paper textures and stains) to mimic degraded scans and historical documents. These were combined with a corpus of the 714,876 most frequent words from the Bulgarian Wikipedia (Kostov 2011). All single-letter words except the high-frequency prepositions/conjunctions were excluded. This exposes the model to Bulgarian vocabulary and orthographic patterns rather than relying only on foreign-language Cyrillic corpora.

For real-world handwriting, we incorporated the Cyrillic Handwriting Dataset (Werner 2021) (primarily Russian cursive) and the Kazakh Offline Handwritten Text Dataset (KOHTD) (Toiganbayeva et al. 2022). The KOHTD corpus was further processed to remove examples

containing characters specific to the Kazakh alphabet that were outside our defined support set.

Training a single model to recognise both styles requires careful curriculum learning. We utilised a four-phase strategy:

1. **Phase 0 (Alphabet Adaptation):** The primary goal of this initial phase was to warm up the newly initialised weights of the expanded classification head. The base PARSeq model, pre-trained on Latin text, was adapted to the Cyrillic alphabet. Weights for homoglyphs were transferred from the Latin model, while new Cyrillic characters were initialised randomly. Training utilised a 20% Latin and 80% Cyrillic printed mix for just 2 epochs to prevent the random weights from corrupting the pre-trained feature extractor.
2. **Phase 1 (Stabilisation):** With the classification head adapted, this phase performed full-network fine-tuning to solidify the model's understanding of Cyrillic typography. The model was trained for 10 epochs on a 25% Latin and 75% Cyrillic printed mix, establishing a stable, high-accuracy baseline on printed text before introducing the complex variance of handwriting.
3. **Phase 2 (Handwritten Introduction):** Introducing synthetic handwriting. The dataset mixture included the custom Bulgarian synthetic fonts alongside printed data to prevent forgetting. The model successfully learned to read synthetic handwriting despite heavy augmentations.
4. **Phase 3 (Real Handwritten Adaptation):** Fine-tuning exclusively on real-world handwritten datasets (KOHTD and Cyrillic Handwriting Dataset) to minimise the domain gap and handle the high variance of real human handwriting.

### 3.5 Implementation Details

The entire system was developed to run under resource constraints, ensuring it remains accessible to the general public, not just well-funded research institutions. The system is formally constrained to run locally on consumer-grade hardware on the order of 16 GB RAM and an 8-core CPU. This constraint guarantees privacy, as sensitive documents never leave the user's device.

To maximise extensibility, the backend exposes its functionality via a REST API. This allows the recognition engine to be integrated into broader document management systems or archival workflows. The system supports multiple standardised output formats: plain text for simple search and retrieval tasks, ALTO XML for describing the layout and content of physical text documents (preserving bounding box coordinates and reading order), and HOCR for embedding layout information directly into HTML.

Training deep learning models on millions of high-resolution images required a deliberate data pipeline. An initial attempt to *stream* examples directly from remote storage proved too slow for our setup and made proportional mixing across corpora difficult. We therefore converted all training corpora into compact *WebDataset* tar shards, stored them locally on the training machine, and fed them through PyTorch `Dataset`/`DataLoader` with multiple worker processes to overlap decoding, on-the-fly augmentation, and GPU compute.

## 4. Experiments and Results

We run two experiments. The first measures the final model on the held-out test splits of the training corpora, under conditions close to the training distribution, to establish an upper bound. The second measures it on real, unseen Bulgarian documents and compares it against widely used OCR tools and a local vision-language model, to test how far that accuracy carries to practical use.

We report CER and Word Error Rate (WER) throughout. Both are standard for OCR and HTR: CER captures character-level mistakes, and WER reflects whole-word usability, which is what matters for downstream search and indexing.

### 4.1 Test Set Performance

This first experiment measures the final model on the held-out test split of every training corpus, so accuracy is assessed under conditions that match training. To quantify uncertainty, we computed Wilson 95% confidence intervals (CIs) and two-proportion $z$-tests between the Phase 2 and final Phase 3 checkpoints on the real handwritten corpora. The final PARSeq model achieved strong results on the test splits of the training data. Table 2 summarises point estimates. Held-out sample sizes ranged from 14,831 characters (2,156 words) on the Cyrillic Handwriting Dataset to 5,398,419 characters (822,035 words) on Synthetic Cyrillic Large (PUMB AI 2024); normal approximations for the $z$-tests were appropriate throughout. Table 3 reports CIs at the final checkpoint. Table 4 summarises phase-to-phase significance. The model achieves high accuracy on stylistically controlled synthetic data, and scores on real-world sets provide an adequate baseline for generalisation to unseen Bulgarian cursive.

| Dataset | CER (%) | WER (%) |
|---|---|---|
| MJSynth (Latin) | 5.63 | 10.61 |
| OCR Cyrillic Printed (Combined) | 3.87 | 17.70 |
| BulgarianPacifico | 0.03 | 0.29 |
| BulgarianMarckScript | 0.03 | 0.23 |
| BulgarianComforter | 0.05 | 0.41 |
| BulgarianGNUTypewriter | 0.03 | 0.27 |
| Synthetic Cyrillic Large | 6.15 | 27.20 |
| Cyrillic Handwriting Dataset | 9.03 | 39.28 |
| KOHTD | 7.38 | 25.46 |

**Table 2**
Detailed Character Error Rate (CER) and Word Error Rate (WER) of the final PARSeq model on the test splits of the training corpora.

### 4.2 Real-World Comparison

To evaluate practical applicability, we constructed a custom test set. For printed text, we randomly selected highly degraded documents from a small subset of the Bulgarian National

| Dataset | CER | 95% CI | WER | 95% CI |
|---|---|---|---|---|
| Cyrillic Handwriting | 9.03% | [8.58, 9.50] | 39.28% | [37.24, 41.35] |
| KOHTD | 7.38% | [6.98, 7.80] | 25.46% | [23.91, 27.07] |

**Table 3**
Wilson 95% CIs for the final Phase 3 checkpoint on real handwritten test splits.

| Dataset | Metric | Phase 2 | Final | $p$ ($z$-test) |
|---|---|---|---|---|
| Cyrillic Handwriting | CER | 28.31% | 9.03% | $7.37 \times 10^{-7}$ |
| Cyrillic Handwriting | WER | 82.42% | 39.28% | $6.88 \times 10^{-4}$ |
| KOHTD | CER | 37.65% | 7.38% | $\approx 0$ |
| KOHTD | WER | 86.58% | 25.46% | $\approx 0$ |

**Table 4**
Phase 2 vs. final Phase 3 checkpoints on real handwritten corpora (two-proportion $z$-tests).

Radio's archive.[1] For handwritten text, we collected paragraphs randomly chosen from Wikipedia, featuring three distinct cursive styles. The ground truth for these samples was manually transcribed and verified by a native Bulgarian speaker. For these multi-line documents, predicted and ground-truth lines are aligned via the Hungarian algorithm on edit-distance costs before CER/WER are computed; unpaired lines count as full errors. We compared our PARSeq implementation against Tesseract and EasyOCR (for printed) and Qwen3-VL-4B-8bit (for handwritten).[2] To ensure a fair zero-shot evaluation on real-world document conditions, Tesseract and EasyOCR were evaluated out-of-the-box using their standard Bulgarian extensions and dictionaries without any further fine-tuning. Similarly, our PARSeq model was trained strictly on the training corpora splits and was not fine-tuned on any samples from this custom test set.

**Results on Printed Text:** As shown in Table 5, PARSeq outperformed both Tesseract and EasyOCR on degraded typewritten documents. The bidirectional context of PARSeq allowed it to correctly infer characters obscured by noise, whereas traditional OCR tools failed at the segmentation stage.

**Results on Handwritten Text:** Recognising real-world Bulgarian handwriting proved considerably more challenging. Table 6 compares PARSeq against Qwen3-VL-4B-8bit. While dedicated HTR architectures like TrOCR exist, they typically lack out-of-the-box support for Bulgarian cursive and would require resource-intensive pre-training. Instead, we selected Qwen3-VL-4B-8bit as our primary baseline because it was the strongest open-weights Vision-Language Model we identified that supports Bulgarian and still fits within our strict

1 These typewritten transcripts were provided by the Bulgarian National Radio through private correspondence for thesis research; their public disclosure would require separate permission.
2 Specifically, the MLX 8-bit checkpoint `https://huggingface.co/mlx-community/Qwen3-VL-4B-Instruct-8bit`, an 8-bit quantisation of `Qwen/Qwen3-VL-4B-Instruct`.

| Model | CER (%) | WER (%) | Time (s) |
|---|---|---|---|
| Tesseract | 12.9 | 33.5 | 8.2 |
| EasyOCR | 14.1 | 43.8 | 3.1 |
| **Ours** | **9.4** | **28.1** | 5.1 |

**Table 5**
Average results on degraded printed documents.

hardware constraints. Furthermore, benchmarking against a VLM reflects the current real-world trend of relying on general-purpose assistants for document understanding tasks. In this comparison, PARSeq outperformed the local VLM on the handwritten test set. Qwen3-VL suffered from context explosions and hallucinations on high-resolution images and sometimes stopped reading abruptly. Resource consumption was also much lower: roughly 4–6 GB vRAM for DocRec versus 12–16 GB vRAM for Qwen3-VL-4B-8bit.

| Model | CER (%) | WER (%) |
|---|---|---|
| Qwen3-VL-4B-8bit | 56.4 | 84.1 |
| **Ours** | **17.2** | **45.3** |

**Table 6**
Average results on handwritten documents across three cursive styles.

## 5. Discussion and Limitations

**The Synthetic vs. Real Data Discrepancy:** Loss curve analysis (Figure 5) reveals a negative correlation between synthetic and real handwriting data: while they share useful structure, synthetic cursive often fails to capture the variability, slant, and stroke continuity of real handwriting. This is reflected in the phase-wise accuracy trends (Table 7), where performance on real data improves substantially only during the final adaptation stage. Overall, data scarcity, rather than model complexity, remains the primary constraint, limited by the lack of large-scale, annotated Bulgarian HTR datasets.

| Stage | Printed CER (%) | CHD CER (%) | KOHTD CER (%) |
|---|---|---|---|
| Phase 0 | 7.40 | – | – |
| Phase 1 | 3.10 | – | – |
| Phase 2 | 3.43 | 28.31 | 37.65 |
| Phase 3 | 3.98 | 6.11 | 6.68 |
| Final Test | 3.87 | 9.03 | 7.38 |

**Table 7**
Phase-wise development of recognition accuracy. Real-handwriting performance improves sharply only in the final adaptation stage, justifying the staged curriculum.

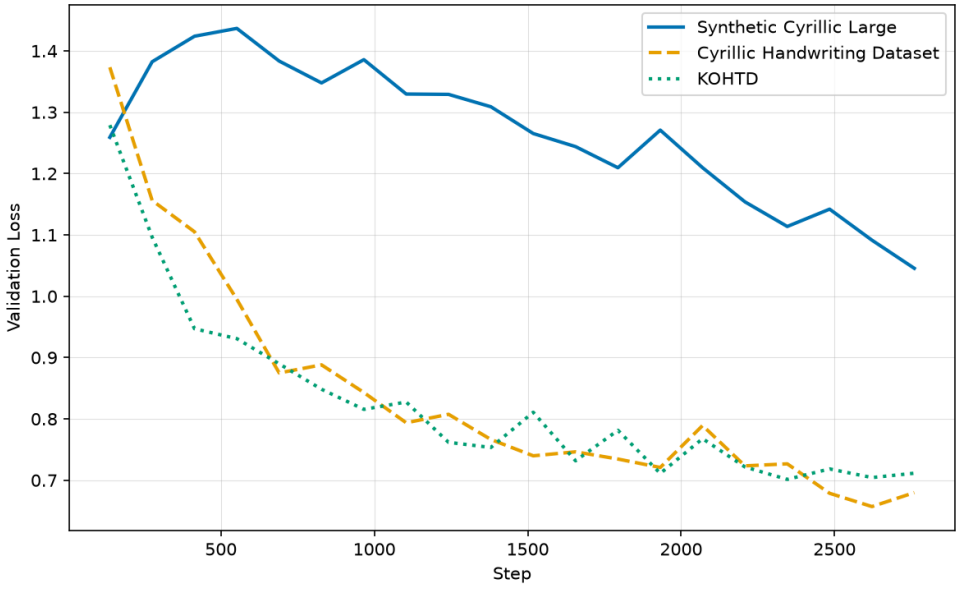


**Figure 5**
The observed discrepancy between validation loss on synthetic datasets and real-world handwritten datasets, highlighting the limitations of purely synthetic training.

**Error Analysis:** The real-world results are lower than the test-split results for two main reasons:

1. The real-world set is genuinely unseen Bulgarian handwriting; no real-world corpus of handwritten Bulgarian was available for training.
2. Even the real handwriting datasets suffer from limited variance. Their training, validation and test splits are mostly words segmented from forms, student essays and notebooks. The dataset details are unclear, but despite their large size these corpora were likely built from a relatively small number of independent documents that were then segmented into words. The words are therefore many, but the writers behind them may be few.

Per-character errors were limited mostly to confusions between visually similar letters. On the real-world samples these were о/с (where ink had worn away), г/т in plain handwriting, and ъ/ь. In cursive handwriting, where ligatures are denser, the most frequently confused pairs were а/о, и/ш, е/с, н/п and к/н. The high character and word error rates were mostly driven by ligatures in dense handwriting, where the model struggled to separate individual characters. This is somewhat surprising, considering the model does not rely on explicit segmentation and should, in principle, cope with connected letters. In certain limited cases, the slant of the lines also caused confusion, as words were appended to the wrong line.

**Opportunities for Improvement:** To improve accuracy on highly variable real-world Bulgarian handwriting, addressing the data deficit remains the most critical step. Building large, annotated corpora is essential and can be achieved through institutional data collection. For example, universities could systematically gather student-written materials and integrate lightweight annotation into coursework, enabling continuous corpus growth.

In parallel, exploring larger model architectures, combined with efficiency techniques such as quantisation or knowledge distillation, offers a practical path to increased capacity within fixed hardware constraints. Recent methods such as Google's TurboQuant further highlight the potential for optimisation without prohibitive computational cost.

**The Role of Local VLMs in Document Processing:** Our experiments with Qwen3-VL-4B-8bit highlight a crucial limitation in the current trend toward using general-purpose Vision-Language Models for specialised tasks like dense document OCR. While these models possess remarkable zero-shot capabilities, their autoregressive nature over massive context windows makes them highly inefficient for extracting dense, full-page text. The observed hallucinations and abrupt stopping are likely symptoms of the model losing its spatial grounding within the high-resolution image encoding. For tasks requiring deterministic, high-throughput text extraction, specialised architectures like PARSeq remain more practical in both speed and resource efficiency.

**Threats to Validity:** While the proposed pipeline demonstrates strong potential, several limitations must be acknowledged to contextualise the results fairly:

- **External Validity (Evaluation Size):** The real-world evaluation was conducted on a highly curated but small custom test set (10 printed pages, 12 handwritten paragraphs). While indicative of the model's potential capabilities, results should be interpreted as pilot evidence rather than definitive benchmarking.
- **Construct Validity (Baselines and Metrics):** The custom synthetic datasets lack realistic noise in their validation splits, leading to validation scores that mimic high-quality scans rather than degraded conditions. Furthermore, while Qwen3-VL-4B-8bit serves as a realistic local assistant baseline, it is not a dedicated HTR model.
- **Internal Validity (Pipeline Cascades):** The modular pipeline architecture is susceptible to cascading errors; failures in early stages (e.g. layout recognition fragility on degraded historical data) propagate to and multiply within the text recognition stage.

**Ethical Considerations:** All training corpora are used under their respective open licences. The system's local-only design mitigates privacy risks, but any HTR technology can be misused for bulk transcription of private correspondence and should be deployed with care.

## 6. Conclusion

In this paper, we presented a comprehensive, end-to-end software prototype for extracting printed and handwritten Cyrillic text from documents under real constraints. By integrating advanced preprocessing treated as a configurable search space, document structure recognition, and fine-tuning the PARSeq architecture via a staged training curriculum, the system addresses several practical limitations of existing tools.

Our results indicate that specialised, lightweight architectures can outperform standard baselines like Tesseract and EasyOCR on degraded printed text, and can be more stable and resource-efficient than local VLMs for handwriting recognition. The results suggest that specialisation and careful data design can rival larger general-purpose models under strict hardware and privacy constraints.

Crucially, our experiments show that the primary barrier to further progress in Bulgarian HTR is not algorithmic but the scarcity of real-world annotated data: synthetic data cannot fully bridge the domain gap to natural handwriting. Coordinated, community-driven data collection (e.g. university-sourced essay corpora) could thus be the most impactful next step.

## Use of AI Tools

During the revision of this manuscript, the authors used a general-purpose AI assistant (Anthropic's Claude) to check the bibliography for missing fields, to normalise text formatting and correct spelling mistakes. The study's methods, experiments, results, and conclusions are entirely the authors' own. All AI-assisted changes were reviewed, edited, and verified by the authors, who take full responsibility for the content.

## Acknowledgments

This work is partially supported by the project UNITe BG16RFPR002-1.014-0004 funded by PRIDST and EU NextGenerationEU project, through the National Recovery and Resilience Plan of the Republic of Bulgaria, project SUMMIT, No BG-RRP-2.004-0008.

## References

Atienza, Rowel. 2021. Vision transformer for fast and efficient scene text recognition. In *Proceedings of the 16th International Conference on Document Analysis and Recognition (ICDAR 2021), Lausanne, Switzerland, Part I*, volume 12821 of *Lecture Notes in Computer Science*, pages 319–334, Springer.

Baek, Youngmin, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character region awareness for text detection. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019), Long Beach, CA, USA*, pages 9365–9374, Computer Vision Foundation / IEEE.

Bautista, Darwin and Rowel Atienza. 2022. Scene text recognition with permuted autoregressive sequence models. In *Proceedings of the 17th European Conference on Computer Vision (ECCV 2022), Tel Aviv, Israel*, volume 13688 of *Lecture Notes in Computer Science*, pages 178–196, Springer.

Beshirov, Angel, Milena Dobreva, Dimitar Dimitrov, Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2025. Post-OCR text correction for Bulgarian historical documents. *International Journal on Digital Libraries*, 26.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021), Virtual Event, Austria*.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231, AAAI Press.

Fang, Shancheng, Hongtao Xie, Yuxin Wang, Zhendong Mao, and Yongdong Zhang. 2021. Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition. In *Proceedings of the 2021 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2021)*, pages 7098–7107, Computer Vision Foundation / IEEE.

Graves, Alex, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, Association for Computing Machinery, New York, NY, USA.

Jaderberg, Max, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Synthetic data and artificial neural networks for natural scene text recognition. In *Proceedings of the Workshop on Deep Learning (NIPS)*. ArXiv, abs/1406.2227.

Jocher, Glenn, Ayush Chaurasia, and Jing Qiu. 2024. YOLO by Ultralytics. https://github.com/ultralytics/ultralytics.

Kittinaradorn, Rakpong et al. 2020. EasyOCR. https://github.com/JaidedAI/EasyOCR.

Kostov, Nikolay. 2011. Analiz na balgarskiya ezik chrez Wikipedia (in Bulgarian). https://nikolay.it/Blog/2011/08/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7-%D0%BD%D0%B0-%D0%B1%D1%8A%D0%BB%D0%B3%D0%B0%D1%80%D1%81%D0%BA%D0%B8%D1%8F-%D0%B5%D0%B7%D0%B8%D0%BA-%D1%87%D1%80%D0%B5%D0%B7-Wikipedia/3.

Kratchanov, Ivan, Laska Laskova, and Kiril Simov. 2020. Towards improving OCR accuracy with Bulgarian language resources. In *Proceedings of Twin Talks 3: Understanding and Facilitating Collaboration in Digital Humanities (DH 2020)*, volume 2717 of *CEUR Workshop Proceedings*, pages 115–123.

Li, Minghao, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2022. TrOCR: Transformer-based optical character recognition with pre-trained models. ArXiv, abs/2109.10282.

Pfitzmann, Birgit, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter W. J. Staar. 2022. DocLayNet: A large human-annotated dataset for document-layout segmentation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2022), Washington, DC, USA*, pages 3743–3751, ACM.

PUMB AI. 2024. Synthetic Cyrillic Large. https://huggingface.co/datasets/pumb-ai/synthetic-cyrillic-large. Apache-2.0 license.

Qwen Team. 2025. Qwen3-VL technical report. ArXiv, abs/2511.21631.

Shi, Baoguang, Xiang Bai, and Cong Yao. 2017. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304.

Smith, Ray. 2007. An overview of the Tesseract OCR engine. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Paraná, Brazil*, volume 2, pages 629–633, IEEE.

Smock, Brandon, Rohith Pesala, and Robin Abraham. 2022. PubTables-1M: Towards comprehensive table extraction from unstructured documents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022), New Orleans, LA, USA*, pages 4634–4642, IEEE.

Toiganbayeva, Nazgul, Mahmoud Kasem, Galymzhan Abdimanap, Kairat Bostanbekov, Abdelrahman Abdallah, Anel Alimova, and Daniyar Nurseitov. 2022. KOHTD: Kazakh offline handwritten text dataset. *Signal Processing: Image Communication*, page 116827.

Ultralytics. 2023. Ultralytics YOLO. https://github.com/ultralytics/ultralytics.

Werner, Constantin. 2021. Cyrillic Handwriting Dataset. https://www.kaggle.com/datasets/constantinwerner/cyrillic-handwriting-dataset.

Zhu, Xizhou, Han Hu, Stephen Lin, and Jifeng Dai. 2018. Deformable ConvNets v2: More deformable, better results. ArXiv, abs/1811.11168.

## Appendices

### A. Real-world handwritten examples

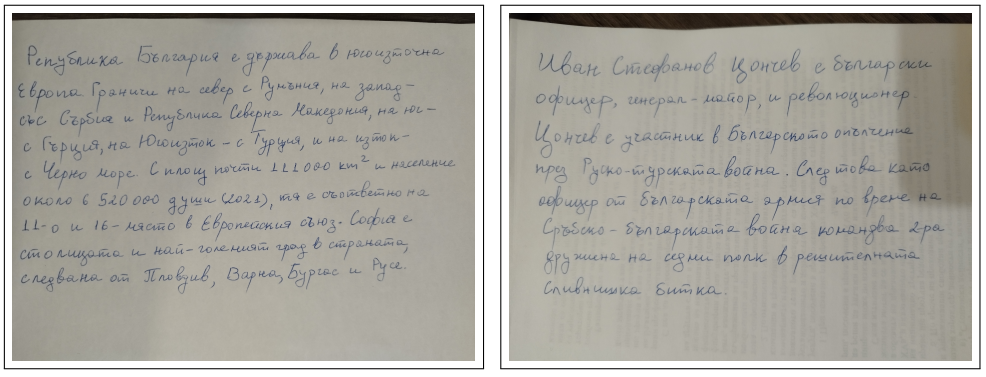


**Figure 6**
Examples of real-world handwritten documents from the custom test set, illustrating variation in slant, stroke connection, spacing, and line regularity.

### B. Detailed Dataset Composition

Table 8 provides a comprehensive breakdown of the datasets utilised for training and testing the text recognition models. The synthetic handwritten datasets were generated using custom scripts to mimic various Bulgarian cursive styles, substantially expanding the available training data for low-resource Cyrillic handwriting.

| Dataset | Style | Type | Train. subset | Test. subset |
|---|---|---|---|---|
| MJSynth | printed | synthetic | 7,220,000 | 892,000 |
| OCR Cyrillic Printed 1, 9, 10 | printed | synthetic | 2,700,000 | 300,000 |
| BulgarianPacifico | handwritten | synthetic | 643,364 | 71,484 |
| BulgarianMarckScript | handwritten | synthetic | 643,364 | 71,484 |
| BulgarianComforter | handwritten | synthetic | 643,364 | 71,484 |
| BulgarianGNUTypewriter | printed | synthetic | 643,364 | 71,484 |
| Synthetic Cyrillic Large | handwritten | synthetic | 3,120,000 | 779,000 |
| Cyrillic Handwriting Dataset | handwritten | real | 72,285 | 1,543 |
| KOHTD | handwritten | real | 49,497 | 5,500 |

**Table 8**
Summary of the datasets used for training and testing the recognition models.

### C. Dataset Sources

Table 8 lists the datasets used in this work. The publicly available datasets can be obtained from the following sources:

- MJSynth:
  https://huggingface.co/datasets/priyank-m/MJSynth_text_recognition
- OCR Cyrillic Printed 1:
  https://huggingface.co/datasets/DonkeySmall/OCR-Cyrillic-Printed-1
- OCR Cyrillic Printed 9:
  https://huggingface.co/datasets/DonkeySmall/OCR-Cyrillic-Printed-9
- OCR Cyrillic Printed 10:
  https://huggingface.co/datasets/DonkeySmall/OCR-Cyrillic-Printed-10
- Synthetic Cyrillic Large:
  https://huggingface.co/datasets/pumb-ai/synthetic-cyrillic-large
- Cyrillic Handwriting Dataset:
  https://www.kaggle.com/datasets/constantinwerner/cyrillic-handwriting-dataset
- KOHTD:
  https://github.com/abdoelsayed2016/KOHTD

The custom synthetic Bulgarian handwriting datasets generated for this work are publicly available:

- BulgarianPacifico:
  https://huggingface.co/datasets/KotakaDanski/BulgarianPacifico
- BulgarianMarckScript:
  https://huggingface.co/datasets/KotakaDanski/BulgarianMarckScript
- BulgarianComforter:
  https://huggingface.co/datasets/KotakaDanski/BulgarianComforter
- BulgarianGNUTypewriter:
  https://huggingface.co/datasets/KotakaDanski/BulgarianGNUTypewriter

### D. Supported Alphabet

The system supports a carefully curated alphabet designed to handle both standard modern Bulgarian and common historical Cyrillic characters, as well as Latin fragments frequently found in administrative documents.

| A–I | | J–R | | S–Z | |
|---|---|---|---|---|---|
| **Uppercase** | **Lowercase** | **Uppercase** | **Lowercase** | **Uppercase** | **Lowercase** |
| A | a | J | j | S | s |
| B | b | K | k | T | t |
| C | c | L | l | U | u |
| D | d | M | m | V | v |
| E | e | N | n | W | w |
| F | f | O | o | X | x |
| G | g | P | p | Y | y |
| H | h | Q | q | Z | z |
| I | i | R | r | | |

**Table 9**
The supported Latin alphabet subset.

| А–Й | | К–У | | Ф–Я + Archaic | |
|---|---|---|---|---|---|
| **Uppercase** | **Lowercase** | **Uppercase** | **Lowercase** | **Uppercase** | **Lowercase** |
| А | а | К | к | Ф | ф |
| Б | б | Л | л | Х | х |
| В | в | М | м | Ц | ц |
| Г | г | Н | н | Ч | ч |
| Д | д | О | о | Ш | ш |
| Е | е | П | п | Щ | щ |
| Ж | ж | Р | р | Ъ | ъ |
| З | з | С | с | Ь | ь |
| И | и | Т | т | Ю | ю |
| Й | й | У | у | Я | я |
| | | | | Ѣ | ѣ |
| | | | | Ѧ | ѧ |

**Table 10**
The supported Cyrillic alphabet subset, including historical characters for archival document processing.

### E. Table Detector Training Results

Table 11 reports training validation metrics on a 100,000-document subset of *PubTables-1M* (Smock, Pesala, and Abraham 2022) (8 epochs).

| Ep. | Precision | Recall | mAP50 | mAP50-95 | Val box | Val cls | Val dfl |
|---|---|---|---|---|---|---|---|
| 1 | 0.7182 | 0.6989 | 0.7322 | 0.5840 | 0.8186 | 0.6128 | 0.0121 |
| 2 | 0.8233 | 0.7897 | 0.8255 | 0.6891 | 0.6472 | 0.4516 | 0.0077 |
| 3 | 0.8778 | 0.8246 | 0.8724 | 0.7480 | 0.5536 | 0.3701 | 0.0060 |
| 4 | 0.9117 | 0.8448 | 0.8954 | 0.7878 | 0.4825 | 0.3132 | 0.0055 |
| 5 | 0.9260 | 0.8609 | 0.9082 | 0.8138 | 0.4457 | 0.2796 | 0.0046 |
| 6 | 0.9232 | 0.8707 | 0.9129 | 0.8254 | 0.4200 | 0.2600 | 0.0044 |
| 7 | 0.9339 | 0.8781 | 0.9207 | 0.8379 | 0.3999 | 0.2428 | 0.0041 |
| 8 | 0.9375 | 0.8821 | 0.9234 | 0.8430 | 0.3924 | 0.2331 | 0.0040 |

**Table 11**
Training validation metrics on a subset of PubTables-1M.

### F. Training Curriculum Dataset Proportions

The four-phase curriculum learning strategy required carefully balancing the proportions of synthetic and real datasets to prevent catastrophic forgetting while adapting to new domains.

| Dataset | Phase 0 | Phase 1 | Phase 2 | Phase 3 |
|---|---|---|---|---|
| MJSynth | 0.20 | 0.25 | 0.05 | 0.05 |
| OCR Cyrillic Printed | 0.80 | 0.75 | 0.25 | 0.10 |
| Synthetic Cyrillic Large | – | – | 0.30 | 0.10 |
| BulgarianPacifico | – | – | 0.10 | 0.05 |
| BulgarianMarckScript | – | – | 0.10 | 0.05 |
| BulgarianComforter | – | – | 0.10 | 0.05 |
| BulgarianGNUTypewriter | – | – | 0.10 | 0.05 |
| Cyrillic Handwriting Dataset | – | – | – | 0.35 |
| KOHTD | – | – | – | 0.20 |

**Table 12**
Proportions of the training datasets across the four phases of the PARSeq training curriculum.

### G. Layout Recognition Algorithms

The system offers two interchangeable families for text detection: CRAFT (Character Region Awareness for Text Detection) and YOLO-based detectors.

CRAFT is a robust text detector that localises characters and links them into lines, which suits unconstrained layouts.

YOLO-family detectors provide fast region proposals; users can swap implementations or checkpoints as needed alongside CRAFT.

### H. Training Curriculum Details

The training of the PARSeq model was conducted in four distinct phases to ensure stable convergence and prevent catastrophic forgetting. The dataset proportions, hyperparameters, and results for each phase are detailed below.

#### H.1 Phase 0: Alphabet Adaptation

| Dataset | Proportion |
|---|---|
| MJSynth | 0.20 |
| OCR Cyrillic Printed | 0.80 |

**Table 13**
Proportions of the training datasets for *PARSeq* in Phase 0.

| Hyperparameter | Value |
|---|---|
| Epochs | 2 |
| Batch Size | 512 |
| Learning Rate | $1 \times 10^{-4}$ |
| Weight Decay | $1 \times 10^{-2}$ |
| Warmup | 5% |
| Patience | 5 |
| Workers | 8 |

**Table 14**
Hyperparameters for training *PARSeq* in Phase 0.

| Dataset | CER (%) | WER (%) |
|---|---|---|
| MJSynth | 7.92 | 13.30 |
| OCR Cyrillic Printed | 7.40 | 23.37 |

**Table 15**
Results from the training of *PARSeq* in Phase 0.

### H.2 Phase 1: Printed Cyrillic Stabilisation

| Dataset | Proportion |
|---|---|
| MJSynth | 0.25 |
| OCR Cyrillic Printed | 0.75 |

**Table 16**
Proportions of the training datasets for *PARSeq* in Phase 1.

| Hyperparameter | Value |
|---|---|
| Epochs | 10 |
| Batch Size | 512 |
| Learning Rate | $1 \times 10^{-4}$ |
| Weight Decay | $1 \times 10^{-2}$ |
| Warmup | 5% |
| Patience | 5 |
| Workers | 8 |

**Table 17**
Hyperparameters for training *PARSeq* in Phase 1.

| Dataset | CER (%) | WER (%) |
|---|---|---|
| MJSynth | 4.29 | 7.74 |
| OCR Cyrillic Printed | 3.10 | 15.76 |

**Table 18**
Results from training *PARSeq* in Phase 1.

### H.3 Phase 2: Handwritten Introduction

| Dataset | Proportion |
|---|---|
| MJSynth | 0.05 |
| OCR Cyrillic Printed (1, 9, 10) | 0.25 |
| Synthetic Cyrillic Large | 0.30 |
| BulgarianPacifico | 0.10 |
| BulgarianMarckScript | 0.10 |
| BulgarianComforter | 0.10 |
| BulgarianGNUTypewriter | 0.10 |

**Table 19**
Proportions of the training datasets for *PARSeq* in Phase 2.

| Hyperparameter | Value |
|---|---|
| Max Epochs | 5 |
| Batch Size | 512 |
| Learning Rate | $5 \times 10^{-5}$ |
| Weight Decay | $1 \times 10^{-2}$ |
| Warmup | 5% |
| Patience | 5 |
| Workers | 8 |

**Table 20**
Hyperparameters for training *PARSeq* in Phase 2.

| Dataset | CER (%) | WER (%) |
|---|---|---|
| MJSynth | 5.17 | 9.69 |
| OCR Cyrillic Printed 1 | 2.69 | 9.67 |
| OCR Cyrillic Printed 9 | 7.10 | 48.83 |
| OCR Cyrillic Printed 10 | 2.56 | 9.81 |
| OCR Cyrillic Printed (concat) | 3.43 | 17.02 |
| Synthetic Cyrillic Large | 5.72 | 25.74 |
| Bulgarian Pacifico | 0.03 | 0.29 |
| Bulgarian MarckScript | 0.02 | 0.19 |
| Bulgarian Comforter | 0.04 | 0.36 |
| Bulgarian GNU Typewriter | 0.03 | 0.25 |
| Cyrillic Handwriting | 28.31 | 82.42 |
| KOHTD | 37.65 | 86.58 |

**Table 21**
Results from training *PARSeq* in Phase 2.

### H.4 Phase 3: Real Handwritten Adaptation

| Dataset | Proportion |
|---|---|
| MJSynth | 0.05 |
| OCR Cyrillic Printed (1, 9, 10) | 0.10 |
| Synthetic Cyrillic Large | 0.10 |
| BulgarianPacifico | 0.05 |
| BulgarianMarckScript | 0.05 |
| BulgarianComforter | 0.05 |
| BulgarianGNUTypewriter | 0.05 |
| Cyrillic Handwriting Dataset | 0.35 |
| KOHTD | 0.20 |

**Table 22**
Proportions of the training datasets for *PARSeq* in Phase 3.

| Hyperparameter | Value |
|---|---|
| Epochs | 10–20 |
| Batch Size | 512 |
| Learning Rate | $5 \times 10^{-4}$ |
| Weight Decay | $1 \times 10^{-2}$ |
| Warmup | 5% |
| Patience | 5 |
| Workers | 8 |

**Table 23**
Hyperparameters for training *PARSeq* in Phase 3.

| Dataset | CER (%) | WER (%) |
|---|---|---|
| MJSynth | 5.50 | 10.49 |
| OCR Cyrillic Printed | 3.98 | 17.67 |
| Synthetic Cyrillic Large | 6.13 | 27.25 |
| Cyrillic Handwriting | 6.11 | 23.92 |
| KOHTD | 6.68 | 24.87 |

**Table 24**
Validation metrics from training *PARSeq* in Phase 3 (final).

### I. Structure and Table Recognition Algorithms

For document structure recognition, we use an off-the-shelf YOLO-family layout detector (no fixed version string in our deployment) whose released weights were trained on *DocLayNet* (Pfitzmann et al. 2022); we did not train this module. It exposes the following classes: Caption, Footnote, Formula, List-item, Page-footer, Page-header, Picture, Section-header, Table, Text, and Title.

For tables lacking borders or complex spanning cells, we integrate a YOLO-based detector using weights trained on a subset of 100,000 documents from *PubTables-1M* (Smock, Pesala, and Abraham 2022) (again: third-party weights; not trained in this work).

### J. Validation Metrics during Training

Figures 7–11 show validation curves by training phase. Each figure stacks two plots vertically (CER on top; WER or training loss below). Phase 3 is shown twice because we validate separately on the *Cyrillic Handwriting Dataset* and on *KOHTD*.

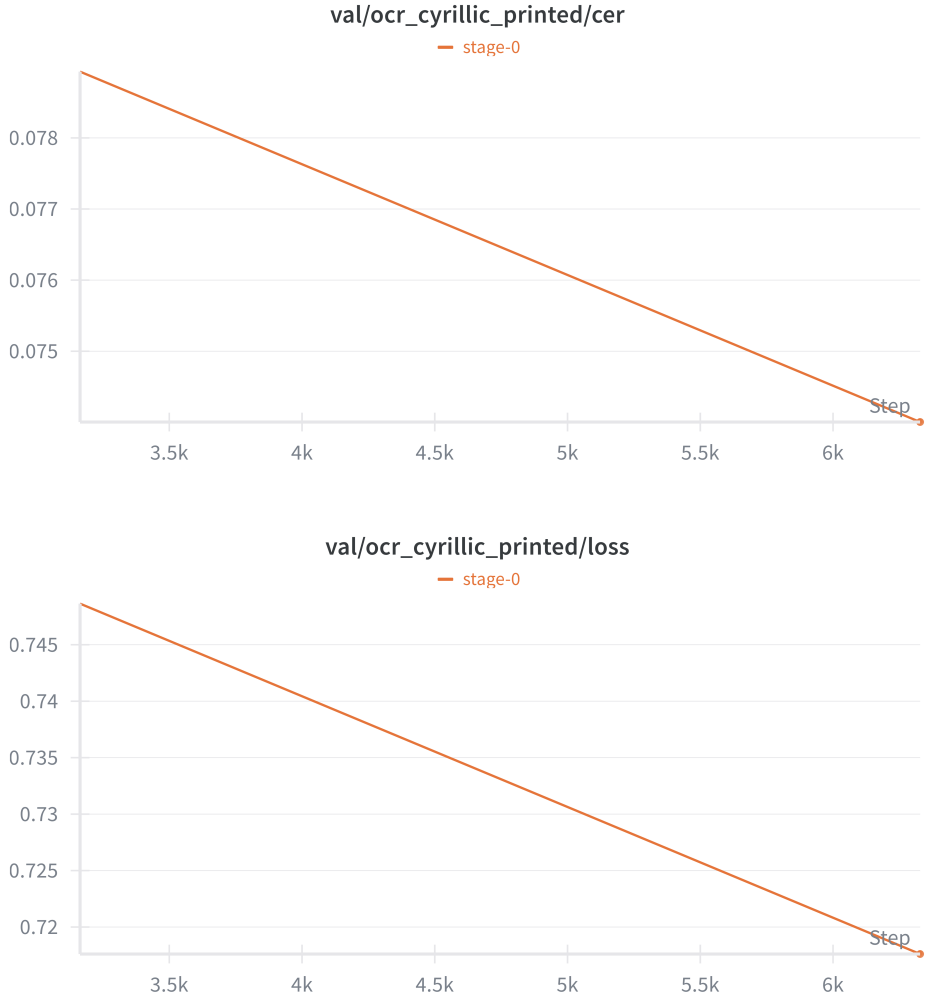


**Figure 7**
Phase 0 (*OCR Cyrillic Printed*): CER (top) and training loss (bottom).


**Figure 8**
Phase 1 (*OCR Cyrillic Printed*): CER (top) and WER (bottom).

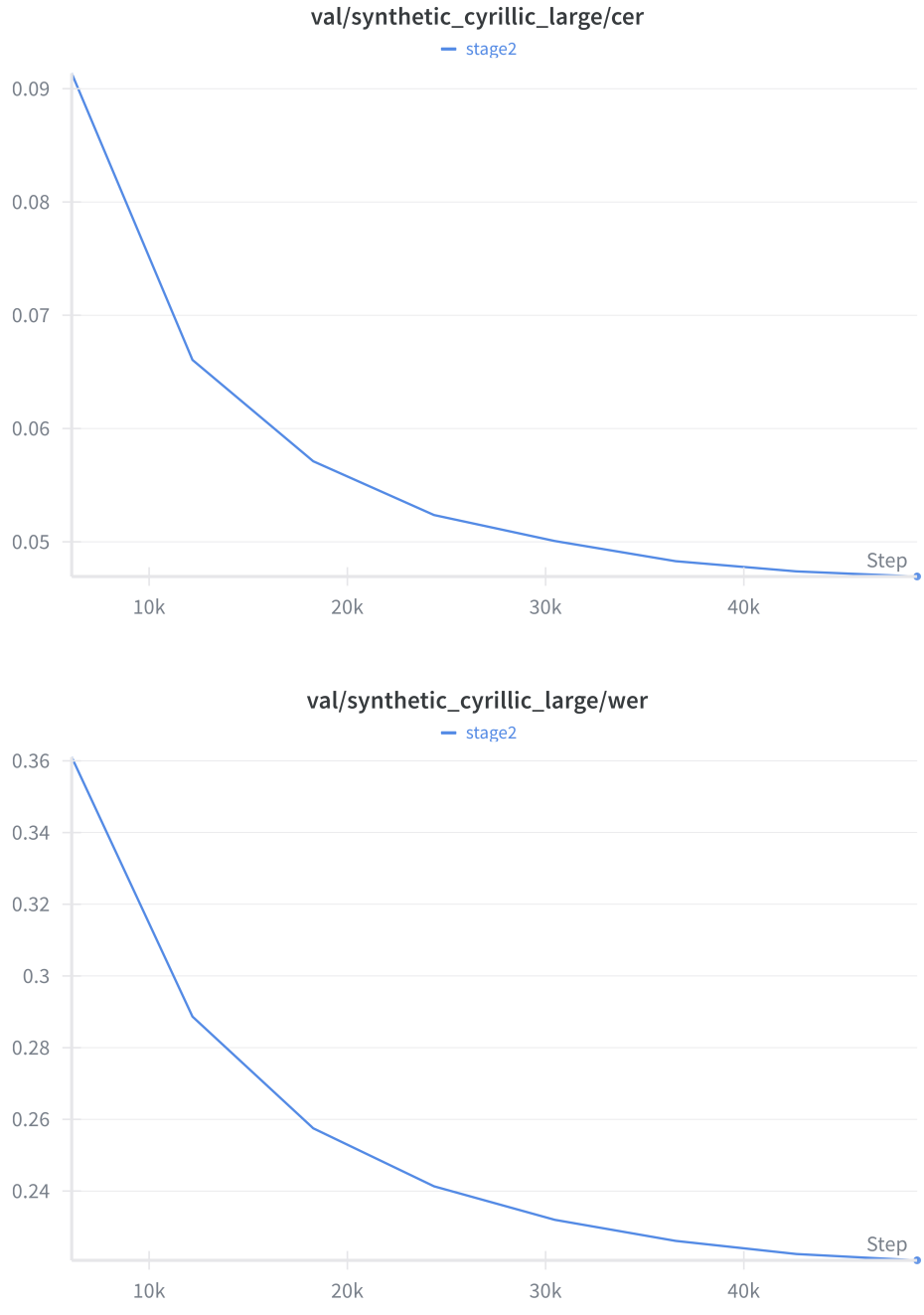


**Figure 9**
Phase 2 (*Synthetic Cyrillic Large*): CER (top) and WER (bottom).

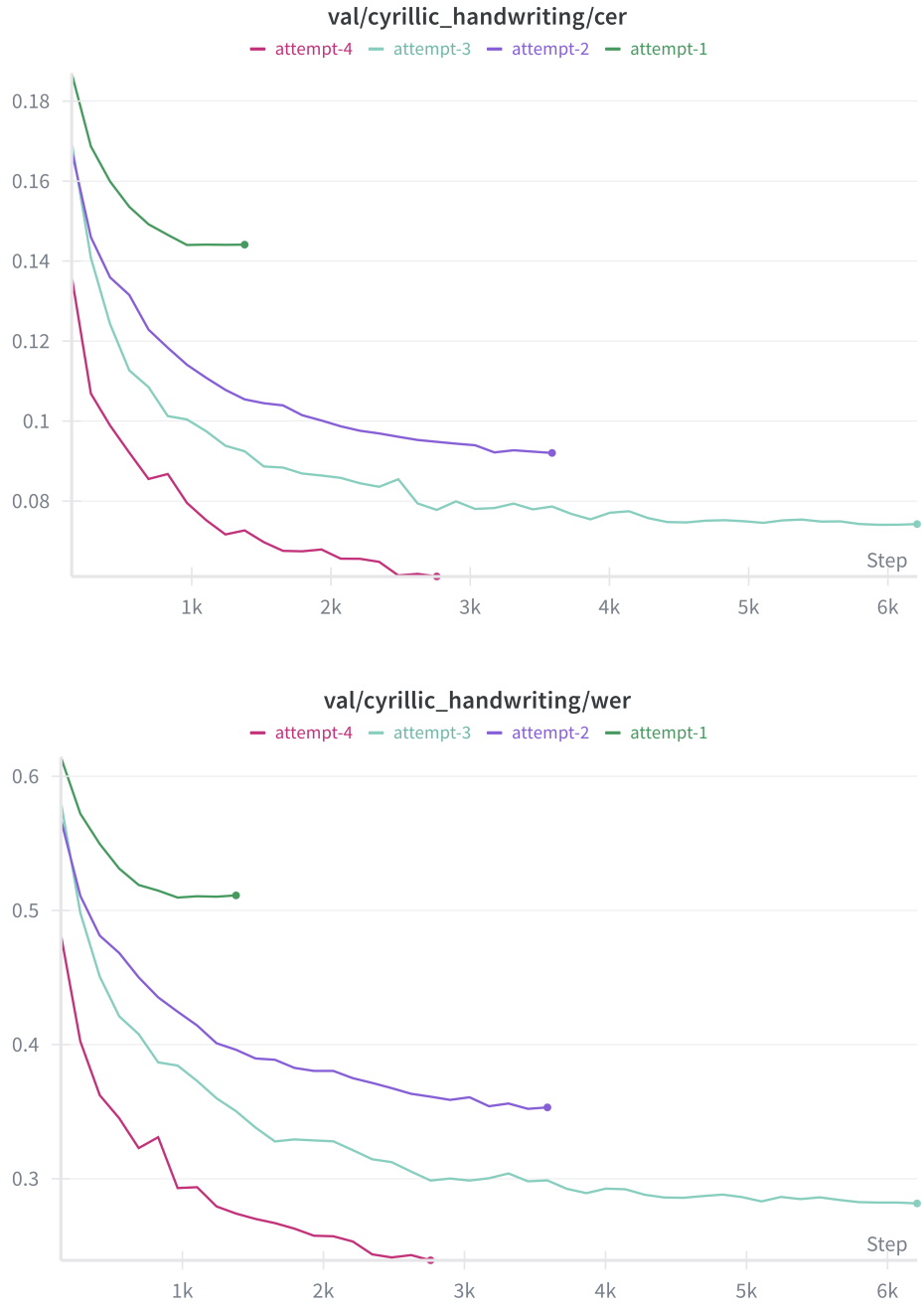


**Figure 10**
Phase 3, *Cyrillic Handwriting Dataset* validation split: CER (top) and WER (bottom).

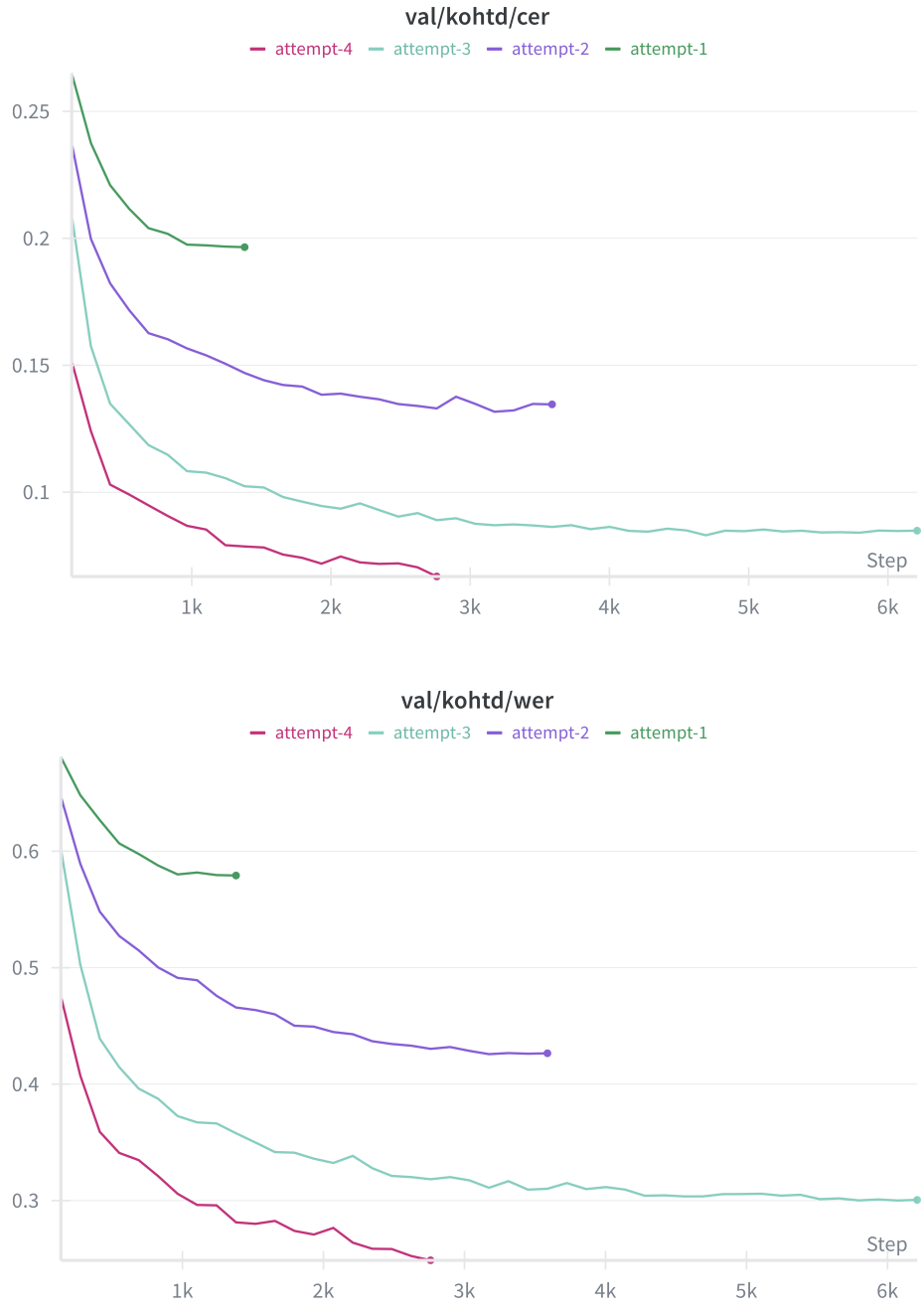


**Figure 11**
Phase 3, *KOHTD* validation split: CER (top) and WER (bottom).

## K. Preprocessing Effects on Printed Pages

The evaluation of individual preprocessing operations on printed test pages confirms that preprocessing should remain optional and configurable. Different algorithms address specific defects, such as blur, skew, or weak contrast.

| Algorithm | Doc. | Metric | Before → After (%) |
|---|---|---|---|
| Average Filter | 3070-5 | WER | 35.66 → 25.19 |
| Gaussian Filter | 3070-3 | CER | 11.76 → 6.49 |
| Hough Transform | 16-3 | CER | 10.69 → 4.77 |
| Projection Profile | 3070-4 | WER | 27.27 → 22.51 |
| Histogram Equalisation | 3070-3 | WER | 31.69 → 27.46 |
| CLAHE | 16-2 | WER | 21.68 → 20.28 |

**Table 25**
Representative best-case improvements from optional preprocessing on printed documents.

### K.1 Additional Architectural Comparisons

During the preliminary experiments, we compared the convergence and performance of several architectures before selecting PARSeq. The following figures illustrate these comparisons.

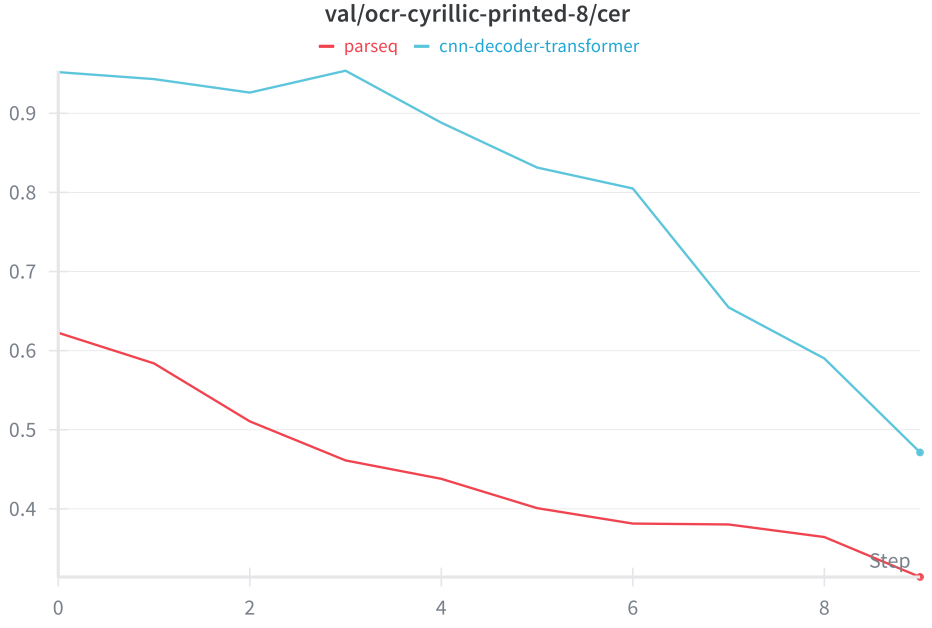


**Figure 12**
Preliminary experiments comparing PARSeq and CNN–Decoder Transformer on character error rate (CER).

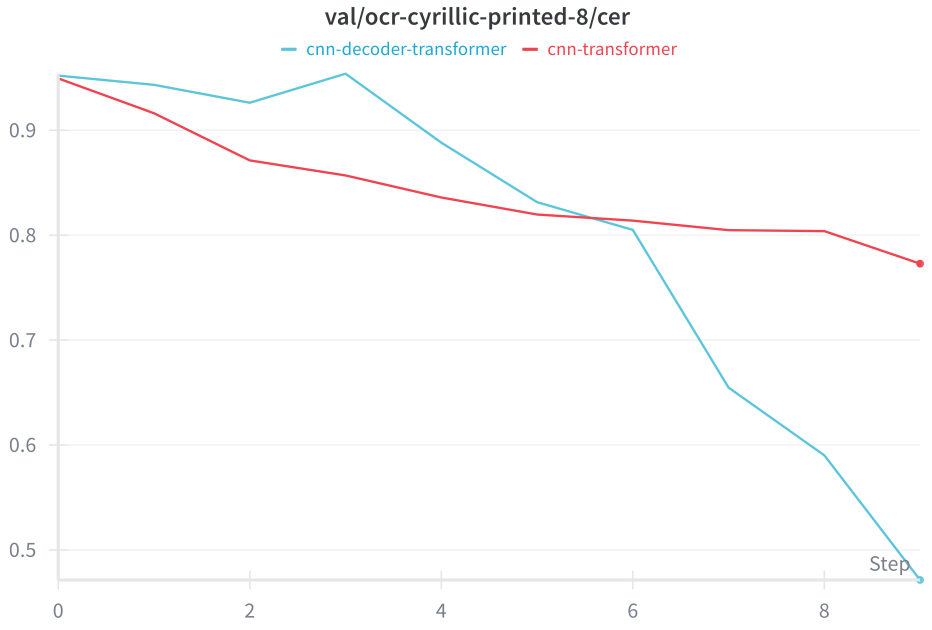


**Figure 13**
Preliminary experiments comparing CNN–Decoder Transformer and CNN–Transformer on character error rate (CER).