Reviving an Endangered Script: Optical Character Recognition for Syriac

Erick Angelo Ramirez
Stanford University
erick25@stanford.edu

Abstract

This project explores optical character recognition (OCR) for Syriac, a historically significant but underrepresented writing system. The task is divided into two subtasks: consonant detection/recognition and word detection/recognition. The dataset was constructed using real Syriac manuscript images as well as clean synthetic renderings, with extensive preprocessing and normalization. I implement and evaluate several deep learning models, including K-Nearest Neighbors (KNN), Convolutional Neural Networks (CNN), CRNN-LSTM, and Transformer-based architectures. For each model, I experiment with hyperparameter configurations and analyze performance using accuracy as the primary evaluation metric. My results show that while CNNs perform well on single-character recognition, Transformer-based models outperform CRNN-LSTMs on the more complex word-level task. This work demonstrates the feasibility of using modern deep learning methods for Syriac OCR and provides insights for further improvements in low-resource language processing.

1. Introduction

The Syrian language, which originates from early and medieval Christian communities in the region of modern-day Syria, is of significant cultural and historical importance. These communities maintained extensive records in Syriac manuscripts. Consequently, the ability to efficiently extract information from these manuscripts is crucial to expand our understanding of these communities.

Optical character recognition (OCR) is a valuable tool in this regard. OCR is a method for extracting text from images, offering a means of automating the transcription of Syriac manuscripts. However, the application of OCR to Syriac text introduces unique challenges.

A major challenge in developing Syriac OCR lies in the script's structure. Syriac is written from right to left and often features connected characters, making it difficult to isolate individual consonants. Consonants are represented by larger base characters, while vowels appear as smaller marks above them, requiring careful modeling of their spatial relationships. Additionally, Syriac manuscripts use three primary fonts—Serto, Estrangela, and East Syrian—each with distinct stylistic variations.

To tackle these challenges, I adopt a structured approach that breaks down the OCR task into a baseline and a proposed method. The baseline method focuses on Syriac detection and recognition at the character level. The proposed method focuses on Syriac detection and recognition for entire syriac words.

Before performing recognition, Syriac text must first be detected and localized within an image. This involves identifying the regions that contain individual characters or lines of text. Due to the handwritten and often degraded nature of Syriac manuscripts, text detection presents a significant challenge. Common issues include uneven line spacing, connected characters, ink bleed, and background noise. In this project, Syriac text detection was applied to both clean synthetic images and historical manuscript scans. The input to the detection system was an image containing Syriac text, and the output was a set of bounding boxes corresponding to either individual consonants or full words.

The input to the Syriac consonant recognition task is an image of a Syriac consonant. I use both a K-Nearest Neighbors (KNN) classifier and a Convolutional Neural Network (CNN) to output the predicted consonant class from a fixed set of 35 Syriac consonants. For full word recognition, the input is an image containing a sequence of 2 to 9 consonants. I explore two architectures: a CRNN-LSTM model trained with Connectionist Temporal Classification (CTC) loss, and a Transformer-based sequence-to-sequence model trained with cross-entropy loss. Both models output a predicted sequence of consonants representing the full word.

Across experiments, this project found that while the CNN consistently outperformed KNN in single-character classification, accuracy improved significantly with more training data and deeper architectures. In the word recognition task, the Transformer model outperformed the CRNN-LSTM model, achieving higher accuracy and generating more coherent predictions on multi-character input. These results suggest that attention-based architectures are better suited for modeling Syriac's positional and morphological complexity, especially in low-resource settings.

2. Related Work

Smith (2007) introduced Tesseract OCR, a widely used open-source OCR engine that has been adapted for multiple scripts, including Syriac [18]. While Tesseract has demonstrated effectiveness in recognizing Latin-based scripts, its application to Syriac remains limited due to the script's cursive nature and diacritic usage, necessitating specialized adaptations for improved recognition.

Majeed and Hassani (2024) presented a study on developing an OCR model for handwritten Syriac texts using the KHAMIS dataset, which consists of 624 handwritten East Syriac sentences from 32 contributors [11]. Their research fine-tuned the Tesseract OCR engine's Syriac model for handwritten text recognition, achieving significantly improved character error rates. This study highlights the importance of dataset creation and adaptation in Syriac OCR development, reinforcing the necessity of tailored datasets for enhancing recognition performance.

Chesley et al. (2024) evaluated Tesseract 4.0's OCR performance on different Syriac font types [4]. Their findings indicate that Tesseract achieves high consonantal accuracy for Estrangela (around 99%) but performs less reliably on Serto (89–94%) and East Syriac (89%). Their work underscores the need of further training and human revision for Serto and East Syriac, aligning with this project's focus on improving accuracy for these less-supported Syriac styles.

Effective text detection is a critical preprocessing step for any OCR pipeline, especially when dealing with complex or degraded images. Cho et al. (2016) discuss the Canny Text Detector, a fast and robust scene text localization algorithm inspired by the Canny edge detector [5]. By leveraging the structural similarities between edge patterns and text—such as spatial cohesion, stroke width, and color similarity—their method uses double thresholding and hysteresis tracking to detect cohesive character regions, improving recall compared to methods that rely heavily on high-confidence character classification. This is especially useful in multilingual or noisy visual environments. Complementing this, Ventzas et al. (2012) address the specific challenges of historical manuscript restoration through denoising and binarization techniques [21]. Their work highlights the importance of converting images to grayscale and applying adaptive thresholding to remove background textures and enhance character visibility. Together, these approaches inform the preprocessing pipeline for my Syriac text recognition model, which must contend with background noise, variable script morphology, and limited training data.

Bush et al. (2024) challenge the traditional binary classification of Syriac scripts into Estrangela and Serto [3]. Using a database of tens of thousands of Syriac letters from 96% of securely dated early Syriac manuscripts, their research demonstrates that this common categorization does not accurately reflect how early scribes actually wrote. By applying digital analysis and visual analytics, their study illustrates the need for a more nuanced classification system that better represents historical script variations. Their findings support the use of large datasets and computational methods in refining Syriac paleographic studies.

Kataria and Jethva (2022) address the challenges of OCR for Sanskrit manuscripts using a CNN-BLSTM architecture with CTC loss [9]. Their work focuses on overcoming issues such as overlapping lines, touching characters, and degraded inputs—common in historical Indian manuscripts written in complex scripts. While existing models primarily focus on isolated character recognition, their approach emphasizes robustness against real-world noise and script irregularities by leveraging sequence modeling with bidirectional LSTMs. Similarly, Mars et al. (2023) propose a hybrid architecture that combines a denoising generative adversarial network (DE-GAN) with a CNN-LSTM-CTC pipeline to improve OCR accuracy for Arabic script on images with colorful and noisy backgrounds [12]. The DE-GAN is used to clean the input image before sequence labeling. These works are particularly relevant to my project, which also employs a CNN-LSTM model with CTC loss to recognize Syriac verb sequences from images. Like Sanskrit and Arabic, Syriac poses challenges such as connected consonants and variation in script shape, making both denoising and sequence-aware architectures valuable comparative strategies for improving OCR performance in low-resource, historical scripts.

Transformer-based OCR models have shown strong potential for handling the complexity and variability of historical scripts. Ströbel et al. (2023) demonstrate that such models can generalize well across multilingual datasets with minimal training data, effectively recognizing both printed and handwritten texts—an ability particularly useful for digitizing diverse archival materials [19]. Building on this promise, Mostafaa et al. (2022) propose an end-to-end OCR framework for Arabic handwriting that eliminates convolutional backbones in favor of a Vision Transformer (BEIT) encoder paired with a standard transformer decoder [13]. Their model incorporates image enhancement, time-space optimization, and post-correction layers, achieving a 4.46% character error rate on a massive 270 million-word corpus with Classical Arabic diacritics. These studies highlight the adaptability and scalability of transformer architectures, offering a compelling path forward for a transformer-based Syriac OCR, where challenges such as morphological richness, diacritics, and stylistic variation closely parallel those in Arabic and other historical scripts.

3. Data Sources

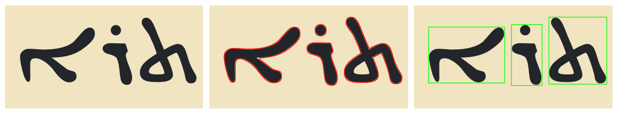
The Digital Analysis of Syriac Handwriting (DASH) project (2010–present) has contributed significantly to the digital study of Syriac manuscripts [16]. This interdisciplinary initiative integrates religious studies, computer science, and visual analytics to analyze Syriac script development and scribal practices. DASH consists of a large image database of 22,089 images of 35 distinct consonants (including subforms) found in 156 early Syriac manuscripts. This data that I scraped is used to train both the single-consonant classifier and the word classifier. Rather than relying on traditional font-based categories (i.e., Estrangela, Serto, and East Syrian), which Bush et al. (2024) argue do not accurately reflect early scribal practice [3], I adopt the DASH categorization system, which emphasizes stylistic variation. This more granular approach aligns with the evidence that early Syriac writing did not conform neatly to rigid script families and allows for more accurate consonant classification.

Moreover, I employ the verb conjugation paradigms available from Beth Mardutho: The Syriac Institute [1]. By scraping their website, I collected paradigms that span various conjugations based on specific verb forms, tenses, and personal-gender-number (PGN) combinations. I then rendered these conjugated forms using Beth Mardutho's Syriac fonts to generate labeled images for the purpose of developing single consonant detection methods. Then, for a more granular approach that would also be used as the input for the syriac word recognizer, I made synthetic images of these verb conjugations be merging the DASH single-consonant images together. These synthetic images, paired with their corresponding textual labels, are used to train the Syriac word recognition models. This dataset enables the model to learn not only individual character shapes, but also how they appear in context within longer textual units, accounting for variations in spacing and ligatures.

4. Methodology and Data Preprocessing

The development of an effective Syriac OCR model involves four key components, the first two being the baseline method and the last two being the proposed method. First, text detection for single Syriac consonants, ensuring that individual characters can be accurately located within manuscript images. Second, text recognition models are built specifically for these single consonants. Third, text detection for entire Syriac words, addressing the complexities of connected script and varied spacing. Finally, methods are explored for constructing robust text recognition models capable of identifying full Syriac words. This multistep methodology aims to address both the granular identification of individual characters and the broader challenge of recognizing meaningful textual sequences.

4.1. Text Detection for Single Consonants



(a) Input img of separated consonants. (b) Contours represented in red. (c) Text regions represented in green.

Figure 1: Steps for text detection for single consonants.

The first stage of the methodology focuses on detecting individual Syriac consonants [Figure 1]. Given a simple, noise-free image of Syriac script where the consonants are separated [Figure 1a], the detection process is as follows. The Canny edge detector is used to identify the edges within the manuscript image. This algorithm detects sharp changes in pixel intensity, which often correspond to the boundaries of individual characters. By applying this method, the primary structure of each character is effectively highlighted. These enhanced boundaries allow for improved contour detection. Contours are continuous curves that trace the outer boundaries of each character [Figure 1b]. By identifying these contours, the system can distinguish individual characters within the text. Finally, bounding boxes are drawn around each detected character [Figure 1c]. These rectangles define the text regions, simplifying the process of isolating characters for subsequent recognition.

4.2. Text Recognition for Single Consonants

Following text detection, the next step involves building effective recognition models for individual Syriac consonants. In constructing the datasets, the small dataset has 35 examples per consonant and the larger dataset has 160 examples per consonant. From these datasets, an 80-10-10 split was used for training, validation, and testing data.

All selected images required preprocessing. Since the original images were grayscale PNGs with an extra dimension for the transparent background, this dimension was removed, and the background was replaced with white pixels. Based on a random sample analysis, the largest images had dimensions of about 400 pixels in width and 200 pixels in height. The images were resized to 200x100 pixels. To prevent warping, each example was resized proportionally to fit within the new dimensions, with white padding added as needed [Figure 2]. The resulting grayscale images were then normalized so that pixel values fell between 0 and 1.

To recognize individual Syriac consonants from image inputs, this project implemented and evaluated two models: a K-Nearest Neighbors (KNN) classifier and a Convolutional Neural Network (CNN).

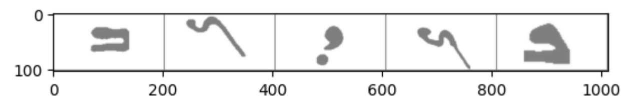


Figure 2: Each example image is resized, keeping its original aspect ratio, to fit in a 200x100 image. This resized image is then used for the training-validation-testing sets.

4.2.1 K-Nearest Neighbors (KNN)

The KNN algorithm is a non-parametric, instance-based learning method that classifies an input image $x \in \mathbb{R}^{H \times W}$ based on the majority label among its k closest neighbors in feature space. For this experiment, the feature vectors were extracted by flattening grayscale image data. The model was evaluated on both small and large training samples. For the smaller dataset, I tested $k \in \{2, 3, 4, 5\}$ and selected the best-performing k to use in the larger dataset to minimize training time, as KNN scales poorly with dataset size. The predicted labels were compared to ground truth labels, and the loss was computed using 0–1 loss, where the loss is 0 if the prediction is correct and 1 otherwise. The overall accuracy, calculated as the number of correct predictions divided by the total number of examples, was used as the evaluation metric

4.2.2 Convolutional Neural Network (CNN)

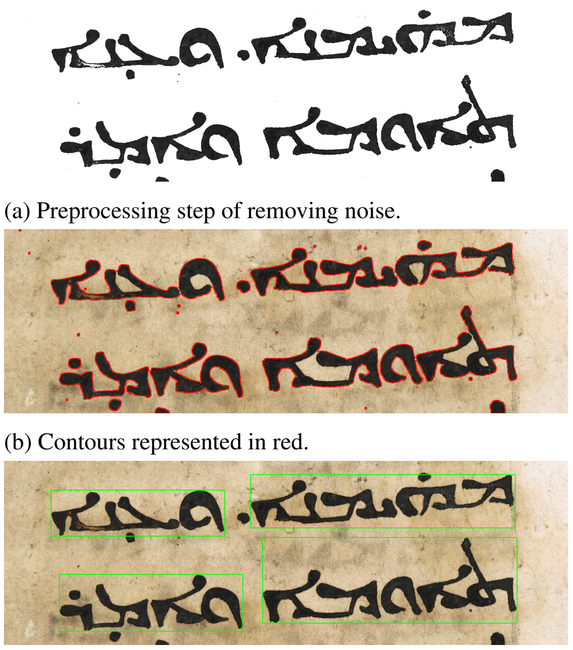
The CNN model takes input images $x \in \mathbb{R}^{C \times H \times W}$ and outputs logits $f(x) \in \mathbb{R}^K$ over K classes. The model was trained to minimize the standard multi-class cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{K} y_i \log \hat{y}_i,$$

where $\hat{y}_i$ is the softmax-normalized output of the network and y_i is the one-hot encoded ground truth label. The Adam optimizer was used to minimize this loss and update the network's weights during training.

The CNN architecture consists of two convolutional layers followed by max pooling, and then three fully connected layers. Specifically, the model includes:

- `Conv2d(3, 6, kernel_size=5)` → ReLU → `MaxPool2d(kernel_size=2)`
- `Conv2d(6, 16, kernel_size=5)` → ReLU → `MaxPool2d(kernel_size=2)`
- Flattening of spatial features
- Fully connected layers: `Linear(16544, 120)` → ReLU → `Linear(120, 84)` → ReLU → `Linear(84, 35)`



(a) Preprocessing step of removing noise.

(b) Contours represented in red.

(c) Text regions represented in green.

Figure 3: Steps for text detection for words.

The output dimension of 35 corresponds to the number of consonant classes in our label set. Training was conducted using a batch size of 5. For the small dataset, the model was trained for 2 epochs. For the larger dataset, I trained the model for both 2 and 10 epochs to examine the impact of extended training.

4.3. Text Detection for Words

Given a manuscript image containing Syriac script where the consonants are connected [Figure 3], the detection process follows the same steps as the single-consonant detection process but with an added preprocessing step to remove noise. The manuscript image is first converted to grayscale, and noise pixels above a specified threshold are set to white [Figure 3a].

Following noise reduction, the process continues with Canny edge detection to identify sharp intensity changes. Contours are subsequently detected to trace text outlines [Figure 3b]. Finally, bounding boxes are drawn around detected text regions, and overlapping or closely aligned text boundaries are merged to ensure entire words are correctly grouped for recognition [Figure 3c].

4.4. Text Recognition for Words

Once text boundaries are identified, these regions can be fed into a Syriac recognition model. To obtain the data, I used the DASH dataset and programmatically merged individual Syriac consonants to form real verb conjugations.

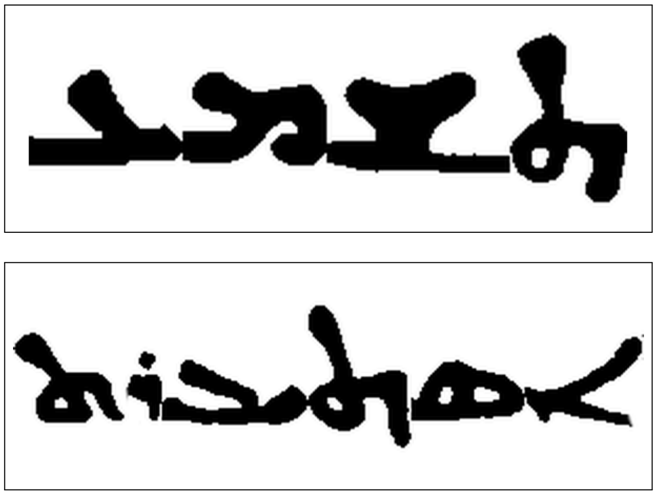


Figure 4: Example input images for the Syriac word recognition models.

I limited the scope to geminate, hollow, and strong verbs, ensuring linguistic coherence. Each resulting image represents a complete conjugation and measures 400×200 pixels to accommodate words ranging from 2 to 9 consonants in length [Figure 5]. Images were normalized to standardize pixel values, and any samples containing visual errors were removed. To prevent overlap between training and evaluation sets, I split the data by verb stem into training (82.02%), validation (8.82%), and testing (9.16%) sets, resulting in 2,864 training examples, 308 validation examples, and 320 test examples—each drawn from distinct conjugations.

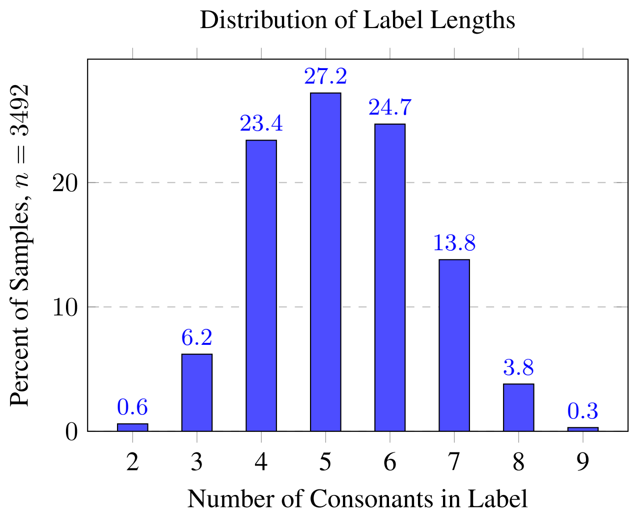


Figure 5: The number of consonants in the data samples range from 2 to 9, inclusive. Therefore, max length parameters for model architectures are set to 9. The majority of samples have 4 to 6 consonants.

To recognize entire Syriac word images containing up to nine consonants, I implemented and evaluated two distinct model architectures: a Convolutional Recurrent Neural Network-Long Short Term Memory (CRNN-LSTM) model trained with Connectionist Temporal Classification (CTC) loss, and a Transformer-based encoder-decoder model trained with cross-entropy loss. Both were trained with the Adam optimizer. Both models were hyperparameter tuned on a smaller data set with 50 examples before using the entire dataset.

4.4.1 CRNN-LSTM with CTC Loss

The first model follows the CRNN structure, which combines convolutional layers for visual feature extraction with recurrent layers for sequence modeling. The input image $x \in \mathbb{R}^{3\times H\times W}$ is first passed through a series of convolutional blocks that include convolution, ReLU activation, and max pooling. These layers extract local features while reducing spatial resolution. The resulting feature maps are flattened and reshaped to form a temporal sequence of length T.

This sequence is then passed to a multi-layer bidirectional LSTM, which captures dependencies in both forward and backward directions. The LSTM outputs are fed into a fully connected layer that projects each timestep to a distribution over the character vocabulary. Let $\mathbf{z}_t \in \mathbb{R}^{K+1}$ denote the logit vector at timestep t, where K is the number of Syriac consonants and the extra class corresponds to the CTC blank token.

I train the model using the CTC loss function:

$$\mathcal{L}_{\text{CTC}} = -\log P(y|x),$$

where $P(y|x)$ is the total probability of aligning the input sequence to the target label y over all valid alignments, and no explicit segmentation of characters is required. This makes it well-suited for unsegmented handwritten text.

4.4.2 Transformer Sequence-to-Sequence Model

The second model adapts a Transformer architecture for OCR by treating the task as an image-to-sequence translation problem. The input image is first encoded using a ResNet-50 backbone, from which I extract global image features via average pooling. These features are linearly projected into a d-dimensional embedding and treated as the encoder memory.

The decoder is a standard Transformer decoder consisting of N layers of multi-head self-attention and cross-attention. During training, the decoder takes as input a tokenized and padded ground truth sequence of length L, shifted by one position. Each token is embedded and combined with a sinusoidal positional encoding. The model is trained to predict the next token in the sequence using the cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{L} \log p(y_t | y_{<t}, x),$$

where y_t is the target character at position t and x is the encoded image. Special tokens including `<PAD>`, `<START>`, and `<END>` are added to the vocabulary to handle sequence boundaries and padding.

This model is capable of modeling complex dependencies and decoding characters autoregressively, making it suitable for handling a wide variety of word lengths and script complexities. Unlike the CTC model, it explicitly generates one token at a time, conditioned on previously predicted characters and the image representation.

5. Experiments

To evaluate the effectiveness of the models for Syriac character and word recognition, I conducted a series of experiments comparing traditional and deep learning-based approaches. The evaluation focuses on accuracy as the primary metric, defined as the proportion of characters correctly classified out of the total ground truth characters. Formally:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Characters}}$$

I report accuracy at both the individual character level and sequence level (i.e., across entire word images). All experiments were conducted using manually constructed image datasets derived from the DASH Syriac corpus, as detailed in the methods section.

5.1. Hyperparameter Selection

For all models, I used the Adam optimizer and experimented with learning rates ranging from 10^{-3} to 10^{-5} based on standard practice for deep learning models with CTC or cross entropy loss. CNN models for single-consonant recognition were trained with a batch size of 5 due to memory constraints and limited training examples. For the word recognition models (CNN-LSTM and Transformer), I used a batch size of 32 and selected hyperparameters through random sampling within a constrained range (e.g., dropout between 0.1 and 0.3, number of decoder layers between 2 and 6). Moreover, I maintained a strict 80/10/10 train-validation-test split to ensure that no overlapping conjugations appeared across data subsets.

5.2. Single Consonant Classification

I began by testing a KNN classifier, which served as a non-neural baseline. As shown in Table 1, the KNN model achieved 70.86% accuracy with $k = 3$ on the small dataset, but its performance plateaued with larger datasets due to its reliance on raw pixel similarity rather than learned representations. In contrast, the CNN baseline demonstrated clear gains from increased data and training time. Table 2 shows that accuracy improved from 2% to 80% when scaling from a small dataset trained for 2 epochs to a large dataset trained for 10 epochs.

Sample Size	K Value	Accuracy
Small	2	56%
Small	3	70.86%
Small	4	64%
Small	5	68.57%
Large	3	79.29%

Table 1: KNN accuracies per sample size and k value. The small sample size has 35 examples per consonant. The large sample size has 160 examples per consonant.

Sample Size	Batch Size	Num Epochs	Accuracy
Small	5	2	2%
Large	5	2	50%
Large	5	10	80%

Table 2: CNN accuracies per sample size, batch size, and number of epochs. The small sample size has 35 examples per consonant. The large sample size has 160 examples per consonant.

To better understand how individual characters were being classified, I analyzed per-consonant CNN performance [Table 3]. Consonants with unique or isolated shapes, such as Alaph and Lamadh-final, achieved near-perfect accuracy (96.9%), while visually similar or variably shaped characters such as Shin and Taw-looped saw significantly lower accuracies (62.5% and 56.2%, respectively). This suggests that errors often stemmed from subtle differences in stroke width or final/medial positioning, which are difficult to resolve in pixel space alone.

5.3. Word Recognition

I next evaluated two sequence recognition models for predicting entire Syriac word images. The first was a CRNN-LSTM architecture that combined convolutional layers with a bidirectional LSTM and was trained using CTC loss. Table 4 reports accuracy across sampled hyperparameter configurations. The best-performing CRNN model achieved 17.31% accuracy on a small dataset and 2.64% on the large dataset. The low performance on the larger dataset is likely due to overfitting on early epochs and the inherent difficulty of aligning long sequences with-

Consonant Name	Script	Accuracy
Alaph (Angular)		96.9%
Lamadh (Final, closed)		96.9%
Nun (Final, connected)		96.9%
Waw		96.9%
Yudh (Stand-alone)		96.9%
Zain		96.9%
Gamal		90.6%
Nun		90.6%
Pe		90.6%
Rish (Round)		90.6%
Qaph		87.5%
Ayin		84.4%
He (Angular)		84.4%
He (Round)		84.4%
Lamadh (Final, open)		84.4%
Teth		84.4%
Beth		81.2%
Heth		81.2%
Kaph		81.2%
Mim		81.2%
Mim (Final)		81.2%
Semkath		81.2%
Taw (L-shaped)		81.2%
Dalath (Angular)		78.1%
Kaph (Final)		78.1%
Nun (Final, unconnected)		78.1%
Dalath (Round)		75.0%
Rish (Angular)		75.0%
Taw (Triangular)		71.9%
Alaph (Round)		68.8%
Sadhe		68.8%
Shin		62.5%
Lamadh		59.4%
Taw (Looped)		56.2%
Yudh (Connected)		40.6%

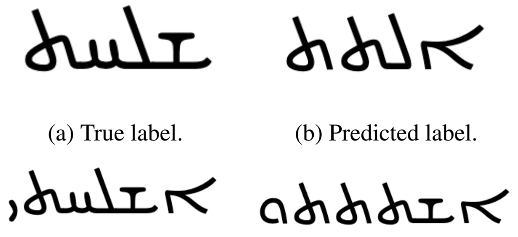
Table 3: CNN large sample accuracies per consonant.

out sufficient regularization.

The second model was a Transformer-based encoder-decoder. This model used a ResNet-50 encoder to extract global image features, and a Transformer decoder to autoregressively predict a sequence of consonants, trained with cross-entropy loss. As shown in Table 5, the Transformer outperformed the CRNN-LSTM model on the small dataset, reaching 26.33% accuracy. Moreover, the large dataset reach an accuracy of 43.56%.

5.4. Qualitative Observations and Failure Modes

I observed a number of recurring qualitative error patterns across models. At the character level, confusion often occurred between morphologically similar consonants (e.g., Dalath vs. Rish, Taw variants), especially when characters were connected or appeared in final forms. At the word level, the CRNN-LSTM model struggled with correct alignment, often predicting repeated characters or collapsing sequences prematurely. The Transformer model frequently made valid character predictions out of order or similarly mistook consonants with similar characteristics [Figure 6].


(a) True label. (b) Predicted label.

(c) True label. (d) Predicted label.

Figure 6: True labels and their respective predicted label from the Transform model. (b) is the predicted label of (a). (d) is the predicted label of (c). Both sets have an accuracy of 50%—the first with 2 correct out of 4, and the second with 3 correct out of 6.

Example misclassifications showed that both models were sensitive to handwriting variation, especially when multiple consonants were connected with little inter-character spacing. Overfitting was a concern in smaller models, as training loss decreased steadily while validation accuracy plateaued early. To mitigate this, dropout and early stopping were used in subsequent runs, although further regularization (e.g., data augmentation or synthetic expansion) remains a promising future direction.

While the CNN baseline performed robustly on single-character classification, sequence models require more data and more refined architecture tuning to generalize well on the Syriac verb corpus. The results underscore the importance of dataset size and character morphology in developing OCR systems for historical scripts.

Sample Size	Hidden Size	Num Layers	Learning Rate	Batch Size	Num Epochs	Accuracy
Small	256	3	0.01	32	15	17.31%
Small	512	2	0.0001	64	15	12.24%
Small	512	3	0.01	16	15	11.76%
Small	512	3	0.01	32	15	12.76%
Small	256	1	0.001	16	15	10.20%
Large	256	3	0.01	32	4	2.64%

Table 4: Hyperparameter configurations and accuracy results for the CRNN-LSTM model. The table displays combinations with the highest accuracies through random sampling of the hyperparameter space. The small sample size has 50 examples: 40 for training and 10 for validation. The large sample size has 3,492 examples: 2,864 for training, 308 for validation, and 320 for testing.

Sample Size	Model Dim	Num Heads	Num Decoder Layers	FeedForward Dim	Dropout	Learning Rate	Num Epochs	Accuracy
Small	256	4	3	1024	0.1	0.0001	5	26.33%
Small	256	4	3	1024	0.2	0.0001	5	24.04%
Small	256	8	3	1024	0.2	0.0001	5	23.28%
Small	512	4	3	1024	0.1	0.0001	5	21.42%
Small	256	4	3	1024	0.1	0.001	15	15.19%
Large	512	8	6	2048	0.1	0.0001	10	19.04%
Large	256	4	3	1024	0.1	0.0001	10	43.56%

Table 5: Hyperparameter configurations and accuracy results for the Transformer model. The table displays combinations with the highest accuracies through random sampling of the hyperparameter space. The small sample size has 50 examples: 40 for training and 10 for validation. The large sample size has 3,492 examples: 2,864 for training, 308 for validation, and 320 for testing.

6. Conclusion

This study highlights the effectiveness of KNN and CNN models in recognizing Syriac consonants. The KNN model performed relatively well with smaller datasets but lacked scalability, while the CNN model demonstrated significant improvements when trained with larger datasets and more extensive training periods. This study also demonstrates that Transformer Sequence-To-Sequence models outperform CRNN-LSTM models for Syriac recognition. Despite both architectures struggling with limited data, Transformers achieved higher overall accuracy and produced more coherent predictions on multi-consonant word images. These results suggest that attention-based models are better suited for capturing the structural and positional complexity of Syriac script.

A valuable direction for future work is the development of Handwritten Text Recognition (HTR) models for Syriac. Unlike OCR methods that focus on printed texts in traditional fonts, HTR systems address the challenges presented by modern handwritten Syriac. Modern handwriting differs significantly from ancient calligraphic styles, often appearing less uniform. Resources like the KHAMIS dataset [11] along with other contemporary handwritten datasets could provide a strong foundation for training HTR models.

Lastly, further improvements to OCR model performance may involve grouping visually similar consonants—such as *Dalath* and *Rish* or the multiple stylistic variants of *He* and *Taw*—into shared classification buckets to reduce confusion and enhance recognition accuracy for characters with subtle visual differences. Additionally, more rigorous hyperparameter tuning—such as grid searches for learning rates, batch sizes, and model depths—may yield performance gains, especially given the long training times observed. Incorporating techniques like data augmentation, character context modeling, or ensemble methods could also help further improve accuracy, especially for degraded manuscript images.

The implications of this work extend beyond improving Syriac OCR models alone. By demonstrating effective recognition techniques for Syriac consonants and words, this study contributes to preserving and studying the rich cultural and historical manuscripts written in Syriac. Accurate text recognition can enable researchers to analyze, translate, and digitize ancient texts more efficiently, advancing research in history, theology, and linguistics. Furthermore, improved OCR techniques for Syriac may serve as a foundation for expanding OCR capabilities for other ancient, endangered, and low-resourced languages that share similar script complexities.

7. Contributions

All work for this project—including data preprocessing, model implementation, experimentation, and writing—was completed by the author. This project uses Pytorch [14], TorchVision [10], Tensorflow [2], Scikit-Learn [15], Python [20], BeautifulSoup [17], Pillow [6], NumPy [7], and Matplotlib [8].

References

[1] Beth mardutho: The syriac institute. https://sedra.bethmardutho.org/about/openapi. ⟨arabic⟩

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. ⟨arabic⟩

[3] K. Bush, M. Penn, R. J. Crouser, N. Howe, and S. Wu. Challenging the estrangela / serto divide: Why the standard model of syriac scripts just doesn't work. *Hugoye: Journal of Syriac Studies*, 21(1):43–80, 2018. ⟨arabic⟩ , ⟨arabic⟩

[4] E. Chesley, J. Marcantonio, and A. Pearson. Towards digital syriac corpora: Evaluation of tesseract 4.0 for syriac ocr. *Hugoye: Journal of Syriac Studies*, 22(1):109–192, 2019. ⟨arabic⟩

[5] H. Cho, M. Sung, and B. Jun. Canny text detector: Fast and robust scene text localization algorithm. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. ⟨arabic⟩

[6] A. Clark. Pillow (pil fork) documentation, 2015. ⟨arabic⟩

[7] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. ⟨arabic⟩

[8] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. ⟨arabic⟩

[9] B. Kataria and H. B. Jethva. Optical character recognition of indian language manuscripts using convolutional neural networks. *Design Engineering*, (3), 2021. ⟨arabic⟩

[10] T. maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016. ⟨arabic⟩

[11] A. Majeed and H. Massani. Ancient but digitized: Developing handwritten optical character recognition for east syriac script through creating khamis dataset. *arXiv preprint arXiv*, 2408(13631), 2024. ⟨arabic⟩ , ⟨arabic⟩

[12] A. Mars, K. Dabbabi, S. Zrigui, and M. Zrigui. Combination of de-gan with cnn-lstm for arabic ocr on images with colorful backgrounds. *Advances in Computational Collective Intelligence*, 1864, 2023. ⟨arabic⟩

[13] A. Mostafa, O. Mohamed, A. Ashraf, A. Elbehery, S. Jamal, A. Salah, and A. S. Ghoneim. An end-to-end ocr framework for robust arabic-handwriting recognition using a novel transformers-based model and an innovative 270 million-words multi-font corpus of classical arabic with diacritics. *arXiv preprint arXiv*, 2208(11484), 2022. ⟨arabic⟩

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Desmaison, A. Kopf, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8024–8035, 2019. ⟨arabic⟩

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. ⟨arabic⟩

[16] M. Penn. Digital analysis for syriac handwriting. https://dash.stanford.edu/. ⟨arabic⟩

[17] L. Richardson. Beautiful soup documentation. *April*, 2007. ⟨arabic⟩

[18] R. Smith. An overview of the tesseract ocr engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2:629–633, 2007. ⟨arabic⟩

[19] P. B. Ströbel, T. Hodel, W. Boente, and M. Volk. The adaptability of a transformer-based ocr model for historical documents. *Digital Studies in Language and Literature*, 1(1), 2023. ⟨arabic⟩

[20] G. Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020. ⟨arabic⟩

[21] D. Ventzas, N. Ntogas, and M.-M. Ventza. Digital restoration by denoising and binarization of historical manuscripts images. *INTECH Open Access Publisher*, 2012. ⟨arabic⟩