# Transcription of Historical Documents using Seq2Seq Models

**Anonymous ACL submission**

## Abstract

In this work, we apply a Seq2Seq architecture, originally developed for machine translation tasks, to the transcription of low-quality scanned machine-printed historical documents from the *Berrutti Archive*, a collection of about three million documents generated by the Uruguayan Armed Forces between 1968 and 1985, during the last civic-military dictatorship in Uruguay (1973-1985). We present an implementation of an encoder-decoder model with an attention mechanism, alongside techniques for performance enhancement such as image augmentation and beam search, leveraging manual transcriptions as training data. Enhancement techniques improve our baseline CER by 16.3%, while our reported LCS is 42% higher than previous works for similar dataset. These results demonstrate the potential of carefully optimized Seq2Seq models with attention mechanisms in improving OCR tasks for challenging historical datasets.

## 1 Introduction

The *LUISA*[1] project (Etcheverry et al., 2021) aims to transcribe digitized documents from the *Berrutti Archive*, a significant collection of nearly three million scanned pages from the last military civic dictatorship in Uruguay (1973-1985), digitally scanned from microfilm reels. This archive includes press clippings, interrogation reports, personal records, and various other documents deemed useful by security agencies, sourced from heterogeneous origins.

This archive is an invaluable resource for uncovering information related to state terrorism and human rights violations that remain inadequately documented. LUISA is part of CRUZAR[2], an initiative that aims to develop tools and methodologies to facilitate analysis and research over historical documents from that period.

Manual transcription of these documents is impractical due to the immense volume and poor quality of the scans. To address this challenge, *LUISA* features a collaborative platform that enables volunteers to transcribe segments of the digitized images. This process generates a substantial annotated dataset, which is essential for training automated transcription models, specifically Optical Character Recognition (OCR) systems.

Through the efforts of LUISA platform volunteers, over 1500 documents have been fully transcribed. Although this number is relatively small compared to the total in the *Berrutti Archive*, it provides a critical resource for automation purposes. LUISA has previously applied standard OCR tools, such as Tesseract, to these documents, and the need for a more tailored approach has become evident. The transcriptions created by volunteers serve as a corpus which can be utilized to train advanced automatic transcription models.

In this work, we present an implementation[3] of a Seq2Seq encoder-decoder architecture with an attention mechanism to address the transcription of historical documents from the *Berrutti Archive*. Additionally, we assess the impact of several performance enhancement techniques, including teacher forcing, vocabulary tuning, image augmentation, and beam search, on transcription quality. Given that standard augmentation libraries may not effectively handle the specific noise types within our images, we developed custom augmentation methods using OpenCV and Numpy, allowing for greater control and flexibility in the augmentation process.

[1] *Leyendo Unidos para Interpretar loS Archivos* (Reading Together to Interpret the Archives)

[2] cruzar.edu.uy

[3] The code used for this work will be made available via git upon publication.

## 2 Related Work

The OCR field has evolved through various approaches, starting from traditional techniques such as heuristic feature extraction and statistical models (Trier et al., 1996) to more recent deep learning-based methods (Subramani et al., 2020; Memon et al., 2020). Early OCR systems relied heavily on manually designed features, which limited their ability to handle diverse handwriting and document layouts effectively (Borovikov, 2014). With the advent of deep learning, Recurrent Neural Networks (RNNs) became a popular choice for sequence modeling due to their ability to retain contextual information across timesteps (Elman, 1990). RNN-based OCR systems, particularly those using Long Short-Term Memory (LSTM) units, were capable of learning dependencies across longer sequences by mitigating the issues of vanishing and exploding gradients that affected earlier RNN architectures (Hochreiter et al., 2001).

The introduction of the Connectionist Temporal Classification (CTC) loss function (Graves et al., 2006) helped overcome the alignment problem, where the alignment between the input and output sequences were unknown, requiring models to determine the optimal alignment during training. CTC allowed RNNs to be trained end-to-end without requiring pre-segmented data, by interpreting the network output as a probability distribution over all possible output sequences conditioned on the input sequence. CTC-based approaches were successfully applied to OCR tasks, but they imposed constraints on the monotonic alignment between input and output sequences, which works well in cases where the sequences are generally monotonic (e.g., text follows the order of characters in an image or audio), but may not be enough for tasks where the output sequence length significantly differs from the input, or when non-monotonic relationships between input and output exist.

Seq2Seq models, initially proposed for machine translation (Cho et al., 2014; Sutskever et al., 2014), provided a more flexible approach for OCR by utilizing an encoder-decoder architecture that processes input sequences into a latent representation via the encoder and generates an output sequence through the decoder, allowing for variable-length transcriptions. This approach was further improved by incorporating attention mechanisms, which significantly improved the performance of machine translation tasks by enabling the model to focus on specific parts of the input during decoding (Bahdanau et al., 2015). The encoder-decoder framework and attention mechanism enabled Seq2Seq models to overcome limitations of CTC-based approaches, particularly in handling complex layouts and non-monotonic alignments.

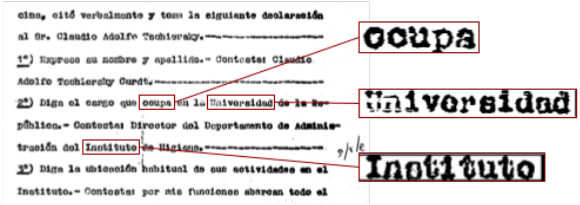


Figure 1: A digitized document with blocks containing individual words. Blocks that cannot be accurately transcribed by Tesseract v4.1 were assigned to human annotators via LUISA. Each block was labelled by at least three annotators, and heuristics were applied to combine them into coherent lines of text.

In the context of OCR for historical documents, several studies have explored the use of Seq2Seq models. Michael et al. (2019) proposed a Seq2Seq architecture with a hybrid attention mechanism for handwritten text recognition, achieving state-of-the-art results on challenging datasets. Other studies have highlighted the effectiveness of combining convolutional layers for feature extraction with bidirectional LSTM (Graves and Schmidhuber, 2005) based encoders and attention-based decoders for transcribing historical and degraded texts (Chowdhury and Vig, 2018; Poulos and Valle, 2021), .

More recent works have started utilizing Transformer models for OCR tasks. Diaz et al. (2021) explored different model combinations, including Transformer decoders, highlighting their effectiveness for text line recognition, while Li et al. (2021) achieved state-of-the-art results across various OCR domains by leveraging Transformer architecture for both image understanding and end-to-end text generation.

## 3 Dataset

The dataset utilized in this study comprises documents produced between 1968 and 1985 during the last civic-military dictatorship in Uruguay[4]. Discovered by the Ministry of Defence in 2006-2007, these documents were digitized into binary images. However, they present several challenges due to the quality of the digitized images, which suffer from

[4]The dataset is not publicly available due to privacy concerns. Further details in Section 10.

Figure 2: The LUISA crowdsourcing labelling interface. Users receive blocks of words along with a contextual image to assist in their transcription tasks.

| Split | N. examples | N. of tokens |
|---|---|---|
| Train | 66900 | 23.0 ± 24.1 |
| Validation | 9600 | 23.3 ± 24.2 |
| Test | 19114 | 23.0 ± 24.4 |

Table 1: This table summarizes the dataset's splits, detailing the number of examples and tokens for each. High variability in images is noted, with average dimensions of 675.7 ± 710.6 pixels wide and 23.0 ± 24.1 pixels high.

age-related damage, poor conservation conditions, and distortions introduced during the digitization process. The text is in Spanish and contains numerous names, codes, and local jargon, particularly military terminology from that era. These factors significantly diminish the effectiveness of standard OCR tools on this collection of documents. Initial transcriptions were performed using off-the-shelf Tesseract v4.1, an open-source OCR tool (Kay, 2007). However, documents that could not be accurately transcribed were earmarked for manual labelling.

Before starting the labelling process, a preprocessing stage was conducted to enhance image quality, correct text orientation, and segment the images into smaller blocks containing individual words, as depicted in Figure 1. These segments were transcribed by anonymous volunteers as part of the LUISA crowdsourcing project (Etcheverry et al., 2021). Volunteers received these blocks along with contextual images to assist in their transcription efforts, as shown in Figure 2. Additionally, they were provided with a detailed user guide[5] that outlines how to label the instances, ensuring a self-service experience on the platform. Each block was labelled by at least three annotators, with the final transcription determined by a majority vote.

The labelling process described thus far is detailed in Etcheverry et al. (2021). The dataset we received included blocks of text with their transcriptions and corresponding page images, obtained from previously described process. From this, we constructed a corpus of text lines instead of individual blocks. We combined adjacent blocks that were close enough to form a complete line, similar to the reconstruction method outlined in Etcheverry et al. (2021). However, we enhanced this process by incorporating logic to handle multi-column formats present in several documents. This involved applying heuristics based on vertical alignment and proximity between blocks based on a distance threshold.

From these constructed lines, we generated the final dataset, which was then divided into three splits: training (70%), validation (10%), and testing (20%). Table 1 summarizes the dataset's properties, including the number of examples, number of tokens, and average image dimensions for each split.

## 4 Methodology

The OCR task is framed as a Seq2Seq problem where input is a sequence of pixels representing a binary image that contains a few words on it, and output is a sequence of characters corresponding to the transcription of the text present in the image.

### 4.1 Metrics

The metrics employed in this study serve to evaluate the model's performance in accurately transcribing text from images. A Negative Log Likelihood Loss guides the optimization process by assessing prediction accuracy during training. The Character Error Rate (CER) quantifies transcription errors, while the Longest Common Subsequence (LCS) measures structural similarity between predicted and target texts. Additionally, Soft CER and Soft LCS ensures the evaluation reflects true transcription quality without being overly punitive for case discrepancies. They are defined as follows:

- **Negative Log Likelihood (Loss)**: This loss function penalizes the model inversely based on the probability assigned to the correct class, summed over the predicted sequence. Longer sequences lead to larger losses.

- **CER**: Measures the fraction of incorrect characters in the predicted sequence, implemented as the Levenshtein distance. It is defined as:

[5] *Guia para uso de LUISA* (Guide for Using LUISA)

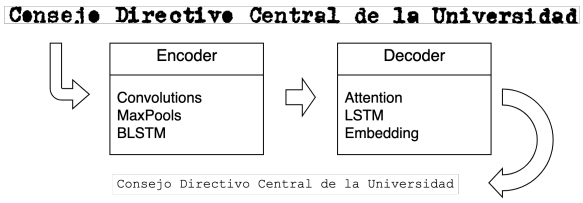


Figure 3: A high level view of the model, which receives a binary image interpreted as a sequence of pixels, and outputs plain text, as a sequence of characters. More details and low level description are given in Appendix A.

$$CER = \frac{I + D + R}{N}$$

where $I$, $D$, and $R$ are the counts of insert, delete, and replace operations, and $N$ is the character count in the largest alignment.

- **LCS ratio**: Measures the ratio of the longest common subsequence between two strings:

$$LCS_{ratio}(P, T) = \frac{2 * len(C)}{len(P) + len(T)}$$

where $P$ and $T$ are the predicted and target strings, and $C$ is their longest common subsequence. This metric is named as LCS for simplicity.

- **Soft CER** and **Soft LCS**: These metrics account for case insensitivity by converting both predicted and ground truth strings to uppercase before computing CER and LCS.

### 4.2 Model

The proposed architecture for the OCR task employs a Seq2Seq encoder-decoder framework that integrates Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, enhanced by an attention mechanism.

**Architecture Overview**

The architecture follows a sequence to sequence approach where the input is an image, interpreted as a sequence of pixels, while output is text which is a sequence of characters. Figure 3 shows an schematic simplification of the OCR model. Further details are included in Appendix A.

The model's inputs are binary single-channel images, preprocessed and then used to feed an Encoder: some convolutive layers followed by an Bidirectional Long Short-Term Memory (BLSTM) (Graves and Schmidhuber, 2005) that computes a context vector into an internal state. Once the image has been encoded, the model moves on to decoding the text output using an LSTM with an attention mechanism. The attention mechanism allows the model to weight all the information from the input, given the output produced up to the moment. On each step, a single character is predicted.

**Feature Extraction**

Based on Chowdhury and Vig (2018) and Michael et al. (2019), the initial extraction process begins with a CNN composed of two convolutional layers followed by max pooling, one additional convolutional layer, and another max pooling layer. This series of operations encodes the input image into a sequence of feature vectors, capturing spatial hierarchies and patterns within the image. Each feature vector corresponds to a vertical segment of the original image, preserving the order necessary for subsequent decoding.

**Encoder**

The sequence of feature vectors is fed into a BLSTM encoder, which processes the sequence in both forward and backward directions, generating a comprehensive representation of the input by capturing dependencies across the entire sequence of features.

**Attention Mechanism**

The attention mechanism employed is a content-based attention model (Bahdanau et al., 2015), which computes a context vector at each decoding step. This vector is derived from the encoder's hidden states, allowing the decoder to selectively focus on relevant parts of the input sequence based on the current decoding context.

**Decoder**

The decoding process is done by a unidirectional LSTM. At each time step, the decoder predicts the next character of the output sequence, informed by the context vector generated by the attention mechanism in the current step, together with the embedding of the previous predicted character and the previous context vector. The feed of the previous step context vector is a mechanism known as *input feeding* and is used to inform the model of past alignment decisions (Luong et al., 2015).

During training, prediction cycle repeats until an End of Sequence (⟨EOS⟩) token is reached or the number of predictions matches the ground truth sequence. During evaluation, predictions also stop at ⟨EOS⟩ but can extend to a maximum number

of predictions[6]. This limit mitigates information leakage from ground truth to validation, although allowing more predictions can inflate loss due to accumulating errors.

This Seq2Seq architecture effectively integrates CNNs for robust feature extraction, LSTMs for sequence modeling, and attention mechanisms to enhance the decoder's predictive capabilities, making it well-suited for OCR tasks.

### 4.3 Image Preprocessing

The input to the model consists of binary single-channel images, resized to a fixed height of 64 pixels while maintaining the aspect ratio, with a minimum width of 24 pixels. Images shorter than this width are padded with ones (representing the white background) to ensure uniform dimensions within each batch.

### 4.4 Training and evaluation process

Each model is trained for a fixed number of epochs using the Stochastic Gradient Descent (SGD) algorithm with the Adam variation (Kingma and Ba, 2015). After each epoch, the model is evaluated using the validation dataset to compute performance metrics. The model weights corresponding to the lowest CER are retained, ensuring that the best-performing instance is selected as the final trained model.

### 4.5 Performance Enhancement

In this section, we present the techniques implemented to enhance the performance of our model, including teacher forcing, vocabulary tuning, image augmentation, and beam search.

**Teacher Forcing**

During training, at each time step, the model may replace the predicted character with the actual character from the ground truth sequence, thereby enhancing learning efficiency (Bahdanau et al., 2015). This replacement occurs randomly for each character in the sequence, with the probability of replacement controlled by the hyperparameter `teacher_forcing`.

**Vocabulary Tuning**

We transitioned from a fixed char-set of 215 characters to a dynamic set of 132, retaining only those characters present in the training data. An out-of-vocabulary (⟨OOV⟩) token was introduced to handle unknown characters in validation or test sets. Upon examining infrequent characters, many were found to stem from labelling or encoding errors, particularly with accent marks; for instance, *ì* was used instead of *í*, and *ù* instead of *ú* in the following example:

*honestos de todas las condiciones sociales, de las convicciones políticas, sin ningún tipo de*

Errors related to incorrectly accented vowels were corrected, which not only reduced the overall vocabulary size but also improved the representation of accurately accented vowels. Additionally, characters appearing only once in the training set were replaced by ⟨OOV⟩.

**Image Augmentation**

Data augmentation is a widely adopted technique for artificially increasing the size of the training set, particularly in image processing tasks (Krizhevsky et al., 2012). This approach involves making minor modifications to images while ensuring that their corresponding labels remain valid. The main objective of data augmentation is to improve the model's generalization capabilities by exposing it to a more diverse set of inputs during training.

In our experiments, we used a comprehensive image augmentation strategy tailored to the unique characteristics and noise present in the dataset.

The following augmentation techniques were implemented, each with defined parameters to control their effects:

- **`random_horizontal_shift`**: Shifts the image horizontally, with maximum shifts set at 2.5% of the image width in either direction.
- **`random_vertical_shift`**: Vertically shifts the image, allowing for a 5% shift of the image height.
- **`random_zoom`**: Applies zooming in or out, constrained to 10% of the largest dimension of the image.
- **`random_rotation`**: Introduces small random rotations (up to ±1.5 degrees) to help address misalignments without significantly affecting the content of sequences.
- **`salt_and_pepper`**: Adds noise to the images by randomly altering up to 5% of the pixels, which can help the model learn to handle noise effectively.

[6] We set this maximum to 105 characters that corresponds to the average plus two standard deviations of sequence lengths in the dataset

- **median_blur**: Applies a median blur filter to reduce noise, selecting kernel sizes randomly: a size of 3 is chosen with a probability of 2/3, and a size of 5 is chosen with a probability of 1/3.

- **erode**, **dilate**, **open** and **close**: Those are morphological transformations. In this case, any of those operations are applied to the image, with kernel size of 3 or 5 that is picked at uniform random. Those transformations are really similar to the noise observed in the dataset.

- **no_op**: this is just no operation, and is a special case to let the chance that images are not modified at all.

The augmentations are organized into two groups based on their effects: the first group includes transformations that impact the entire image structure, such as shifts, zoom, and rotation; the second group consists of local transformations, such as blurring and morphological operations, which alter specific regions of the image. The implementation was designed to ensure that only one operation from the second group is applied per image to avoid conflicting effects.

This process is controlled by a single hyperparameter, that specifyies the fraction of training examples that will undergo augmentation. The hyperparameter ranges for each transformation were manually selected after a thorough examination of various cases, aiming to identify maximum and minimum values that maximize image distortion while ensuring the text remains accurately represented.

**Beam Search**

Beam search enhances sequence generation by evaluating multiple candidates at each decoding step, rather than selecting the single most probable token. At each step, it considers the best `beam_size` candidate tokens, computes their probabilities, and maintains the top `beam_size` for subsequent steps. This process continues until the ⟨EOS⟩ token is reached or a predefined maximum length is achieved.

The probability of each sequence is calculated as the product of its tokens' probabilities, which tends to decrease with longer sequences. To counteract this, probabilities are normalized by $\frac{1}{len(sequence)^{\alpha}}$, where $\alpha$ adjusts the penalty for longer sequences (Wu et al., 2016).

| Params | LCS | CER | T. loss | V. loss |
|---|---|---|---|---|
| lr=1e-4 tf=0.8 | 0.742 | 0.275 | **0.211** | 2.447 |
| lr=1e-4 tf=0.85 | **0.749** | **0.269** | 0.318 | 2.243 |
| lr=3e-4 tf=0.8 | 0.748 | 0.271 | 0.421 | **1.966** |
| lr=3e-4 tf=0.85 | 0.728 | 0.29 | **0.308** | 2.218 |

Table 2: This table shows results for different values of `learning_rate` (`lr`) and `teacher_forcing` (`tf`) hyperparameters, without any of the other proposed improvements, trained during 20 epochs.

### 4.6 Parameter Tuning

To determine an initial configuration, we performed a grid search on `learning_rate` and `batch_size` hyperparameters, excluding any of the proposed enhancements. The `learning_rate` was set to 5e-4, and the `batch_size` was initially determined to be 8 based on this preliminary validation. This configuration, with `teacher_forcing` set to 0.8 and run for 20 epochs is considered the baseline and reported in Table 6.

Each of the performance enhancement techniques discussed above was validated, and its hyperparameters were selected by modifying only the parameters specific to each technique at a time. Then, all of the techniques were combined into a single execution using the best hyperparameters found in each step. It is important to note that when changes validated independently are applied together, their effects may overlap; therefore, the fact that they work well collectively is not a trivial one.

## 5 Experiments

### 5.1 Teacher Forcing

The goal of this experiment is to obtain an appropriate value of `teacher_forcing` and `learning_rate` hyperparameters. The small grid presented in Table 2 was defined after executing for less epochs over a greater grid. Based on those results, the following experiments are run using this `teacher_forcing` = 0.85 and `learning_rate` = 1e-4 parameters.

### 5.2 Vocabulary Tuning

This experiment aims to validate the changes proposed for the vocabulary: we run two versions of

| Vocab. | LCS | CER | T. loss | V. loss |
|---|---|---|---|---|
| fixed | 0.713 | 0.305 | 0.647 | 2.082 |
| dynamic | **0.721** | **0.298** | **0.587** | **2.061** |

Table 3: This table shows the results of running the base model for 10 epochs under different settings with respect to the vocabulary being used, as the incremental improvements proposed over base. Fixed vocabulary corresponds to let the full charset from training + validation + test, dynamic refers to using only the characters present in the training set, fixing the accents and removing those with less than 2 occurrences, in this order.

| Aug. rate | Epoch | LCS | CER | T. loss | V. loss |
|---|---|---|---|---|---|
| 1.0 | 29 | 0.7643 | **0.2534** | 0.311 | **2.151** |
| 0.8 | 31 | **0.7644** | 0.2536 | **0.277** | 2.262 |
| 0.5 | 24 | 0.757 | 0.261 | 0.306 | 2.176 |
| 0.0 | 15 | 0.746 | 0.272 | 0.34 | 2.209 |

Table 4: This table presents the performance metrics at various augmentation rates. Higher rates lead to improved results and longer training epochs, suggesting that augmentation helps mitigate overfitting by consistently altering the training data. All experiments were run for 32 epochs.

the model, one that used a fixed vocabulary and another that dynamically adapted the vocabulary to the training set, reducing its size considerably. The impact of these modifications is summarized in Table 3. Results indicate a consistent improvement in performance metrics, with the best configuration achieving a CER of 0.298 for dynamic vocabulary.

### 5.3 Image Augmentation

This experiment varies the augmentation rate from 0 (no augmentation) to 1 (where all examples undergo augmentation pipeline). Table 4 summarizes the results obtained. The findings indicate a clear relationship between augmentation strength and model performance, supporting the hypothesis that higher augmentation rates lead to improved transcription accuracy. Additionally, the experiment was conducted over 32 epochs, revealing that higher augmentation rates often correspond to later optimal epochs. This is closely related to the tendency of the model to overfit during the early epochs when little or no augmentation is applied, while increased augmentation helps mitigate overfitting, leading to better performance at later stages. Consequently, when using image augmentation, extending the number of training epochs is advisable.

| `beam_size` | $\alpha$ | LCS | CER |
|---|---|---|---|
| 1 | - | 0.756 | 0.260 |
| 3 | 0.5 | 0.772 | **0.244** |
| 3 | 0.6 | 0.770 | 0.247 |
| 5 | 0.5 | **0.773** | **0.244** |
| 5 | 0.6 | 0.770 | 0.247 |

Table 5: This table shows the results for different configurations over 30% of the validation set randomly sampled without replacement. The first line, with beam size 1 corresponds to not using beam search, and is reported for comparison. As can be seen, all of them improves the results with respect to not using beam search. However, there is no major difference between `beam_size` of 3 and 5, and same thing is for $\alpha$.

### 5.4 Beam Search

After training the model, we select the weights corresponding to the best validation CER and run inference on this specific model instance using various `beam_size` and $\alpha$ parameters to identify the most suitable configuration.

For each configuration, only 30% of the validation set was used, randomly sampled, for two primary reasons. First, beam search is computationally expensive, especially with larger beam sizes, requiring significant memory and processing power. More importantly, since this technique is applied only during inference, results derived from the training set may not accurately reflect the model's performance on unseen examples. Using the full validation set could risk overfitting the hyperparameters to this dataset. By restricting our evaluation to a small fraction of the validation set, we can later trust the metrics obtained on the full validation dataset with beam search, ensuring that its parameters are not overfitted.

Our results in Table 5 indicate that using beam search significantly improves performance, achieving a CER reduction of approximately 6.1% compared to not using it. This improvement underscores the effectiveness of beam search in refining sequence predictions, making it a valuable component of our decoding strategy.

### 5.5 Integrated Improvements

In this section, we present the results of our model after collectively applying all previously validated improvements.

In summary, we use `teacher_forcing` with 0.85, `learning_rate` in 1e-4, do the vocabulary tuning and set `augmentation_rate` to 1. Finally,

| Split | Model | LCS | CER |
|---|---|---|---|
| validation | Baseline* | 0.74 | 0.284 |
| | Proposed | 0.781 | 0.236 |
| test | Baseline* | 0.74 | 0.282 |
| | Proposed | 0.781 | 0.236 |

Table 6: Evaluation over both validation and test sets using the proposed model with all enhancements, executed for 50 epochs. The Baseline* model was executed for 20 epochs with different values for `teacher_forcing` (0.85) and `learning_rate` (5e-4), without any of the proposed techniques.

beam search is applied with `beam_size`=3 and $\alpha$=0.5. This last experiment was run for 50 epochs, with `batch_size`=8.

Table 6 shows the results. As can be seen, putting all techniques together and training for more epochs considerably improved the results, reporting a CER of 0.236 in validation set: in Tables 2, 3, 4 and 5 the best CER achieved was 0.244, which implies an improvement of 3.3%.

## 6 Results

Table 6 presents the results for the validation and test datasets. The proposed model shows a 17% improvement in validation CER (from 0.284 to 0.236) and a 16.3% improvement in test CER (from 0.282 to 0.236) compared to the base model, which did not utilize enhancements such as vocabulary tuning, image augmentation, or beam search.

Additionally, the Soft CER for the proposed model on the test set was 0.228, indicating a 3.4 percentage point difference from the CER metric for the same dataset. This discrepancy is primarily due to case sensitive errors, which highlight both legitimate misclassifications by the model and inconsistencies in the annotation process. Further investigation is needed to assess the prevalence of these case-related annotation errors, which could guide future improvements in the training and evaluation phases.

In Etcheverry et al. (2021), the authors report applying Tesseract 4.1 to individual blocks of text from a dataset similar to ours. They then concatenate blocks aligned at the same height to obtain line transcriptions, achieving an LCS of 0.543. With the implementation of n-gram based post-processing, they improved this score to 0.550. Our proposed model achieves an LCS of 0.781, representing a 42% enhancement over their results.

## 7 Discussion

Since our dataset is not a public one, there is no a clear baseline to take as reference. Since the very first parameters we defined were `teacher_forcing` and `learning_rate` in Section 4.6, where image augmentation, beam search nor vocabulary tuning was considered, we can establish this as a reference value.

In Section 5.2 we can see that by tuning the vocabulary the CER decreased in 2%. This change is related to the quality of the dataset, and suggests that further improvements in dataset labelling quality may improve the quality of the results.

With respect to image augmentation, Section 5.3 we can conclude that the more augmented images, the better. In terms of CER, the usage of image augmentation reduced it in 6.8%. We set `augmentation_rate` to 1, meaning that all examples are processed by the augmentation pipeline. However, 1/25 of them are not modified: this evidence indicates that the decision of not to modify those images may be limitating its impact.

Finally, Section 5.4 exposes the improvements in the usage of beam search: it shows that CER is reduced in 6.1% when beam search is used.

## 8 Conclusion

In this work, we implemented a robust OCR architecture utilizing a Seq2Seq framework that integrates CNNs and LSTM networks, enhanced by an attention mechanism to extract plain text from digitalized historical documents. Our model processes binary images to produce text sequences, achieving strong performance metrics including a CER of 0.236 on the test set.

Key enhancements such as teacher forcing, vocabulary tuning, image augmentation, and beam search were systematically evaluated, demonstrating their individual and integrated effects on model performance. Notably, we proposed and implemented an augmentation pipeline tailored to this dataset, addressing its unique challenges. In fact, image augmentation turned out to be the enhancement that increased performance the most when evaluated in isolation.

## 9 Limitations

Due to Ethical Considerations outlined in Section 10, the dataset used in this work remains private as a way to preserve sensible information of affected individuals. To our knowledge, this dataset is the first

of its kind reported in the literature. Consequently, we lack an independent baseline for comparison, and our inability to publish a "shareable" version restricts the scientific community's capacity to validate our results and hampers reproducibility in future research.

Another limitation is the presence of noise and poor-quality labels in the dataset, as it was annotated by non-specialized volunteers. This noise has not been quantitatively assessed or studied in depth; this may affect our results. It is our understanding that a cleaner version of the dataset is being built through manual correction of the labels by specialized volunteers. Additionally, our approach did not include specific considerations for handling the noise present in the dataset.

All experiments were executed only once, which may affect results robustness and reliability. Single-run evaluations can introduce variability due to factors such as random initialization and dataset shuffling, limiting our ability to assess consistency and obscuring the true stability of outcomes. Future work should involve multiple repetitions of experiments to strengthen the validity of the findings.

Lastly, our approach to image augmentation retained a probability of 1/25 for instances undergoing no modifications. While higher augmentation rates correlate with improved model performance, allowing some images to remain unchanged may hinder the model's ability to fully benefit from the augmentation process. Although we aimed to maximize training diversity, unaltered images could dilute the impact of the augmentation pipeline. The transformations were also not evaluated in isolation, meaning some may not contribute significant value relative to their computational cost. Future experiments should explore enforcing modifications for all instances to ensure every training example contributes to learning.

## 10 Ethical Considerations

The dataset used in this study contains highly sensitive content, including interrogation reports, press clippings, personal records, and other documents related to security agencies. Many of these records likely stem from periods of political oppression, and some involve instances of torture. The presence of names, codes, and other identifiers adds a layer of complexity and sensitivity, as these documents are tied to individuals, places, and events that carry significant historical and emotional weight (Etcheverry et al., 2021).

Given the nature of this content, we recognize the ethical obligation to protect the privacy and dignity of individuals mentioned in the documents. Many of these individuals are still alive, while others remain missing, and all have families. They all have a right to privacy. Consequently, the dataset cannot be released publicly, as doing so would pose serious risks to the privacy and safety of those involved.

Furthermore, access to the dataset was granted under a Non-Disclosure Agreement (NDA), as the data owners retain legal and ethical responsibilities regarding its use. As we are not the owners of the data, we commit to handling it with the utmost care, ensuring that our research adheres to ethical standards while contributing to the understanding of this unique and sensitive subject matter.

## 11 Acknowledgements

The experiments presented in this paper were carried out using ClusterUY[7].

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA*.

Eugene Borovikov. 2014. A survey of modern optical character recognition techniques. *ArXiv*, abs/1412.4183.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar*, pages 1724–1734. Association for Computational Linguistics.

Arindam Chowdhury and Lovekesh Vig. 2018. An efficient end-to-end neural model for handwritten text recognition. In *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK*, page 202. BMVA Press.

Daniel Hernandez Diaz, Siyang Qin, R. Reeve Ingle, Yasuhisa Fujii, and A. Bissacco. 2021. Rethinking text line recognition models. *ArXiv*, abs/2104.07787.

[7] Site: https://cluster.uy

Jeffrey L. Elman. 1990. Finding Structure in Time. *Cognitive Science*, 14(2):179–211.

Lorena Etcheverry, Leopoldo Agorio, Virginia Bacigalupe, Sofía Barreiro, Elena Bing, Samuel Blixen, Daniel Calegari, Lautaro Cardozo, Fernando Carpani, Felipe Chavat, Diego Garat, Alvaro Gómez, Fabián Hernández, Víctor Marabotto, Guillermo Moncecchi, Ignacio Ramírez Paulino, Aiala Rosa, Jorge Tiscornia, Dina Wonsever, Gregory Randall, Guillermo Zorron, Lia Rivero, Nilo Patiño, Javier Stabile, Ernesto Fernández, Federico Fioritto, and Rodrigo Laguna. 2021. A computational framework for the analysis of the uruguayan dictatorship archives. In *Qurator 2021 - Conference on Digital Curation Technologies, Berlin, Germany*.

Alex Graves, 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, Pittsburgh, Pennsylvania*, pages 369–376, Pittsburgh, Pennsylvania. ACM Press.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*, pages 237–243. IEEE Press.

Anthony Kay. 2007. Tesseract: An open-source optical character recognition engine. *Linux J.*, 2007(159):2.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei A. F. Florêncio, Cha Zhang, Zhoujun Li, and Furu Wei. 2021. Trocr: Transformer-based optical character recognition with pre-trained models. In *AAAI Conference on Artificial Intelligence*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Jamshed Memon, Maira Sami, and Rizwan Ahmed Khan. 2020. Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr). *IEEE Access*, 8:142642–142668.

Johannes Michael, Roger Labahn, Tobias Grüning, and Jochen Zöllner. 2019. Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition. *arXiv:1903.07377 [cs]*. ArXiv: 1903.07377.

Jason Poulos and Rafael Valle. 2021. Character-based handwritten text transcription with attention networks. *Neural Comput. Appl.*, 33(16):10563–10573.

Nishant Subramani, Alexandre Matton, Malcolm Greaves, and Adrian Lam. 2020. A survey of deep learning approaches for ocr and document understanding. *ArXiv*, abs/2011.13534.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Øivind Due Trier, Anil K. Jain, and Torfinn Taxt. 1996. Feature extraction methods for character recognition-a survey. *Pattern Recognit.*, 29:641–662.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

## A Low level model definition

The proposed Seq2Seq model for OCR transcription, which can be seen in Figure 4 follows the standard encoder-decoder approach, utilizing both convolutional and recurrent networks, with an attention mechanism to aid the decoder in focusing on relevant input features at each step of prediction. Chowdhury and Vig (2018) and Michael et al. (2019) were the two main motivations for the proposed architecture.

**Encoder**

The encoder is composed of a three-layer CNN interleaved with two max-pooling layers, followed by three bidirectional LSTM (BLSTM) layers. The encoder architecture is defined as follows:

$$C_{4x2}^{6x4}[8] \rightarrow C_{1x1}^{6x4}[32] \rightarrow P_{2x2}^{2x2} \rightarrow C_{1x1}^{4x3}[64] \rightarrow P_{2x2}^{2x2}$$

$$\rightarrow R[64] \rightarrow R[256] \rightarrow R[256]$$

Where $C$ refers to convolutional layers, $P$ to max pooling layers, and $R$ to BLSTM layers. $C/P_{K_y x K_x}^{S_y x S_x}$ represents the kernel dimensions ($K$) and stride values ($S$) in the x and y axes. Between

brackets are the number of filters (or output channels) for the convolutional layers and the number of hidden units for each LSTM in the recurrent network. This configuration, together with fixing the input image height to 64 pixels (while maintaining the aspect ratio), ensures that the feature vectors extracted by the CNN maintain consistency in their receptive field dimensions, making each output vector represent features from a section of the input containing the complete vertical dimension.

**Attention Mechanism**

The attention mechanism enables the decoder to focus on relevant parts of the encoded sequence during transcription. Instead of a fixed context vector, it calculates a unique context vector for each time step by assigning weights to the encoder outputs based on their relevance to the current decoder state. This allows the model to selectively attend to specific inputs, improving accuracy.

The proposed architecture uses *content-based attention* (Bahdanau et al., 2015), implemented through a fully connected network that takes the decoder hidden state and encoder outputs to compute an energy vector. A softmax function then converts this energy vector into weights associated with each encoder output. This weights vector ($c_t$) is used to perform a weighted sum of the encoder outputs. Along with the previous attention vector ($\bar{c}_{t-1}$), it produces the new context vector ($\bar{c}_t$), which is fed to the decoder to predict the next symbol. Both the fully connected network for the energy vector and the network for combining context have hidden sizes of 512.

**Decoder**

The decoder is a unidirectional LSTM with a single layer and a hidden size of 256.

At each time step, the decoder takes the previous predicted symbol embedding, the context vector from the attention mechanism, and the previous decoder state. Additionally, the decoder employs *input feeding*, as described by Luong et al. (2015), meaning that the context vector from the previous time step is also used as an input to the LSTM. This helps the model maintain a better sense of local context, thereby improving the quality of the predictions.

A fully connected layer with a softmax activation function is used to generate a probability distribution over possible output symbols. The symbol with the highest probability is selected as the predicted character.

## B Computational Resources

The experiments were conducted on Cluster.uy, with the following hardware specifications:

- RAM: 32 GB
- Storage: 10 GB SSD
- GPU: NVIDIA Tesla P100 with 12 GB of GPU memory

During the early development of the model, it was optimized for a batch size of 8, as this was the maximum size that could be accommodated by the available memory. While implementing mixed precision allowed us to increase the batch size to 12, we ultimately maintained a batch size of 8 to achieve better performance metrics. This decision was supported by observations that training times remained unchanged, as the GPU was underutilized. The LSTM's code and decoding processes were executed for each batch instance on the CPU, alongside the image augmentation and beam search implementations.

| Layer Type | Number of Parameters |
|---|---|
| **Encoder** | |
| Conv2d (C1) | 200 |
| Conv2d (C2) | 6,176 |
| Conv2d (C3) | 24,640 |
| LSTM (L1) | 2,236,416 |
| **Total Encoder** | **2,267,432** |
| **Decoder** | |
| Embedding | 13,760 |
| LSTM (L2) | 2,232,320 |
| Linear (Output Layer 1) | 524,800 |
| Linear (Output Layer 2) | 220,375 |
| Linear (Output Layer 3) | 524,800 |
| **Total Decoder** | **3,516,055** |
| **Total Parameters** | **5,783,487** |

Table 7: Summary of the number of parameters per layer in both the encoder and decoder components of the model.

The encoder model contains close to 2.3 million parameters, while the decoder model contains around 3.5 million parameters, resulting in an architecture with a total of nearly 5.8 million parameters. A detailed breakdown of the parameter count per layer is provided in Table 7.

Based on logs from 70 training experiments, the average time per epoch was 78 minutes.

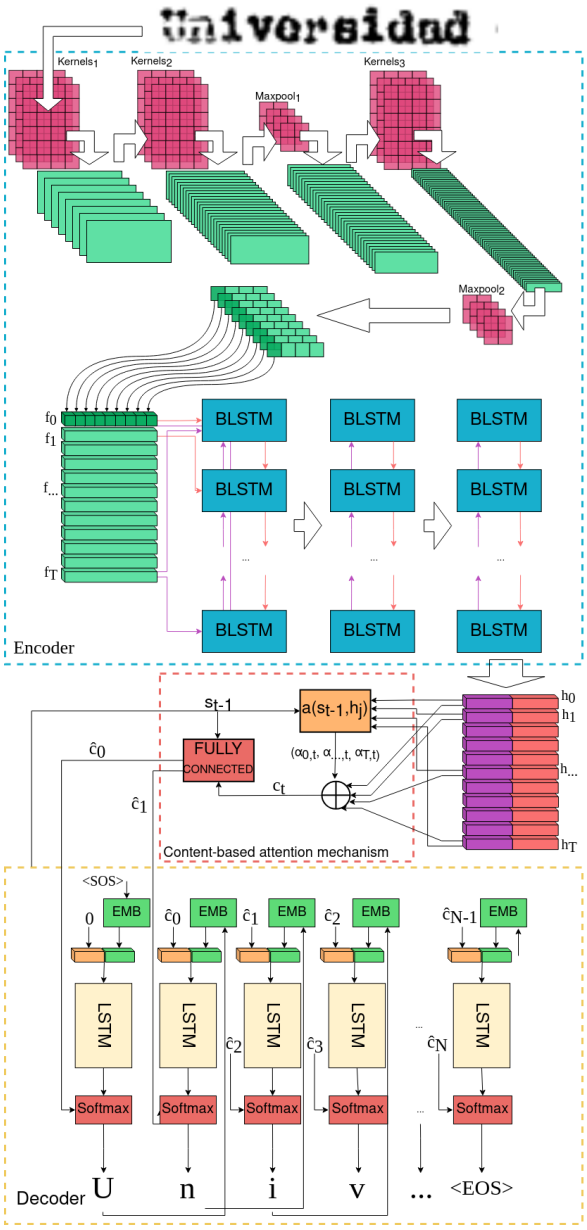


Figure 4: Architecture of the proposed model. The input image is first processed by the convolutional network, consisting of three convolutional layers and two pooling layers. The vectors $f_0, \ldots, f_T$ obtained, called feature vectors, are then processed by the bidirectional recurrent network with LSTM units. The state of the recurrent network is stored while processing each of the feature vectors, resulting in the state vectors $h_0, \ldots, h_T$. These state vectors are then processed by the attention mechanism, which generates the attention vector ($\overline{c}_t$) at each time step $t$ used by the decoder to predict the next symbol in the final sequence.