HTR

HANDWRITTEN TEXT RECOGNITION SYSTEM

IDATHA

MEMBER OF pyxis ECOSYSTEM

Zuhair Abbas

CONTENTS

INTRODUCTION

Handwriting recognition is a trending application of Artificial Intelligence with extensive research going on around it. It can be classified as a subfield of Optical Character recognition. Almost every major industry enjoys the perks of OCR. Still, there is tremendous space for improvement in this sphere. Expectations leading to more advanced needs will always surpass the quality of current results. This gap between the quality and expectations can only be filled by the constant research in the area of Computer Vision and Deep Learning.

Within OCR lies a field of Handwriting detection and recognition. Although computer-generated text can be easily recognized using various OCR tools available with remarkable accuracy, handwritten documents are usually hard to recognize because there are a number of factors involved apart from the text itself. The data might contain a lot of variation in terms of handwriting/text size/page texture, picture quality, and ink. In this project, we explore a State-of-the-art technology to solve a handwritten detection problem for a historical transcript written in Spanish.

In Complex cases like Handwriting recognition of Historical documents, most OCR tools/libraries miserably fail to detect the handwritten text. Offline Handwritten text recognition for Historical documents is considered the most challenging problem in this sphere due to:

- Cursive nature of handwriting
- Quality of pages and pictures
- Variety of character size and shape
- Large vocabulary.

This is where ICR (Intelligent Character Recognition) comes to the rescue. ICR comprises advanced Machine learning and Deep learning techniques. These techniques usually require a huge amount of continuous and stable data preferably coming from the same source. This data is then used to train complex models for text recognition. Further using these trained models for a different handwritten dataset with very high accuracy is still a hot topic of research that goes under the umbrella of Domain adaptation. In our project, we also utilize a subfield of domain adaption called Transfer learning. We will shed some light on it in the Training section.

LITERATURE REVIEW

A considerable amount of research has been carried out on Offline handwriting recognition in the last decade. We will briefly go through some of the well-known approaches.

Jose Carlos Aradillas and Juan Jose Murillo-Fuentes crafted a technique to boost Offline Handwritten Text Recognition in Historical Documents with Few Labeled Lines by combining the benefits of transfer learning and advanced data augmentation techniques. They proposed an algorithm to mitigate the effects of incorrect labeling in the training set. The method significantly reduces the required number of annotated lines. However, their network takes many days to train on large datasets.

Curtis Wigington and Chris Tensmeyer proposed Start, Follow, Read: An End-to-End Full-Page Handwriting Recognition pipeline using 3 separate models. It Jointly learns text detection, segmentation, and recognition using mostly images without detection or segmentation annotations. SFR exceeds the performance of the winner of the ICDAR2017 handwriting recognition competition. Nevertheless, it has some drawbacks like it requires a small number of images that have SOL, segmentation, and line-level transcription annotations. The overall pipeline is fairly complex to understand, and the best recognition results are still achieved by the systems working at the line level.

Adeline Granet and Emmanuel Morin utilized transfer learning techniques. Their goal was to train a recognition system without involving the annotation step. Also, make use of Transfer learning from heterogeneous datasets with ground truth and share common properties with a new dataset that has no ground truth. They used RCNN with CTC loss for training. The paper didn't explain how they used the text-line segmentation algorithm. Moreover, it requires a manual validation of the collected transcriptions of each block, each line, as well as the segmented lines themselves.

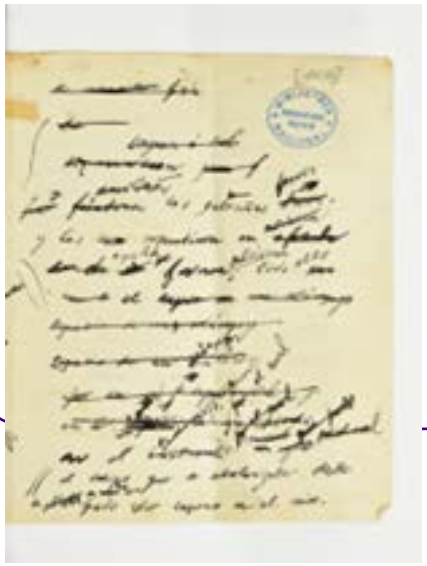
Edgard Chammas and Chafik Mokbel demonstrate how to train an HTR system with few labeled data. Specifically, we train a deep convolutional recurrent neural network (CRNN) system on only 10% of manually labeled text-line data from a dataset and propose an incremental training procedure that covers the rest of the data. Performance is further increased by augmenting the training set with specially crafted multi-scale data. They used a Contour based line segmentation approach that is not feasible for complex data. Therefore, more advanced segmentation algorithms are needed to improve the selection/training process, like the ones based on the Seam Carving technique. Also, the network architecture was very complex.

All these trending HTR approaches have something in common. They all use a combination of RNNs and CNNs and most importantly they use CTC loss for training. We will explore its benefits in the following section. In our project, we also utilized a CNN-RNN model along with CTC loss and also integrated the benefits of data augmentation techniques and transfer learning.

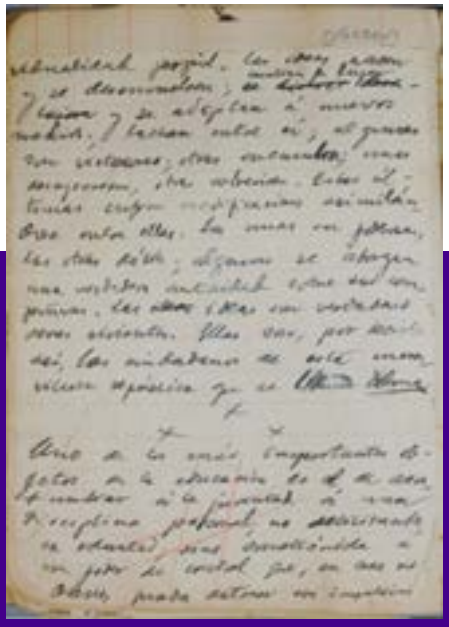
DATASETS

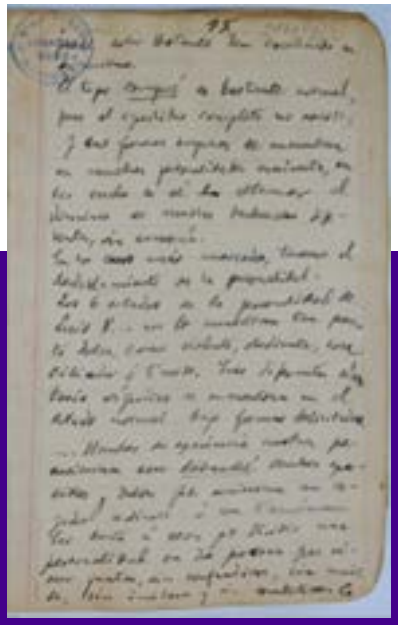
Our dataset consists of 40 pages taken from 2 different transcripts of Jose Enrique Camilo Rodo, a famous Uruguayan essayist. This dataset was provided by The National Library of Uruguay. Although the data was coming from the transcript of the same author, it was still one of the most complex data to be recognized due to a lot of variation in handwriting/text size/page texture and picture quality, and ink.

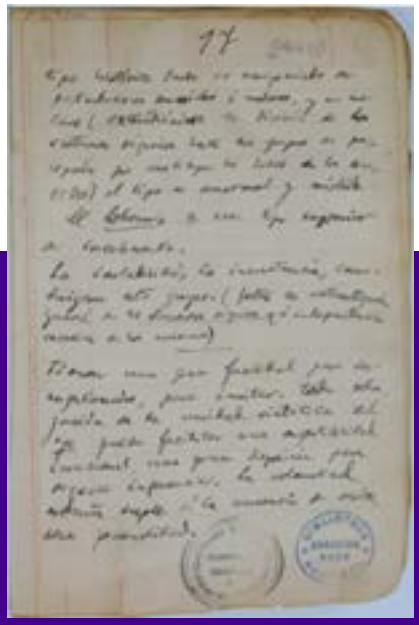


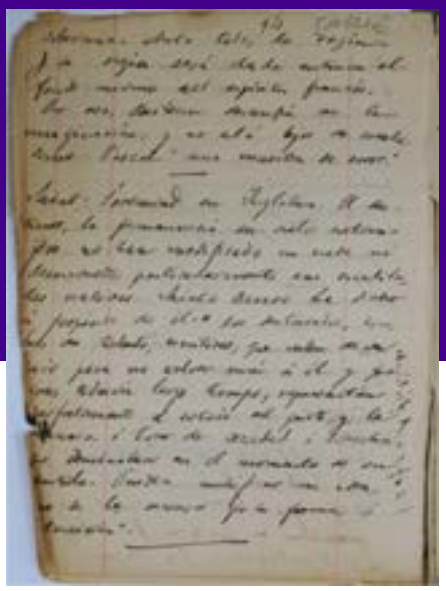



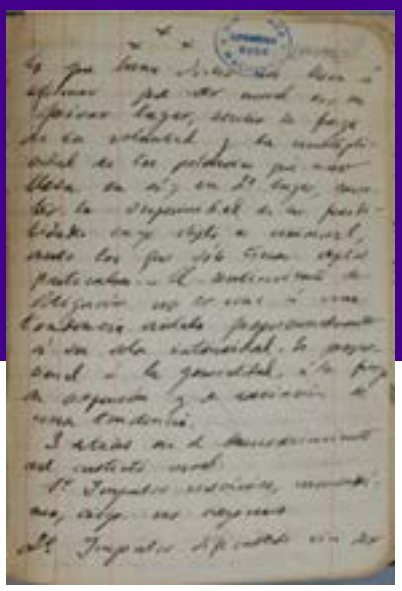
And of course, there were some readable images, from the other book, like the ones below:

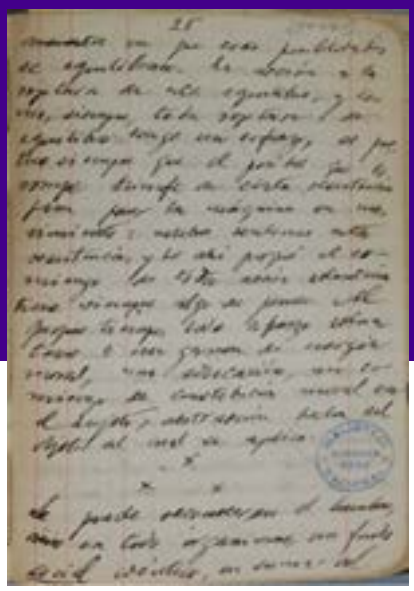












DATA PREPARATION

Most recognition models work on a word level or the line level. Thus, it is essential to segment the page into lines/words. It's been observed that the annotation accuracy remarkably affects the performance of the recognition model. In order to evaluate the performance of HTR (handwriting recognition system), the safest bet is to use an annotation tool that can annotate and label the text lines using polygons. We explored many annotation tools and found the VGG annotator to be the most suitable for our project.

VGG Annotator

VGG Image Annotator is a simple and standalone manual annotation software for images, audio, and video.

Pros

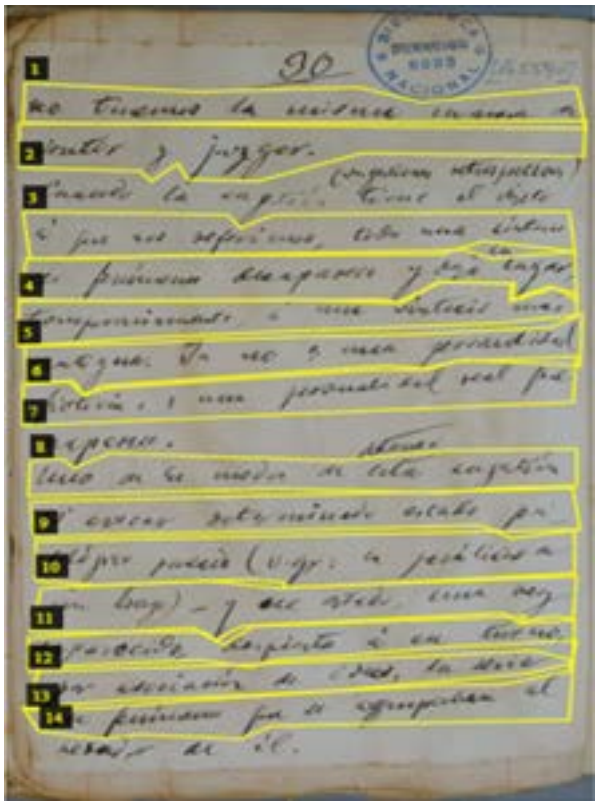
- Free and easy to use.
- Runs in the browser and no installation/dependencies are required.
- Only 400kb, works offline.
- Allow polygonal segmentation. Allow basic editing tools. Support output formats like JSON, CSV, etc.
- Compatible with all modern browsers.

Cons

- A very basic tool but works well in our case.

Requirements for Annotation

- Exclude the margins on the pages.
- The line images should be concise and clear containing only one line.
- Make sure that the words are not cut.
- The line images should be straight (not rotated).
- If a word is not clear, better replace it with a blank character '-'. (The model learns to identify the individual characters, so it might be a problem).
- It is ok to have just one word in the line, but the horizontal length of the image should be somewhat uniform.
- Avoid lines that are too damaged (contains crossed words, unidentified words, lines, etc.)

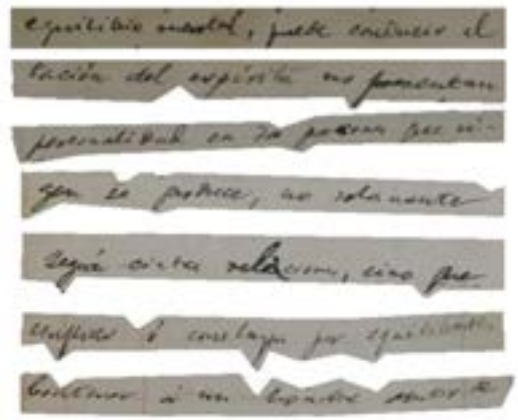


Once all the images are annotated like the example shown above, we can download the annotations in a CSV / JSON file containing the metadata, polygonal data, as well as the labels.

Line Segmentation

The next step is to segment the lines using polygonal data. Our algorithm crops each line using coordinates of the polygon and place it over a white background for the sake of similarity.

624 line images were generated.



UNet Segmentation

While manual line segmentation and annotation are more precise and reliable, it's often not suitable for production. Thus, we also came up with an automatic line segmentation solution using UNet model. Since we are dealing with a difficult dataset set consisting of historical handwritten documents, naive approaches like contour detection, simple Deep neural networks, and fully connected networks fail against the complexities of such data. UNet is a fully convolutional network that was initially designed for biomedical applications, but it can be used for a vast variety of segmentation tasks and text line segmentation is one example.

We can divide the line segmentation pipeline into 3 steps:

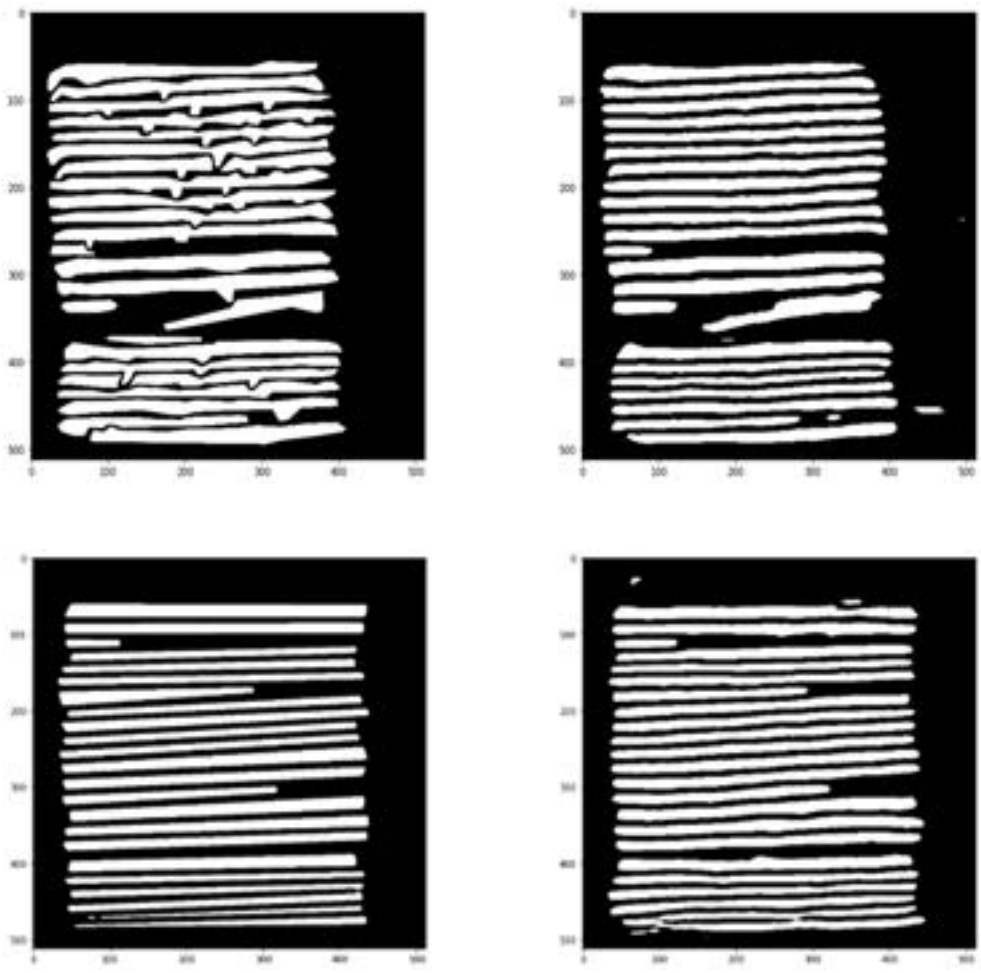
- **Pre-processing**
  - Performing **binarization**, we perform OTSU thresholding since it automatically determines an optimal global threshold value from the image histogram.
  - Since most of the images in our dataset have overlapping lines, we apply **Erosion** using a horizontally dominated kernel, to create a clear boundary between the lines.
  - Finally, we apply **median blur** to further smooth out the edges and remove any unwanted noise.

- **Training / Prediction**

We used a typical UNet architecture, where the image is initially downsampled and then upsampled. This process preserves the most important features of the image, which helps in segmentation. The output is a complete image that not only classifies each pixel but also retains the information on precise localization of each pixel. Our model comprises of 2D convolutional, layers along with alternating Max pooling layers and some upsampling layers. The upsampling is done using transposed convolution. Transpose convolution boils down to learning parameters through back propagation in order to convert a low-resolution image to a high-resolution image. During the upsampling stage, which is also referred as decoding, we use skip connections by concatenating the output of the transposed convolution layers with the feature maps from the Encoder at the same level. In this way, the model learns to assemble a more precise output.

The model is compiled with Adam optimizer, and we use binary cross entropy loss since we are working with a binary image.

We train for 50 epochs and achieved a validation loss of 70 %. Let's look at the results below containing the original masked image and the predicted mask respectively.

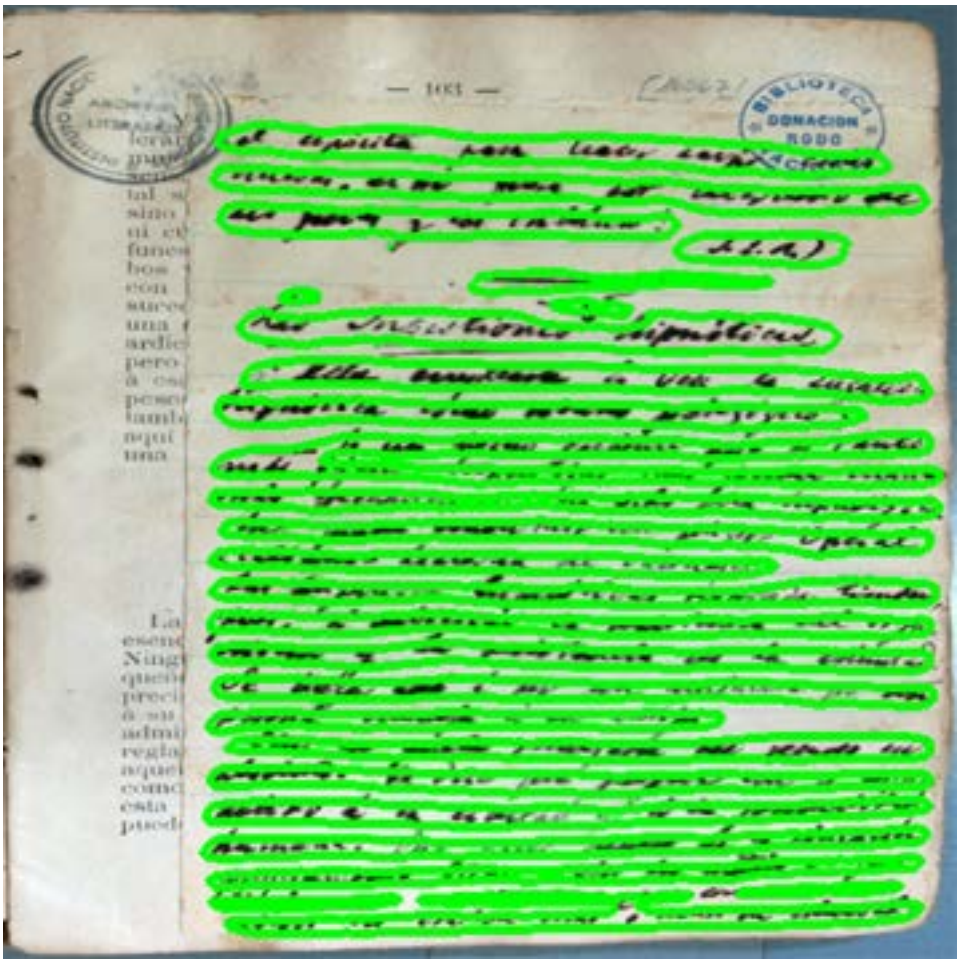


Although UNet performed considerably well, it also failed in some extreme cases where the page was too noisy, or lines were heavily overlapped.

- **Post processing**

To clean up the prediction image we apply a low degree erosion and median blur to clear the noisy edges. After that, we apply the CV2 contour detection technique. The active contours/snake method can also be used as an alternative here.

The output from the contour detection is shown in the following figure:

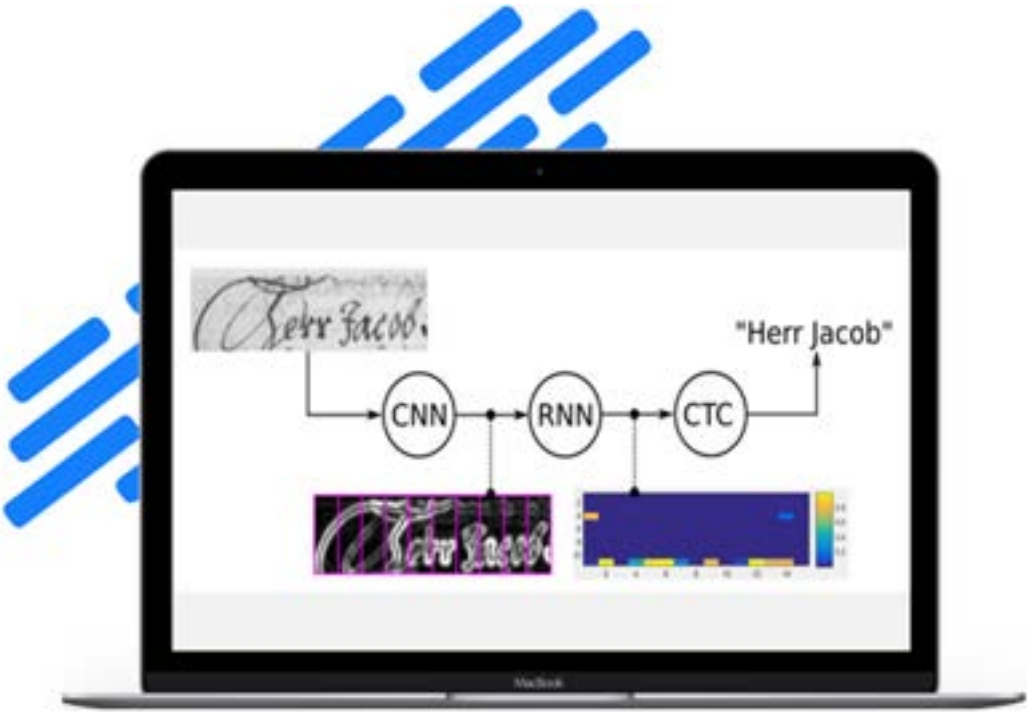


We consider the outer edge of contours while cropping it out of the lines. These lines are then passed as the input to our CNN-RNN-CTC network for text recognition.

CNN-RNN-CTC NETWORK

The HTR system consists of 3 components:

1. **CNN.** The Convolutional neural network works as a feature extractor, the input image is first fed into CNN, and the output is regarded as a series of features. 512 features are extracted from the convolution part.
2. **RNN.** A BI-LSTM model takes the features as input and outputs a matrix containing the character probabilities at each timestamp. There are 4 LSTM layers with 512 hidden units.
3. **CTC.** CTC computes the loss for RNN and also decodes the output sequence.



Basic Intuition

First the Convolutional Recurrent Neural Network is used to extract the important features from the handwritten line text Image. Its output (512 features) is passed to the BLSTM which is for sequence dependency and time-sequence operations. The output of BLSTM is 96 x length(character_dictionary) i.e., 96 timesteps and number of a certain number of characters including blank. Then CTC loss is used to train the RNN which eliminates the Alignment problem in Handwritten since handwritten has a different alignment for every writer. We will discuss the CTC loss in more detail in the upcoming section.

CTC LOSS

CTC finds out the possible paths from the given labels. Loss is given by for (X, Y) pair is .

$$p(Y \mid X) = \sum_{A \epsilon A_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC Conditional **probability**

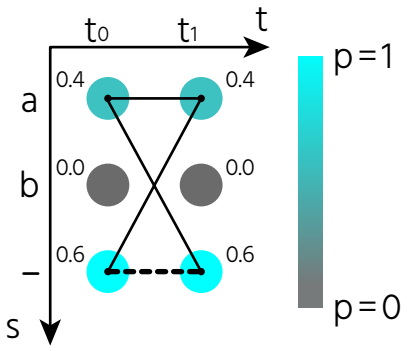
marginalizes over the set of valid alignments

computing the **probability** for a single alignment step-by-step

Motivation

- We don't need to annotate the data at the Character level, we only pass the GT text, ignoring both width and position of the character.
- No post-processing is needed to get the final text. CTC handles the decoding for us.

CTC is very useful for variable-size input as well as outputs. Optimize two functions at the same time: The length of the output sequence and the classes of the output sequence.



Output matrix of NN. The character-probability is color-coded and is also printed next to each matrix entry. Thin lines are paths representing the text "a", while the thick dashed line is the only path representing the text "".

It is calculated by summing up all scores of all possible alignments of the GT text, this way it does not matter where the text appears in the image.

The score for one alignment (or path, as it is often called in the literature) is calculated by multiplying the corresponding character scores together. In the example shown above, the score for the path "aa" is 0.4·0.4=0.16 while it is 0.4·0.6=0.24 for "a-" and 0.6·0.4=0.24 for "-a".

Best Path Decoding

Take the most likely character per timestamp. Remove all the duplicates and then blanks '-' from the path.

Best path decoding is a very simple yet effective decoding scheme, there are still some scenarios that can't be handled by it, for those cases we have more advanced decoding techniques such as beam-search decoding, prefix-search decoding, or token passing, which also use information about language structure to improve the results.

TRAINING

1. DATA PREPROCESSING

This is a crucial step in any Deep learning problem. Our data is pre-processed and augmented before passing to the model. Augmentation techniques are used to increase the size of training data. These pre-processing techniques include

- Adaptive thresholding
- Deslanting
- Random transformations like
  - Stretching, Rescaling
  - Rotation
  - Padding
  - Rescaling.
  - These random transformations result in batch-level data augmentation.

2. TRANSFER LEARNING

It is a subdomain of Domain adaptation that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

Motivation

- Lack of training data
- To save time and space complexity

We used a CNN-RNN-CTC model, pre-trained on IAM Dataset (A benchmark for HTR tasks) The CNN layers are frozen and BILSTM is re-trained on test datasets.

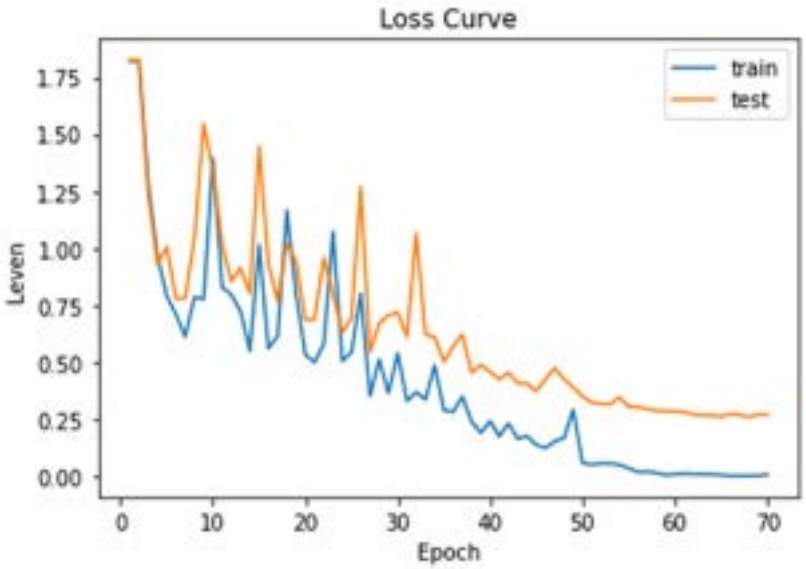
The model was trained for 70 epochs using the Batch size of 16, the Base learning rate of 1e-4, the Max learning rate of 1e-2, and the decay value of 0.1. All these hyper-parameters were tuned using grid search, an exhaustive search over specified parameter values for an estimator.

EVALUATION

We use Levenshtein distances as the main matrix for model evaluation metrics. The sum of the Levenshtein edit distances divides by the sum of the target lengths, for both the training and validation set. At each epoch, these distances are printed along with the losses for both the training and validation set.

Losses for the last epoch:

EPOCHS	TRAIN LOSS	VAL LOSS	TRAIN LEVEN	VAL LEVEN
70	10.198	859.392	0.0073	0.27

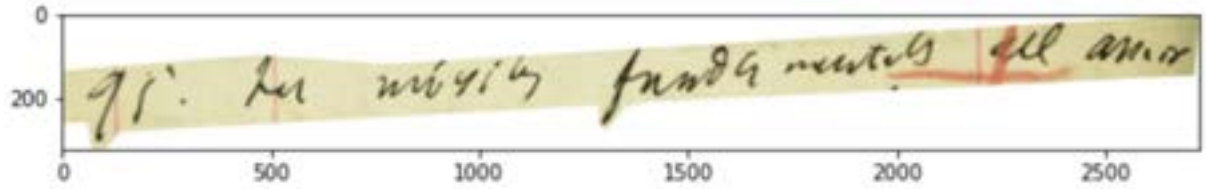


Here we can see a descending trend in both the training and validation losses which tells us that the model is learning and denying the presence of over / underfitting. Although there are fluctuations in the first half of the epochs which make sense since we have relatively few training images and a considerable amount of variation within the images.

We successfully trained and tested our model and achieved a validation error rate as low as 0.2 % for Rodo's dataset after 70 epochs using both training approaches. This can be further reduced by using more data for training and increased learning time.

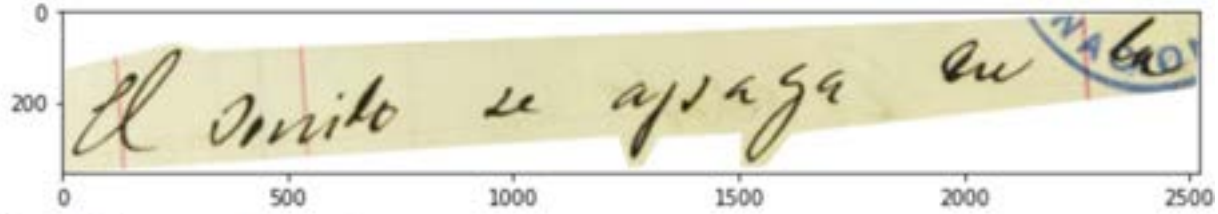
TESTING

We tested our model on a variety of images from both transcripts. Here are a few examples of predictions:

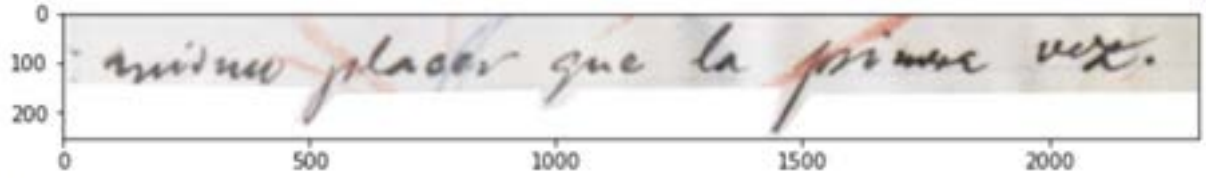


95. Los méritos fundamentales del amor

934. Naturaleza siempre cambiante de los pueblos



El sonido se apaga en la



mismo placer que la primera nvez.

Vid: el Dante, y demás poetas

CONCLUSION

In light of results obtained from training and testing we can conclude that:

- The model is sufficient and strong enough to learn from such complex data given the input images are well annotated and segmented.
- The text recognition performance depends both on the quality as well as quantity of the data we have. More data results in better learning and performance of the model.
- It is evident that it is possible to recognize text from an author (Rodo in our case) even if the data is discontinued (coming from different sources of the same author), given that we have enough carefully annotated lines to train the model.

FUTURE WORK

We can have a single model, good enough to recognize Rodo's text even if it's coming from different transcripts given that the model is trained with a lot more well-segmented line images from various textbooks of Rodo. We can try to use a language model or a spell checker to correct minor mistakes in the predictions.

AFTERWORD

Unlock the power of handwriting recognition

By converting handwritten text into a digital format, our solution offers a range of powerful capabilities for organizations and businesses. It increases the accuracy and efficiency of information processing, simplifies the search, location and retrieval of specific information or documents from large collections, and enables accurate indexing and metadata tagging.

Institutions such as libraries, archives and museums can benefit from handwriting recognition. By leveraging this technology, they can digitize and transcribe their valuable archives and manuscripts, making them easily accessible to researchers, scholars, journalists and students worldwide. Once documents are in digital format, users can efficiently extract information, perform textual analysis, and conduct linguistic or historical research. The technology also makes it easy to compare and cross-reference multiple documents. In addition, it enables remote collaboration and annotation sharing, contributing to the collective knowledge of various fields of study, fostering a vibrant academic community and encouraging interdisciplinary research.

Other organizations, including banks and public and private institutions with an archive of handwritten documents, can benefit from our technology by extracting structured data such as names, dates, locations, and specific content elements. This data extraction enhances the metadata, making it easier to categorize, classify and organize the digitized materials within the institution's databases.

Our handwriting recognition technology enables companies and organizations to unlock the full potential of their handwritten collections by improving accessibility and searchability. It also adds value to research, education, cultural preservation and information management. We invite you to work with our dedicated team of engineers to find a solution tailored to your specific handwriting recognition needs. Our technology supports multiple languages and is designed to accurately recognize and adapt to variations in handwriting, including differences in handwriting size, text size, page texture, image quality and ink.

REFERENCES

- https://openaccess.thecvf.com/content_ECCV_2018/papers/Curtis_Wigington_Start_Follow_Read_ECCV_2018_paper.pdf
- https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c
- https://hal.archives-ouvertes.fr/hal-01681126/document
- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8395169
- https://medium.com/swlh/learn-and-use-handwritten-line-text-recognition-using-deep-learning-with-tensorflow-b661434b5e3b
- https://www.robots.ox.ac.uk/~vgg/software/via/via.html
- https://idus.us.es/bitstream/handle/11441/125558/Boosting%20Offline%20Handwritten%20Text%20Recognition.pdf?sequence=1&isAllowed=y
- https://medium.com/geekculture/detecting-text-lines-in-a-document-image-using-deep-learning-5a21b480bc4c
- https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47