Leverage Deep Learning to digitize old Vietnamese handwritten for historical document archiving

Hoang-Quan Dang[12], Duc-Duy-Anh Nguyen[12], Trong-Hop Do[12]
[1]Faculty of Information Science and Engineering, University of Information Technology, Ho Chi Minh City, Vietnam.
[2]Vietnam National University, Ho Chi Minh City, Vietnam.
{18520339, 18520455}@gm.uit.edu.vn, hopdt@uit.edu.vn

***Abstract*—In this article, we introduce the NomNaOCR dataset for *Hán-Nôm* texts based on 3 tremendous and valuable historical works of Vietnam, including *Lục Vân Tiên*, *Truyện Kiều*, and *Đại Việt Sử Ký Toàn Thư*. With 2953 handwritten Pages collected from the Vietnamese Nôm Preservation Foundation for analyzing and annotating the bounding boxes to generate additional 38,318 Patches containing text along with *Hán-Nôm* strings in digital form. This makes NomNaOCR currently become the biggest dataset for *Hán-Nôm* script in Vietnam, serving 2 main problems in Optical Character Recognition: Text Detection and Text Recognition. A difference here is that our implementations will all be done at the sequence level, which not only saves the cost for annotation but also helps us retain the context in the sequence instead of just performing on each individual character as in most previous works. From that, we have a premise to focus on solving the 2 problems above by many novel Deep Learning methods. With Detection, we used 2 models, EAST and DBNet, corresponding 2 approaches Regression-based and Segmentation-based. Along with that are 4 other approaches to the Recognition problem, including Image Captioning, CRNN, Seq2Seq model in Machine Translation, and Transformer architecture. Evaluation on the validation set of the NomNaOCR dataset shows that the DBNet model excels in the Detection task with an F1-score up to 99.65%. For the Recognition task alone, the CNN model combined with Transformer architecture after Fine-tuning and being added a Skip connection that we proposed, has achieved quite good results with a character accuracy of 84.90% and a low CER of 13.35%. In addition, we also performed an End-to-End evaluation for the above 2 problems and got a surprising result when the combination of EAST and CRNN was the one that outperformed, with 87.45% for F1-score. Besides, many other detailed analyses such as Detection results on poetic and prose images, or error analysis for the best models will also be clarified.**

***Index Terms*—Optical Character Recognition, Text Detection, Text Recognition, Digitization, History, Hán-Nôm, Vietnamese.**

I. INTRODUCTION

Language is an innate ability of humans, and the writing system is a sign of a country's civilization, a creative invention of a nation. Wonderful Vietnamese with extremely rich phonetics and the most affluent writing system in East Asia. Throughout history, Vietnamese writing has gone through a journey from *Hán* (Sino) or *Nho* letters to *Nôm* letters, and finally, the *Quốc Ngữ* alphabet based on the Latin writing system, and accompanying each type of script is glorious history worth reviewing of the nation. After ending the thousand years of Northern domination in 939 CE, Vietnamese people, with a sense of language autonomy, created the *Nôm* script based on the *Hán* characters that are read according to the Sino-Vietnamese sounds, so it can be said that *Hán* is a subset of the *Nôm* script. And over the next 1000 years, from the 10th to the 20th century, in parallel with the use of *Hán* characters, the *Nôm* script was used to write most of the literature documents, medicine, philosophy, religion, and cultural history of the Vietnamese people. However, this heritage is now in danger of being lost by the shift to the *Quốc Ngữ* modern script.

According to the Vietnamese Nôm Preservation Foundation - VNPF [1]: "Today, less than 100 scholars worldwide can read *Nôm*. Much of Việt Nam's vast, written history is, in effect, inaccessible to the 80 million speakers of the language." Due to the great value of historical documents for study, especially the social aspects and lifestyle of the past and the messages left by predecessors, the preservation of this cultural heritage is imperative. Vietnamese agencies and organizations around the world have collected thousands of volumes of *Hán-Nôm* documents. Recently, thanks to the development of information technology in preserving, managing, researching, and exploiting *Hán-Nôm* documents, over 4,000 texts have been scanned into digital images. However, to make use of this vast source of knowledge, they must be indexed, in the form of fully searchable text, with comments, annotations and translation into modern *Quốc Ngữ*. Due to the intricate and time-consuming translation work, along with the limited number of experts, these efforts could not be accomplished in a short time. Optical Character Recognition (OCR) techniques will speed up this digitization process making all significant works in *Hán-Nôm* available online.

Therefore, we will focus our research on the process of building a good dataset, image processing techniques, natural language processing, and deep learning approaches to solve 2 main problems of OCR for handwritten *Hán-Nôm* characters in old documents: Text Detection and Text Recognition.

In this article, we first present the topic of digitizing old documents written in *Hán-Nôm* script, especially the practical application of this topic in archiving historical documents. In section II, we will introduce the information related to OCR. In section III, We will present some researches worldwide in general and Vietnam in particular related to the data construction process, methods used to solve the problem of Detection and Recognition for handwriting text in old documents. Section IV, we will introduce to our NomNaOCR dataset, especially present the process of building a standard and good quality dataset as well as performing detailed statistics and analyzing the characteristics of the data, thereby serving as a premise to develop methods for the problem posed on this dataset. Section V will describe the approaches that we have researched and applied to the NomNaOCR dataset for the problem of detecting and recognizing handwritten *Hán-Nôm* characters in old documents. The setup steps and detailed settings for the hyperparameters as well as the hardware used to train the models will be shown in section VI and section VII is the discussion for the results that we have achieved as well well performing the evaluation, explanation for those results, and analyzing the errors of the best model when predicting on the NomNaOCR dataset. Finally is our conclusion and future approaches VIII to improve the model performance.

II. Optical Character Recognition

Optical Character Recognition (OCR) refers to a set of computer vision problems that convert digital or scanned document images into machine-readable text in a form computers can process, store, and edit as a regular text file or as a part of manipulation software. The images can be documents, invoices, legal forms, identity cards, or things in the natural scene (OCR in the wild) like street signs, license plates, etc. This is a complex topic with a harmony of 2 large areas in AI: Computer vision and Natural language processing.

A. Types of text image

Challenges in OCR arise mainly due to the tasks or types of data that are performing. In general, we divide these types into 3 main categories:

- Structured: images that contain text in typed documents has a clean, uniform background, standard fonts, low noise, explicit lines, has a certain order, and often has a high text density.
- Unstructured: this is the most difficult type, the text will be in random positions in the image or a natural scene, often with sparse text density, irregular line structure, complex background, and no standard font. This data type can also be divided into 3 other subcategories:
  - Graphic text: images that have text added after the image is already there, like a video subtitle.
  - Scene text: images that have text appears in natural scene, the text is an existing element in the image. Data of this type have many challenges, such as orientation, font type, lighting conditions, or skewness of the text. The OCR task for this data type can be called Scene Text Recognition (STR).
  - Synthetic text: quite similar to Graphic text, this is a pretty good idea to improve the performance of the training process by creating artificial data. Generating random characters or words onto an image will seem much more natural than any other object because of the flat nature of the text. The datasets of this data type also excel in their ability to generate in different languages, even difficult ones, such as Chinese, Thai, or Arabic.
- Hybrid between structured and unstructured: usually scans, photocopies not in the typed form of documents, books, with good structure, less complicated background, neat, and often have a higher density of text than Unstructured type but has a slight orientation and skewness. In addition, for documents such as historical documents, there may be significant and ambiguous changes due to differences in handwriting between people, or image quality may decrease over time, such as images of historical documents used in the NomNaOCR dataset we built.

B. Pipeline

There are 3 levels for implementing OCR tasks: line level, word level, and character level. Today, line and word levels are often preferred (we call them sequence level in general for this article) since character levels require too much annotation cost because we need to draw bounding box for each character, which leads to more dilemmas such as the bounding box between 2 consecutive characters may be unclear or overlap. Next, the OCR is also divided into 2 main sub-problems:

- **Text Detection**: detect the image regions that contain text. The input is an image (or a *Page* for this article), and the output is a bounding box surrounds the text area found.
- **Text Recognition**: after detecting boxes or image regions that contain text, each of these regions will be cropped from the original image, forming small image parts called *Patch*. The input will now be a *Patch*, and the output will be the text existed in that *Patch*.

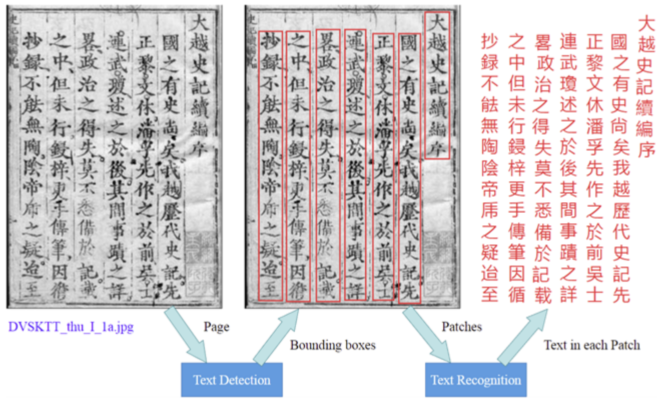

DVSKTT_thu_I_1a.jpg
Page
Bounding boxes
Patches
Text in each Patch
Text Detection
Text Recognition
大越史記續編序
國之有史尚矣我越歷代史記先
正黎文休潘孚先作之於前吳士
連武瓊述之於後其間事蹟之詳
畧政治之得失莫不悉備於記載
之中但未行鋟梓更手傳筆因循
抄録不能無陶陰帝帍之疑迨至

Figure 1: Pipeline for 2 OCR sub-problems in this article. Input: Scanned image of a handwritten Hán-Nôm text page. Output: Digital form of Hán-Nôm strings in that input image.

III. Related Works

The need for digitization is growing rapidly in the modern era. Due to the growth of information, connectivity technology, and the widespread availability of mobile devices, people generally prefer digital content over printed materials, including books and newspapers. In addition, it has become easier to organize digitized data and analyze them for different purposes thanks to advanced techniques such as AI. So to keep up with the development of technology, it is necessary to convert all existing information from print format to digital format.

For historical documents alone, digitization is crucial to preserve cultural heritage. Furthermore, extracting digital text from text images during digitization is necessary to provide efficient information access to the content of these documents. So, there have been typical researches on this topic for building datasets and developing models. From that *Text Detection* and *Text Recognition* are attracting lots of researchers and businesses. However, combining them is challenging but plays important role in many fields. Therefore, many works have also been published to address these problems.

In Korea, Kim et al. developed a system to digitize more than 10 million handwritten Hanja documents - a type of script that was common in Korea until the late 9th century [2]. In China, Digital Heritage Publishing Ltd. digitized more than 36,000 volumes or 4.7 million pages of Siku Quanshu - the largest book series composed by 361 Qianlong scholars [3].

For solving the OCR problems of digitization, A.D. Le et al. introduced the Attention-based Encoder-Decoder model to recognize Japanese historical documents [4] using 2 main modules: DenseNet for features extraction and a built-in LSTM decoder to generate output text. The model requires the input image as a text line to produce the output characters. Therefore, the authors do not need to label the characters, saving a lot of time. In addition, the authors also introduce a method to create artificial text lines to solve the imbalance problem of the dataset. The model achieved a Character error rate (CER) of 23.76% and 22.52%, corresponding to with and without training with the artificial text stream. Moreover, this recognition system outperforms the CNN-LSTM model.

In Vietnam, a huge number of documents were written in Nôm script and are still stored in communal houses, pagodas, and libraries. Currently, these priceless historical works are in danger of being lost, difficult to reach for the next generations, and are gradually degraded, most of which have not been digitized. Besides, the number of experts understand and read these texts is also decreasing. Therefore, there have also been several research works in Vietnam, such as [5] or [6], aimed at solving the above problems, but there are still some limitations. Especially, no work has provided a good enough dataset to serve both the Detection and Recognition of handwritten Hán-Nôm characters to digitize Vietnamese historical documents.

Most notable recently, M.T. Vu et al. introduced a new handwritten database IHR-NomDB [7] for the old writing system of Vietnam. More than 260 pages of Nôm documents have been collected from the VNPF [1] for analysis and labeling. The author has manually determined the bounding boxes to create more than 5000 cuts for handwritten images with corresponding digital Nôm characters and translations into Quốc Ngữ script. In addition to this handwritten dataset, the author also created a Synthetic Nom String dataset consisting of 101,621 images automatically generated using the collected bank of Nôm sentences. It became the first and largest public database for the study of the old Vietnamese script. For the basic results, the author experimented on the validation set of the handwriting dataset by using the CRNN model combined with the CTC loss pre-trained on the Synthetic Nom String dataset and achieved an accuracy of 42.70% at sentence level and 82.28% at character level.

Another method based on text segmentation to digitize Nôm documents using deep CNNs [8] has been proposed by K.C. Nguyen et al. Nôm pages are preprocessed, segment into separate characters, and then recognized. The U-Net architecture was applied here to create the segmentation map and extract the image region containing the characters. Next, the author also proposed classifiers to recognize each character using a language model. Compared to the traditional segmentation method, which only achieves 81.23% for IoU, the deep CNN network method yield better results with an IoU of 92.08% in detecting image regions containing characters. For character recognition, the author's proposed CNN models are better than the traditional models, with a recognition rate of 85.07%.

IV. The NomNaOCR Dataset

Since this is a peculiarity dataset for old Hán-Nôm script, we first need to do a search for a reputable data source to increase the reliability of this dataset after completion. We chose the website of the VNPF [1] as the sole data source because this is a reputable association on Nôm script led by many professors and PhDs from Vietnam and worldwide with the aim of preserving the cultural heritages written in the form of Nôm script that are gradually disappearing from the nation by developing software tools to work with digitization, printing, research, physically preserving and sharing on the Internet many of the documents in the national library and Buddhist temple collections. After nearly 20 years, VNPF has been working to promote the study of Hán-Nôm documents and the culture created among the community. The construction of our dataset using semi-manual method is described below.

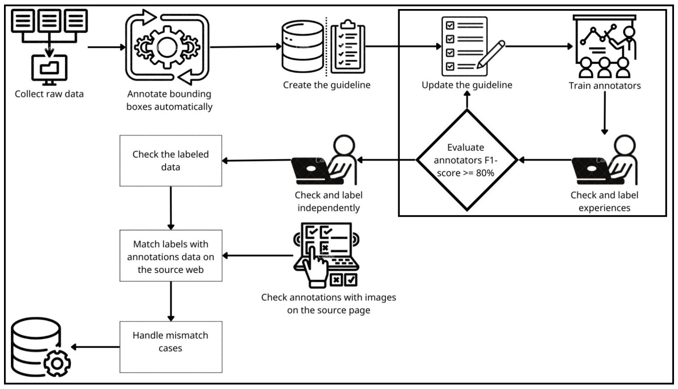
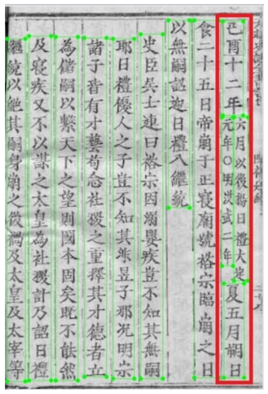

Collect raw data
Annotate bounding boxes automatically
Create the guideline
Update the guideline
Train annotators
Check the labeled data
Check and label independently
Evaluate annotators F1-score >= 80%
Check and label experiences
Match labels with annotations data on the source web
Check annotations with images on the source page
Handle mismatch cases

Figure 2: Construction process of the NomNaOCR dataset

Due to the high cost of labeling, we do not perform the character level annotation. For Hán-Nôm Pages, each column in the image can have multiple bounding boxes (wrapped by the red box in Figure 3). So we treat each bounding box as a word and process them at word level, not a column, and process them at line level. Another benefit to this approach is that the context in a Hán-Nôm string will not lost. From that, we can develop Recognition models that not only recognize Hán-Nôm letters but also learn the context around them.



己酉十二年六月以後篡日禮大定元年○明洪武二年夏五月朔日

食二十五日帝崩于正寢廟號裕宗臨崩之日

以無嗣詔迎日禮入繼統

史臣吳士連曰裕宗因溺嬰疾豈不知其無嗣

耶日禮優人之子豈不知其非昱子耶況明宗

諸子皆有才藝苟念社稷之重擇其才德者立

爲儲嗣以繫天下之望則國本固矣既不能然

及寢疾又不以謀之太皇爲社稷計乃詔日禮

繼統以絶其嗣身崩之後禍及太皇及太宰等

Figure 3: Illustrate a column can have multiple boxes

After the actual implementation, the NomNaOCR dataset was processed and obtained 2953 Pages from these documents: *Lục Vân Tiên*, *Tale of Kiều* or *Truyện Kiều* (version 1866, 1871, 1872), and a full set of 5 parts in *History of Greater Vietnam* or *Đại Việt Sử Ký Toàn Thư* (ĐVSKTT). By semi-manually annotating the above documents, we got additional 38,318 Patches. In addition to poetic documents (*Lục Vân Tiên* and *Truyện Kiều*) accounting for about 17.03% of Pages in the dataset, the rest is *Đại Việt Sử Ký Toàn Thư* which accounts for about 82.97%, so the length of texts are mainly 6, 8, 17, and 18 characters.

Table I: Number of strings by text length in NomNaOCR

Text length	Number of strings
1	631
2	1318
3	1371
4	1232
5	727
6	6023
7	406
8	5824
9	396
10	432
11	436
12	519
13	331
14	290
15	254
16	588
17	2679
18	10506
>18	4355

For data splitting, we used by the formula of IHR-NomDB [7] to get the similarity distribution between the Train set and the Validate set. We defined an R value for each data points in the dataset D, where N is the sum of how often each character in s occurs in D/s. R score is calculated by the formula below:

$$R_s = N_{distinct(s)}.max_i^D N_i + N_s \qquad (1)$$

The R score emphasizes the number of characters in string s appearing in D and the occurrence frequency of those characters. To select k samples for the Validate set, we only need to take k samples with the highest R score. Here, we chose k = 7664, equivalent to 20% of the dataset. Table II shows the frequency of data points and the *character intersection* between the Train set and Validate set. The term *character intersection* refers to the lexical coverage of one set relative to the other. From Table II, it can be seen that by applying the R score for data splitting, we get 93.24% different characters in the Validate set that exists in the Train set, which means the Train set includes almost every character in the Validate set.

TABLE II: Intersection between Train set and Validate set

	Number of records	Character intersection
Train set	30654	93.24%
Validate set	7664	64.41%

V. Approaches

One of the most straightforward approaches to the OCR problem is to use traditional computer vision techniques or basic image processing methods, which have been used for a long time. They can be encapsulated in the following steps:

- Apply filters to highlight characters from the background.
- Perform contour detection to detect image regions that containing characters.
- Run a classifier for those image regions to determine what characters in them are.

Obviously, if step 2 is done well, step 3 can be easily done by pattern matching or training a simple CNN. However, contour detection will be pretty difficult for generalization, not being able to adapt to variations in images, especially in heterogeneous environments. It requires a lot of manual adjustment, so this becomes unfeasible in most cases. Another disadvantage can also be seen that most of these methods is only effective with character level, so if we solve our problems in this direction, in addition to the high labeling cost when building dataset, we will also lose the context in a Hán-Nôm string. Today, Deep Learning approaches can be considered mature, excel in their generalization, and have become the dominant solutions in both research and practice, which can help us achieve good results for the digitization.

A. Text Detection

Firstly, it can be seen that the Text Detection is quite similar to the Object Detection problem in that the object need to be detected is text. Therefore, we can apply some Deep Learning architectures of this problem, such as YOLO, SSD, or Faster R-CNN, to achieve higher accuracy than manual image processing methods. However, these models seem to yield good results only with large objects as well as with high-resolution images. It is pretty ironic because in practice, these models seem to have more difficulties and tend not to achieve the desired accuracy when detecting digits and letters than when detecting complex objects such as dogs, cats, or human.

Specialized methods based on Deep Learning have recently solved most of the above problems and achieved good results on many standard benchmark datasets such as ICDAR 2013 [9] and 2015 [10]. These are also the approaches we will use for the Text Detection problem in this article. Specifically, they are divided into 2 main categories as follows:

- Regression-based method: roughly, after the bounding box is predicted, the bounding box is finally filtered through by Non-maximum Suppression (NMS). Most bounding boxes of this method have only 4 coordinate points, so this method will be limited to representing bounding boxes or texts that have irregularly shaped (e.g. curved) in the image. Some prominent models for this method can be mentioned like EAST or TextBoxes.
- Segmentation-based method: consider the Text Detection problem from the perspective of an Object Segmentation problem, aiming to find the image regions containing text at the pixel level. This approach detects text by estimating the bounding area of the text. Bounding boxes of text will be formed through the segmentation result so that they can represent irregular shapes. Some typical representatives of this method are DBNet, PSENet, etc.

1) Regression-based method with EAST:

EAST (Efficient and Accurate Scene Text Detector) [11] uses a single neural network to predict text at word level or line level. It can detect text in all directions depending on the quadrilateral shapes of the bounding box. In 2017, this model outperformed modern methods at that time. This model consists of a fully convolutional network that plays a key role in detecting text in images, combined with NMS to make predictions by merging many detected but not yet correct bounding boxes into one correct bounding box.

The architecture of EAST was created to consider different sizes of Patches. The idea is that detecting large Patches will require features extracted from a later stage of the network, while detecting smaller Patches will require features from an early stage. of the model. Therefore, the authors of EAST proposed an architecture with 3 branches or 3 parts to combine into a neural network, including Feature Extractor Stem, Feature Merging Branch, and finally the Output Layer.

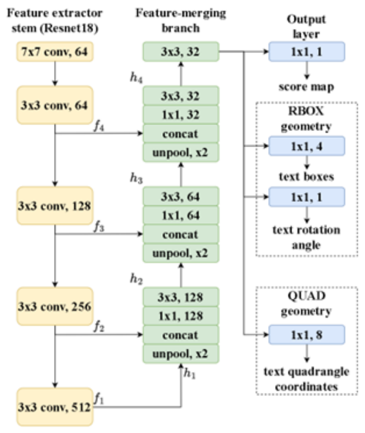


Figure 4: EAST architecture

2) Segmentation-based method with DBNet:

DBNet (Differentiable Binarization) [12] uses a CNN as backbone to extract image features. This network can be VGG, ResNet, etc. These features will go through the Un-sample layer at the same scale and cascade to create feature maps F. The feature map layers F are used to predict both the Probability map P and the Threshold map T, from that obtaining Approximate binary map $\hat{B}$. During training, supervision will be applied to the 3 maps above, in which the Probability map and the Approximately binary map share the same observations. During the inference phase, bounding boxes will be obtained from Approximately binary map or Probability map after passing through a box generation module.

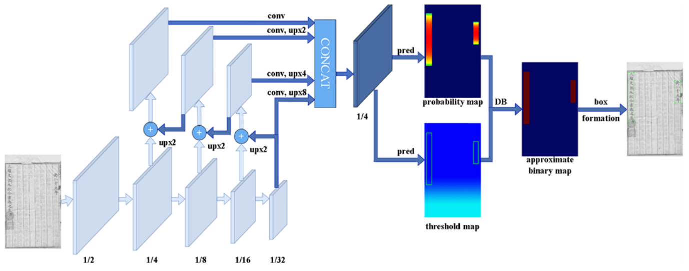


Figure 5: DBNet architecture

B. Text Recognition

Suppose, if we chose to implement our article from scratch at the character level, in this Text Recognition problem, to know what characters the obtained bounding boxes contain after performing Text Detection on a Hán-Nôm Page, we just simply apply a CNN architecture. This approach is similar to the handwritten digit recognition problem on the MNIST dataset, which is familiar to those new to Deep Learning. Besides, there was also a contest with the same digitization purpose as our article for Kuzushiji (an ancient Japanese character) on Kaggle [13] with many helpful solutions from different candidates. For the most part, they used familiar architectures in Object Detection combined with a CNN.

However, because of the disadvantages of character level implementations mentioned in the previous sections, we aimed for approaches that not only help Recognition models to recognize Hán-Nôm letters but also can learn the context around them, instead of simply using a CNN to recognize individual characters one by one.

1) Image captioning approach:

The simplest and easiest to see, we considered the OCR as a problem of generating a natural language description for the content of an image or Image Captioning. In the process of forming a caption, we simultaneously observed the image and tried to find ways to generate a sequence of meaningful words. Thus, there are 2 types of information will need to be processed: the image and the caption corresponding to that image. For image input, we used a CNN for feature extraction and for the caption input, RNN was applied to process string data. The question here is how can the information between these 2 data types be combined, i.e., how should these information input to the model or in what order?

A research by authors from the University of Malta [14] presented a systematic comparison of different combinations, or more specifically, evaluating whether it is more effective to process the information collectively or separately. The authors uses the following 2 types of architecture:

- Injection architecture: the feature vector of the image obtained when passing through the CNN will be inserted into the RNN, so every step of the RNN is affected by the image data. In addition, processing the image and caption simultaneously will provide additional information for each other. Furthermore, Injection architecture is also divided into 3 sub-types: Init-inject, Pre-inject, Par-inject.
- Merging architecture: process the image and its caption separately, then combine their results by Concatenation or Addition. Thus, RNN will not be exposed to the image vector at any step. This is a Late Binding architecture, it will not modify the image representation for every step.

With the architectures presented above, the authors showed that the Init-inject architecture achieves many good results based on various evaluations and metrics, so this will be the first model we choose for the Image Captioning approach and we call it *Init-inject Captioner* from here on.

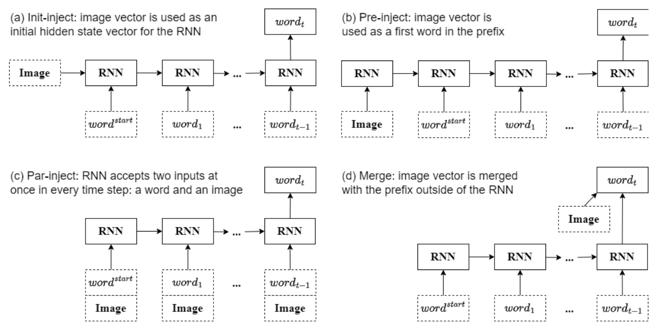

(a) Init-inject: image vector is used as an initial hidden state vector for the RNN
(b) Pre-inject: image vector is used as a first word in the prefix
(c) Par-inject: RNN accepts two inputs at once in every time step: a word and an image
(d) Merge: image vector is merged with the prefix outside of the RNN
Image
RNN
word_t
word^start
word_1
word_{t-1}

Figure 6: Injection and Merging in Image Captioning

Our other workaround for the Image Captioning approach is to take advantage of the Attention Mechanism, inspired by 2 related works [15] and [16]. By using an Attention-based model, we can observe which part of the image will be focused on by the model when it generates the caption. We construct this model according to the Encoder-Decoder architecture and call it *Attention-based Captioner* from here.

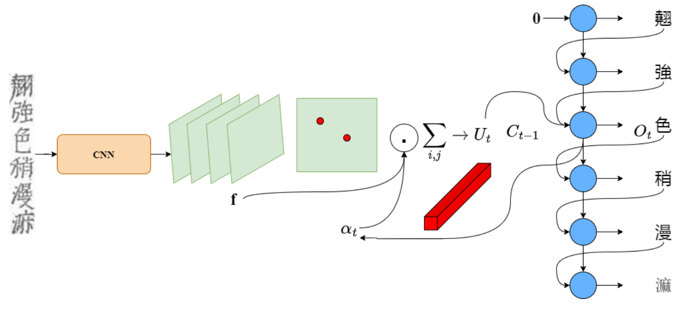


Figure 7: Attention-based Image Captioning model

In general, in the flow of the *Attention-based Captioner* model, we first use a CNN as an Encoder to extract image features. Then this feature vector along with the hidden state (initialized at 0) and the character at the first position will be passed to the Decoder, which returns the prediction for the next character along with its hidden state. This hidden state will then be passed back into the Decoder along with the image and character vector at the next location. The process will be repeated continuously until the end of the sentence or the end of the input string. In addition, we also use Teacher Forcing to decide the next input, a technique in which the target character is passed as the next input to the Decoder.

The specific task of the Decoder is to generate a prediction for the next character. Once it receives the image vector or feature map from the Encoder and the hidden state of the character under consideration, the Decoder will use the Attention Mechanism to transform them into a context vector as input for the RNN along with the embedding of the current character later after focusing the "attention" on a particular region of the image. Thus, the input of the RNN at timestep *t* will be calculated by the following formula:

$$x_t = W_c c_{t-1} + W_u u_{t-1}$$

Where c_{t-1} is the embedding of the previous character and u_{t-1} is the context vector. The distribution for final prediction on characters at timestep *t* will be calculated by a softmax function, and the character with the highest probability will be the result of the prediction. This is also known as Greedy Search. However, using the Attention Mechanism here can lose the sequential nature of the pixels. Therefore, we will concat a one-hot vector of coordinates to the feature maps. This technique is proposed in the paper [15], thereby helping the model to have additional location information of pixels.

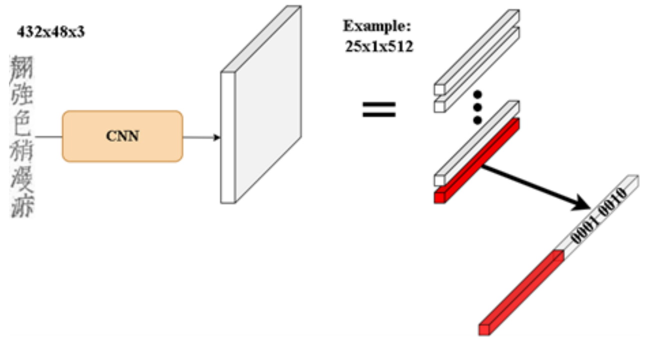


Figure 8: Add pixel coordinates to Attention-based Captioner

2) *Convolutional Recurrent Neural Network:*

Convolutional Recurrent Neural Network (CRNN) [17] was created based on the most fundamental views: simple to say, Text Recognition is just a problem of recognizing the sequence of text from the input image. For image processing, the most suitable network is usually CNN; for the sequence processing problem, the most suitable is usually RNN. From that idea, CRNN was born and is a combination of a CNN and a Deep Bidirectional RNN along with CTC [18] loss function, a mechanism to convert each step output to the final output and vice versa. The CRNN model is designed with the aim of solving image-based sequence recognition tasks, such as Scene Text Recognition problems. This is also a popular model for recognizing print as well as handwriting and has very satisfactory results despite its extremely simple architecture. We will call this model *CRNNxCTC* from here.

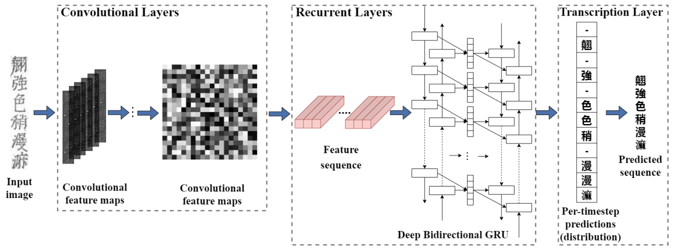


Figure 9: CRNN architecture

One thing to note here is that the feature maps will be divided into columns of 1 pixel width or a series of feature vectors. The question is, why divide these feature maps by columns? The answer related to the concept of the receptive field. This is the region in the input image that a particular CNN feature map sees. The receptive field of each feature vector will correspond to a rectangular region in the input image as shown below, i.e., we can consider each feature vector as a descriptor of that rectangular image area.

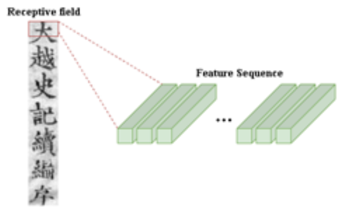


Figure 10: Receptive field in CRNN

The goal of CTC is to take the messy strings, which may have some extra characters or other blanks, and use a probabilistic approach to unify and make them meaningful. So, to map the input sequence X = $[x_1, x_2, ..., x_T]$ to the corresponding output sequence Y = $[y_1, y_2, ..., y_U]$, CTC will sum all the probabilities that can happen when aligning the text in the image as shown below:

$$p(Y|X) = \underbrace{\sum_{A \in A_{X,Y}}}_{\text{Marginalizes over the set of valid alignments}} \underbrace{\prod_{t=1}^{T} p_t(a_t|X)}_{\text{Compute the probability for a single alignment step-by-step}}$$

CTC conditional probability

Figure 11: The CTC conditional probability

3) Sequence-to-Sequence in Machine Translation: The CRNN architecture using CTC Loss has a limitation is that we must carefully tune the model architecture so that the size of the receptive field to match the maximum number of characters that can be predicted. A popular addition to the CRNN model that can be used to improve the prediction of text in input images is an Attention Mechanism. In this approach, as usual, we will first use CNN for image feature extraction. These features are then converted into strings and transmitted through RNN to obtain the results for the Attention Mechanism. The model we use in this process is inspired and works pretty similar to the Attention-based Sequence-to-Sequence (Seq2Seq) model in Machine Translation [19]. For example with a Machine Translation problem from Vietnamese to English, we need to encode a Vietnamese text sequence into a vector; but with this model, the input will be an image. We will call this model *CNNxSeq2Seq* from here.

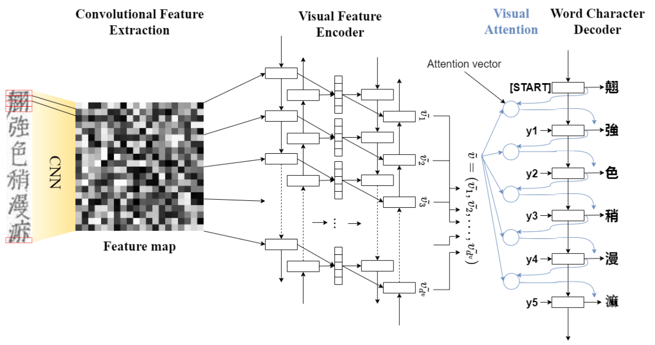


Figure 12: Combined model of CNN & Attention-Seq2Seq

The Decoder will use an Attention head to selectively make "attention" (attn) on parts of the input sequence. This Attention layer is similar to an Average Pooling layer but represents a weighted average. The Decoder's job now is to generate predictions for the next character by taking in the complete Encoder output, then it will use the RNN to keep track of what it has generated up to the current step. The RNN output will be used as a query to do the "attention" on Encoder output, thereby forming a context vector. Next, Decoder will combine the RNN output and this context vector to generate an Attention vector. The prediction result for the next character will be based on this vector. The formulas for above process of the model in this approach will be in the following order:

$$\begin{array}{ll} score(h_t, h_s)) = v_a^t tanh(W_1 h_t + W_2 h_s) & \text{[Bahdanau attn]} \\ a_{ts} = softmax(score(h_t, h_s)) & \text{[Attn weights]} \\ c_t = \sum_s a_{ts} h_s & \text{[Context vector]} \\ a_t = f(c_t, h_t) = tanh(W_c[c_t; h_t]) & \text{[Attn vector]} \end{array}$$

The training of this model is similar to the training of the Seq2Seq model in Machine Translation, they use Cross-entropy as a loss function instead of CTC loss like CRNN. That means at each timestep, the model will predict one character to calculate the loss against the actual label and update the model's weight. However, for this model, short sequences will often perform better, but if the input is too long, the model will literally lose focus and stop providing reasonable predictions. There are 2 main reasons for this:

- The model was trained using Teacher Forcing will provide the correct character at each step, regardless of the prediction outcome. Therefore, the model can become more robust if it is fed with its own predictions.

- The model can only know its previous output through the state of the RNN. If this state is corrupted, there is no way for the model to recover. Transformer solved this by using Self-Attention in Encoder and Decoder, which also plays a key role to model long-term dependencies.

4) *TransformerOCR:*

Another way we used to overcome the disadvantages of the Seq2Seq architecture and increase the accuracy of Recognition models is to leverage the Transformer [20], an architecture that changed the NLP world. Transformer will function similarly to RNNs, but the difference is that it does not require processing input data in sequence. This can also significantly reduce the time required to train such a Recognition architecture. We break the model down into 2 sub-modules: a feature extraction module and a Transformer module. The feature map is used as an Embedding and as input to the Encoder to create a new representation of the input. Decoder uses the Encoder output along with the target text as input and learns how to produce the correct output. While the model predicts each position in the input sequence, Self-Attention allows looking at previous positions to make a better prediction for the next position. We'll call this model *CNNxTransformer* from here on.

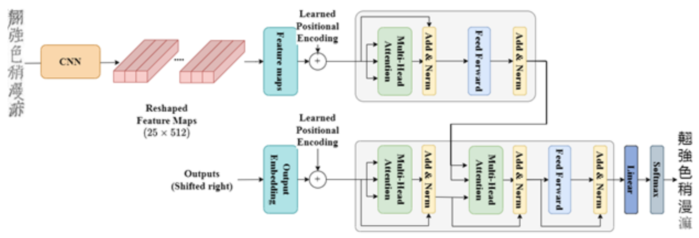


Figure 13: Combined model of CNN & Transformer

Besides the model using CNN that combines the entire Transformer architecture above, we also build another model based on the Transformer architecture without using intermediate sequence representations. We make a direct connection of features from the CNN with the Transformer Decoder acting as an Attention-based sequence Decoder. We will call this model *CNNxTransformerDecoder* from here.

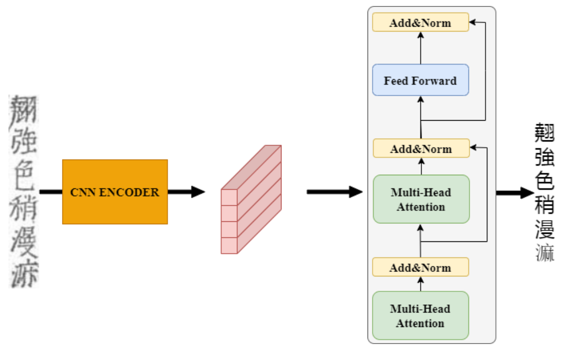


Figure 14: Combined model of CNN & Transformer Decoder

Transformer is an auto-regressive model, which means it makes partial predictions and uses the output from the first step to the current step to decide what to do next. Therefore, the target text needs to be divided into 2 parts: one part is kept as input to the Decoder, and the other part will be shifted by 1 unit to serve as the learning target. In addition, we also use Teacher Forcing to pass the correct output to the next timestep, regardless of what the model predicts at the current step.

VI. Experimental Setup

A. *Implementation for Text Detection*

With this problem, we will experiment 2 models for Pages in the NomNaOCR dataset, EAST and DBNet, which correspond to 2 approaches, Regression-based and Segmentation-based. We used ResNet 18 as the backbone to extract image features for the above models with the same batch size set to 8 and Adam for the optimizer. These experiments were performed on the Google Colab Pro environment using a configuration of 25GB RAM and an NVIDIA T4 GPU.

For EAST, we trained this model for 115 epochs, using a learning rate of 0.001, and warm-up after 10 epochs with the input image resized to 512x512. Finally, in the Post-process step to form the final result from the model output, the geometries that are retained after satisfying a certain threshold of 0.2 will be merged using the NMS algorithm.

For DBNet, we just trained this model for 15 epochs, using a cosine scheduler for learning rate with an initial value of 0.001, and warm-up after 2 epochs. All input images will be reduced to a general size of 960x960 and performed some data augmentation including: random rotation with angle amplitude in the range (-10, 10) and random resize. Finally, with the Post-process step for DBNet, we made the settings that this model requires for capturing bounding boxes efficiently:

- Thresh = 0.3: the threshold required for the binarization result of the segmentation map.
- Box thresh = 0.6: the threshold to filter output boxes (boxes below this threshold will not be output).
- Max candidates = 1000: the maximum number of output text boxes.
- Unclip ratio = 1.5: the unclip ratio of the text box.

B. *Implementation for Text Recognition*

We will train the models with a batch size of 32 and apply Early Stopping to stop the training process if the loss of the Validate set has not improved by at least 0.001 units after 5 epochs. In addition to CRNN using the objective function CTC, the remaining models use Cross-entropy to calculate the loss and update the weights. Experiments were performed on a hardware system with a 64GB RAM configuration and an NVIDIA RTX 2080 SUPER 8GB GPU.

1) *Training phases:*

We did the training of Recognition models in 3 phases:

- Pre-training: while building the NomNaOCR dataset, we also constructed Text Recognition models and used the Synthetic Nom String dataset, a part of IHR-NomDB (mentioned in section III) for training, as well as to find the optimal settings for the above models in while the real dataset is still being finalized. This phase can also be seen as a way of image augmentation.

- Fine-tuning: after NomNaOCR had been completed, we trained the model on this real dataset from the weights learned in the Pre-training phase.
- Retraining: train the model from scratch on NomNaOCR dataset and compare results with the Fine-tuning phase.

2) CNN backbone:

In general, the models we implemented include 2 main parts: a *backbone part* for image feature extraction and a *language processing part*. In this Recognition problem, we built a simple CNN as backbone, with the height dimension of the receptive field adjusted to match the maximum text length in the NomNaOCR dataset. From that, we modeled the feature maps as a sequence of features and used it as input to the language processing part. In addition, the authors of IHR-NomDB [7] also showed that using RGB images in higher resolution will help Deep Learning models for handwritten images have better performance. Thus, the input images or Patches will be processed as 3 channels RGB and resized to a general size, but the aspect ratio is still guaranteed, and the content of the Patch is not affected.

Specifically, we used the recommended size 432x48x3 in the IHR-NomDB to obtain the input for the first Convolution layer of the CNN. Besides, we also stack a series of blocks including *ConvBnRelu* sets with 3 layers of Convolution, Batch Normalization, and ReLU, along with the corresponding Pooling layers into the above CNN to ensure the output width is 1. Finally, the output of the CNN will be reshaped as a sequence of features or feature maps to match the input for the language processing part of the model in use.

TABLE III: Text Recognition CNN backbone setup

Layers		**Configurations**	**Output size**
Input		432x48 RGB image	(432,48,3)
Block1	ConvBnRelu	64 filters, 3x3 kernel, padding "same"	(432,48,64)
	MaxPooling	2x2 kernel	(216,24,64)
Block2	ConvBnRelu	128 filters, 3x3 kernel, padding "same"	(216,24,128)
	MaxPooling	2x2 kernel	(108,12,128)
Block3	ConvBnRelu	256 filters, 3x3 kernel, padding "same"	(108,12,256)
	ConvBnRelu	256 filters, 3x3 kernel, padding "same"	(108,12,256)
	MaxPooling	2x2 kernel	(54,6,256)
Block4	ConvBnRelu	512 filters, 3x3 kernel, padding "same"	(54,6,512)
	ConvBnRelu	512 filters, 3x3 kernel, padding "same"	(54,6,512)
	MaxPooling	2x2 kernel	(27,3,512)
Block5	ConvBnRelu	512 filters, 2x2 kernel, no padding	(26,2,512)
	ConvBnRelu	512 filters, 2x2 kernel, no padding	(25,1,512)

3) Language processing part setup:

Text inputs will be preprocessed by retaining only strings containing Hán-Nôm characters (no Latin letters, numbers, punctuations, or unnecessary characters). In general, Machine Learning or Deep Learning models usually do not process text directly, so we encoded characters in the above strings into numeric forms or tokens. Except for *CRNNxCTC*, which uses feature maps as direct input to the language processing part, the remaining models will add 2 "[START]" and "[END]" tokens to represent the beginning and end of the text. Finally, all tokens will continue to be encoded to vectors by an Embedding layer with an output dimension of 512. Besides, the input strings will be padded to the maximum length using the "[PAD]" token, so they all have a fixed length.

In addition to models that leverage Transformer architecture for the language processing part, for models that use RNNs, we used GRU [21] for our experiments because it is simply a powerful RNN with only 1 hidden state vector instead of 2 state vectors like LSTM (hidden state and cell state). This will make the implementation of architectures more complicated, as well as the total parameters and the time needed to converge can be higher.

4) Optimizers:

We will experiment with 2 optimizers, Adam and Adadelta. For Adam, this is a popular optimizer and often works well when training Deep Learning models. We initialized a learning rate of 2e-4 when using Adam along with a strategy of halving the learning rate every time the loss of the Validate set not improved after 2 epochs. For Adadelta [22], this is a more powerful extension to Adagrad that adjusts the learning rate. As a result, Adadelta can continue to learn even after many updates have been made. Adadelta tends to benefit from higher initial learning rate than other optimizers. Here, we initialized it to exactly 1.0 in the original paper. Thus, when using Adadelta, we do not need to care about adjusting the learning rate to suit the model as well as the training process.

5) Other hyperparameters:

Besides the general settings presented in the above sections, RNN-based models still have another important hyperparameter that needs to be adjusted individually to achieve good performance for the training process: the number of neurons or units of the GRU, which plays a key role in deciding the tokens for the output.

TABLE IV: GRU units setup for Text Recognition models

Models	**GRU units**
Init-inject Captioner	512
Attention-based Captioner	1024
CRNNxCTC	256
CNNxSeq2Seq	256

For Transformer-based Recognition models, we made the tuning to some hyperparameters mentioned in the original paper "Attention is all you need" of this architecture [20]:

TABLE V: Transformer-based Recognition models setup

TransformerOCR models	N	d_{model}	d_{ff}	h	P_{drop}
CNNxTransformer	2	512	512	1	0.1
CNNxTransformerDecoder	2	512	512	1	0.1

6) Experiment with Skip Connection:

During the *Pre-training* phase, we found that having the feature map skip-connected to a layer X that has the same dimensions and is located as far away from the feature map as possible significantly improved model performance. Therefore, we also experimented with 2 more fundamental Skip Connection methods, Addition and Concatenation, for the most feasible models (existing above the X layer). Specifically, these models include: *CRNNxCTC* and *CNNxSeq2Seq*, with layer X being the second Bidirectional GRU of the models, and *CNNxTransformer* with X being the final Encoder layer.

VII. Result and Discussion

A. Evaluation metrics

To evaluate more accurately not just for Text Detection but also for End-to-End, we used a new method called CLEval [23]. Specifically, it can evaluate correctly in the case of 2 combined bounding boxes are the labels need to be detected. Meanwhile, the IoU method may not accept either or both boxes because the threshold chosen is usually greater than 0.5. At the text recognition stage, the method will return different metrics depending on the similarity of the prediction with the label. In addition, this method can also be used to evaluate the Text Detection problem only because the specificity of this problem is that there are no text labels in the bounding boxes, so depending on the problem, CLEval will be different in its computational components. The CLEval evaluation method is calculated according to the following formulas:

$$Recall = \frac{\sum_{i=1}^{|G|}(CorrectNum_i^G - GranulPenalty_i^G)}{\sum_{i=1}^{|G|} TextLength_i^G}$$

$$Precision = \frac{\sum_{j=1}^{|D|}(CorrectNum_j^D - GranulPenalty_j^D)}{\sum_{j=1}^{|D|} TextLength_j^D}$$

$$F1 = 2 \times \frac{Recall \times Precision}{Recall + Precision}$$

With Text Recognition problem only, we used the same evaluation metrics as previous related works at sequence level. Evaluation methods were used include:

- Sequence Accuracy: $seq\_acc = \frac{\sum_i TP_i(sequence)}{Total\ sequence}$
- Character Accuracy: $char\_acc = \frac{\sum_i \sum_j TP_{i,j}(character)}{\sum_i Total_i(character)}$
- Character Error Rate: $cer = \frac{Substitute+Delete+Insert}{Target\ text\ length}$

B. Text Detection experimental results

The NomNaOCR dataset is built on historical documents that are poetry and prose. As for poetry, the structure of these documents is quite similar among Pages because they are all written in six-eight meters. Meanwhile, prose documents often have a much more complex and heterogeneous structure. Besides, Text Detection can simply be understood as a problem of interest in the placement of the text appearing in the image, so we evaluated this problem using the above metrics in 3 ways: on the entire images, on the images of the poetic documents, and on prose images of the Validate set. In addition, we also found that there are differences between each document in the NomNaOCR dataset regarding the condition of the image or the text structure, so we conducted a detailed analysis for each document too.

1) General results:

TABLE VI: General experimental results for Text Detection

Types	Models	Precision	Recall	F1-score
Entire	DBNet	**0.9991**	**0.9939**	**0.9965**
	EAST	0.9910	0.9844	0.9877
Poetry	DBNet	**1.0000**	**0.9998**	**0.9999**
	EAST	0.9994	0.9959	0.9976
Prose	DBNet	**0.9989**	**0.9926**	**0.9957**
	EAST	0.9891	0.9818	0.9855

Table VI presents the general results for the 3 mentioned evaluation ways (Entire, Poetry, and Prose). Text Detection models we used here were DBNet and EAST, which give excellent results with the efficiency of all metrics over 98%. Specifically, DBNet give better results than EAST when it excelled in all 3 evaluation ways for the Validate set of the NomNaOCR dataset with an F1-score of 0.9965 on entire images, 0.999 on poetic images, and 0.9957 on prose images. In addition, it can also be seen that both DBNet and EAST models predict poetic images better than prose but they are not significantly different. Therefore, we conclude that the complex structure of prose affects the experimental results.

2) Results by documents:

TABLE VII: Detailed results by documents of DBNet

Documents	Precision	Recall	F1-score
Lục Vân Tiên	1.0000	0.9991	0.9996
Truyện Kiều version 1866	1.0000	1.0	**1.0000**
Truyện Kiều version 1871	1.0000	1.0	**1.0000**
Truyện Kiều version 1872	1.0000	0.9998	0.9999
ĐVSKTT-Quyển Thủ	0.9974	0.9822	0.9898
ĐVSKTT-Ngoại kỷ toàn thư	0.9997	0.9915	0.9956
ĐVSKTT-Bản kỷ toàn thư	0.9992	0.9945	**0.9968**
ĐVSKTT-Bản kỷ thực lục	0.9986	0.994	0.9963
ĐVSKTT-Bản kỷ tục biên	0.9988	0.9882	0.9934

Table VII presents the results of DBNet for each document in the Validate set. Again, the results obtained on poetry documents are better than prose but they are not significantly different. Specifically, the poetry documents are Truyện Kiều versions 1866 and 1871, giving an impressive F1-score up to 1.0. For prose, although Bản kỷ of ĐVSKTT has the largest number of Pages, DBNet still gives better results compared to the 4 remaining parts of ĐVSKTT with an F1-score of 0.9968. In contrast, Quyển Thủ of ĐVSKTT has the smallest number of Pages but has the lowest F1-score with 0.9898, which is still a very good value, though. In general, DBNet gives very high results on poetic and prose images.

TABLE VIII: Detailed results by documents of EAST

Documents	Precision	Recall	F1-score
Lục Vân Tiên	0.9988	0.9997	0.9993
Truyện Kiều version 1866	0.9997	0.9997	**0.9997**
Truyện Kiều version 1871	0.9989	1.0000	0.9994
Truyện Kiều version 1872	1.0000	0.9864	0.9931
ĐVSKTT-Quyển Thủ	0.9760	0.9531	0.9644
ĐVSKTT-Ngoại kỷ toàn thư	0.9730	0.9616	0.9672
ĐVSKTT-Bản kỷ toàn thư	0.9901	0.986	0.9880
ĐVSKTT-Bản kỷ thực lục	0.9929	0.9867	**0.9898**
ĐVSKTT-Bản kỷ tục biên	0.9889	0.9773	0.9831

As for the EAST model, it can also be found that the results obtained on poetic documents are better than prose but there is no significant difference. In which, the poem Truyện Kiều version 1866 give the highest result with an F1-score of 0.9997. With prose, Bản kỷ thực lục of ĐVSKTT has the highest F1-score of 0.9898. Besides, Quyển Thủ also has the lowest results for this model as for DBNet with an F1-score of 0.9644. In general, EAST gives high results on poetic and prose images.

C. Text Recognition experimental results

1) Pre-training results:

TABLE IX: Text Recognition Pre-training results on Synthetic Nom String part of IHR-NomDB

Models	Optimizer	seq_acc	char_acc	cer
Init-inject Captioner	Adam	0.1021	0.6927	0.2908
Attention-based Captioner	Adam	0.7759	0.9729	0.0252
CRNNxCTC	Adadelta	0.9150	0.9892	0.0105
CRNNxCTC + *Concat Connection*	Adadelta	**0.9485**	**0.9936**	**0.0062**
CNNxSeq2Seq	Adam	0.6854	0.9580	0.0401
CNNxSeq2Seq + *Concat Connection*	Adam	0.8449	0.9825	0.0168
CNNxTransformer	Adadelta	0.8726	0.9845	0.0133
CNNxTransformer + *Add Connection*	Adadelta	0.8895	0.9859	0.0115
CNNxTransformer Decoder	Adadelta	0.6414	0.9034	0.0620

As can be seen with the Validate set of the Synthetic Nom String dataset, the *Init-inject Captioner* model, the results are pretty bad, with only 0.1021 for seq_acc, even with a metric that can easily reach a high value like char_acc, this model also just gives a weak result with only 0.6927, which obviously also leads to a relatively bad CER value of 0.2908. All remaining models have quite high char_acc (> 0.9).

About the optimizers, *CRNNxCTC* models give the best results using Adadelta, similar to the experiment proposed in its original paper [17]. Another noticeable thing is that the experiment with Adadelta proposed by us for models that need to schedule the learning rate carefully for efficient convergence like Transformer, is a right thing to do when these models have pretty good results. Thanks to that, when training Transformer-based models, we do not need to use traditional scheduler nor worry about tuning the learning rate.

Besides that, from the 3 possible models for experimenting with our proposed Skip Connection, including *CRNNxCTC*, *CNNxSeq2Seq*, and *CNNxTransformer*, we can easily see the effectiveness of this experiment. For the original version (no Skip Connection) of these 3 models, *CRNNxCTC* has the best results but is not significant compared to *CNNxTransformer*, and it takes a long time to train (more than 5 hours) to get that result. With our proposed Skip Connection, these models all give better results than their original versions. In which, the most notable is *CRNNxCTC* using Concatenation that outperformed all metrics with seq_acc reaching 0.9485 and an almost absolute char_acc of 0.9936, so it is easy to understand when the CER value is just as good with only 0.0062. Nearly 6% of seq_acc omitted probably belonged to sequence types containing characters that appear only once in the entire dataset. The *CRNNxCTC* model also has quite a long difference in results compared to the remaining models. This is also the best model for our Pre-training phase with parameters just greater than its original version, which is also the model with the fewest parameters.

2) Fine-tuning and Retraining results:

From the results of Pre-training phase in Table IX, we compare models of the 4 approaches in Section V and remove ones with bad results, but the above comparison will not apply between the models experimented with our Skip Connection and its original version. Specifically, we removed *Init-inject Captioner* and *CNNxTransformerDecoder*, the remaining models will be used in both Fine-tuning and Retraining phases.

TABLE X: Text Recognition Fine-tuning and Retraining results on the NomNaOCR dataset

Models	Phases	seq_acc	char_acc	cer
Attention-based Captioner	Finetune	0.1326	0.7680	0.2234
	Retrain	0.1024	0.7121	0.2714
CRNNxCTC	Finetune	0.2627	0.8325	0.1617
	Retrain	**0.2941**	0.8473	0.1508
CRNNxCTC + *Concat Connection*	Finetune	0.2266	0.8060	0.1823
	Retrain	0.2423	0.8189	0.1699
CNNxSeq2Seq	Finetune	0.1095	0.7067	0.2711
	Retrain	0.1057	0.6995	0.2868
CNNxSeq2Seq + *Concat Connection*	Finetune	0.1837	0.8176	0.1728
	Retrain	0.0979	0.6915	0.2815
CNNxTransformer	Finetune	0.1963	0.7910	0.1771
	Retrain	0.1452	0.7013	0.2521
CNNxTransformer + *Add Connection*	Finetune	0.2714	**0.8490**	**0.1335**
	Retrain	0.2423	0.8016	0.1664

In general, except for *CRNNxCTC* models, the remainings have better results when performing Fine-tuning from the learned weights of the Synthetic Nom String dataset than when Retraining from scratch. In addition, when considering the above Fine-tuning on 3 possible models for Skip Connection experiments, our proposal is once again reinforced when these experiments are all better compared to their original versions but once again exclude *CRNNxCTC*. We conclude that with the NomNaOCR dataset, CRNNxCTC only really gives good results when it is retrained from scratch and has no Skip Connection. Specifically, with these settings, *CRNNxCTC* achieved the highest seq_acc in all models with 0.2941.

Another noticeable thing is that the *CNNxSeqSeq* model performed quite badly compared to the other models when retrained from scratch, with only 0.0979 for seq_acc and 0.6915 for char_acc. Even when using the Skip Connection we proposed, the model's CER remained unchanged and had the highest value compared to the others, up to 0.2868. However, the implementation of Fine-tuning combined with our Skip Connection experiments has significantly improved this model performance, as mentioned in the paragraph above. *CNNxSeqSeq* is also architecture with a long convergence time. For the 3 metrics used, in addition to *CRNNxCTC* having the highest seq_acc, the model with the best results for 2 remaining metrics is *CNNxTransformer* when fine-tuned and used the Addition to skip-connect, with 0.8490 for char_acc and 0.1335 for CER respectively. However, these metrics do not significantly differ between the 2 models above. So, we consider them to be the 2 best models for the Text Recognition problem on the NomNaOCR dataset we built.

D. End-to-End experimental results

In this section, we evaluate the combination of 2 problems, detecting and recognizing Hán-Nôm sequences in the Validate images of the NomNaOCR dataset using the End-to-End metrics mentioned in VII-A. To perform this evaluation, we put the Pages into the Text Detection models, which are DBNet and EAST, to obtain the output Patches. Then, we use these Patches as input for the 4 best models corresponding to our 4 approaches for the Recognition problem. Finally, we get the output labels of the bounding boxes and the corresponding Hán-Nôm sequences in each box.

TABLE XI: End-to-End results on NomNaOCR Validate set

Combinations		Finetune	Precision	Recall	F1-score
DBNet	Attention-based Captioner	✓	0.6750	0.6741	0.6745
	CRNNxCTC		**0.8356**	**0.8293**	**0.8324**
	CNNxSeq2Seq + *Concat Connection*	✓	0.7524	0.7469	0.7496
	CNNxTransformer + *Add Connection*	✓	0.8065	0.8062	0.8064
EAST	Attention-based Captioner	✓	0.7077	0.7144	0.7110
	CRNNxCTC		**0.8735**	**0.8755**	**0.8745**
	CNNxSeq2Seq + *Concat Connection*	✓	0.7886	0.7981	0.7933
	CNNxTransformer + *Add Connection*	✓	0.8519	0.8603	0.8561

As shown in Table XI, the combinations of EAST give much better results than the combinations of DBNet, about 3-5% F1-score. It can be seen that the *CRNNxCTC* model has the best results in all combinations of DBNet, with an F1-score of 0.8324. However, when combined with EAST, this model outperforms all other combinations with an F1-score of 0.8745. Besides, the combinations involving the *Attention-based Captioner* when Fine-tuning had the worst results, with F1-scores of 0.6745 (DBNet) and 0.7110 (EAST), respectively. In addition, other combinations, including DBNet/EAST and *CRNNxCTC*, EAST and *CNNxTransformer* when Fine-tuning along with using our proposed Skip Connection, give pretty good results when all have the F1-scores above 0.8. In general, the combinations of EAST model are much better than DBNet. The surprising thing here is that DBNet gives better results than EAST in the evaluation results for Text Detection only, as seen in VII-B1. But when combined with Text Recognition models, EAST is outperformed.

This can be explained that the generated bounding boxes of the DBNet are usually smaller than the labels (but not too much). Or in other words, the DBNet bounding boxes wrap the image regions containing the text very tight or close. With peculiarity dataset like NomNaOCR, being wrapped so tightly leads to the Hán-Nôm characters in Patch being very easily out of focus when they are cut from the original image. This is highly harmful to these characters because losing just a small stroke can turn one Hán-Nôm character into another.

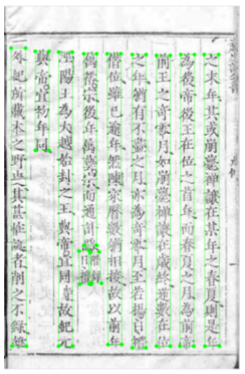


Figure 15: The output bounding boxes of DBNet

In contrast, the predicted bounding box of EAST is much larger than the label, so it leads to the fact that EAST has a lower Text Detection performance than DBNet when the generated boxes are not as tight as DBNet. But also because of this weakness, EAST is more informative and helps to avoid missing strokes, which is extremely important not only for handwritten but also for hieroglyphs.

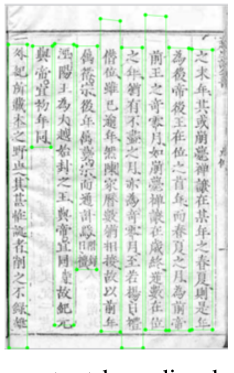


Figure 16: The output bounding boxes of EAST

As well as the contrasting results between the specific evaluation of the Text Detection problem and the End-to-End evaluation of EAST and DBNet, for the Text Recognition problem only, when evaluating the fine-tuned and skip-connected *CNNxTransformer* on Validate set of NomNaOCR specifically for the Recognition task, the results are not much different and mostly better than *CRNNxCTC* as seen in VII-C2. But when combined with Detection models, *CRNNxCTC* has better results. Especially with the combination of EAST and *CRNNxCTC*, although it is not much different from the combination of EAST and *CNNxTransformer* (Fine-tuning and Skip Connection), the results still completely outperformed to remaining combinations. With this result, it can be seen that *CRNNxCTC* is more suitable for reality when the bounding boxes are not as good and accurate as when annotated.

E. Error analysis

1) For Text Detection:

For poetic documents, DBNet is our best model when detecting almost completely correct. There is only one case type, like in Figure 16, which is incorrect when detecting the second sentence of a Page in Lục Vân Tiên. This happened because Patch was blurred and affected by large ink stains.

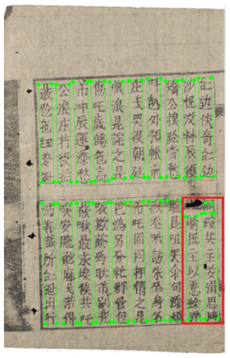


Figure 17: Example for false detection on poem of DBNet

For prose images, DBNet is often wrongly detected 3 cases:

- The model is often confused with texts outside the content of the Page, as shown in Figure 18. Because these Patches not only have the same shape as the Patches in the main content but also have the same size of the characters.

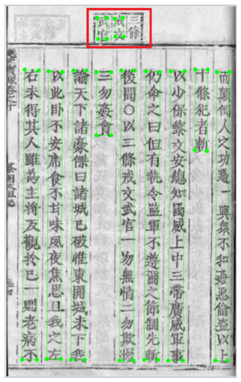


Figure 18: Example for outside texts detection

- Characters are very small and close to each other, as shown in Figure 19, which also confuses the model, leading to missed detection. These very small characters will result in some loss of stroke due to reduced resolution.

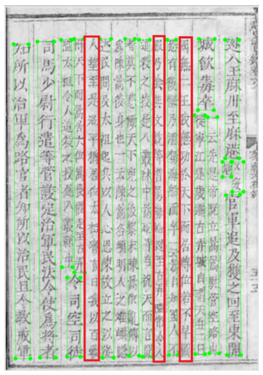


Figure 19: Example for missed detection of texts

- The model is also often undetectable for the characters "一" appear in the first positions of the Patch (Figure 20)

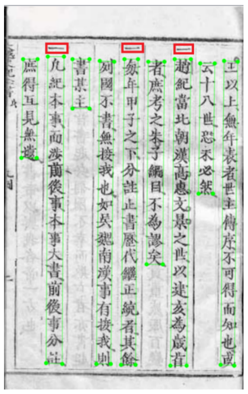


Figure 20: Example for undetectable "一" characters

2) For Text Recognition:

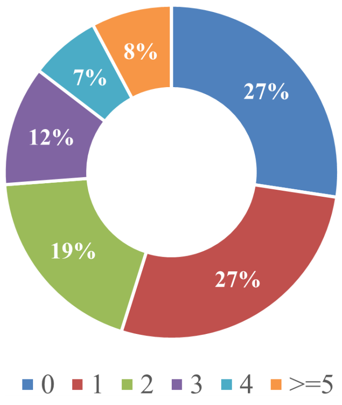


Figure 21: Distribution of wrongly predicted characters in Validate set

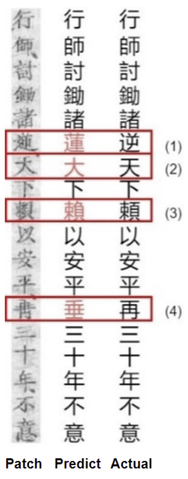

行師討鋤諸蓮大下賴以安平垂三十年不意
行師討鋤諸逆天下賴以安平再三十年不意
(1)
(2)
(3)
(4)
Patch
Predict
Actual

Figure 22: Example of error predictions

With the fine-tuned *CNNxTransformer* using Addition to skip-connect, we have obtained good results for the Text Recognition problem in particular. This model almost only incorrectly predicts 1 to 2 characters in each Patch, as shown in Figure 21. In addition, from Table I, most of text lengths in each Patch are 17 or more. Both of the above also explains why while CER has good results but Sequence Accuracy is very low, as seen in VII-C2. In addition, only a small number of Patches are incorrectly predicted over 3 characters. Figure 22 shows an example of errors that appear in predictions when evaluated on a Patch of Bản kỷ tục biên in Đại Việt Sử Ký Toàn Thư. It can be seen that the cases of incorrect predictions (1), (2), (3), and (4) are very similar in shape or letterfaces to the actual characters. Or with the case (3) in particular, a glance can cause a normal person to confuse these 2 characters. This can happen due to changes in writing between characters (cases 2 and 3).

Besides, we also performed a statistic on the number of characters that appear less (from 1 to 3 times) in the Train set, but the model still correctly predicts on the Validate set, as shown in Table XII. Specifically, in the Validate set with 764 characters that rarely appear in the Train set (appearing from 1 to 3 times), the *CNNxTransformer* model when fine-tuned with our proposed Skip Connection has predicted a total of 187 characters, equivalent to 24.48%.

TABLE XII: The number of characters is still correctly predicted even though it rarely appears in the Train set

Frequency of appearance	**Amount**
1	32
2	70
3	85

VIII. Conclusion and Future Works

A. Conclusion

In this article, we have completed a good and largest dataset in Vietnam at present for Hán-Nôm script named NomNaOCR, including 2953 Pages which are scanned images of old documents, and 38,318 Patch were extracted from these Pages along with corresponding digital Hán-Nôm strings, to serve the 2 problems of detecting and recognizing handwritten Hán-Nôm texts at sequence level.

Besides that, we have successfully implemented Deep Learning models to solve the 2 problems above in many approaches. And also had certain achievements when reaching very high results on the Validate set of the NomNaOCR dataset for the Detection problem with 2 models, EAST and DBNet. They achieved more than 0.98 points for all 3 metrics of Precision, Recall, F1-score, and on all 3 evaluation ways, including Entire images, Poetry, or Prose.

In addition, with 4 proposed approaches to the Recognition problem, besides performing experiments on 3 phases of Pre-training, Fine-tuning, and Retraining, we also got very positive results with Character Accuracy values above 84% for both *CRNNxCTC* models when retrained and *CNNxTransformer* when fine-tuned with our proposed Skip Connection, which is also the model with smallest CER values we achieved, with 13.35% on the Validate set of the NomNaOCR dataset for the Recognition problem.

Moreover, we also performed the combined evaluation (End-to-End) for the above 2 problems and got a quite surprising result when the combination of EAST and *CRNNxCTC* was not much different from EAST and *CNNxTransformer* (Fine-tuning and Skip Connection) but completely outperformed the other combined models with an F1-score of 87.45%. In addition, the error analysis for the best models has also been provided and clarified.

B. Future approaches

Based on the achieved results, it can be seen that for only the Text Detection problem, EAST and DBNet have achieved excellent values with over 98% for all 3 measures of Precision, Recall, and F1-score. Therefore, in the future, we will focus on improving the Text Recognition problem only. From the obtained results of this problem, we see very good performance on the NomNaOCR dataset of 2 models, *CRNNxCTC* and *CNNxTransformer* (Fine-tuning and Skip Connection), so we will research for a way to combine these 2 models by training *CNNxTransformer* in 3 phases mentioned in VI-B1 as normal, but the model will be optimized by CTC Loss like CRNN. Or further is to build an End-to-End model like STN-OCR [24], which can train both tasks of text detection and recognition simultaneously.

Besides, the current sequence generation of all models is simply using Greedy Search to find the tokens with the highest conditional probability at each timestep. However, with this approach, the generated sequence may not be the most probable one. This can be solved by Exhaustive Search, which checks all possible output sequences and returns the one with the highest conditional probability, but the computational cost for this algorithm is too high. So, in the future, we may use an improved algorithm named Beam search to balance computational cost and search quality by keeping N strings with high probability by the best path. Finally, the selected sequence will be the sequence with the highest probability among those N sequences. Or further is to integrate the language model to decode the contextual output of the Text Recognition problem.

In addition, the built NomNaOCR dataset will also be further improved in quantity with many other historical documents of Vietnam. And from what we have collected, we can also expand this dataset to include other problems besides OCR, such as translating old literary style into modern, etc.

References

[1] B. John C. Lee L. Stephen P. John S. D. Neil and N. T. Việt. The vietnamese nôm preservation foundation. http://www.nomfoundation.org. Accessed: 2022-07-01.

[2] Min-Soo Kim, Man-Dae Jang, Hyun-Il Choi, Taik-Heon Rhee, Jin-Hyung Kim, and Hee-Kue Kwag. Digitalizing scheme of handwritten hanja historical documents. In *First International Workshop on Document Image Analysis for Libraries, 2004. Proceedings.*, pages 321–327, 2004.

[3] Cheng-Lin Liu and Yue Lu. *Advances in chinese document and text processing*, volume 2. World Scientific, 2017.

[4] Anh Duc Le, Daichi Mochihashi, Katsuya Masuda, Hideki Mima, and Nam Tuan Ly. Recognition of japanese historical text lines by an attention-based encoder-decoder and text line generation. In *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*, HIP '19, page 37–41, New York, NY, USA, 2019. Association for Computing Machinery.

[5] Truyen Phan, Kha Nguyen, and Masaki Nakagawa. A nom historical document recognition system for digital archiving. *Int. J. Doc. Anal. Recognit.*, 19(1):49–64, mar 2016.

[6] Truyen Van Phan, Bilan Zhu, and Masaki Nakagawa. Development of nom character segmentation for collecting patterns from historical document pages. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, page 133–139, New York, NY, USA, 2011. Association for Computing Machinery.

[7] Manh Tu Vu, Van Linh Le, and Marie Beurton-Aimar. Ihr-nomdb: The old degraded vietnamese handwritten script archive database. In *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part III*, page 85–99, Berlin, Heidelberg, 2021. Springer-Verlag.

[8] Kha Cong Nguyen, Cuong Tuan Nguyen, and Masaki Nakagawa. Nom document digitalization by deep convolution neural networks. *Pattern Recognition Letters*, 133:8–16, 2020.

[9] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazàn Almazàn, and Lluís Pere de las Heras. Icdar 2013 robust reading competition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1484–1493, 2013.

[10] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, Faisal Shafait, Seiichi Uchida, and Ernest Valveny. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160, 2015.

[11] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: An efficient and accurate scene text detector. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2642–2651, 2017.

[12] Minghui Liao, Yan Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-time scene text detection with differentiable binarization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:11474–11481, 04 2020.

[13] Rois-ds center for open data in the humanities, "kuzushiji recognition," 2019. [online]. Available: https://www.kaggle.com/competitions/kuzushiji-recognition. Accessed: 2022-02-25.

[14] Albert Gatt Marc Tanti and Kenneth P. Camilleri. Where to put the image in an image caption generator. *Natural Language Engineering*, 24(3):467–489, 2018.

[15] Zbigniew Wojna, Alexander N. Gorban, Dar-Shyang Lee, Kevin Murphy, Qian Yu, Yeqing Li, and Julian Ibarz. Attention-based extraction of structured information from street view imagery. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 844–850, 2017.

[16] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2048–2057. JMLR.org, 2015.

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

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

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

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[21] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics.

[22] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012.

[23] Youngmin Baek, Daehyun Nam, Sungrae Park, Junyeop Lee, Seung Shin, Jeonghun Baek, Chae Young Lee, and Hwalsuk Lee. Cleval: Character-level evaluation for text detection and recognition tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2404–2412, 2020.

[24] Christian Bartz, Haojin Yang, and Christoph Meinel. Stn-ocr: A single neural network for text detection and text recognition. *ArXiv*, abs/1707.08831, 2017.