Localization of Sino-Nom characters on woodblock-printed images

Chi Thanh Nguyen
Faculty of Information Technology
VNU University of Engineering and Technology
Hanoi, Vietnam
22021123@vnu.edu.vn

Ngoc Hiep Nguyen
Faculty of Information Technology
VNU University of Engineering and Technology
Hanoi, Vietnam
22021169@vnu.edu.vn

***Abstract*—There are many documents written in Sino-Nom capturing a wide range of historical, political, and literary facets of Vietnamese culture. To contribute to the process of preserving those documents, we focus on the task of building Optical Character Recognition models for Sino-Nom characters. Our OCR model was evaluated on the given dataset, and it achieves mAP50-95 scores of over 0.86 and mAP25-75 scores of over 0.99.**

***Index Terms*—Sino-Nom OCR, Sino-Nom text localization**

I. Introduction

Sino-Nom, an ancient ideographic vernacular script, was adopted as the national script of Vietnam after its independence from China in 939 CE. Throughout the next millennium from the 10th century to the 20th, Sino-Nom became the primary script for Vietnamese literature, philosophy, history, law, medicine, religion, and government policy. Sino-Nom woodblock is a type of document record created by carving Sino-Nom document content in reverse onto a board of wood. These woodblocks were used to print books in Vietnamese in the 19th and early 20th centuries, as depicted in Figure 1. The contents of these woodblocks vary from official literature to classic and historical books.

Digitization provides a way to preserve these Sino-Nom documents for future generations and to make them accessible to scholars and students around the world. One major problem with the digitization process is that it is a very time-consuming task and there is a shortage of Sino-Nom specialists. Thus, Optical Character Recognition (OCR) systems can be applied to contribute to this digitization process. To make use of those OCR systems, there is also a need for a considerable amount of labeled data to effectively train machine learning models. In this paper, we are proposing the way we build a model for our OCR systems to localize Sino-Nom from document images.

II. Dataset

We are provided a dataset with 80 document images with their corresponding labels. Our dataset is divided into two groups: train and val. The train group has 70 images and the val group has just 10 images. Figure 2 is an example of how we label an arbitrary image.

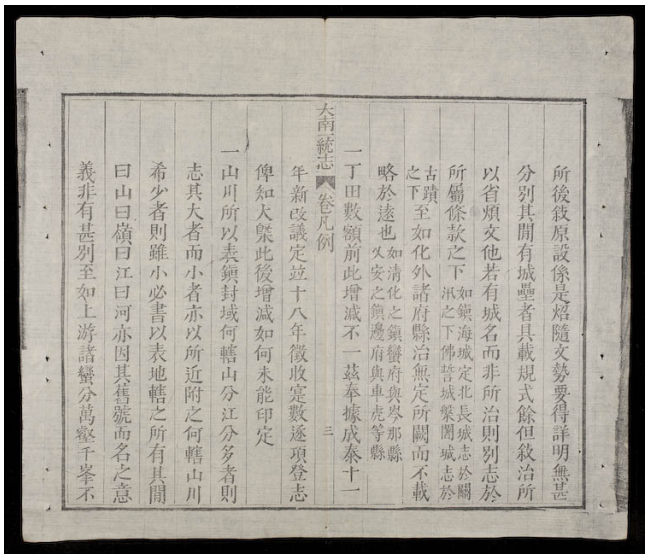

Fig. 1. Example of an image.

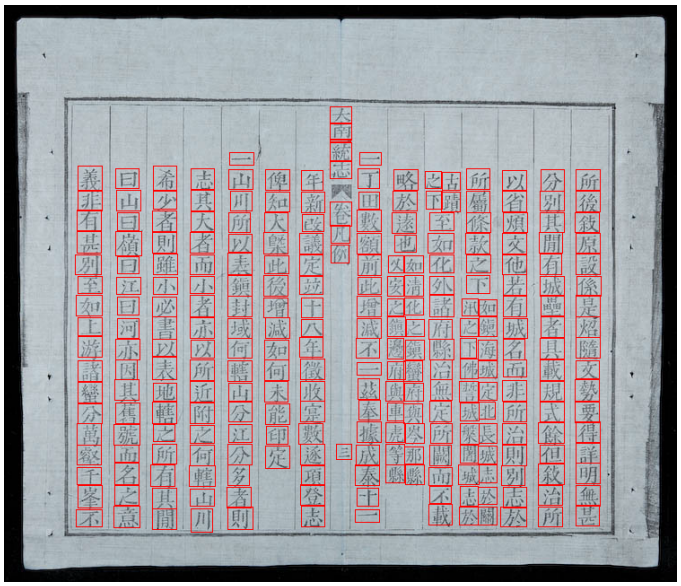

Fig. 2. Labeling an arbitrary image.

Details of our dataset are noted in Table I.

TABLE I
STATISTICS OF DATASET

Group	Avg Width	Avg Height	Avg Letter Frequencies
Train	852.86	619.46	195.27
Val	850	620.7	195.6

III. METHODS

Our OCR model has to localize Sino-Nom in the given dataset, so we try and conduct experiments to compare two major approaches to the general object detection problem: the regression-based approach and the segmentation-based approach. While the regression-based approach aims to train a regression model to directly achieve the locations of objects, segmentation-based approaches tend to calculate the probability of each pixel being the object needed for detection. We use YOLOv8 as a representative of the two above approaches.

YOLO (You Only Look Once) is a state-of-the-art object detection that was introduced in 2016 by Joseph Redmon. The YOLOv8 (YOLO version 8) architecture is a fully convolutional neural network that uses a backbone feature extractor and a set of detection heads to predict object bounding boxes and class probabilities. YOLOv8 models are implemented on the wisely-used platform PyTorch, making them easier to use.

A summary of YOLOv8 architecture is shown in Figure 3.

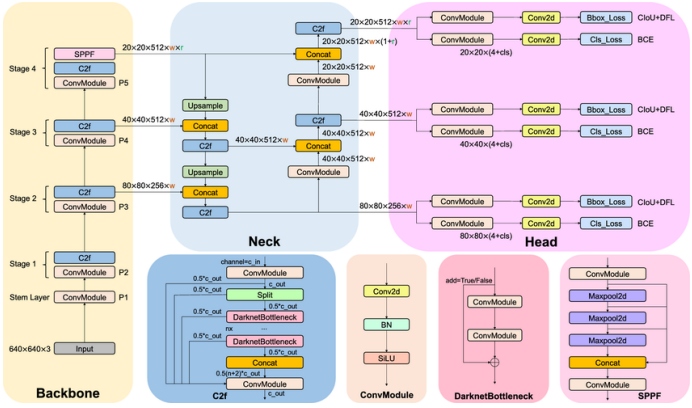

Fig. 3. YOLOv8 Architecture.

IV. EXPERIMENTS AND RESULTS

A. Environments

All experiments run on the Google Colaboratory platform. Some of its hardware specifications are listed as follows:

- CPU: Intel Xeon with 2 vCPUs.
- RAM: 13GB.
- GPU: NVIDIA Tesla T4 (12 GB VRAM).
- Storage: 78GB of SSD storage.

B. Localization experiments and results

We conduct different experiments using various pre-trained models of YOLOv8. All models are trained for 100 epochs over 80 labeled document images. We use five pre-trained models and fine-tune them:

- Nano model: YOLOv8n.pt
- Small model: YOLOv8s.pt
- Mediium model: YOLOv8m.pt
- Large model: YOLOv8l.pt
- XLarge model: YOLOv8x.pt

The hyperparameters we choosing are as follows:

- Batch size: 8
- Learning rate: 0.002
- Momentum: 0.9
- Weight decay: 0.0005
- Optimizer: Adam
- Image size: $640 * 640$
- IoU threshold: 0.7
- Epochs: 100

The metrics used for model evaluation are mAP50-95 and mAP25-75. Table II shows the detailed results of those experiments on the train dataset, and Table III shows the detailed results on the val dataset.

TABLE II
RESULTS ON THE TRAIN DATASET

Model	Memory	mAP25-75	Avg Time per Image
YOLOv8n	6.5MB	0.990099	0.9s
YOLOv8s	22.6MB	0.994079	1.3s
YOLOv8m	52.1MB	0.995257	2.5s
YOLOv8l	87.8MB	0.995018	3.9s
YOLOv8x	136.9MB	0.993491	5.7s

TABLE III
RESULTS ON THE VAL DATASET

Model	mAP50-95	mAP25-75	Avg Time per Image
YOLOv8n	0.832	0.986705	1s
YOLOv8s	0.854	0.991148	1.4s
YOLOv8m	0.867	0.993076	2.5s
YOLOv8l	0.868	0.993294	3.9s
YOLOv8x	0.871	0.995837	5.8s

As we can see, our mAP25-95 scores reach roughly 0.99, and an image needs less than 10 seconds to be localized.

Details of our training progress are contained in these links, because of limited resources.

- https://colab.research.google.com/drive/1rs5D-pISvZ6c8yzIdACeUTjhTb6K5ndi?usp=sharing
- https://colab.research.google.com/drive/1rIuonfmxxEOmieFFvgiRgkOB0oE0fuJB?usp=sharing

C. Encapsulation

Because of the 10MB constraint, we choose the smallest model YOLOv8n with just 6.5 MB of memory to detect the

document images. We have published our code at https://github.com/flashhhhh/INT3404---CharacterLocalization/. The instructions for running our code are attached in README.md file.

Also, we have encapsulated the necessary code into zip file, and it costs only 7.1MB. Our requirement libraries are listed in the requirement.txt file.

D. How to run our code

1) Evaluate models score: .

You need to prepare a dataset with images folder and corresponding labels folder, that is, an arbitrary image in the images folder must have the same name label file in the labels folder. Then type this command:

$$python3\ main.py\ /path/to/images\ /path/to/labels$$

Then the console will display the mAP score with your dataset and your execution time.

2) Get the localization images for your images: .

Prepare your images folder. Then type this command:

$$python3\ display.py\ /path/to/images$$

Then the localization images will be in the folder named 'predict'.

V. Conclusion

In this paper, we propose a way to localize Sino-Nom in the given images with YOLOv8 architecture. Our localization models achieve mAP25-75 scores of over 0.99, and mAP50-95 scores of over 0.87.

Since the mAP50-95 scores are not as high as we expected, our work could be extended in several ways. Here are some ways we can approach in the future:

- Generate more images in our dataset. Our dataset has just 80 images, so it affects significantly in our models.
- More experiments with various architectures should be carried out to improve our localization models.
- With the boundary boxes after localization, we can generate classification models to detect what these letters are. That helps to translate and understand these images.

References

[1] *YOLOv8 documentation*. URL: https://docs.ultralytics.com/

[2] *YOLOv8 Github repository*. URL: https://github.com/ultralytics/ultralytics/

[3] Trong Tuan Dao, Cong Thuong Le, Thi Duyen Ngo, and Thanh Ha Le. "Detection and Recognition of Sino-Nom Characters on Woodblock-Printed Images," in 2023 15th International Conference on Knowledge and Systems Engineering (KSE).

[4] *Mean Average Score library*. URL: https://lightning.ai/docs/torchmetrics/stable/detection/mean_average_precision.html