

# ĐẠI HỌC QUỐC GIA HÀ NỘI

## TRƯỜNG ĐẠI HỌC CÔNG NGHỆ



# BÁO CÁO BÀI TẬP LỚN XỬ LÝ ẢNH INT3404E 20 NĂM HỌC 2023 – 2024

## Tên báo cáo:

# OBJECT LOCALIZATION FOR SINO NOM'S CHARACTER WITH YOLO FRAMEWORK

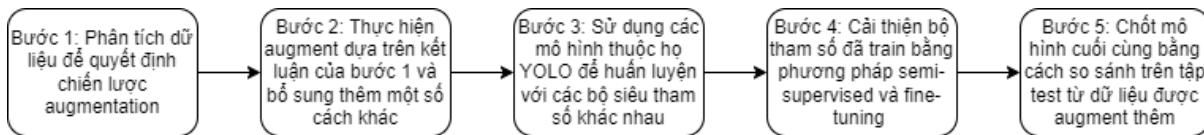
Nhóm 2: Tăng Vĩnh Hà - K67-CA-CLC2  
Lê Thị Hải Anh - K67-CA-CLC2  
Vũ Nguyệt Hằng - K67-CA-CLC2  
Lê Xuân Hùng - K67-CA-CLC2

Giảng viên hướng dẫn: GS. Lê Thanh Hà  
ThS. Nguyễn Công Thương

HÀ NỘI - 2024

# 1 Tóm tắt

Nhóm đã có một số thay đổi về kế hoạch so với tiến độ được đề ra trong proposal reports vì gặp khó khăn trong việc thử nghiệm nhiều mô hình nên đã quyết định sử dụng chiến lược: chỉ sử dụng các mô hình thuộc họ YOLO nhưng bổ sung thêm nhiều dữ liệu được augment, cải thiện bằng cách thử nghiệm phương pháp semi-supervise và để các thành viên train độc lập với các bộ hyperparameters khác nhau để tìm ra bộ tham số tốt nhất. Các bước thực hiện project của nhóm được tiến hành như hình dưới tóm tắt:



Hình 1: Sơ đồ pipeline cơ bản của nhóm

## 2 Phương pháp

### 2.1 Phân tích dữ liệu

Trước khi train model, nhóm đã thực hiện phân tích qua về dữ liệu của label trong tập train để đưa ra các chiến lược augmenting data. Nhóm đã thực hiện 3 hướng chính để phân tích dữ liệu được tóm tắt như sau:

- **Hướng tiếp cận 1:** Khảo sát các thông tin chung của bộ dữ liệu nhận được: số lượng ảnh, kích cỡ ảnh, đặc điểm chung các bức ảnh.
- **Hướng tiếp cận 2:** Khảo sát các tham số của các bounding box: chiều dài - rộng và tọa độ. Thực hiện bằng cách: vẽ các biểu đồ histogram từng aspect (tọa độ x - y, chiều dài - rộng của bounding box) trên toàn bộ tập ảnh để rút ra 2 insight: đặc điểm của từng tập dữ liệu val - train và so sánh distribution giữa các tham số đó giữa tập train - test.
- **Hướng tiếp cận 3:** Tập trung vào các ảnh đặc biệt khiến kết quả model kém. Trong quá trình train model thành nhiều lần, nhóm nhận ra model ở các giai đoạn đầu (khi được train với số lượng epoch ít) thì làm kém trên những bức ảnh "mất cân bằng" về mức độ tập trung của ký tự trong ảnh. Nên nhóm đã đưa ra giả thuyết là: model sẽ được cải thiện hơn khi được "học" trên nhiều bức ảnh như vậy.

#### 2.1.1 Kết quả phân tích 1: Khảo sát các thông tin chung của bộ dữ liệu

Kết quả phân tích như sau:

- Số lượng ảnh: 70 ảnh tập train, 10 ảnh tập val.
- Kích cỡ ảnh: quan sát các ảnh trên tập dữ liệu train, kích cỡ các ảnh có khác nhau nhưng chủ yếu có 3 nhóm kích cỡ dựa theo tên của các ảnh (Ví dụ tên ảnh là "nlvnpf-0137-01-001" thì ảnh thuộc nhóm kích cỡ "0137"). Cụ thể hơn 3 nhóm có kích cỡ W x H như sau:

- Nhóm 0137: 900 x 608
- Nhóm 0140: 750 x 640
- Nhóm 0174: 800 x 632

Mỗi nhóm có chung chỉ số W còn H giao động quanh số đã nêu ở trên.

- Đặc điểm chung các bức ảnh: nhìn chung các bức ảnh rõ nét - không bị nhiễu - không bị xoay (được chụp thẳng đứng), bình thường và cùng có chung format là 2 trang giấy.
- Đặc điểm của file label:
  - Tọa độ x, y của các bounding box. Các tọa độ đã được chuẩn hóa nên  $x, y \in [0, 1]$ .
  - Chiều dài, rộng w, h của các bounding box. Chiều dài, rộng cũng đã được chuẩn hóa nên  $w, h \in [0, 1]$ .

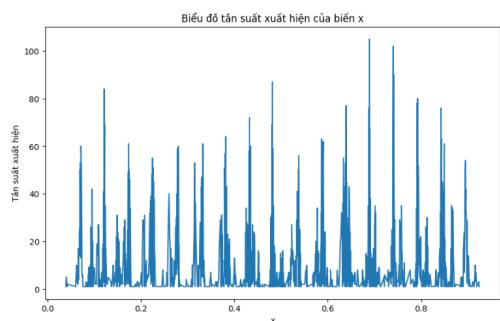
### Kết luận rút ra sau khi phân tích:

- Số lượng ảnh train và val còn quá ít, nhóm quyết định sẽ thêm vào tập train với số lượng ảnh được augment là: 70 (bằng số lượng ảnh trong tập train ban đầu).
- Các bức ảnh có các kích cỡ khác nhau theo từng cách đặt tên khác nhau, nhóm quyết định sẽ thử nghiệm 2 cách cùng resize về 1 kích cỡ (thực hiện bởi framework Ultralytics khi train mô hình YOLO): 640 x 640 và 1280 x 1280.
- Thực hiện các chiến lược augment đa dạng được nhắc đến phía sau với thư viện Albumentation để bổ sung vào tập train.

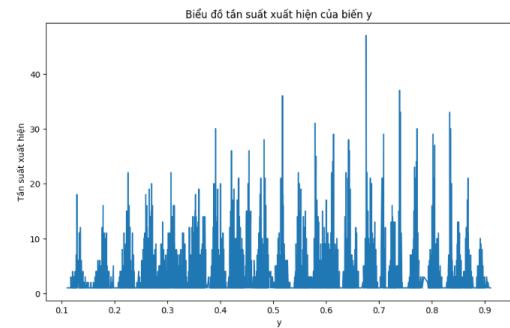
#### 2.1.2 Kết quả phân tích 2: Khảo sát các tham số của các bounding box

**Kết quả phân tích như sau:** các tham số của bounding box trên toàn bộ 70 ảnh tập train được thể hiện qua 4 biểu đồ histogram như sau: trục Ox là giá trị của các tham số và trục Oy là giá trị tần suất xuất hiện (số lần xuất hiện từng thuộc tính đang xét trên toàn bộ 70 ảnh):

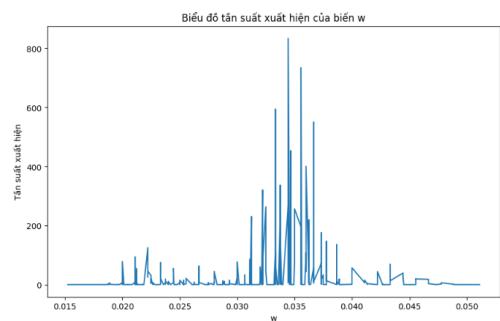
- Phân bố tọa độ x, y: tọa độ  $x, y$  của các kí tự phân bố khá đều, nhìn biểu đồ ta thấy tập trung nhiều nhất trong tọa độ  $x \in [0.6, 0.7]; y \in [0.65, 0.7]$  thể hiện qua hình 2 và hình 3.
- Phân bố của chiều dài và chiều rộng của tất cả các bounding box: tập trung từ 0.03 đến 0.045 thể hiện qua hình 4 và hình 5.
- So sánh các tham số nói trên giữa tập train và val: nhóm đã vẽ các biểu đồ Histogram của các tham số trên của tập labels train và tập labels val, và thấy 2 tập có distribution không quá khác nhau (so sánh phân bố của tọa độ x qua hình 2 và hình 6; phân bố của tọa độ y qua hình 3 và hình 7).



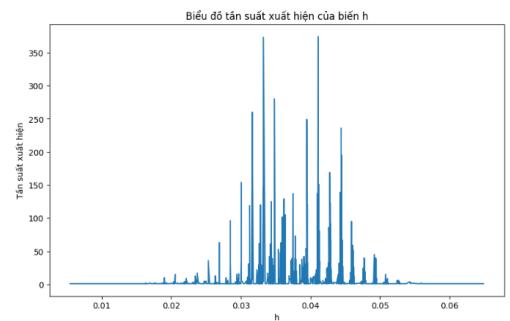
Hình 2: Phân bố của tọa độ x của tập train



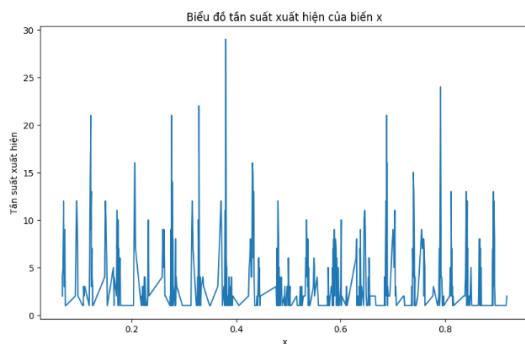
Hình 3: Phân bố của tọa độ y của tập train



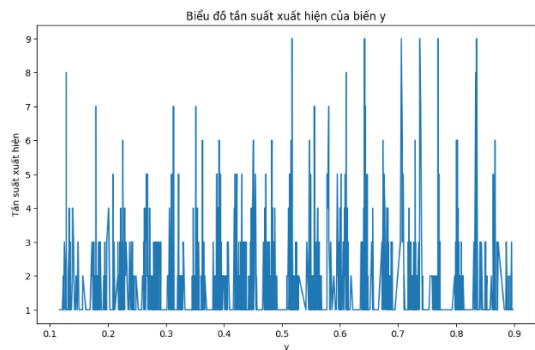
Hình 4: Phân bố của chiều rộng các bounding box trong tập train



Hình 5: Phân bố của chiều dài các bounding box trong tập train



Hình 6: Phân bố của tọa độ x của tập validation



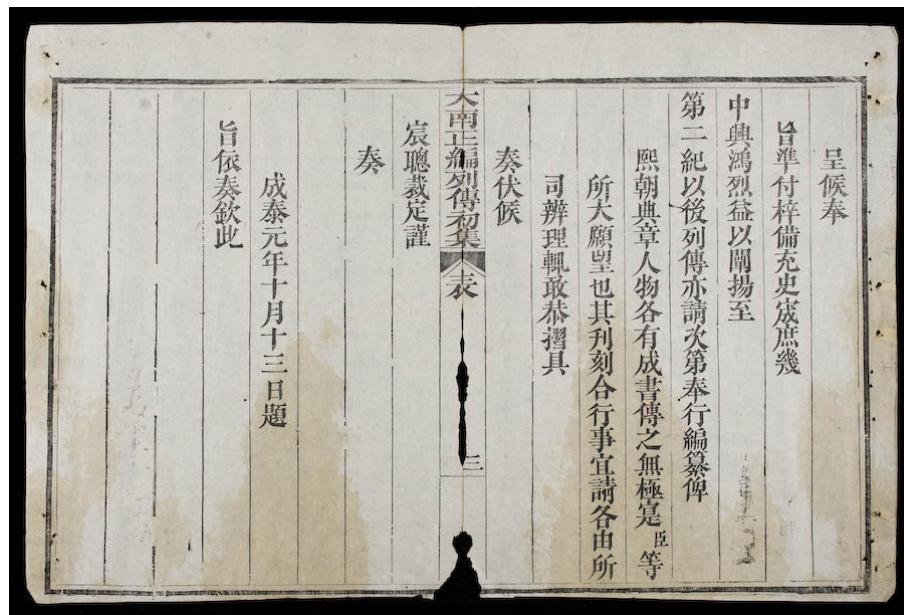
Hình 7: Phân bố của tọa độ y của tập validation

**Kết luận rút ra sau khi phân tích:** các tham số của bounding box vừa phân tích ổn, phân bố đều, không phải tác động quá nhiều bằng các phương pháp augment như: crop một phần của ảnh.

### 2.1.3 Kết quả phân tích 3: Tập trung vào các bức ảnh có khả năng khiến model kém

Như đã giải thích ở trên, nhóm đã nghiên cứu và đưa ra giải pháp sau:

- **Ý tưởng chung:** xét trên từng ảnh xem các bounding box của ảnh có tập trung quá nhiều vào 1 nửa của ảnh hay không. Nếu có thì khả năng model "học kém" trên ảnh đó sẽ cao hơn.
- **Cụ thể hơn,** đầu tiên ta chia ảnh thành 2 nửa (có 2 cách chia: chia theo chiều dọc và chiều ngang) đồng thời ta xây dựng 1 ngưỡng **rate**. Ở đây, **rate** là tỉ lệ độ thưa thể hiện nửa ảnh có số kí tự ít hơn **rate%** hay không, miền giá trị của **rate** là [0, 0.5].
  - Từ đó, ta xây dựng 1 thuật toán như sau: nếu một nửa ảnh có số kí tự nhỏ hơn **rate%** của ảnh thì ảnh đó sẽ được phân vào danh sách những ảnh có nguy cơ "học kém", ta gọi các ảnh này là các **ảnh bị lệch** - hình 8. Danh sách các **ảnh bị lệch** sẽ tăng khi ta tăng **rate** lên.
  - Tuy nhiên, đôi khi **rate** cao quá thì điều kiện lọc sẽ bị quá lỏng. Vậy nên nếu ta không điều chỉnh độ **rate** sao cho hợp lý thì sẽ có rất nhiều ảnh bị phân nhầm vào nhóm **ảnh bị lệch**. Đây cũng là điều lưu ý quan trọng mà nhóm rút ra được sau khi thử phân tích với nhiều độ **rate** khác nhau.



Hình 8: Minh họa cho "ảnh bị lệch"

### Kết quả rút ra được sau khi phân tích:

- Sau khi thử với nhiều độ rate khác nhau, nhóm quyết định chọn các **rates** phù hợp nhất đó là 0.3 và 0.35.

- Sau đó, nhóm quyết định sinh ra 2 bộ dữ liệu được augment bằng cách phương pháp phía bên dưới tương ứng với 2 độ **rate** trên kết hợp với bộ dữ liệu gốc tạo ra 2 bộ training data hoàn chỉnh rồi đưa vào train model xem bộ dữ liệu nào cho kết quả tốt hơn.

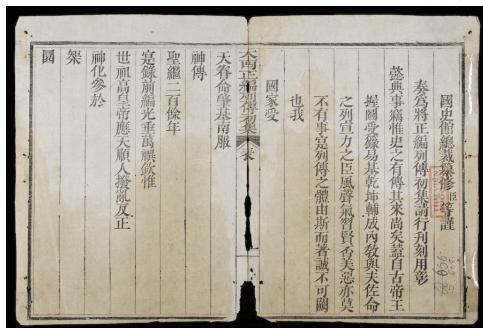
## 2.2 Tiền xử lý dữ liệu

Trong phần này, nhóm xin trình bày về việc tiền xử lý dữ liệu bằng cách tăng cường dữ liệu (augmentation) cho tập dữ liệu ảnh và nhãn. Việc tăng cường dữ liệu ảnh đã được thực hiện trên dữ liệu ảnh ban đầu **original data** và dữ liệu ảnh đặc biệt có nguy cơ bị lỗi theo hướng tiếp cận 3 đã đề cập với **rate = 0.3** và **rate = 0.35**.

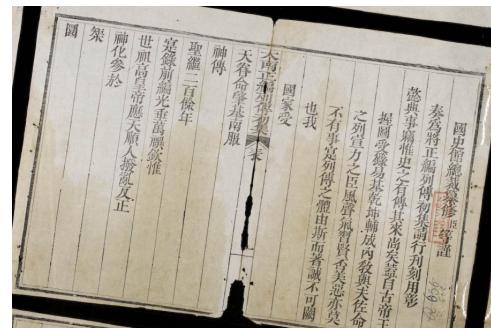
### 2.2.1 Các kỹ thuật augmentation cơ bản

Nhóm đã sử dụng thư viện **Albumentations** của Python hỗ trợ để thực hiện các kỹ thuật augmentation cơ bản sau đây:

- Xoay ảnh theo nhiều góc:** nhóm đã sử dụng phép xoay ảnh với nhiều góc khác nhau trong đó giới hạn là  $15^\circ$ . Quá trình này giúp tăng cường dữ liệu bằng cách tạo ra các ảnh có góc nhìn khác nhau, từ đó làm tăng tính đa dạng của tập dữ liệu.



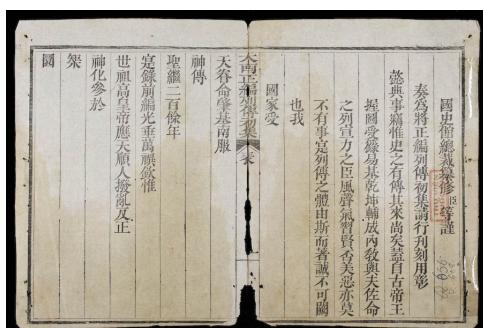
(a) Ảnh gốc



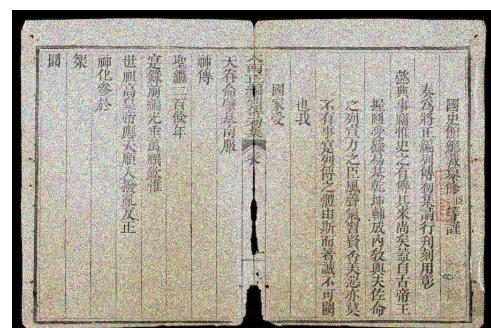
(b) Ảnh sau khi xoay góc

Hình 9: Kết quả phép xoay ảnh theo góc giới hạn  $15^\circ$

- Add noise:** Nhóm đã áp dụng kỹ thuật thêm nhiễu vào ảnh để tạo ra các phiên bản mới với độ nhiễu khác nhau giúp tạo ra sự đa dạng trong dữ liệu và làm tăng khả năng tổng quát hóa của mô hình. Nhóm sử dụng 2 loại nhiễu để thêm vào ảnh chính là **gaussian noise** và **multiplicative noise**



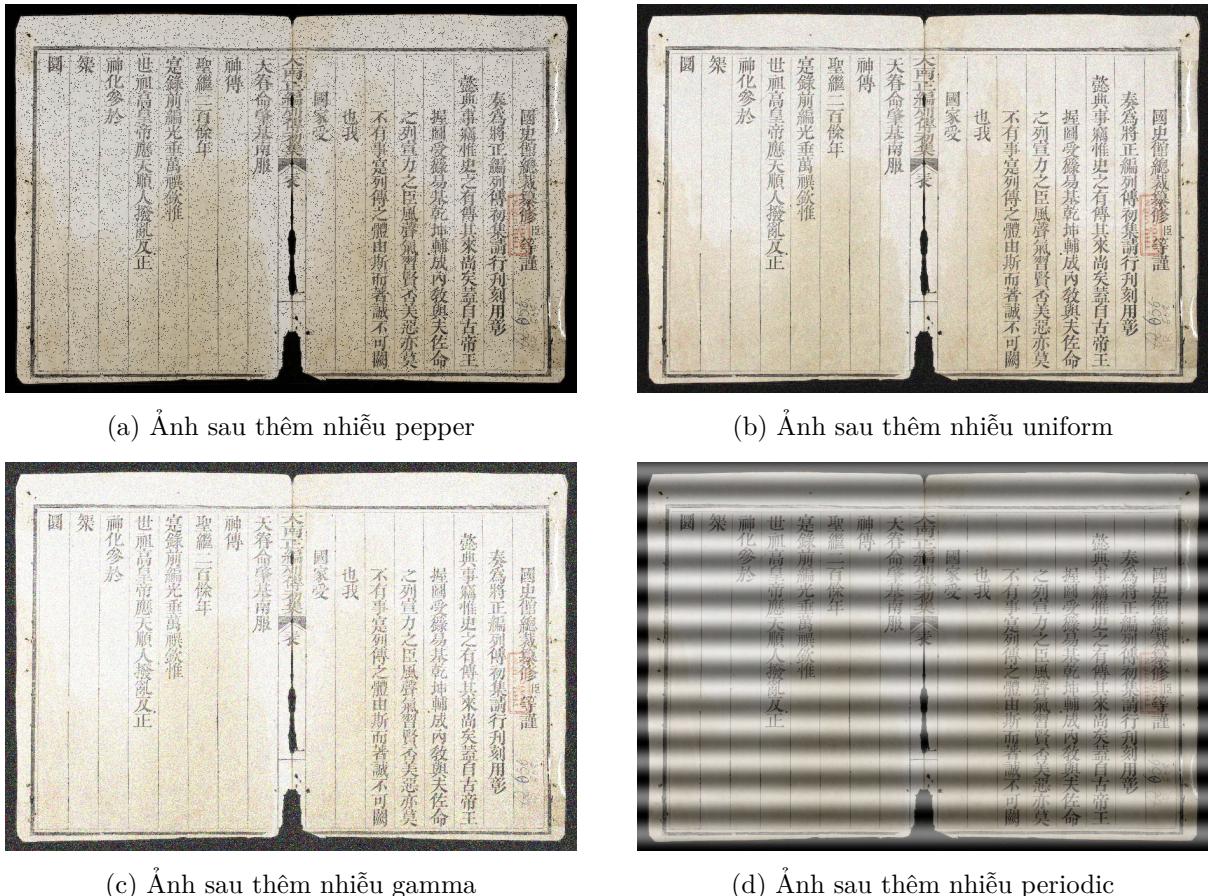
(a) Ảnh sau thêm nhiễu gaussian



(b) Ảnh sau thêm nhiễu multiplicative

Hình 10: Kết quả phép thêm nhiễu

Ngoài ra, nhóm đã sử dụng thêm 4 loại nhiễu phổ biến được đề cập trong chương trình học mà thư viện [Albumentations](#) chưa hỗ trợ, bao gồm **pepper noise**, **uniform noise**, **gamma noise**, và **periodic noise**, để thêm vào ảnh bằng sự hỗ trợ của thư viện [numpy](#).



Hình 11: Kết quả phép thêm các nhiễu khác

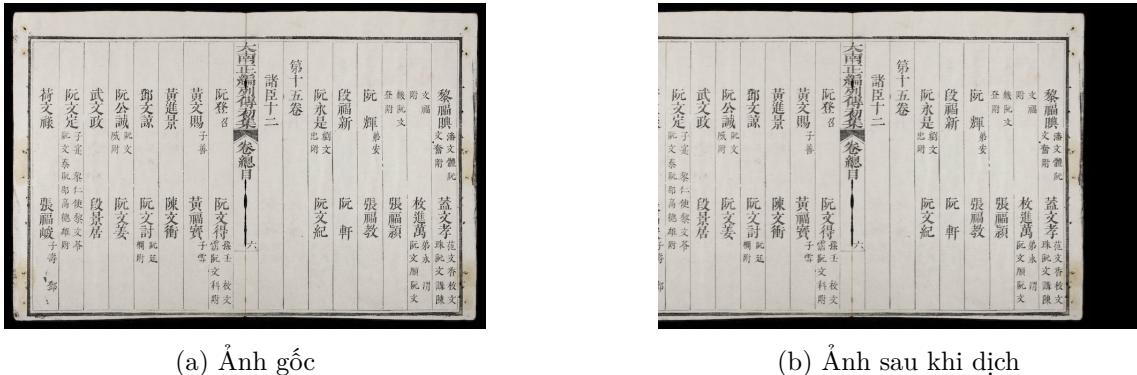
- Scale ảnh:** nhóm đã thực hiện phép biến đổi scale ảnh để tạo ra các phiên bản mới có kích thước khác nhau giúp tạo ra các ảnh có kích thước và tỷ lệ khác nhau.



Hình 12: Kết quả phép scale ảnh

- Dịch chuyển ảnh sang trái phải:** nhóm thực hiện dịch ảnh trên trục  $x$  và thiết lập giới hạn là 0.3 - tức là ảnh có thể dịch chuyển tối đa 30% chiều rộng của nó

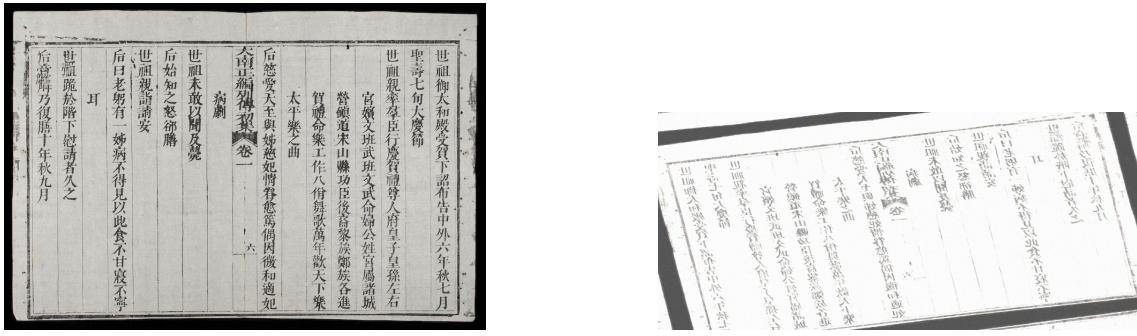
sang trái hoặc phải. Các vùng khoảng trống sau khi dịch ảnh sẽ được lấp đầy bằng màu đen. Điều này giúp mô hình học giảm sự phụ thuộc vào vị trí và tạo ra một tập dữ liệu phong phú hơn.



Hình 13: Kết quả phép dịch ảnh

### 2.2.2 Các kỹ thuật augmentation nâng cao

- Kết hợp nhiều kỹ thuật augmentation trong cùng 1 bức ảnh:** nhóm đã kết hợp nhiều kỹ thuật augmentation cơ bản trong cùng một bức ảnh. Trong đó, ngoài 3 phép biến đổi đã đề cập, nhóm đã kết hợp thêm các phép biến đổi như **Horizontal\_flip**, **Blur** và **BrightnessContrast**, trong đó mỗi phép biến đổi con được thực hiện với xác suất  $p=0.5$



Hình 14: Kết quả phép biến đổi kết hợp nhiều kỹ thuật

### 2.2.3 Tập dữ liệu được bổ sung thêm

Sau khi áp dụng các kỹ thuật trên, từng thành viên đã tự thiết kế tập dữ liệu thêm của mình để train ngoài tập dữ liệu gốc ban đầu. Vì quá trình train nhiều lần của từng thành viên khác nhau nên cũng có nhiều tập dữ liệu. Một ví dụ về tập dữ liệu đầu tiên mà nhóm bổ sung thêm trong những lần thử nghiệm là tập **train** 70 ảnh như sau:

- Các ảnh được coi là **biased** theo **threshold = 0.3** bị rotate đi: 9 ảnh
- Các ảnh bị thêm nhiễu **multiplicative**: random 9 ảnh trong 70 ảnh gốc ban đầu
- Các ảnh được coi là **biased** theo **threshold = 0.35** cho trước là 0.35 bị **scaled**, **rotated** và thêm nhiễu **Gaussian**: 21 ảnh.

Tập **val** gồm 15 ảnh như sau:

- 10 ảnh ngẫu nhiên trong toàn 70 ảnh **train** ban đầu bị **scaled**, **rotated** và thêm nhiễu **Gaussian**.
- 5 ảnh ngẫu nhiên trong 70 ảnh train ban đầu bị thêm nhiễu **multiplicative**.

## 2.3 Thuật toán xác định bounding box

YOLO là một trong những mô hình nổi tiếng làm tốt bài toán xác định bounding box cho vật thể. Nhóm đã quyết định thử nghiệm các phiên bản YOLO được implement bởi thư viện Ultralytics vì tính dễ dùng, dễ cài đặt của chúng.

### 2.3.1 Tổng quan thư viện Ultralytics

**YOLOv5** : phiên bản YOLOv5 được cài đặt bởi Ultralytics đã tự động implement một số phương pháp cho người dùng như: augment data đa dạng (Mosaic Augmentation, Copy-Paste Augmentation,...) sử dụng các chiến lược đào tạo nâng cao như: Multiscale Traning, AutoAnchor, Hyperparameter Evolution....) và các cải tiến quan trọng trong việc tính hàm loss, target.

**YOLOv8** : phiên bản YOLOv8 có nhiều cải tiến so với YOLOv5, tuy nhiên vì hạn chế về thời gian và kinh nghiệm trong học máy, học sâu của nhóm nên nhóm chưa tìm hiểu được cụ thể kiến trúc của YOLOv8 ra sao, tuy nhiên phiên bản YOLOv8 được cài đặt bởi Ultralytics cơ bản vẫn sử dụng một số phương pháp trong quá trình train như YOLOv5

### 2.3.2 Một số lưu ý khác

- Cả 2 phiên bản YOLO trên đều có các phiên bản như nano, small, medium, ... nhưng vì tài nguyên GPU hạn chế trên Google Colab nên nhóm chỉ có thể thử nghiệm đến bản large của 2 mô hình YOLO
- Tại sao lại thử nghiệm 2 mô hình này: bởi vì YOLOv5 nổi tiếng với performance vượt trội của nó và YOLOv8 là một trong những phiên bản mới nhất được Ultralytics cài đặt, tuy nhiên theo tìm hiểu của nhóm thì trong một số trường hợp, YOLOv8 có accuracy và precision hơn YOLOv5 trong lúc train và validate, nhưng YOLOv8 lại kém hơn YOLOv5 trong lúc test nên nhóm quyết định chia nhau thử cả 2 phiên bản.

### 2.3.3 Áp dụng vào bài toán detect chữ Nôm

Chi tiết các bộ hyperparameters được nhóm thử có thể tìm thấy trong phần Supplementary (dưới dạng Google Docs). Sau khi cùng nhau thử nhiều bộ hyperparameters khác nhau thì nhóm đã cùng đánh giá các mô hình train được trên cùng 1 bộ test được thiết kế với số lượng ảnh như sau:

- 5 ảnh thêm nhiễu Gamma
- 5 ảnh thêm nhiễu Multiplicative

- 5 ảnh thêm nhiễu Periodic
- 5 ảnh thêm nhiễu Uniform
- 5 ảnh bị dịch chuyển
- 5 ảnh kết hợp cả 3 kỹ thuật augmentation cơ bản
- 10 ảnh bị xoay theo nhiều góc

Và kết quả là: mô hình với bộ hyperparameters: batch=5, optimizer=SGD, epochs=100, lr0=0.01, momentum=0.973, weight\_decay=0.0005 là hoạt động tốt nhất trên tập ảnh test trên, nên nhóm đã quyết định lấy mô hình đó làm mô hình cuối cùng.

## 2.4 Cải thiện mô hình với kĩ thuật semi-supervised learning

Dựa trên tập NomNaOCR, nhóm đã cho model tốt nhất mà nhóm train được inference thêm những bức ảnh đó và lựa chọn ra những bức ảnh có nhãn chính xác nhất bổ sung vào tập dữ liệu. Tuy nhiên, phương pháp này không được hiệu quả cho lắm vì sau khi train thêm với những bức ảnh này thì độ chính xác của model không tăng lên đáng kể (nên nhóm không keep track lại).

## 3 Kết quả và thảo luận

- Với chiến lược huấn luyện chủ yếu tập trung vào dữ liệu và tận dụng những kiến trúc nổi tiếng sẵn có, nhóm đã có model có độ chính xác là 86.7% trên tập val gốc ban đầu của thầy và 83.67% trên tập val kiểm tra của thầy. Dễ thấy nhóm có kết quả kém so với các nhóm khác do quá trình huấn luyện nhóm đã tập trung với nhiều ảnh augment hơn so với ảnh bình thường, tuy nhiên nhìn chung về khả năng generalize với các ảnh xoay, scale thì mô hình với bộ tham số nhóm đã huấn luyện vẫn làm tốt hơn.
- Nhóm đã học được nhiều thứ như: thiết kế machine learning pipeline, chạy các mô hình deep learning,... trong thời gian chạy project này. Nhóm xin cảm ơn thầy Thương đã luôn nhiệt tình hỗ trợ và giải đáp các thắc mắc của nhóm.