

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

-----o0o-----



BÁO CÁO ĐỒ ÁN CUỐI KỲ
ĐỀ TÀI 006: SinoNôm OCR VỚI PaddleOCR
CHO THIẾT BỊ DI ĐỘNG

Thành viên thực hiện:	Phạm Nguyên Khánh	22127190
	Đặng Văn Kỳ	22127227
	Nguyễn Anh Hoàng	22127124
	Lê Phước Thạnh	22127392
Lớp:	22CNTThuc-CLC_2024	
Học phần:	Nhập môn xử lý ngôn ngữ tự nhiên	CSC15006
Giảng viên hướng dẫn:	PGS.TS. Đinh Điền	
	TS. Nguyễn Hồng Bửu Long	
	TS. Lương An Vinh	

Mục lục

I) Tổng quan đề tài:	3
1) Giới thiệu đề tài:	3
2) Các đầu công việc chính đã thực hiện:	3
II) Xây dựng bộ dữ liệu SinoNôm:	4
1) Tổng quan bộ dữ liệu đã xây dựng:	4
2) Quy trình trích xuất dữ liệu trên tập dữ liệu thô:.....	4
3) Quy trình gán nhãn:	4
III) Xây dựng mô hình:	6
1) Tổng quan mô hình PaddleOCR:.....	6
2) Các thành phần chính của PaddleOCR:	6
3) Quy trình fine-tuning mô hình:	7
IV) Xây dựng ứng dụng OCR trên điện thoại:.....	14
1) Xây dựng API xử lý yêu cầu chạy mô hình:.....	14
2) Xây dựng giao diện ứng dụng điện thoại:.....	16
3) Tìm hiểu thêm về app OCR đã được xây dựng sẵn dựa trên PaddleLite:	18
V) Kết quả thực nghiệm:	26
VI) Tổng Quan về thư mục nộp bài:	27
VII) Phân công việc nhóm:.....	28
VIII) Tham khảo:.....	29

I) Tổng quan đề tài:

1) Giới thiệu đề tài:

- Việt Nam ta đã trải qua hơn nghìn năm những trang sử sách hào hùng với những di sản về văn hóa vô cùng phong phú. Rất nhiều trong số những trang sử sách ấy được lưu lại thông qua hệ thống chữ Nôm do ông cha ta sáng tạo ra sau thời kỳ Bắc thuộc. Với việc chữ Nôm ngày càng bị mai một, cũng như việc dịch thuật thủ công các tài liệu Nôm cổ rất tốn thời gian, việc số hóa chữ Nôm một cách tự động là vô cùng cần thiết.
- Sự trợ giúp của các công cụ OCR ngày nay sẽ góp phần giúp thúc đẩy việc phát hiện và nhận dạng chữ Nôm được nhanh hơn từ đó giúp thúc đẩy việc số hóa các văn bản chữ Nôm.
- Với việc xây dựng một ứng dụng di động giúp hỗ trợ OCR chữ Nôm thành dạng text, nhóm hi vọng sẽ giúp việc thu thập dữ liệu chữ Nôm được đơn giản hóa hơn.

2) Các đầu công việc chính đã thực hiện:

- Trích xuất và xây dựng bộ dữ liệu OCR Hán-Nôm với hơn 8000 box từ dữ liệu thô.
- Fine-tuning lại các mô hình Text Detection và Text Recognition có sẵn của PaddleOCR với bộ dữ liệu nhóm tự xây dựng được và bộ dữ liệu NomNaOCR có sẵn.
- Xây dựng App Mobile cho mô hình đã fine-tuned được.

II) Xây dựng bộ dữ liệu SinoNôm:

1) Tổng quan bộ dữ liệu đã xây dựng:

- Bộ dữ liệu chứa hơn 8000 box chữ Hán Nôm trải dài trên 4 tác phẩm Sách Nôm công giáo:
 - + Tứ Nguyên Yếu Lý - phần I: khoảng 2300 box.
 - + Tứ Nguyên Yếu Lý - phần II: khoảng 3100 box.
 - + Tứ Chung Quốc Ngữ: khoảng 1500 box.
 - + Tứ Chung Lược Thuyết: Khoảng 1800 box.

2) Quy trình trích xuất dữ liệu trên tập dữ liệu thô:

a) Trích xuất ảnh từ file PDF:

- Sử dụng một số thư viện python được hỗ trợ để tự động trích xuất và lưu các trang sách từ PDF.
- Đồng thời có thể thực hiện việc chia ảnh sách thành từng trang cụ thể để tăng độ chính xác, cũng như dễ quản lý khi đóng hàng trang.
- Việc đóng hàng trang Quốc Ngữ và trang Hán-Nôm tương ứng có thể thực hiện dựa trên số thứ tự trang trên sách. Số thứ tự trên trang sách có thể được xác định tự động thông qua các công cụ OCR khác như Google Vision, Tesseract, ...

b) Xử lý ảnh:

- Tùy vào các đặc tính của các ảnh dữ liệu thô trong PDF mà nhóm sẽ có các phương pháp xử lý ảnh phù hợp, một số phương pháp bao gồm:
 - + Cắt ảnh (tự động một cách tương đối hoặc bán tự động) để giới hạn, xác định phạm vi OCR cũng như giảm các chi tiết gây nhiễu bên ngoài khu vực chứa câu Hán-Nôm.
 - + Giảm bóng giữa hai trang sách (biến ảnh thành dạng trắng đen).
 - + Tăng độ nét của chữ.
 - + Làm mờ các chi tiết gây nhiễu.
 - + Chỉnh lại các trang bị nghiêng.

3) Quy trình gán nhãn:

a) Sử dụng công cụ OCR API CLC để gán nhãn tự động:

- API CLC là bộ công cụ được nghiên cứu và phát triển được bởi một số giảng viên Khoa Công nghệ Thông tin và Trung tâm Ngôn ngữ học Tính toán thuộc Trường ĐH Khoa học Tự nhiên - ĐHQG - HCM. Bộ công cụ cung cấp các chức năng về

nhận dạng và dịch các văn bản chữ Hán, và đặc biệt là chữ Hán-Nôm với độ chính xác tương đối cao.

- Nhóm em sẽ viết script để tự động upload ảnh và gọi API image-ocr và lưu kết quả OCR tương ứng vào JSON file.

b) Một số kỹ thuật alignment sau khi dán nhãn:

- Sắp xếp lại thứ tự các bounding box thu được từ công cụ CLC API.
- Vì phần chữ Quốc Ngữ được trích từ ảnh của cả trang sách, nên các câu Quốc Ngữ trích được từ ảnh sẽ không được chia thành các câu với số lượng từ khớp với câu Hán-Nôm tương ứng.
- Vì câu Quốc Ngữ sẽ được dùng làm chuẩn, ta cần giải quyết vấn đề xác định được câu và độ dài của câu Quốc Ngữ làm mẫu tương ứng với từng câu Hán-Nôm trong hình. Nhóm em có một số cách thức xử lý như sau:
 - + Lấy box Hán-Nôm đầu tiên trong trang làm gốc, đếm số lượng chữ Hán-Nôm trong box, và đếm tương ứng số lượng chữ Quốc Ngữ để làm mẫu đóng hàng. Với các box tiếp theo, ta tiếp tục lấy số lượng tương ứng các chữ Quốc Ngữ tiếp theo. Và sau đó là thực hiện đóng hàng trong câu đối với từng cặp box Hán-Nôm/câu Quốc Ngữ tương ứng.
 - + Gộp toàn bộ chữ Hán-Nôm từ tất cả các box trong một trang thành một đoạn dài (và cài đặt danh sách ghi nhớ chữ nào thuộc về box nào), hay nói cách khác là xem toàn bộ các box trong một trang là một box duy nhất. Thực hiện đóng hàng với toàn bộ phần Quốc Ngữ của trang tương ứng. Cuối cùng là áp dụng danh sách ghi nhớ để xác định câu Quốc Ngữ nào thuộc về câu Hán-Nôm nào.

III) Xây dựng mô hình:

1) Tổng quan mô hình PaddleOCR:

- PaddleOCR là một bộ công cụ mã nguồn mở toàn diện, được phát triển bởi **Baidu** trên nền tảng học sâu PaddlePaddle. Nó tập trung vào việc giải quyết các bài toán về nhận diện ký tự quang học (**OCR** - Optical Character Recognition), cung cấp các giải pháp mạnh mẽ cho nhận diện văn bản từ hình ảnh, tài liệu và biển báo.

2) Các thành phần chính của PaddleOCR:

a) Text Detection (Phát hiện văn bản):

- Mục tiêu của giai đoạn này là xác định vị trí của văn bản trong hình ảnh, thường được biểu diễn dưới dạng bounding box hoặc polygon.
- Mô hình chính
 - + **DBNet**: Một trong những mô hình phổ biến trong PaddleOCR, được thiết kế để phát hiện các đoạn văn bản trong nhiều hình dạng khác nhau (chữ ngang, chữ dọc, chữ cong, v.v.).
- Đặc điểm nổi bật:
 - + Phát hiện chính xác cả các đoạn văn bản phức tạp (nhỏ, cong, hoặc nghiêng).
 - + Tốc độ xử lý nhanh nhờ kỹ thuật giảm mẫu và tối ưu hoá kiến trúc mạng.

b) Text Recognition (Nhận dạng văn bản):

- Giai đoạn này chuyển đổi các đoạn văn bản đã phát hiện thành dạng chuỗi ký tự.
- Mô hình chính:
 - + **CRNN** (Convolutional Recurrent Neural Network): Kết hợp CNN để trích xuất đặc trưng và RNN để xử lý chuỗi ký tự.
 - + **Transformer-based models**: Các mô hình hiện đại như SAR (Show, Attend and Read) sử dụng cơ chế attention.
 - + **SVTR**: Một mô hình hiện đại cho nhận diện văn bản hiệu quả trên các hình ảnh có độ phân giải cao.
- Đặc điểm nổi bật:
 - + Hỗ trợ đa ngôn ngữ (hơn 80 ngôn ngữ, bao gồm tiếng Việt).
 - + Hỗ trợ các kiểu ký tự như tiếng Trung, tiếng Ả Rập, chữ in nghiêng, hoặc chữ viết tay.

c) Post-processing (Xử lý hậu kỳ):

- Kết hợp kết quả phát hiện và nhận dạng để trả về văn bản cuối cùng. Giai đoạn này thường bao gồm:
 - + Ghép văn bản từ các bounding box.

3) Quy trình fine-tuning mô hình:

a) Train Text Detection Model:

1. Tìm pretrained model thích hợp:

- + Sau khi đánh giá và khảo sát thì nhóm chọn mô hình ch_PP-OCRv3_det. Vì đây là mô hình nhỏ gọn nhưng hiệu suất cao thích hợp để fine tuning và dùng cho thiết bị mobile.

2. Chuẩn bị dữ liệu:

- Bước 1: Từ bộ dữ liệu của nhóm đã chuẩn bị từ trước và dữ liệu NomNaOCR mà thầy đã giao trong file đồ án cuối kỳ.
- Bước 2: Trích ra file dữ liệu riêng của nhóm từ dữ liệu giữa kì.
 - + Folder **gts** gồm những những file txt chứa tọa độ các box và nội dung text của từng box trong 1 trang.
 - + Folder **imgs** gồm những hình ảnh trong dữ liệu giữa kì mà nhóm đã chuẩn bị.
 - + Label.txt là file text gồm tên những hình ảnh được OCR và nội dung được trích xuất trong ảnh đó.

Name	Date modified	Type	Size
gts	12/27/2024 5:31 AM	File folder	
imgs	12/27/2024 5:31 AM	File folder	
Label	12/27/2024 5:27 AM	Text Source File	1,489 KB

- Bước 3: Nhóm 2 folders và file này lại và đưa vào folder Nhóm 6. Sau đó bỏ vào folder Pages của nhóm tác giả mà thầy đã gửi.

C > GIAI TRI (E:) > Pages >			
Sort View ...			
Name	Date modified	Type	Size
DVSKTT-1 Quyên thu	12/27/2024 4:39 AM	File folder	
DVSKTT-2 Ngoại ky toan thu	12/27/2024 4:39 AM	File folder	
DVSKTT-3 Ban ky toan thu	12/27/2024 4:40 AM	File folder	
DVSKTT-4 Ban ky thuc luc	12/27/2024 4:41 AM	File folder	
DVSKTT-5 Ban ky tuc bien	12/27/2024 4:41 AM	File folder	
Luc Van Tien	12/27/2024 4:42 AM	File folder	
Nhom 6	12/27/2024 5:31 AM	File folder	
Tale of Kieu 1866	12/27/2024 4:42 AM	File folder	
Tale of Kieu 1871	12/27/2024 4:42 AM	File folder	
Tale of Kieu 1872	12/27/2024 4:42 AM	File folder	
PaddleOCR-Train	12/27/2024 5:35 AM	Text Source File	4,971 KB
PaddleOCR-Validate	12/27/2024 5:36 AM	Text Source File	1,073 KB
Train	12/27/2024 5:56 AM	Text Source File	271 KB
Validate	12/27/2024 5:58 AM	Text Source File	64 KB

3. Sử dụng Google Colab để tiến hành train model Text Detection.
 - Đầu tiên, nhóm em sẽ git clone project <https://github.com/PaddlePaddle/PaddleOCR> về máy tính.
 - Sau đó sẽ tiến hành đẩy lên drive để dễ dàng cho việc huấn luyện mô hình.

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)

- Tiến hành tải pretrained model vào thư mục pretrain_model.

```
Building wheel for Fire (setup.py) ... done

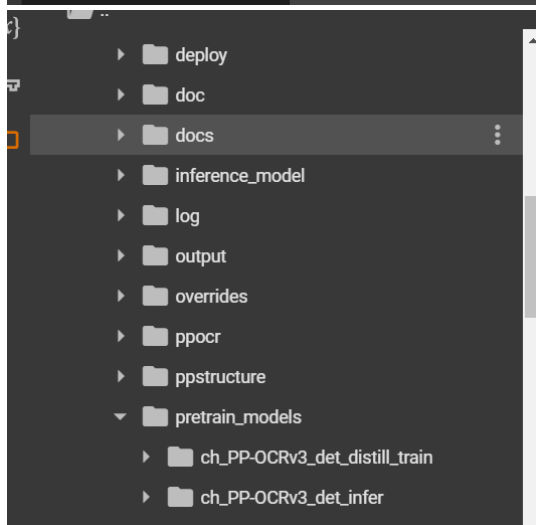
!apt-get install wget

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
wget is already the newest version (1.21.2-2ubuntu1.1).
0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.

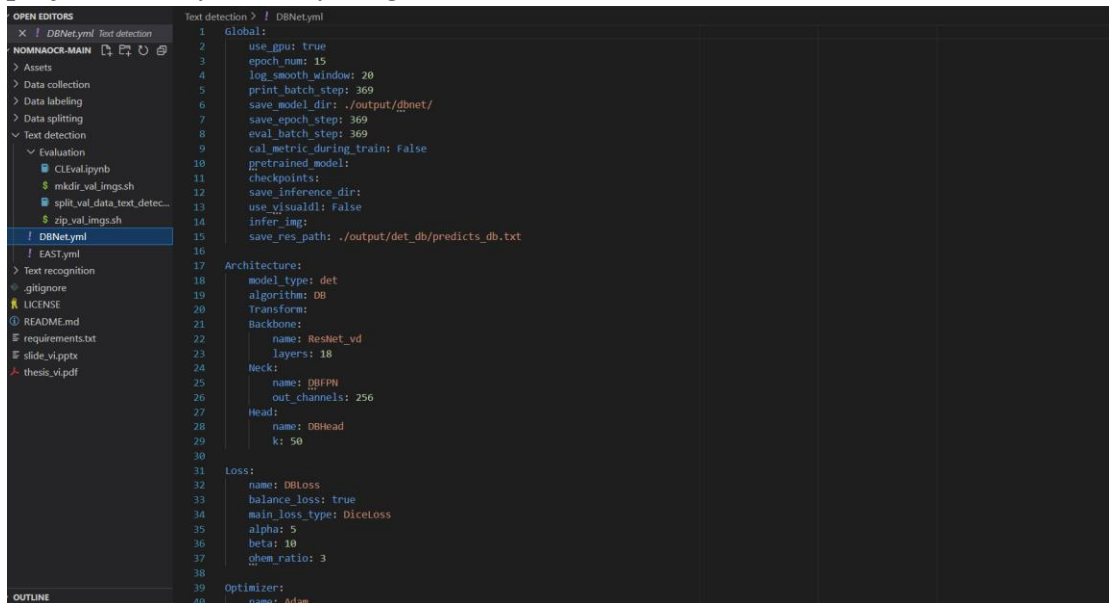
:/ https://paddleocr.bj.bcebos.com/PP-OCRv3/chinese/ch_PP-OCRv3_det_distill_train.tar

ers

listill_train.tar && rm -rf ch_PP-OCRv3_det_distill_train.tar
```



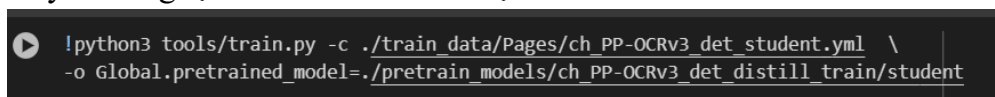
- Tiếp theo nhóm em sẽ thêm vào folder data_training toàn bộ folder Pages. Và sẽ copy thêm file ch_PP-OCRv3_det_student.yml trong thư mục config để tiến hành config.
- Sau khi thêm xong thì nhóm sẽ tiến hành sửa file config theo đúng file det.yml trong project cuối kỳ mà thầy đã gửi.



```

1 Global:
2   use_gpu: true
3   epoch_num: 15
4   log_smooth_window: 20
5   print_batch_step: 369
6   save_model_dir: ./output/dbnet/
7   save_epoch_step: 369
8   eval_batch_step: 369
9   cal_metric_during_train: false
10  pretrained_model:
11  checkpoint:
12  save_inference_dir:
13  use_visualdl: false
14  infer_img:
15  save_res_path: ./output/det_db/predicts_db.txt
16
17 Architecture:
18  model_type: det
19  algorithm: DB
20  Transform:
21  Backbone:
22    name: ResNet_vd
23    layers: 18
24  Neck:
25    name: DBFPN
26    out_channels: 256
27  Head:
28    name: DBHead
29    k: 50
30
31 Loss:
32  name: DBLoss
33  balance_loss: true
34  main_loss_type: DiceLoss
35  alpha: 5
36  beta: 10
37  ghm_ratio: 3
38
39 Optimizer:
40  name: Adam
  
```

- Sau đó chúng em sẽ tiến hành thay đổi đường dẫn các file quan trọng như save_model dir, data_dir, label_File_list.
- Tiếp theo, chúng em sẽ upload ch_PP-OCRv3_det_student.yml lên trên thư mục data_training/Pages của drive.
- Giờ đây thì chúng ta đã có thể bắt đầu train được mô hình.
- Đây là dòng lệnh để có thể train được mô hình:

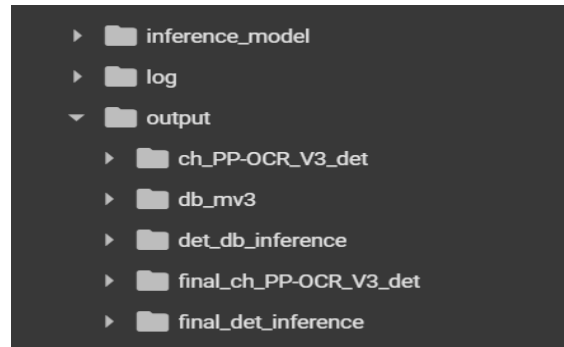


```

!python3 tools/train.py -c ./train_data/Pages/ch_PP-OCRv3_det_student.yml \
-o Global.pretrained_model=./pretrain_models/ch_PP-OCRv3_det_distill_train/student
  
```

- Đầu tiên chúng em sẽ thực thi file train.py và có 2 tham số truyền vào đó chính là 2 đường dẫn. Đường dẫn thứ nhất chính là đường dẫn đến file cấu hình yml mà em đã trình bày ở trên. Đường dẫn thứ hai chính là đường dẫn đến mô hình pretrained mà chúng em đã tải về lúc trước.

- Sau khi chạy xong thì trong thư mục output sẽ được lưu kết quả mô hình được train ra:



- Nhưng tới đây thì chúng ta vẫn chưa sử dụng được mô hình và cần phải chuyển sang dạng inference để có thể áp dụng được.
- Chúng ta cần phải export ra file **inference** để có thể sử dụng model đã train.

```
[2024/12/27 18:19:48] ppocr INFO: inference model is saved to ./output/final_det_inference/inference

[ ] export_model.py -c ./train_data/Pages/ch_PP-OCRv3_det_student.yml -o Global.pretrained_model="./pretrain_models/ch_PP-OCRv3_det_d

W1227 07:45:08.961128 120791 gpu_resources.cc:119] Please NOTE: device: 0, GPU Compute Capability: 8.9, Driver API Version: 12.2
W1227 07:45:08.962388 120791 gpu_resources.cc:149] device: 0, cuDNN Version: 8.9.
[2024/12/27 07:45:09] ppocr WARNING: The pretrained params backbone.conv.conv.weight not in model
[2024/12/27 07:45:09] ppocr WARNING: The pretrained params backbone.conv.bn.weight not in model
[2024/12/27 07:45:09] ppocr WARNING: The pretrained params backbone.conv.bn.bias not in model
[2024/12/27 07:45:09] ppocr WARNING: The pretrained params backbone.conv.bn.mean not in model
[2024/12/27 07:45:09] ppocr WARNING: The pretrained params backbone.conv.bn_variance not in model
[2024/12/27 07:45:09] ppocr WARNING: The pretrained params backbone.stage0.expand_conv.conv.weight not in model
```

- Tham số truyền vào là 2 đường dẫn, đường dẫn 1 là đường dẫn đến model cần phải export ra file, đường dẫn thứ 2 là đường dẫn đến mô hình inference.
- Chúng ta cũng có thể xem được kết quả huấn luyện thông qua file eval.py

```
[2024/12/27 07:34:06] ppocr INFO: loader :
[2024/12/27 07:34:06] ppocr INFO: batch_size_per_card : 8
[2024/12/27 07:34:06] ppocr INFO: drop_last : False
[2024/12/27 07:34:06] ppocr INFO: num_workers : 0
[2024/12/27 07:34:06] ppocr INFO: shuffle : True
[2024/12/27 07:34:06] ppocr INFO: profiler options : None
[2024/12/27 07:34:06] ppocr INFO: train with paddle 2.5.0 and device Place(gpu:0)
[2024/12/27 07:34:06] ppocr INFO: Initialize indexes of datasets:['./train_data/Pages/PaddleOCR-Validate.txt']
W1227 07:34:06.154464 117913 gpu_resources.cc:119] Please NOTE: device: 0, GPU Compute Capability: 8.9, Driver API Version: 12.2
W1227 07:34:06.155689 117913 gpu_resources.cc:149] device: 0, cuDNN Version: 8.9.
[2024/12/27 07:34:06] ppocr INFO: resume from ./output/ch_PP-OCR_V3_det/best_accuracy
[2024/12/27 07:34:06] ppocr INFO: metric in ckpt *****
[2024/12/27 07:34:06] ppocr INFO: hmean:0.9779118752134806
[2024/12/27 07:34:06] ppocr INFO: is_float16:False
[2024/12/27 07:34:06] ppocr INFO: precision:0.9760227272727273
[2024/12/27 07:34:06] ppocr INFO: recall:0.9798083504449008
[2024/12/27 07:34:06] ppocr INFO: fps:55.19710931734243
[2024/12/27 07:34:06] ppocr INFO: best_epoch:8
[2024/12/27 07:34:06] ppocr INFO: acc:0.0
[2024/12/27 07:34:06] ppocr INFO: start_epoch:9
eval model:: 100% 673/673 [00:36<00:00, 18.26it/s]
[2024/12/27 07:34:43] ppocr INFO: metric eval *****
[2024/12/27 07:34:43] ppocr INFO: precision:0.975668541260186
[2024/12/27 07:34:43] ppocr INFO: recall:0.9697695642254164
[2024/12/27 07:34:43] ppocr INFO: hmean:0.9727101092739859
[2024/12/27 07:34:43] ppocr INFO: fps:55.49320944237924
```

- Cuối cùng chúng em chỉ cần tải folder **final_det_inference** trong folder **inference** để có thể sử dụng.

b) Train Text Recognition Model:

1. Chuẩn bị bộ dữ liệu:

- Bộ dữ liệu dành cho việc train mô hình recognition được lấy từ bộ dữ liệu *NomNaOCR**([NomNaOCR / Kaggle](#)) và bộ dữ liệu do nhóm tự chuẩn bị trong đó án giữa kỳ - bao gồm hơn 8000 cặp câu song ngữ.
- Để tiến hành chuẩn bị bộ dữ liệu, ta thực hiện như sau:
 - + Chuẩn bị dữ liệu theo format:
 - o Tạo folder dataset chứa các folder con và file train.txt và validate.txt:
 - o Triển khai file .txt theo format:

§ Trong mỗi dòng được định dạng : <đường dẫn tới ảnh tương ứng> \tab
<Chữ Nôm tương ứng trong ảnh đó>

- + Sau khi bộ dataset đã chuẩn bị xong, ta tiến hành đẩy dataset lên cloud (google drive) để tiến hành sử dụng thông qua Google Colab để thực hiện quá trình train model.

2. Chuẩn bị File Config:

- File Config là một phần không thể thiếu trong việc train model, để chuẩn bị file config ta tiến hành theo các bước sau:
 - + Truy cập vào đường dẫn: [PaddleOCR2Pytorch/configs/rec_at_main · frotms/PaddleOCR2Pytorch](#) tải về file config phù hợp với model đã chọn, ở đây ta sử dụng file : “[ch_PP-OCRv3_rec.yml](#)”.
 - + Cuối cùng ta đẩy file config này lên google drive chuẩn bị cho quá trình train model.

3. Chuẩn bị môi trường train model:

- Để chuẩn bị môi trường train model, ta tiến hành theo các bước sau:
 - + Kết nối colab với tài khoản google drive chứa bộ dataset ta upload:

```
- from google.colab import drive  
- drive.mount('/content/drive')
```

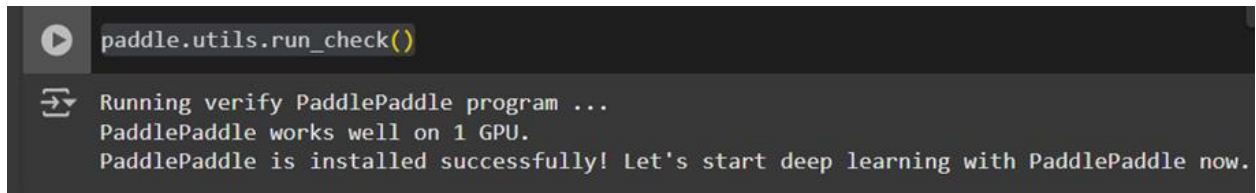
- + Tải thư viện PaddleOCR về máy thông qua github:

```
- !git clone https://github.com/PaddlePaddle/PaddleOCR.git
```

- + Tải các thư viện cần thiết:

```
- !pip install -q pylclipper lmbd rapidfuzz paddlepaddle-gpu paddleocr
```

- + Sau khi đã tải xong, ta kiểm tra trước khi bước vào giai đoạn train model, nếu kết quả hiện như ảnh thì việc chuẩn bị đã xong:



```
paddle.utils.run_check()  
  
Running verify PaddlePaddle program ...  
PaddlePaddle works well on 1 GPU.  
PaddlePaddle is installed successfully! Let's start deep learning with PaddlePaddle now.
```

- + Tải model về máy và giải nén model:

- o Truy cập vào đường link và copy link model cần tải:
[PaddleOCR2Pytorch/doc/doc_en/models_list_en.md at main · frotms/PaddleOCR2Pytorch](https://paddleocr2pytorch.readthedocs.io/en/latest/models_list_en.html)

```
- !wget -P <link model muốn tải>
```

- o Tiến hành giải nén model:

```
- !tar -xf <đường dẫn đến file tar> -C <đường dẫn đến folder chứa model>
```

4. Tùy chỉnh file Config:

- Ở đây, ta có thể chỉnh file config phù hợp với nhu cầu thực tế của model, tuy nhiên trước tiên ta phải chỉnh lại một vài tham số cần thiết:
 - + `save_model_dir`: <đường dẫn đến nơi lưu model >: model sau khi train sẽ được lưu ở đường dẫn này.
 - + `epoch_num`: xxx: số lượng epoch muốn train.
 - + `use_gpu`: true – tùy thuộc vào nhu cầu bản thân.
 - + `character_dict_path`: <đường dẫn đến file character>: có thể tự tạo, tìm kiếm hoặc sử dụng file character được PaddleOCR cung cấp sẵn.
 - + `pretrained_model`: <đường dẫn đến model pretrained>: đường dẫn đến model pretrained được giải nén bước bước 3.
 - + `learning_rate`: xxx: chỉ số learning trong quá trình học.
 - + `Train/data_dir`: đường dẫn đến folder chứa dataset.
 - + `Train/label_file_list`: Đường dẫn đến file .txt của tập train (đề cập ở bước 1. Chuẩn bị bộ dữ liệu).
 - + `Eval/data_dir`: đường dẫn đến folder chứa dataset.
 - + `Eval/label_file_list`: Đường dẫn đến file .txt của tập Validate (đề cập ở bước 1. Chuẩn bị bộ dữ liệu).

5. Train model:

- Bước cuối cùng, ta tiến hành train model bằng dòng lệnh duy nhất:

`!python3 PaddleOCR/tools/train.py -c <đường dẫn đến file config>`
- Tiếp đó là chờ đợi, ta có thể mua các gói pro của google colab để train nhanh chóng hơn. Cuối cùng khi model đã train xong, truy cập vào folder được lưu trong `save_model_dir`, để lấy model ra sử dụng.

IV) Xây dựng ứng dụng OCR trên điện thoại:

1) Xây dựng API xử lý yêu cầu chạy mô hình:

a) Framework:

- Vì đây là một ứng dụng chỉ cần có một API nên nhóm quyết định sử dụng ngôn ngữ Python và framework Flask để có thể xây dựng API một cách nhanh chóng và hiệu quả nhất.

b) Tiến hành xây dựng API:

```
Tabnine | Edit | Test | Explain | Document | Ask
@app.route('/upload', methods=['POST'])
def upload_image():
    try:
        # Ensure the temp directory exists
        print("Hello, world!")

        if 'image' not in request.files:
            return jsonify({'error': 'No image provided'}), 400

        image_file = request.files['image']
        image = Image.open(image_file).convert('RGB')

        result = process_image(image)

        return jsonify(result)

    except Exception as e:
        return jsonify({'error': str(e)}), 500

Tabnine | Edit | Test | Explain | Document | Ask
@app.route('/', methods=['POST'])
def EmptyAPI():
    return jsonify({
        'result': 'connect successfully',
    })
```

- Chúng em xây dựng hệ thống chỉ với 1 API đó chính là upload hình ảnh.
- Client có thể gửi hình ảnh thông api này và server sẽ trả về những câu văn hay từ ngữ trong ảnh đó cho người dùng.

```
Tabnine | Edit | Test | Explain | Document | Ask
def process_image(image):
    # Perform OCR on the image
    result = ocr.ocr(np.array(image), det=True, rec=True)

    # Extract coordinates, text, and confidence scores
    boxes = [line[0] for line in result[0]]
    texts = [line[1][0] for line in result[0]]
    scores = [line[1][1] for line in result[0]]

    # Sort the boxes and reorder texts and scores accordingly
    sorted_indices = sorted(range(len(boxes)), key=lambda i: sort_box(boxes[i]))
    boxes = [boxes[i] for i in sorted_indices]
    texts = [texts[i] for i in sorted_indices]
    scores = [scores[i] for i in sorted_indices]

    # Draw bounding boxes on the image
    draw = ImageDraw.Draw(image)
    for box in boxes:
        draw.polygon([tuple(point) for point in box], outline='red')

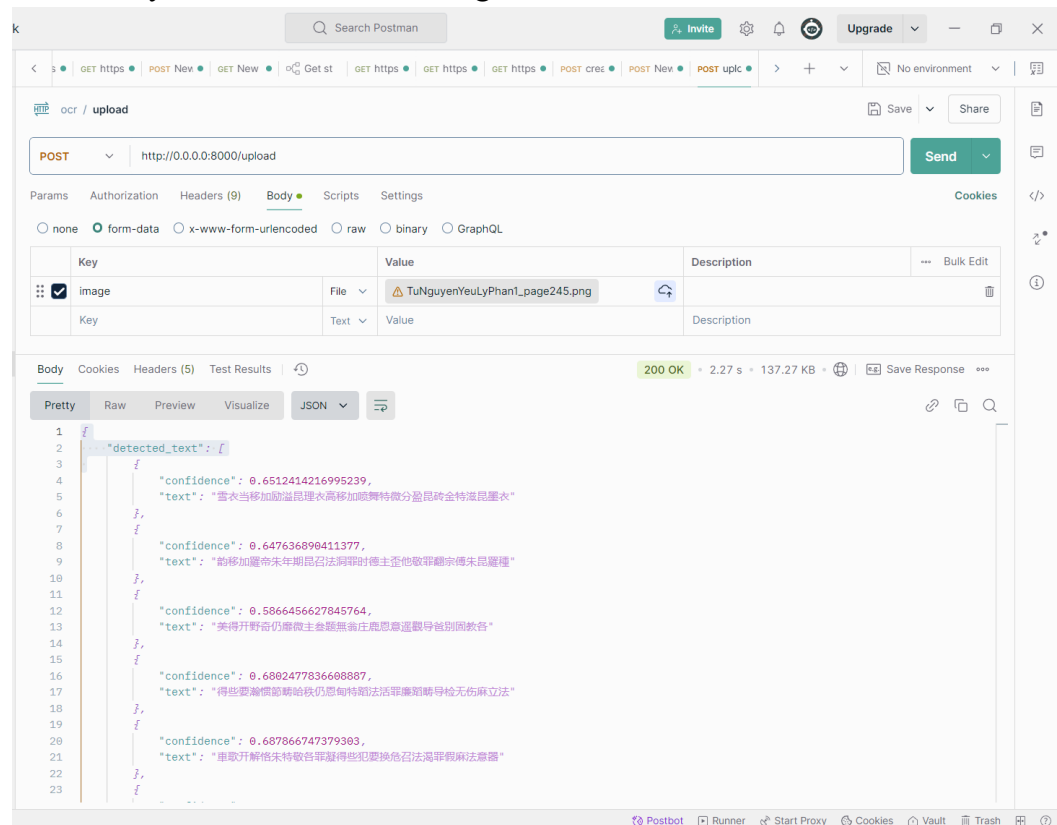
    # Convert the result image to Base64
    buffered = BytesIO()
    image.save(buffered, format="JPEG")
    result_image_base64 = base64.b64encode(buffered.getvalue()).decode('utf-8')

    # Return the result as a dictionary
    return {
        'detected_text': [{'text': text, 'confidence': score} for text, score in zip(texts[:-1], scores[:-1])],
        'image': result_image_base64 # Image returned as Base64
    }
```

- Đề thuận tiện cho việc xử lý thì khi nhận ảnh của người dùng thì em sẽ lưu vào bộ nhớ tạm hình ảnh đó, trong folder temp. Sau đó sẽ dùng chính hình ảnh đã lưu để đem đi OCR và nhận kết quả.
- Sau khi OCR và nhận kết quả được một list String trả về thì em sẽ tiến hành phục dựng lại hình ảnh (thêm các box tìm được vào trong hình ảnh) và trả về người dùng hình ảnh đó.
- Tuy nhiên nhóm em nhận thấy rằng text được trả về không đúng với định dạng chữ Nôm (trên xuống dưới và phải qua trái) thì trước khi trả về kết quả text cho người dùng chúng em tiến hành sort lại các box đó.
- Sau khi đã xong thì chúng em cũng sẽ xóa file hình ảnh trong tmp để giúp tối ưu hóa bộ nhớ cho server.

c) Test API.

- Chúng em sử dụng POSTMAN để test API.
- Sau khi khởi chạy server thì bắt đầu dùng POSTMAN để kiểm tra API.



- Kết quả: server nhận được API và trả về phản hồi cho người dùng.

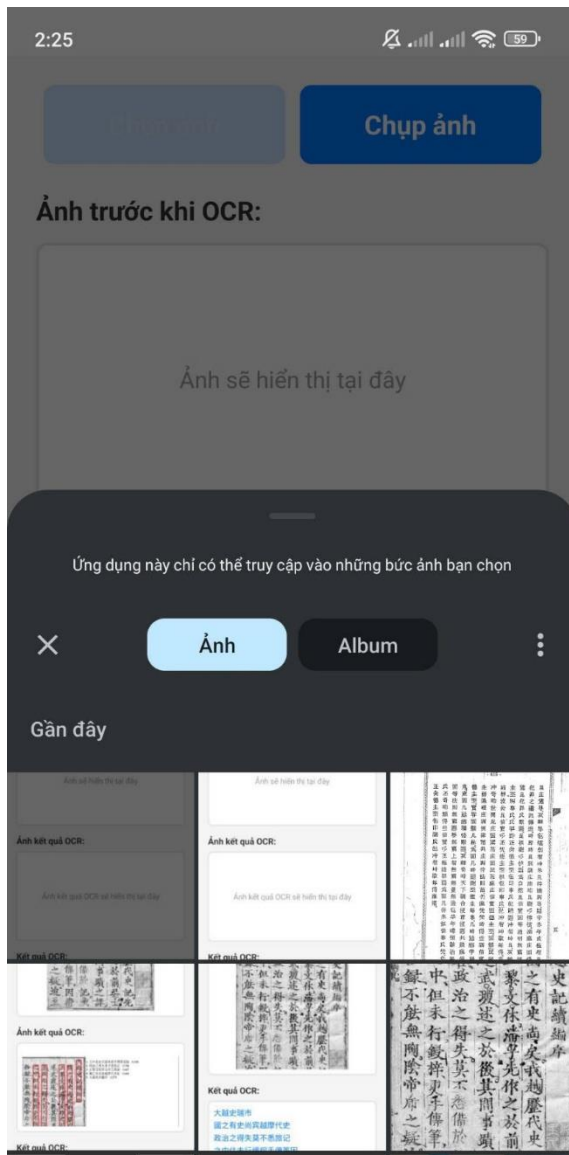
2) Xây dựng giao diện ứng dụng điện thoại:

a) Tổng quan về giao diện:

- Về xây dựng giao diện, nhóm sử dụng React Native - một trong những frame work khá mạnh về mảng lập trình ứng dụng di động, để xây dựng nên phần giao diện app, kết hợp với đó, nhóm sử dụng thêm expo để kết nối điện thoại với ứng dụng thay vì sử dụng thiết bị giả lập tiêu tốn rất nhiều tài nguyên của máy.
- Về giao diện, giao diện được thiết kế đơn giản nhưng vẫn đảm bảo các chức năng của một ứng dụng OCR: Nút tải hình ảnh, Nút chụp ảnh, Khu vực hiển thị ảnh đã upload/chụp, khu vực hiển thị ảnh đã bounding box và cuối cùng là khu vực hiển thị kết quả OCR:



- Về chức năng: ứng dụng cho phép người dùng nhậ vào một bức ảnh được lưu trữ trong thư viện ảnh của người dùng, đồng thời cho phép người dùng chụp vào một bức ảnh để OCR:



b) Cách xây dựng giao diện của người dùng:

- Thiết lập môi trường code:
 - + Bước 1: tải về nodeJ: [Node.js — Run JavaScript Everywhere](#)
 - + Bước 2: tải về Expo CLI: `npm install -g expo-cli`
 - + Bước 3: tạo một project mới: `expo init PaddleOCRApp`
 - + Bước 4: Chạy ứng dụng: `npx expo start`
 - + Bước 5: Trên thiết bị điện thoại, ta tải về ứng dụng expo, sau đó mở ứng dụng và quét mã hiển thị khi chạy ứng dụng
 - + Bước 6: khi việc kết nối hoàn thành thì tiến hành code giao diện.
- Code Xây dựng giao diện: sử dụng react native để tiến hành xây dựng giao diện, nguồn tài liệu: [Introduction · React Native](#)
- Sử dụng dịch vụ API của backend được xây dựng ở trên để tiến hành gửi ảnh và nhận về kết quả OCR của ảnh.

3) Tìm hiểu thêm về app OCR đã được xây dựng sẵn dựa trên PaddleLite:

a) Tổng quan về PaddleLite:

- PaddleLite là một framework open-source nằm trong cùng hệ sinh thái với PaddleOCR. Đây là bản nâng cấp của Paddle-Mobile, giúp hỗ trợ trong việc sử dụng, deploy mô hình inference trên các thiết bị di động, thiết bị IoT và các hệ thống nhúng.
- PaddleLite hỗ trợ cho đa dạng ngôn ngữ lập trình như Java, C++ và Python; cũng như các nền tảng như Android, iOS, Linux, Windows, MacOS.
- Chi tiết có thể xem thêm tại:
 - + [Website chính thức.](#)
 - + [Github.](#)

b) Chuẩn bị mô hình light-weight (.nb):

- Sau khi fine-tuning cho mô hình ch_PP-OCRv3 cho text detection và text recognition, ta sẽ dùng framework PaddleLite để chuyển mô hình đã fine tune thành mô hình lightweight để có thể chạy trên thiết bị di động.
- Quy trình sẽ chuyển các thành phần pdiparams (phần parameter của mô hình) và pdmodel (phần cấu trúc của mô hình) của các mô hình thuộc hệ sinh thái PaddleOCR thành một file duy nhất (.nb) dành cho nền tảng mà ta muốn deploy mô hình.
- Tổng quan quá trình chuyển đổi:
 - + Biến đổi format lưu trữ tương ứng với thiết bị sẽ được deploy để giảm kích thước mô hình.
 - + Sử dụng các thao tác tối ưu hóa như: quantization, subgraph fusion, hybrid scheduling, and Kernel optimization để giảm thời gian, tài nguyên vi xử lý và tài nguyên RAM cần trong quá trình chạy mô hình.

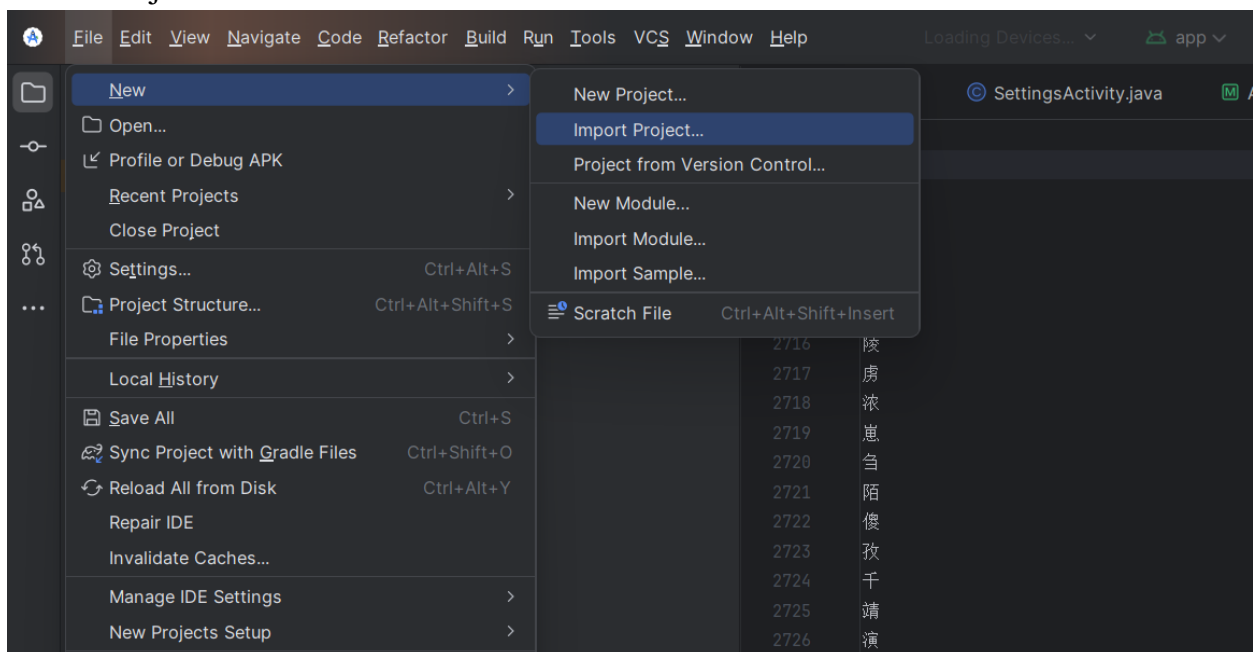
- Các bước chuyển đổi (sử dụng phiên bản PaddleLite 2.10):
 - + Cài đặt framework PaddleLite phiên bản 2.10 (cần đảm bảo phiên bản python = 3.7):
 - + `pip install paddlelite==2.10`
 - + Đảm bảo thư mục model chứa 2 thành phần:
 - + `model.pdmodel` + `model.pdiparams`
 - + Dùng lệnh `paddle_lite_opt` để tiến hành chuyển đổi, với cú pháp bên dưới (nếu không thể chạy với cú pháp bên dưới, ta cần thêm “python <liệt kê cụ thể đường dẫn đến file `paddle_lite_opt` trong thư mục Scripts của môi trường python đã cài `paddlelite`>”):

`paddle_lite_opt --model_file=<đường dẫn đến model.pdmodel> --param_file=<đường dẫn đến model.pdiparams> --optimize_out=<đường dẫn đến model sau khi chuyển đổi> --valid_targets=<loại môi trường sẽ deploy> --optimize_out_type=naive_buffer`

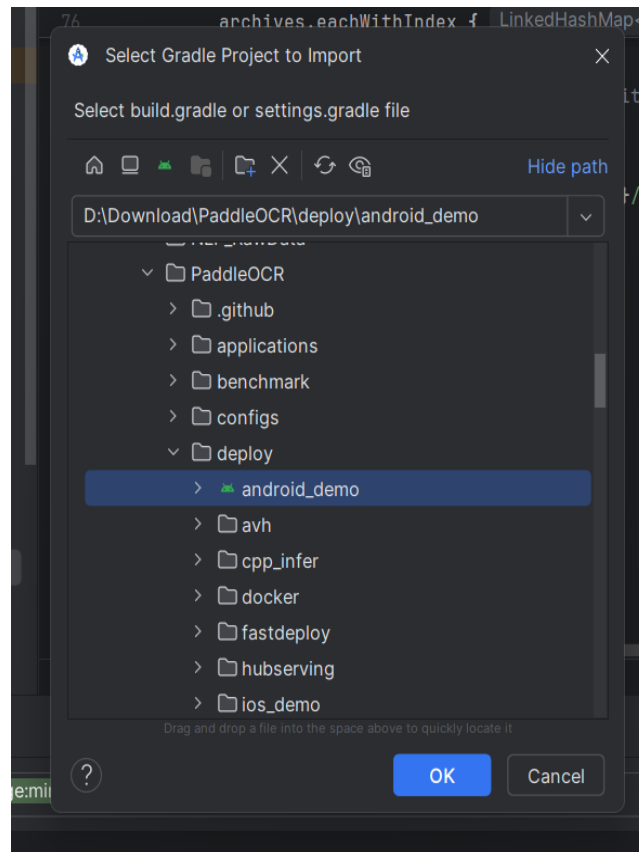
- Chi tiết các bước thực hiện quá trình chuyển đổi có thể xem thêm tại:
 - + [Web chính thức.](#)
 - + [Github.](#) (tại mục 2.1 trong link).

c) Cài đặt demo app có sẵn cho thiết bị android với mô hình vừa thu được:

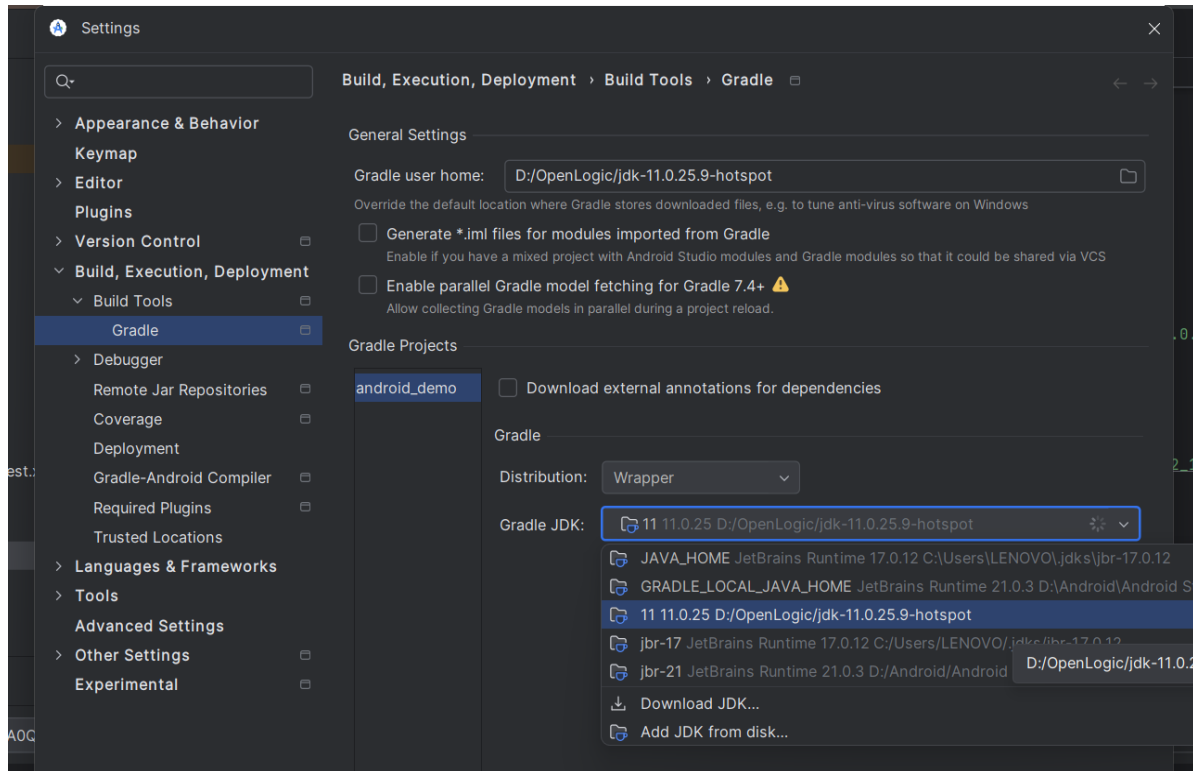
- Bước 1: Cài đặt Android Studio. Cài đặt Java 11.
- Bước 2: Clone git của PaddleOCR về máy.
- `git clone https://github.com/PaddlePaddle/PaddleOCR.git`
- Bước 3: Mở Android Studio, tạo một project mới, vào phần File>New>Import Project.



- Bước 4: Trong cửa sổ chọn project cần import ta tìm đến đường dẫn của git PaddleOCR đã clone trên máy PaddleOCR>deploy và chọn android_demo.



- Bước 5: Sau khi import project thành công thì vào phần File>Settings...>Build, Execution, Deployment>Build Tools>Gradle.
 - + Ở cả “Gradle user home” và “Gradle JDK”, ta chọn đường dẫn tới vị trí Java11 mà ta đã cài trước đó. (Gradle JDK nếu không tìm thấy Java11 thì ta chọn Add JDK from disk).



- Bước 6: Kết nối với thiết bị cần deploy app.
- Bước 7: Điều chỉnh lại các font/ chữ tiếng Trung Quốc trong các file giao diện trong các thư mục “android_demo/app/src/main/java” (hoặc “res”) thành tiếng Anh để dễ thao tác và sử dụng.

- Bước 8: Điều chỉnh lại file “android_demo/app/src/build.gradle” ở phần android, sửa “compileSdkVersion 29” và “targetSdkVersion 29” thành 30. Cũng như liệt kê cụ thể kiến trúc bộ xử lý của thiết bị sẽ deploy (ở đây em sử dụng arm64-v8a) thông qua sdk.

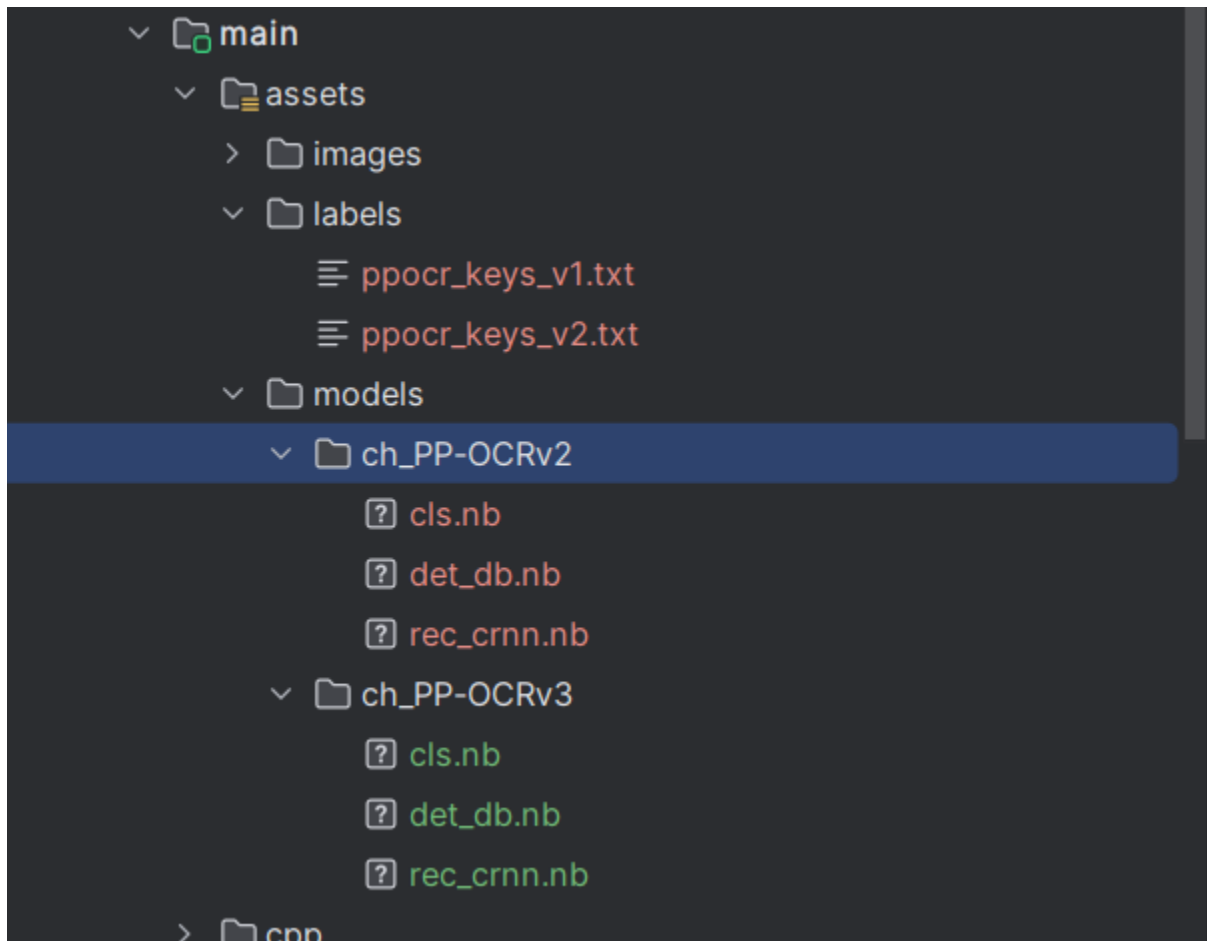
```

4
5 android {
6     compileSdkVersion 30
7     defaultConfig { DefaultConfig it ->
8         applicationId "com.baidu.paddle.lite.demo.ocr"
9         minSdkVersion 23
10        targetSdkVersion 30
11        versionCode 2
12        versionName "2.0"
13        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
14        externalNativeBuild { ExternalNativeBuildOptions it ->
15            cmake { ExternalNativeCmakeOptions it ->
16                cppFlags "-std=c++11 -frtti -fexceptions -Wno-format"
17                arguments '-DANDROID_PLATFORM=android-23', '-DANDROID_STL=c++_shared', '-DANDROID_ARM_NEON=TRUE'
18            }
19        }
20        ndk { NdkOptions it ->
21            abiFilters "arm64-v8a"
22        }
23    }

```

- Bước 9: Chọn Sync Project with Gradle Files, sau khi Sync hoàn tất, ta chạy thử lần đầu (hãy đảm bảo thiết bị đang được kết nối với máy tính).
- Bước 10: Cấp quyền và cài đặt trên điện thoại.
- Bước 11: Chuẩn bị thư mục tên ch_PP-OCRv3 chứa cả 2 model .nb det và rec đã chuẩn bị và copy vào thư mục android_demo/app/src/main/assets/ models:
 - + Đổi tên tương ứng 2 model thành “det_db.nb” và “rec_crnn.nb”, copy model cls.nb từ thư mục assets/ch_PP-OCRv2 sang ch_PP-OCRv3 (vì bọn em không fine tuning cho phần mô hình classification nên có thể dùng tạm mô hình có sẵn).

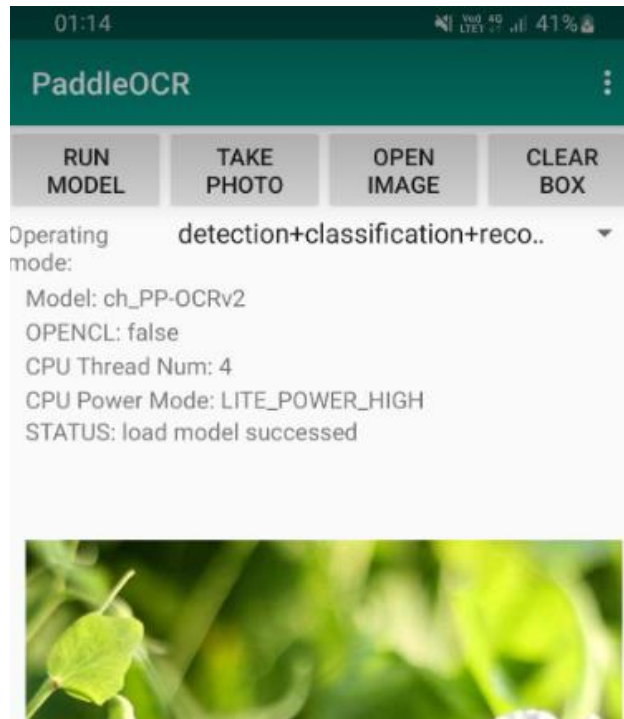
- Bước 12: Chuẩn bị từ điển Hán-Nôm, tại đây em trích các từ Hán Nôm từ file excel SinoNôm dictionary từ bài Lab01 sang file text với tên ppocr_keys_v2.txt và đưa vào thư mục assets/labels.



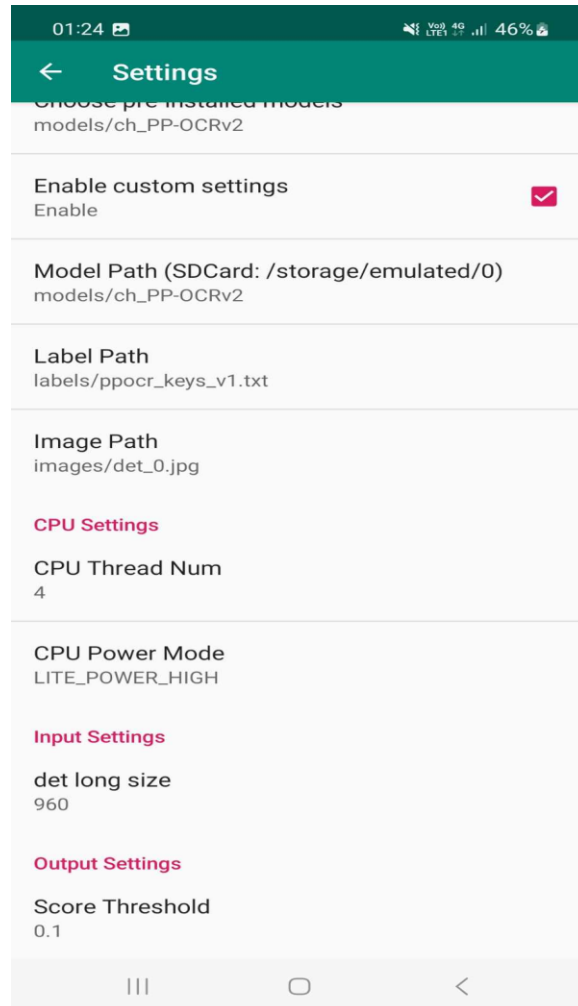
- Bước 13: Xóa app hiện tại đã cài trên thiết bị di động và tiến hành build lại và deploy app trực tiếp lên thiết bị đang kết nối hoặc build>build APK để tạo file cài đặt.

d) Hướng dẫn sử dụng (cần cấp quyền khi sử dụng):

- Có 4 nút chức năng ở màn hình chính:
 - + Run model: dùng để chạy model.
 - + Take photo: dùng để chụp ảnh.
 - + Open image: dùng để chọn ảnh cần OCR từ album ảnh trên máy.
 - + Clear box: dùng để clear các bounding box được vẽ lên ảnh.
- Có thể điều chỉnh loại mô hình khi chạy ở Operating mode:



- Trong phần setting (ba dấu chấm ở góc phải trên). Bật Enable custom settings để có thể điều chỉnh:
 - + Model Path: sửa v2 thành v3 để dùng model.
 - + Lable Path: sửa v1 thành v2 để dùng từ điển SinoNôm (lưu ý chỉ có tác dụng khi model cũng được chuyển và phải chuyển trước khi chuyển model).
 - + Image Path là đường dẫn đến ảnh mặc định (do vấn đề bảo mật của Android phiên bản mới, các đường dẫn này không thể xem được).
 - + CPU Thread Num: điều chỉnh số lượng luồng CPU sẽ dùng để chạy mô hình.
 - + CPU Power mode: mode chạy của CPU.
 - + det long size: độ dài tối đa của bức ảnh (ảnh sẽ được scale lại nếu vượt quá độ dài này).
 - + Score Threadhold: mức điểm tối thiểu mà một box được tính là OCR thành công.



- Chi tiết demo có thể xem thêm: [Tại Github](#).

V) Kết quả thực nghiệm:

- Dưới đây là kết quả kiểm tra thực nghiệm ứng dụng trong việc OCR một ảnh có chữ Nôm (video demo nằm trong thư mục được nhóm gửi cho thầy):



- Kết quả thực nghiệm:
 - + Tương đối tốt trong việc bounding các box: chỉ bounding những chữ Nôm, không bounding những vật thể lạ: khung viền, hoặc những đốm đen như model cũ.
 - + Tuy nhiên, đối với những ảnh có chất lượng thấp thì việc bounding vẫn chưa chính xác.

VI) Tổng Quan về thư mục nộp bài:

- **Root** : Thư mục chính
 - **Additional_Midterm_Data** : Thư mục chứa bộ dữ liệu bổ sung GK
 - **Khanh_Additional_Midterm_Data** : Thư mục chứa bộ dữ liệu bổ sung GK của Khánh
 - **Thanh_Additional_Midterm_Data** : Thư mục chứa bộ dữ liệu bổ sung GK của Thanh
 - **Mobile** : Thư mục chứa ứng dụng mobile
 - **PaddleOCRApp_usingReactNative/PaddleOCRApp** : Thư mục chứa code ứng dụng mobile được code bằng React Native.
 - **PaddlePadlleLiteApp** : Thư mục chứa ứng dụng mobile sử dụng PaddleLite tìm hiểu thêm được.
 - **PaddleOCR** : Thư mục chứa model và API để sử dụng model
 - **Det** : Thư mục chứa model Detection.
 - **Python-api-project/src** : Thư mục chứa code API để sử dụng model
 - **rec** : Thư mục chứa model Recognition.
 - **TrainingModel** : Thư mục chứa code quá trình fine tuning của nhóm trên 2 mô hình : Detection và Recognition.
 - **prebuilt_android_demo_from_PaddleOCR_git**: thư mục chứa các file liên quan đến app PaddleOCR được xây dựng sẵn cho android trên trang git chính thức.
 - **android_demo**: thư mục project chứa code để build và deploy app.
 - **video_demo** : Thư mục chứa 2 video demo về app xây dựng bằng React Native và PaddleLite.

VII) Phân công việc nhóm:

MSSV	Tên thành viên	Nhiệm vụ hoàn thành
22127190	Phạm Nguyên Khánh	<ol style="list-style-type: none">1. Chuẩn bị bộ dữ liệu.2. Tìm hiểu và Fine tuning model Recognition.3. Code giao diện ứng dụng sử dụng sử dụng React Native.
22127227	Đặng Văn Kỳ	<ol style="list-style-type: none">1. Chuẩn bị bộ dữ liệu.2. Tìm hiểu và Fine tuning model Detection.3. Code API backend để nhận và OCR ảnh.
22127392	Lê Phước Thạnh	<ol style="list-style-type: none">1. Chuẩn bị bộ dữ liệu.2. Tìm hiểu fine tuning model Recognition.3. Tìm hiểu và triển khai app OCR đã được xây dựng sẵn dựa trên PaddleLite.
22127124	Nguyễn Anh Hoàng	<ol style="list-style-type: none">1. Chuẩn bị bộ dữ liệu.2. Tìm hiểu fine tuning model Detection.3. Tìm hiểu thêm về app OCR đã được xây dựng sẵn dựa trên PaddleLite.

VIII) Tham khảo:

- Cách fine tuning PaddleOCR recognition :
 - [Hướng dẫn fine-tuning mô hình PaddleOCR recognition với custom dataset - THI GIÁC MÁY TÍNH](#)
 - [Fine-tune - PaddleOCR Documentation](#)
- Cách fine tuning PaddleOCR detection:
 - [Hướng dẫn Finetuning Text Detection model](#)
- Cách xây dựng ứng dụng di động sử dụng react native và expo : [\(42\) Code ứng dụng di động siêu dễ trong 15 phút với JavaScript và React-Native - YouTube](#)