

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH



ĐỒ ÁN CUỐI KỲ
THỊ GIÁC MÁY TÍNH NÂNG CAO (CS331.L22.KHCL)

VIETNAMESE RECEIPT EXTRACTION

Giảng viên hướng dẫn:
Ts. Mai Tiến Dũng

Thành viên nhóm:
Trương Minh Sơn 19522143
Nguyễn Thành Trung 19522432
Nguyễn Khả Tiến 19522337

Thành phố Hồ Chí Minh



Mục lục

1 TỔNG QUAN	3
1.1 Sơ lược về đề án	3
1.2 Mục tiêu của đề án	3
1.3 Ngữ cảnh bài toán	3
1.4 Thách thức bài toán	4
2 Xây dựng bộ dữ liệu	5
3 Các bước thực hiện chính và quá trình huấn luyện mô hình	6
3.1 Input Image	7
3.2 Phát hiện hóa đơn (Yolov4), cắt phần hóa đơn (OpenCV)	7
3.2.1 Mô hình YOLO Darknet	7
3.2.2 Giới thiệu	7
3.2.3 Thông số của bộ dữ liệu	8
3.2.4 Xử lý dữ liệu	10
3.2.5 Cài đặt mô hình	11
3.2.6 Các bước để train	11
3.3 Tiền xử lý ảnh (Cắt phần hóa đơn)	16
3.4 Phát hiện văn bản (Paddle):	16
3.5 VietOCR	18
3.5.1 CNN Của Mô Hình OCR:	19
3.5.2 Language model: AttentionOCR	20
3.5.3 Language model: TransformerOCR	20
3.5.4 Huấn luyện mô hình VietOCR:	22
3.6 Model PICK (KIE)	23
4 Kết quả huấn luyện và đánh giá mô hình:	25
4.1 Các độ đo được sử dụng:	25
4.1.1 Cơ chế IOU	25
4.1.2 Non max suppression	26
4.1.3 Các metric khác	27
4.1.4 Công thức tính AP	29
4.1.5 Accuracy Full Sequence & Accuracy Per Character	30



4.1.6 Character Error Rate	31
4.2 Kết quả và đánh giá mô hình:	32
5 Hướng phát triển	33



1 TỔNG QUAN

1.1 Sơ lược về đề án

Đề án này hướng đến tìm hiểu quy trình và thực hiện một bài toán Information Extraction (Rút trích thông tin) nói chung và các bài toán con trong đó nói riêng. Cụ thể hơn là đề án này sẽ tìm hiểu và thực hiện quy trình nhận diện và trích xuất thông tin văn bản từ ảnh chụp hóa đơn tiếng Việt trong cuộc sống bằng thiết bị di động.

1.2 Mục tiêu của đề án

Để thực hiện được đề án, chúng tôi đã đặt ra những mục tiêu như sau:

- Hiểu và nắm rõ ý tưởng cũng như vấn đề cần giải quyết của các bài toán con.
- Huấn luyện và đánh giá được một số mô hình con trong pipeline (Yolov4 và VietOCR). Từ đó rút ra được kết luận và nguyên nhân cho kết quả đó để phục vụ cho công việc cải tiến sau này.
- Xây dựng được ứng dụng demo Information Extraction trong thực tế.

1.3 Ngữ cảnh bài toán

Các hóa đơn thường mang lại các thông tin cần thiết cho các công ty, và đa số chúng thuộc dạng giấy hoặc là các văn bản dạng điện tử như tệp tin PDF hoặc dạng ảnh. Để quản lý những thông tin này một cách hiệu quả, các công ty trích xuất và lưu trữ chúng vào cơ sở dữ liệu của họ.

Thêm vào đó các hóa đơn cũng thường mang lại thông tin rất cần thiết trong việc quản lý chi tiêu của một cá nhân, và đa số người dùng thường nhận được hóa đơn ở dạng giấy hoặc dạng ảnh. Để quản lý thông tin chi tiêu thông qua hóa đơn một cách hiệu quả. OCR (optical character recognition) là một công nghệ phục vụ nhận diện chữ viết trong ảnh một cách hiệu quả và tự động.

Việc trích xuất được thông tin từ hóa đơn 1 cách tự động có thể giúp người dùng chỉ cần chụp hình 1 hóa đơn và thông tin cần thiết sẽ được tự động lưu lại và đưa vào cơ sở dữ liệu của người dùng đó. Nhờ vậy mà người dùng khỏi phải lặn lội việc nhập hóa đơn 1 cách thủ công nữa.



Input/Output bài toán


- Input: Ảnh có chứa hóa đơn Tiếng Việt. Ảnh có thể chụp ở bất kỳ background nào xung quanh nhưng nhóm có ràng buộc là hóa đơn phải được chụp đúng hướng nhất có thể và không bị che lấp.

- Output: Các trường thông tin như trong Hình 1

4 trường thông tin cần trích xuất:

- SELLER: tên của cửa hàng
- ADDRESS: địa chỉ của cửa hàng
- TIMESTAMP: thời gian xuất hoá đơn
- TOTAL_COST: tổng chi phí có trong hoá đơn

Những hoá đơn sẽ có nhiều bố cục khác nhau nên vị trí cũng như số lượng thông tin trích xuất được có thể khác nhau.

Input	Output (1 line, 4 fields, separated by).
	SCTC CÔ THỎ 104 TRẦN PHÚ – CẨM PHẢ 104 Trần Phú – phường Cẩm Tây – Thành phố Cẩm Phả Ngày 09/08/2020 (7:44 CH 7:44CH) Tổng thanh toán 115.000đ

Hình 1

1.4 Thách thức bài toán

- Bài toán còn gặp nhiều vấn đề nan giải bởi các tác động bên thứ 3 như: ảnh hưởng độ sáng, méo mó,...
- Các tờ hóa đơn không nằm ở một vị trí nhất định trong ảnh và không phải lúc nào cũng thẳng, đều và dễ đọc, kể cả việc mỗi hóa đơn có kích thước khác nhau.



- Bài toán xử lý trên dữ liệu tiếng Việt nên có thể bị ‘bỏ sót’ các dấu thanh (‘sắc’, ‘huyền’, ‘hỏi’, ‘ngã’, ‘nặng’) vì chúng có kích thước nhỏ hơn so với chữ.

Mặc dù những khó khăn này, các thuật toán deep learning ngày nay hoạt động tương đối tốt và càng phát triển.

2 Xây dựng bộ dữ liệu

Sử dụng bộ dữ liệu MCOCR-2021 [1] để phục vụ huấn luyện (900 ảnh), từ cuộc thi RIVF2021 của Việt Nam bao gồm: 1000 ảnh huấn và 400 ảnh kiểm thử. Bộ dữ liệu bao gồm những tấm ảnh chứa hoá đơn bằng Tiếng Việt được chụp từ điện thoại. Tuy nhiên chúng tôi chỉ lấy hình ảnh làm dữ liệu chứ không lấy label từ ban tổ chức bởi vì:

- Tập dữ liệu từ ban tổ chức có rất nhiều trường hợp label các trường dữ liệu sai, cố tình thêm ký tự... Chủ yếu là mang tính đánh đố khá lớn.
- Thêm vào đó có rất nhiều hình ảnh bị nghiêng với góc lớn (vd: 90, 180, -180 độ ...). Mà như phần **Ngữ cảnh bài toán** đã đề cập, thì chúng tôi chỉ xử lý cho trường hợp ảnh hóa đơn được chụp đúng hướng vì thời gian thực hiện đề tài khá ngắn và việc cần làm khá nhiều nên không thể giải quyết được hết những trường hợp như thế này.

Với những lý do trên, nhóm chúng tôi thực hiện một giai đoạn **xử lý thủ công** với các hình ảnh bị nghiêng từ ban tổ chức thành ảnh có hóa đơn được chụp đúng hướng như 2 hình ảnh dưới đây.



Hình 2: Dữ liệu mẫu từ ban tổ chức

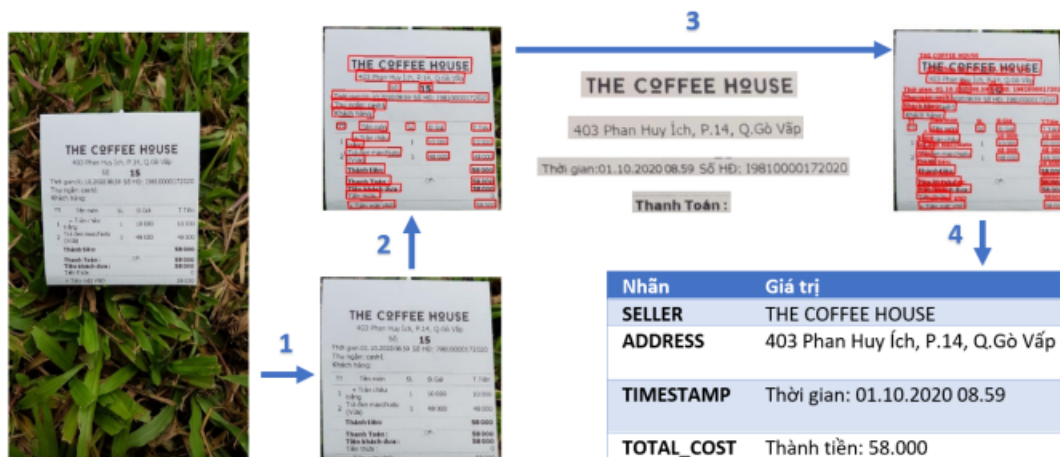


Hình 3: Dữ liệu sau khi lọc và xử lý thủ công

Tóm lại thì nhóm chúng tôi sẽ huấn luyện trên bộ ảnh có hóa đơn được chụp đúng hướng và có thể chứa bất kỳ background nào xung quanh.

3 Các bước thực hiện chính và quá trình huấn luyện mô hình

Pipeline của nhóm sẽ có 4 bước chính như hình vẽ dưới đây:



Hình 4: Mô tả quá trình trích xuất thông tin hóa đơn

1. Phát hiện hóa đơn (Yolov4), cắt phần hóa đơn (OpenCV)
2. Phát hiện văn bản (Paddle)



3. Nhận diện văn bản (VietOCR)

4. Trích xuất thông tin (PICK)

Ở pipeline này, nhóm chúng tôi sẽ chỉ huấn luyện YOLOv4 và VietOCR. Các mục còn lại sẽ sử dụng module có sẵn được public. Cụ thể hơn nhóm chúng tôi sẽ trình bày dưới đây:

3.1 Input Image

Đây là công đoạn chuẩn bị dữ liệu cho bài toán. Ở đây, nhóm đã sử dụng bộ dữ liệu như đã nêu ở **Mục 2**

3.2 Phát hiện hóa đơn (YOLOv4), cắt phần hóa đơn (OpenCV)

3.2.1 Mô hình YOLO Darknet

Bước này sẽ tìm ra vị trí và sẽ cắt hoá đơn trên ảnh. Nhóm đã dùng **Mô hình YOLO Darknet** để thực hiện công đoạn này.

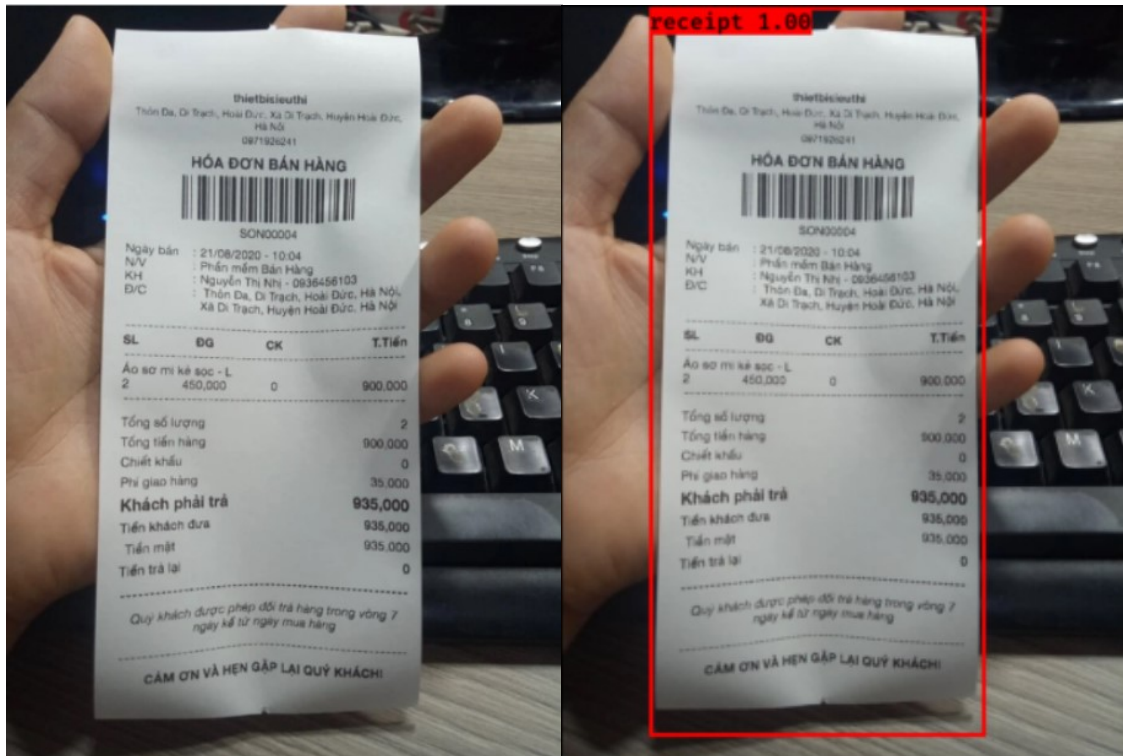
3.2.2 Giới thiệu

YOLO là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. YOLO được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers. Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn full-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

Lý do chọn mô hình:

- Cách train mô hình đơn giản, với các cú pháp đã có sẵn trên mạng nên dễ dàng theo sát quá trình thực hiện.
- Hiểu biết về mô hình qua quá trình tìm hiểu.
- Đã từng sử dụng cho bài toán khác nên có kinh nghiệm làm việc với mô hình.
- Hỗ trợ các thông số giúp đánh giá mô hình mà nhóm đã đề ra.

Mô hình sẽ nhận dữ liệu đầu vào là một ảnh chứa hóa đơn như hình 5. Và kết quả đầu ra của mô hình được thể hiện như ở hình 6.



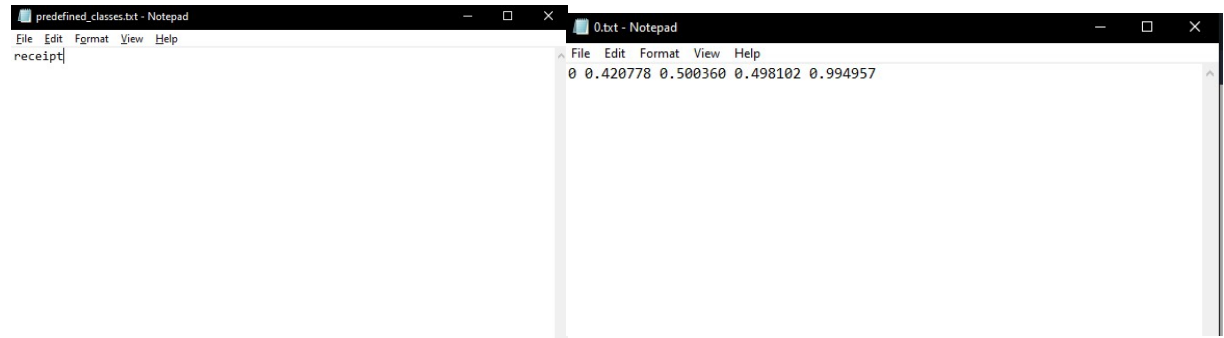
Hình 5: Input

Hình 6: Output

3.2.3 Thông số của bộ dữ liệu

Bộ dữ liệu sẽ có:

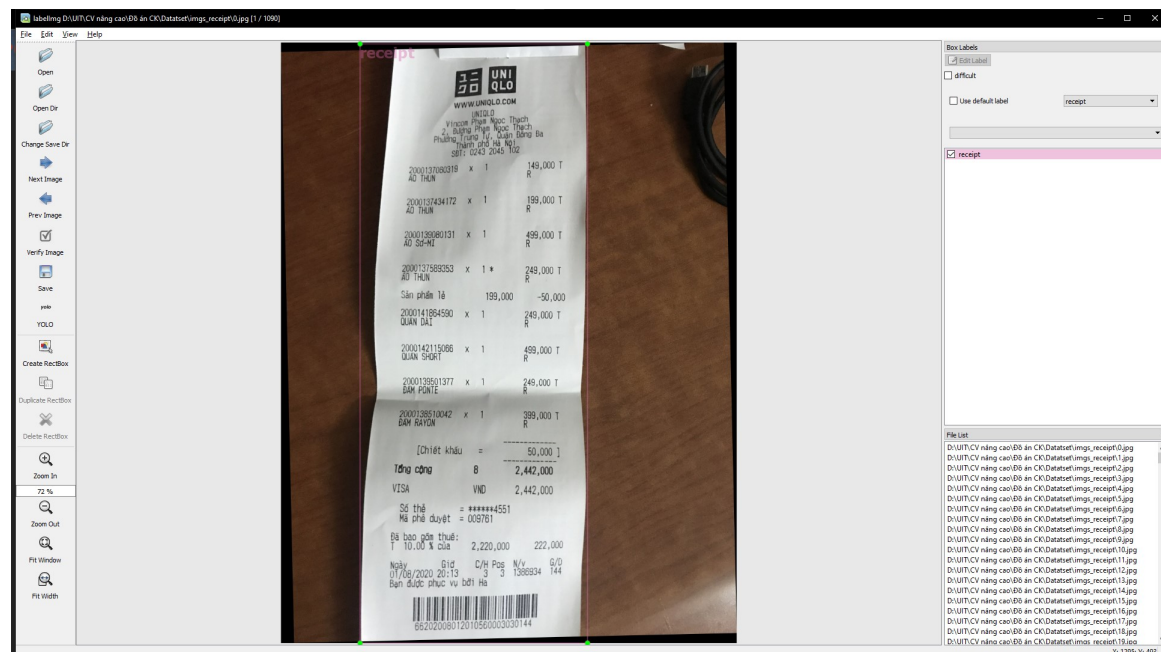
- Phần ảnh gồm các file ảnh gốc: .jpg
- File .txt là file đầu ra, tương ứng với các nhãn ở trên ảnh, tên file .txt sẽ trùng với tên file .png, để biểu thị rằng, các nhãn trong file .txt này sẽ là nhãn của bức hình nào. File .txt có cấu trúc từng dòng là: object - x y w h, với object là classes của chúng ta tương ứng số 0. x và y tương ứng là tọa độ tâm của bounding box. w và h lần lượt là chiều rộng và chiều cao của bounding box.



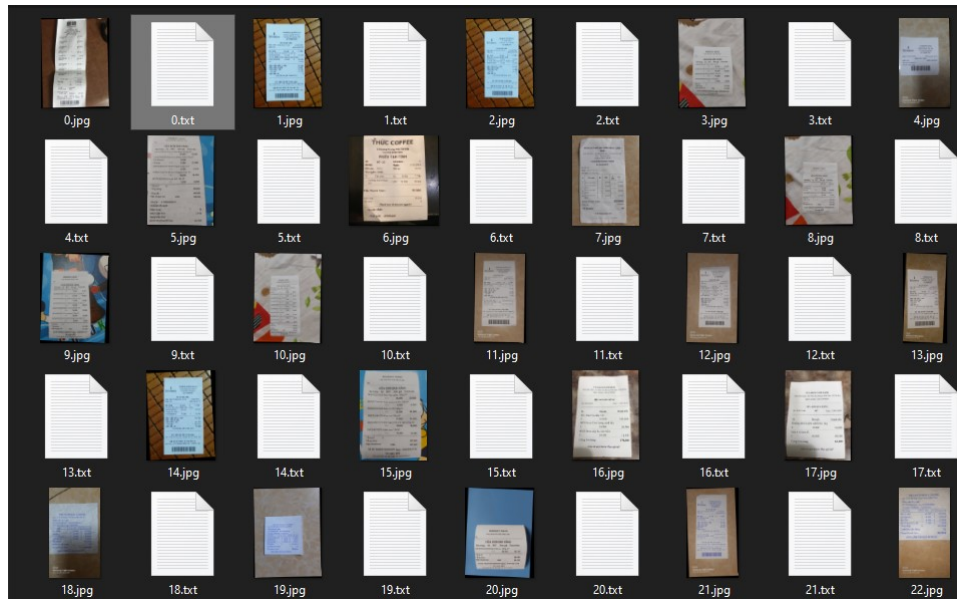
Hình 7: Predefined classes

Hình 8: Class

Một ví dụ trong quá trình gán nhãn



Hình 9: Ví dụ gán nhãn ảnh



Hình 10: Dữ liệu dùng cho việc huấn luyện

3.2.4 Xử lý dữ liệu

Data được xử lý với công cụ LabelImg, là một công cụ dùng để chú thích hình ảnh đồ họa. Cụ thể hơn đó là dùng để gắn nhãn các bounding box cho ảnh. Các bước để xử lý ảnh với công cụ LabelImg như sau:

- Đầu tiên, vì ta sẽ sử dụng mô hình YoloV4 vậy nên ta cần phải thiết lập lại file data/predefined_classes.txt để định nghĩa danh sách các lớp sẽ dùng để train cho mô hình.
- Tiếp đến, ta nhấn chọn Opendir để mở thư mục sẽ làm việc chứa data mà ta lưu trữ
- Sau khi chọn xong thư mục cần làm việc, sẽ hiện lên danh sách các ảnh trong thư mục đó ở góc bên phải.
- Trước khi bắt tay vào thực hiện, ta chọn thư mục lưu trữ các file class - chứa các thông số của mô hình yolov4, bằng cách nhấn chọn vào saveDir và chọn thư mục cần lưu trữ
- Bây giờ, ta tiến hành gắn nhãn trên data bằng cách sử dụng Create Rectbox để vẽ bounding box tương ứng với class ta muốn phân loại trên ảnh
- Sau khi đã vẽ xong các bounding box, sẽ xuất hiện thêm một file .txt tương ứng với các thông số của mô hình yolov4 và ứng với file data đã phân loại



- Cuối cùng, thực hiện vẽ bounding box với các data tiếp theo

3.2.5 Cài đặt mô hình

Để tiếp tục train YOLOv4 trên Colab nhóm thực hiện theo hướng dẫn chủ yếu ở: [\[YOLO Series\] Train YOLO v4 train trên COLAB chi tiết và đầy đủ \(A-Z\)](#) [2] [3] [4] Trong đó, đầu tiên chúng ta cần tải file [yolov4-custom.cfg](#) về máy tính. Mở ra và thiết lập lại sao cho phù hợp với bài toán của chúng ta.

Chuẩn bị file config:

- Đầu tiên cần xác định train mấy class? Chính là bao nhiêu loại đối tượng. Ở đây, nhóm dùng 1 class - receipt.
- Tìm đến dòng 20, sửa `max_batches = max(<số class>*2000,6000)`. Nhóm dùng `max_batches = 6000`.
- Đến dòng 22 sửa thành `steps=80%, 90%` của `max_batches`. Ví dụ ở đây là `steps=4800,5400`.
- Replace toàn bộ các dòng có `"classes=80"` thành `"classes=1"` ở các dòng 970, 1058 và 1146.
- Replace toàn bộ các dòng có `"filters=255"` thành `"filters=18"` ở các dòng: 963, 1051 và 1139.
- Trong trường hợp khi train bị out memory, Chuyển subdivisions (dòng 7) thành 32 (hoặc 64) hoặc có thể giảm size ảnh xuống còn width=416, height=416 (dòng 8,9).

Chuẩn bị file MAKEFILE:

Tải file [Makefile](#). Mở ra và sửa như sau:

- Dòng 1, sửa thành `GPU=1`
- Dòng 2, sửa thành `CUDNN=1`
- Dòng 4, sửa thành `OPENCV=1`

3.2.6 Các bước để train

Sau khi đã hoàn tất chỉnh sửa các file cần thiết chúng ta tiến hành tạo notebook trên colab để train.

Bước 1: Kết nối GG Drive



```
Step 1. Mount drive

[ ] from google.colab import drive
    drive.mount('/content/gdrive')

Mounted at /content/gdrive
```

Hình 11: Mount drive

Bước 2: Tải mã nguồn YOLO về drive.

```
Step 2. Tải mã nguồn YOLO về drive

[ ] %cd /content/gdrive/My\ Drive
    !git clone https://github.com/AlexeyAB/darknet

/content/gdrive/My Drive
Cloning into 'darknet'...
remote: Enumerating objects: 15420, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 15420 (delta 1), reused 6 (delta 1), pack-reused 15413
Receiving objects: 100% (15420/15420), 14.08 MiB | 7.82 MiB/s, done.
Resolving deltas: 100% (10356/10356), done.
Checking out files: 100% (2050/2050), done.

[ ] %cd /content/gdrive/My\ Drive/darknet
    !rm -rf data
    !mkdir data

/content/gdrive/My Drive/darknet
```

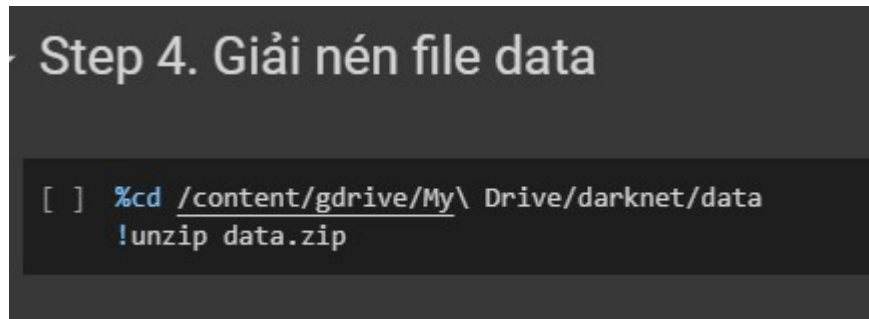
Hình 12: Tải mã nguồn YOLO về drive

Bước 3: Upload file từ máy tính vào thư mục darknet vừa tải về. Tại đây thực hiện những bước sau:

- Upload file data.zip vào trong thư mục data.
- Upload file yolov4-custom.cfg vào thư mục cfg.
- Upload file Makefile vào thư mục darknet.



Bước 4: Giải nén file data



Hình 13: Giải nén file data

Bước 5: Tạo file yolo.names



Hình 14: Tạo file yolo.names

Bước 6: Tạo file train.txt và val.txt



Step 6. Tạo file train.txt và val.txt

```
[ ] %cd /content/gdrive/My Drive/darknet

import glob2
import math
import os
import numpy as np

files = []
for ext in [".png", ".jpeg", ".jpg"]:
    image_files = glob2.glob(os.path.join("data/data/", ext))
    files += image_files

nb_test = math.floor(len(files)*0.15)
nb_val = math.floor(len(files)*0.15)

test_idx = np.random.randint(0, len(files), nb_test)
val_idx = np.random.randint(0, len(files), nb_val)

/content/gdrive/My Drive/darknet

[ ] # Tạo file train.txt
with open("train.txt", "w") as f:
    for idx in np.arange(len(files)):
        if (os.path.exists(files[idx][:-3] + ".txt")):
            f.write(files[idx]+'\\n')

# Tạo file test.txt
with open("test.txt", "w") as f:
    for idx in np.arange(len(files)):
        if (idx in test_idx) and (os.path.exists(files[idx][:-3] + ".txt")):
            f.write(files[idx]+'\\n')

# Tạo file vali.txt
with open("val.txt", "w") as f:
    for idx in np.arange(len(files)):
        if (idx in val_idx) and (os.path.exists(files[idx][:-3] + ".txt")):
            f.write(files[idx]+'\\n')
```

Hình 15: Tạo file train.txt và val.txt

Bước 7: Tạo file yolo.data

Step 7. Tạo file yolo.data

```
[ ] %cd /content/gdrive/My Drive/darknet
!mkdir backup
!echo classes=4 > yolo.data
!echo train=train.txt >> yolo.data
!echo valid=val.txt >> yolo.data
!echo names=yolo.names >> yolo.data
!echo backup=backup >> yolo.data

/content/gdrive/My Drive/darknet
```

Hình 16: Tạo file yolo.data



Bước 8: Make darknet

```
Step 8. Make darknet

%cd /content/gdrive/My Drive/darknet
!rm darknet
!make

/content/gdrive/.shortcut-targets-by-id/1--JewKq2gp-MKIYf9_gDk0f2o3n7nvH0/darknet
```

Hình 17: Make darknet

Bước 9: Tải file pre_trained weight

```
Step 9. Download pretrain weight

%cd /content/gdrive/My Drive/darknet
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137

/content/gdrive/My Drive/darknet
--2022-05-17 15:57:18-- https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/75388965/48bfe500-889d-11ea-819e-c4d182
--2022-05-17 15:57:18-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/75388965/48bfe500-889d-1
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133,
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 170038676 (162M) [application/octet-stream]
Saving to: 'yolov4.conv.137'

yolov4.conv.137 100%[=====] 162.16M 70.7MB/s in 2.3s

2022-05-17 15:57:20 (70.7 MB/s) - 'yolov4.conv.137' saved [170038676/170038676]
```

Hình 18: Download pretrain weight

Bước 10: Thực hiện việc huấn luyện.

```
Step 10. Train

%capture
%cd /content/gdrive/My Drive/darknet
!./darknet detector train yolo.data cfg/yolov4-custom.cfg yolov4.conv.137 -dont_show -map

%capture
%cd /content/gdrive/My Drive/darknet
!./darknet detector train yolo.data cfg/yolov4-custom.cfg backup/yolov4-custom_last.weights -dont_show -map
```

Hình 19: Train



LƯU Ý: Trong trường hợp Google Colab bị disconnect, ta sẽ train tiếp bằng cách chạy lệnh train ở Bước 10 bên trên nhưng trong câu lệnh sẽ thay yolov4.conv.137 bằng file weight mới nhất mà chúng ta có được, chính là yolov4-custom_last.weights

3.3 Tiền xử lý ảnh (Cắt phần hóa đơn)

Sau khi có được model được huấn luyện như đã đề cập ở **Mô hình YOLO Darknet**, model này sẽ trả về được tọa độ của hóa đơn trong ảnh Input. Từ những tọa độ này, nhóm chúng tôi sẽ crop vùng ảnh chứa hóa đơn bằng thư viện OpenCV.



Hình 20: Ảnh input



Hình 21: Ảnh sau khi crop

3.4 Phát hiện văn bản (Paddle):

Sau khi nhận được kết quả output từ bước trước. Bước cần làm tiếp theo đó chính là **phát hiện được vị trí các vùng chữ có trong hóa đơn**. Ở bước này nhóm sử dụng lại module có

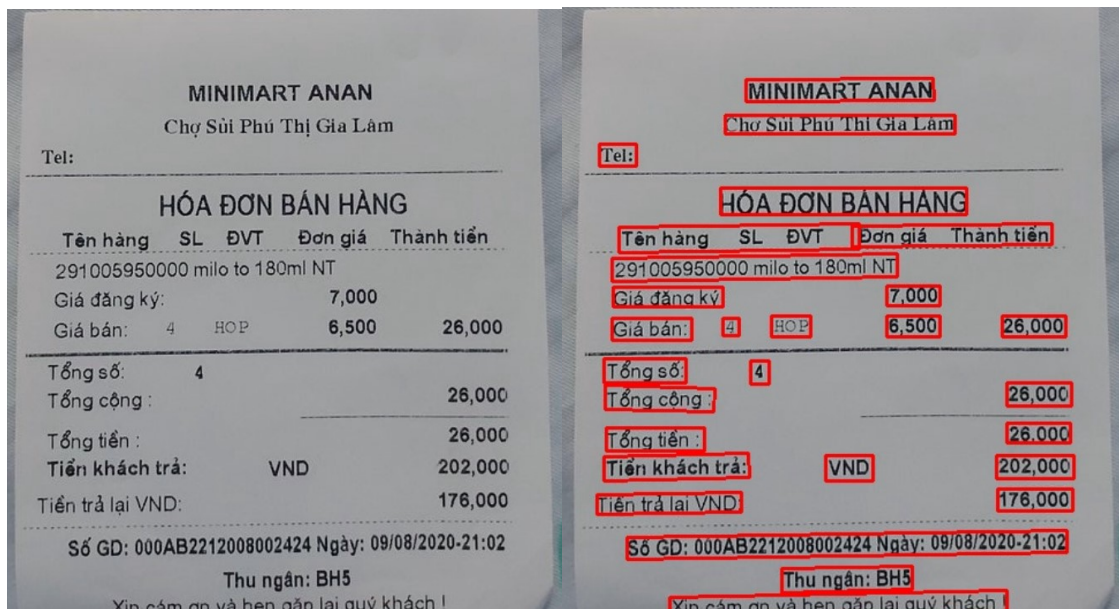


sản text detection của Paddle [PaddleOCR](#). [5]



Hình 22: PaddleOCR

Mô hình sẽ nhận dữ liệu đầu vào là một ảnh chứa hóa đơn đã được crop như hình 23. Và kết quả đầu ra của mô hình được thể hiện như ở hình 24.



Hình 23: Input

Hình 24: Output

PaddleOCR là một mô hình OCR, nó hỗ trợ cả hai task là detect và recognize văn bản có trong hình ảnh. Mô hình này đã có nhiều thành tựu về độ hiệu quả trong bài toán OCR. Những ưu điểm của mô hình này:

- Dung lượng khá nhẹ (3.5-4 Mb): Được xem là mô hình phát triển thiết bị đầu cuối nhẹ nhất.
- Độ linh hoạt: Có thể áp dụng trên nhiều ứng dụng (vé tàu, biển số xe, hóa đơn, ...).



- Độ chính xác: Độ đo F1 sẽ lớn hơn 0.5 ($F1 \text{ score} > 0.5$), gấp 2 lần độ đo những mô hình OCR khác.
- Độ đa dạng: Có thể detect được hơn 27 ngôn ngữ kể cả tiếng Việt.

Tuy nhiên, ở đây vì thời gian thực hiện đồ án ngắn nhóm chỉ tham khảo và sử dụng [pre-train model phần text detection](#) cùng với đó dữ liệu của bài toán là hóa đơn tiếng Việt nên nhóm sử dụng một mô hình khác cho task recognize vùng chữ, được nói kỹ hơn ở mục 4.3.

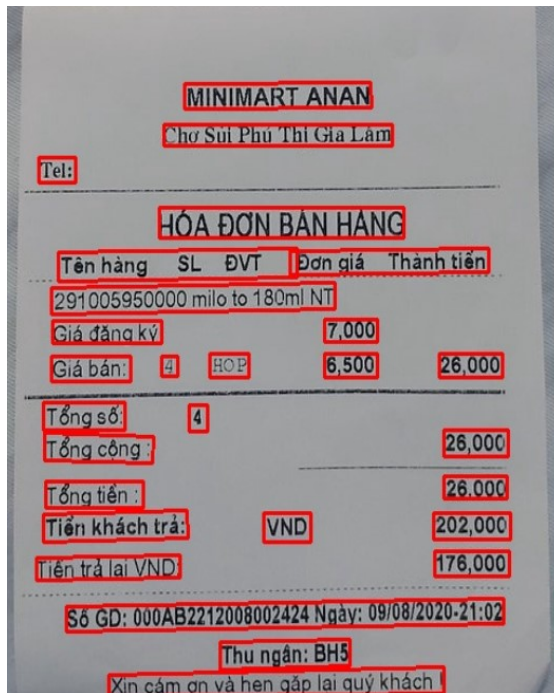
3.5 VietOCR

Sau khi phát hiện được các vùng chứa văn bản, công đoạn tiếp theo sẽ là nhận diện văn bản. Ở đây [VietOCR](#) [6] - một framework nhận dạng văn bản tiếng Việt hoạt động ổn định và hiệu quả. VietOCR là mô hình được tạo ra nhằm giải quyết các bài toán liên quan đến OCR, đặc biệt là nó rất hiệu quả đối với những bài toán nhận diện chữ tiếng Việt và do người Việt làm ra.

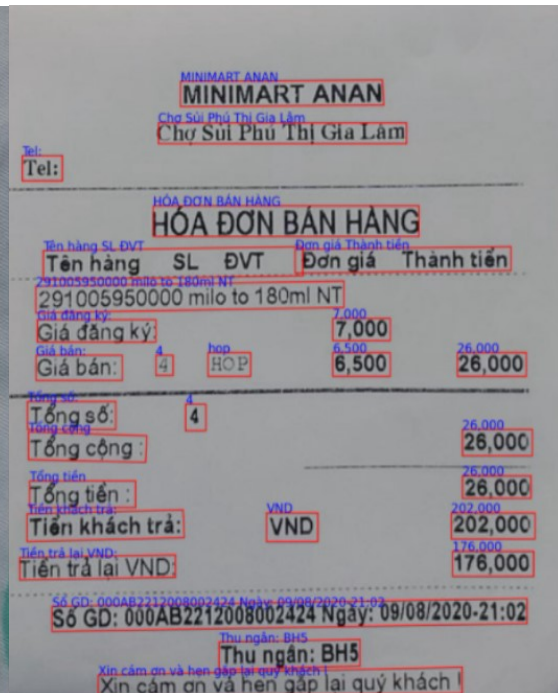
Mô hình VietOCR là sự kết hợp giữa mô hình CNN (VGG-19) và Language Model, sẽ có 2 mô hình OCR để giải quyết:

- Attention-OCR.
- Transformer-OCR.

Mô hình sẽ nhận dữ liệu đầu vào là một ảnh chứa hóa đơn đã được detect vùng chữ như hình 25. Và kết quả đầu ra của mô hình được thể hiện như ở hình 26.



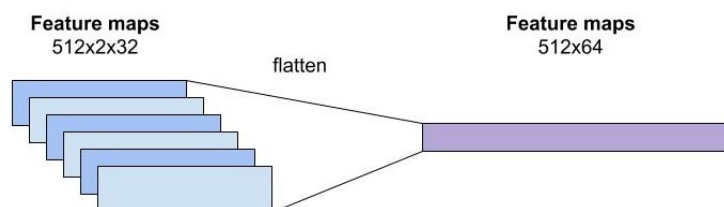
Hình 25: Input



Hình 26: Output

3.5.1 CNN Của Mô Hình OCR:

Mô hình CNN dùng trong bài toán OCR nhận đầu vào là một ảnh, thông thường có kích thước với chiều dài lớn hơn nhiều so với chiều rộng, do đó việc điều chỉnh tham số stride size của tầng pooling là cực kì quan trọng. Thường thì kích thước stride size của các lớp pooling cuối cùng là $w \times h = 2 \times 1$ trong mô hình OCR. Không thay đổi stride size phù hợp với kích thước ảnh thì sẽ dẫn đến kết quả nhận dạng của mô hình sẽ tệ. Đầu ra của VGG sẽ làm đầu vào cho mô hình Language model phía sau.



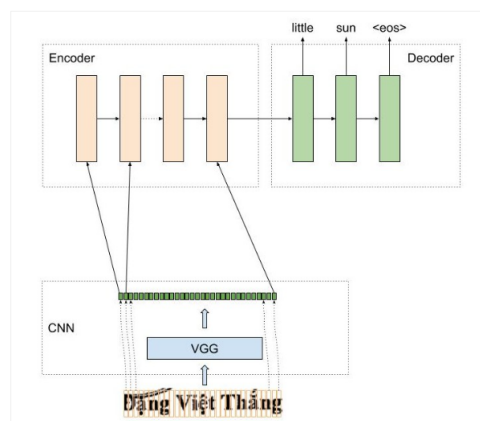
Hình 27: Đặc trưng sau khi được trích xuất từ VGG

3.5.2 Language model: AttentionOCR

AttentionOCR là sự kết hợp giữa mô hình CNN và mô hình Attention Seq2Seq. Cách hoạt động của mô hình này tương tự như kiến trúc của mô hình seq2seq trong bài toán dịch máy. Với bài toán dịch máy từ tiếng việt sang anh, chúng ta cần encode một chuỗi tiếng việt thành một vector đặc trưng, còn trong mô hình AttentionOCR, thì dữ liệu đầu vào này là một ảnh.

Một ảnh qua mô hình CNN, sẽ cho một feature maps có kích thước $channel \times height \times width$, feature maps này sẽ trở thành đầu vào cho mô hình LSTM, tuy nhiên, mô hình LSTM chỉ nhận chỉ nhận đầu vào có kích thước là $hidden \times timestep$. Một cách đơn giản và hợp lý là 2 chiều cuối cùng $height \times width$ của feature maps sẽ được duỗi thẳng. Feature maps lúc này sẽ có kích thước phù hợp với yêu cầu của mô hình LSTM.

Feature maps của mô hình CNN sau khi được flatten thì được truyền vào làm input của mô hình LSTM, tại mỗi thời điểm, mô hình LSTM cần dự đoán từ tiếp theo trong ảnh là gì [6].



Hình 28: VietOCR với language model AttentionOCR

3.5.3 Language model: TransformerOCR

Trước khi đi vào mô hình **TransformerOCR**, chúng tôi thực hiện phân tích một số nhược điểm của các mô hình RNN (LSTM) như sau:

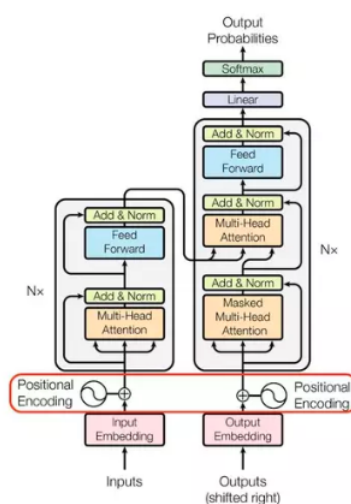
- Thời gian huấn luyện lâu: Khi xử lý một câu văn bằng RNN thì mô hình xử lý câu văn một cách tuần tự theo từng timestep do đó hidden state sau phải phụ thuộc vào hidden state trước thực hiện xong mới tính toán được. Điều này khiến mô hình không thể thực hiện tính toán song song, không tận dụng được khả năng tính toán của GPU khiến thời

gian training lâu hơn nhiều so với cấu trúc như CNN.

- Khả năng ghi nhớ kém : Đây là vấn đề muôn thuở đối với mạng có kiến trúc tuần tự như RNN. Nói đơn giản là mô hình sẽ chỉ học được các từ ở đầu câu, càng về sau những đặc trưng học được càng ít do gradient biến mất (vanishing gradient).

Transformer giải quyết được nhược điểm của mô hình tuần tự truyền thống nhờ chủ yếu vào hai cấu trúc là Multi-head attention - Positional encoding.

Transformer cũng đang là mô hình SOTA hiện tại trong hầu hết các bài toán từ thị giác cho đến xử lý ngôn ngữ. Mô hình này giúp việc trích xuất văn bản từ ảnh một cách hiệu quả. Kiến trúc transformer cũng giống với các mô hình sequence-to-sequence bao gồm hai phần encoder và decoder.



Hình 29: Kiến trúc Transformer

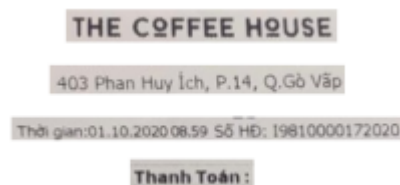
- Phần Encoder, Gồm N block, mỗi block bao gồm hai sub-layer: Multi-Head Attention và Feed forward network. Tác giả dùng một residual connection ở mỗi sub-layer này. Theo sau mỗi sub-layer đó là một lớp Layer Norm có ý nghĩa tương tự như lớp Batch Norm trong CNN. Residual connection cũng góp phần giúp mô hình có thể sâu hơn, deep hơn nhờ giảm tác động của vanishing gradient.
- Phần Decoder cũng tương tự như encoder gồm N block, mỗi block gồm 2 sub-layer. Tuy nhiên, nó có một lớp Masked Multi-Head Attention. Lớp này chính là lớp Multi-Head

Attention. Nó có chức năng chú ý đến toàn bộ những decoder hidden state trước. Lý do mà nó lại được đặt tên như vậy là khi huấn luyện Transformer, ta đưa toàn bộ câu vào cùng một lúc nên nếu ta đưa toàn bộ target sentence cho decoder trước thì mô hình sẽ chẳng học được gì cả. Do đó phải che (mask) bớt một phần token ở decoder hidden state sau trong quá trình decode.

3.5.4 Huấn luyện mô hình VietOCR:

Ở trong đề án lần này, chúng tôi sử dụng pretrained từ repo VietOCR của tác giả Phạm Quốc. Pretrained weight đã được huấn luyện với 10 triệu ảnh, một số lượng rất lớn. Tuy nhiên, theo tác giả thì mô hình **khá nhạy cảm với sự thay đổi nhỏ của ảnh đầu vào** khi sử dụng pretrained model trên tập dữ liệu mới chưa được huấn luyện. Do đó chúng tôi sẽ dùng pretrained này nhưng custom lại với dữ liệu của chúng tôi tạo ra.

Về phần dữ liệu cho VietOCR. Do là để nhận diện tốt trên domain hóa đơn, nên chúng tôi sẽ khai thác từ những ảnh hóa đơn của ban tổ chức. Chúng tôi dùng model có sẵn là Paddle như đã giới thiệu ở mục Phát hiện văn bản (Paddle): để xác định được vị trí vùng chữ trong các ảnh hóa đơn. Sau đó tiến hành cắt ảnh chữ vùng chữ dựa trên thông tin tọa độ như hình dưới đây:



Hình 30: Minh họa data huấn luyện cho VietOCR

Sau khi có các ảnh vùng chữ, chúng tôi tiến hành label khoảng 7000 tấm để phục vụ cho việc huấn luyện mô hình VietOCR. Có khoảng 5000 ảnh huấn luyện và 2000 ảnh kiểm thử.

Như đã đề cập ở mục VietOCR, chúng ta có 2 loại mô hình OCR để giải quyết là Attention-OCR và Transformer-OCR. Chúng tôi huấn luyện cả 2 mô hình này và kết hợp với pretrained của tác giả để ra được những mô hình cuối cùng.

Với model pretrained weight đã được huấn luyện với 10 triệu ảnh từ tác giả. Chúng tôi tiếp tục train trên bộ dữ liệu chúng tôi tự tạo thêm 20000 iters lần lượt với cả Attention-OCR(seq2seq) và Transformer-OCR config. Thông số cụ thể để train mô hình VietOCR như sau:

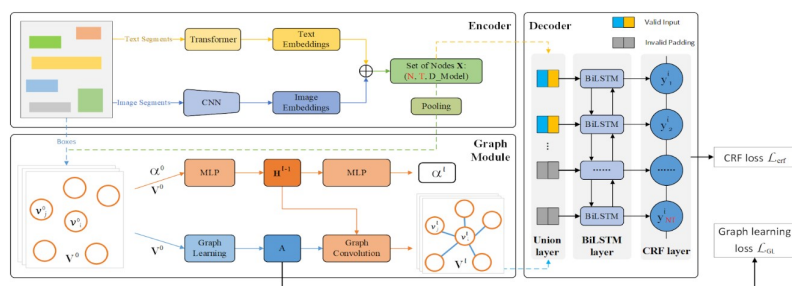
- 'print every':100,
- 'valid every':500,
- 'iters':20000,
- 'batch size': 16

3.6 Model PICK (KIE)

Sau khi nhận dạng được nội dung của các vùng chữ, nhóm sẽ thực hiện việc trích xuất các trường thông tin cần thiết sử dụng **PICK model** của tác giả wenwenyu. [7] [8]

PICK là một framework hiệu quả và mạnh mẽ trong việc xử lý các bố cục tài liệu phức tạp để trích xuất thông tin (Key Information Extraction- KIE) bằng cách kết hợp Graph Learning với Graph Convolution, mang lại một biểu diễn ngữ nghĩa phong phú hơn chứa các tính năng văn bản, hình ảnh và bố cục chung một cách rõ ràng. Bên cạnh đó, việc PICK sử dụng rất nhiều features của văn bản bao gồm text, image và position features làm tăng khả năng biểu diễn của chúng. Việc biểu diễn các features từ rất nhiều thông tin làm cho PICK đạt hiệu năng tốt hơn các mô hình còn lại.

Hướng tiếp cận đơn giản nhất đó là sử dụng Text Classification để phân loại ra những thông tin nào thuộc lớp nào, cách giải quyết này có thể đơn giản và tốt trên những dạng văn bản có sự dạng thấp và phân biệt rõ rệt giữa các trường thông tin, và đặc biệt là cấu trúc văn bản đó đơn giản.



Hình 31: Trích xuất thông tin với model PICK

Giải thích mô hình

- **Encoder**: Sử dụng mô hình Transformer để trích xuất đặc trưng từ văn bản và sử dụng một mạng CNN để trích xuất đặc trưng từ ảnh. Sau đó kết hợp text embeddings và image



embeddings lại thành vector biểu diễn X thể hiện khả năng biểu diễn text và image chứa text đó. X được đưa xem là đầu vào của Graph module.

Set of Node X : X là ma trận sau khi kết hợp 2 ma trận thu được từ Transformer vs CNN và được biểu diễn dưới dạng các Node của Graph và X được dùng làm input của Graph Module \Rightarrow Có thể nói toàn bộ quá trình nói trên đều thuộc hàm Encoder và các set Node chính là các bounding box thu được sau OCR.

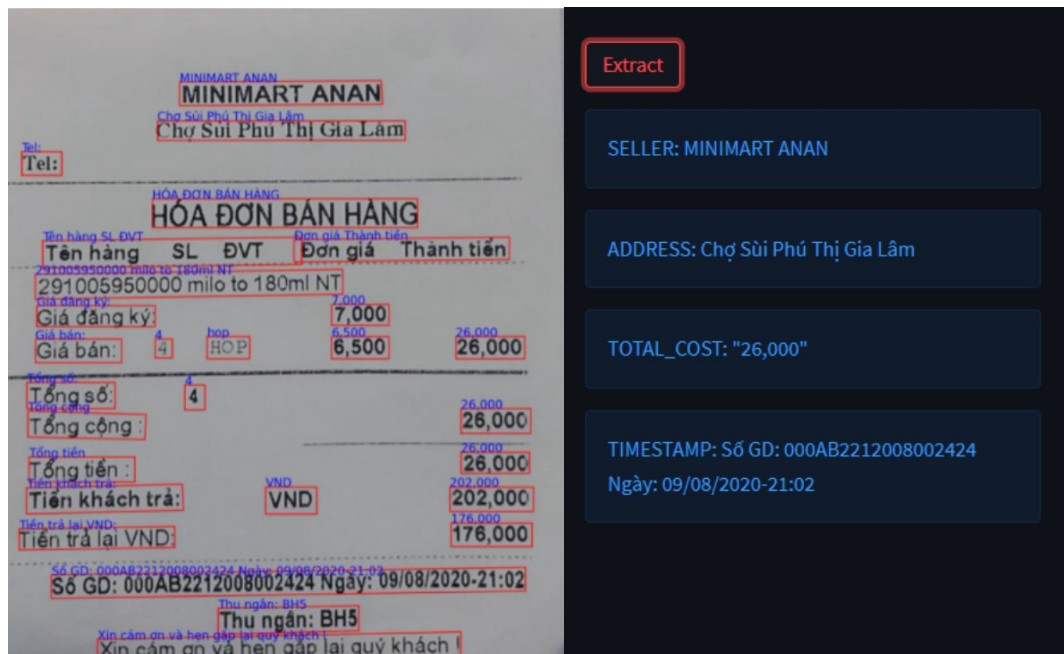
- **Graph module:** Sử dụng một mạng GCN (Graph Convolutional Network) để làm giàu khả năng biểu diễn giữa các node với nhau. Việc các thông tin cần trích xuất có vị trí và nội dung khác nhau, nó không cục bộ và không theo thứ tự nên việc sử dụng Graph giúp mô hình có thể học được khả năng biểu diễn mối tương quan giữa chúng về khoảng cách và vị trí trong văn bản.

Graph Modeling: cách thức xây dựng graph dựa trên các bounding box text đã được OCR.

- **Decoder:** Sau khi kết hợp mô-đun Graph để làm giàu thông tin thì mô hình kết hợp thông tin đó và cả thông tin do mô-đun Encoder sinh ra để đưa vào mô-đun Decoder để nhận dạng và phân loại chúng. Ở mô-đun này mô hình PICK sử dụng BiLSTM layer + CRF layer.

Nhận đầu vào là sự kết hợp giữa đầu ra của Encoder và Graph Module. Cho qua một mạng BiLSTM layer và CRF layer để phân loại chúng. Cuối cùng mô hình sử dụng 2 hàm loss để tối ưu đồng thời chúng, đó là loss của Graph learning và CRF loss.

Mô hình sẽ nhận dữ liệu đầu vào là một ảnh chứa hóa đơn đã được recognize vùng chữ như hình 32. Và kết quả đầu ra của mô hình được thể hiện như ở hình 33.



Hình 32: Input

Hình 33: Output

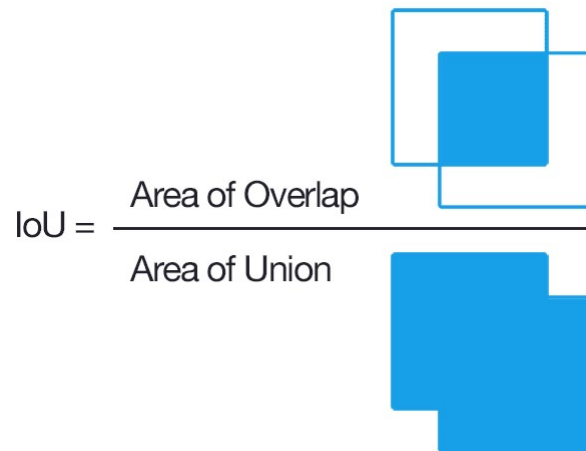
4 Kết quả huấn luyện và đánh giá mô hình:

4.1 Các độ đo được sử dụng:

4.1.1 Cơ chế IOU

Đầu tiên, để xác định được mô hình của chúng ta liệu đã dự đoán chính xác hay chưa, ta cần phải có phương pháp nhằm đối chiếu giữa kết quả thực và kết quả đã có trong quá trình train. Trong bài toán object detection, thì đầu ra của bài toán sẽ là các giá trị khoảng vùng (bounding box). Do đó ta sẽ sử dụng IOU để đo độ chính xác của quá trình phát hiện đối tượng trên tập dữ liệu cụ thể của chúng ta.

IOU (Intersection over union) [9] [10] là tỉ lệ giữa đo lường mức độ giao nhau giữa hai đường bao (thường là đường bao dự đoán và đường bao thực) để nhằm xác định hai khung hình có bị đè chồng lên nhau không. Tỷ lệ này được tính dựa trên phần diện tích giao nhau giữa 2 đường bao với phần tổng diện tích giao nhau và không giao nhau giữa chúng. Được tính cụ thể như sau:



Hình 34: Intersection over Union

4.1.2 Non max suppression

Huấn luyện mô hình và sau đó detect hóa đơn, khi detect nhóm thấy có trường hợp trong một ảnh hóa đơn có nhiều hơn 1 bounding box. Để giải quyết vấn đề này, nhóm áp dụng thuật toán Non max suppression [9] [10], với thuật toán này IoU của các bounding box sẽ được xét với một ngưỡng nhất định (phù hợp với từng bài toán), lớn hơn ngưỡng sẽ bị lược bỏ đi. Thresh_iou trong bài toán nhóm sử dụng là $\text{threshold} = 0.45$.



Hình 35: Hình sau khi đã thực hiện Non max suppression

4.1.3 Các metric khác

Mặc dù độ chính xác - IOU có công thức tường minh và dễ diễn giải ý nghĩa. Tuy nhiên, hạn chế của nó là đo lường trên tất cả các nhãn mà không quan tâm đến độ chính xác trên từng nhãn. Do đó, nó không phù hợp với ngữ cảnh mà bài toán đang sử dụng. Cụ thể hơn, để mô hình được coi là hiệu quả thì bounding càng chính xác càng tốt. Việc sử dụng IOU sẽ làm cho việc dự đoán các đối tượng này là như nhau, làm cho mô hình không được đánh giá cao trong việc xác định khung đối tượng. Vì vậy ta sẽ cần những metrics như precision, recall đánh giá chuyên biệt trên nhóm này. Bởi vì ta có thể thiết lập ngưỡng nhằm xác định và thiên vị cho các đối tượng. Để làm điều này, ta tận dụng chỉ số IOU để xác định như sau: [9] [11] [12]

- Đối tượng được nhận dạng đúng với tỉ lệ $\text{IOU} \geq 0.5$ (True positive : TP).
- Đối tượng được nhận dạng sai với tỉ lệ $\text{IOU} < 0.5$ (False positive : FP)
- Đối tượng không được nhận dạng (False negative: FN)

Từ đó, ta có được công thức:



$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

TP = True positive

TN = True negative

FP = False positive

FN = False negative

Hình 36: Công thức tính Precision và Recall

Precision trả lời cho câu hỏi trong các trường hợp được dự báo là positive thì có bao nhiêu trường hợp là đúng. Và tất nhiên precision càng cao thì mô hình của chúng ta càng tốt trong việc xác định chính xác khung đối tượng.

Recall đo lường tỷ lệ dự báo chính xác các trường hợp positive trên toàn bộ các mẫu thuộc nhóm positive.

Tuy nhiên, ta không thể sử dụng hai giá trị Precision và Recall để đánh giá mô hình một cách độc lập được bởi sẽ phát sinh rất nhiều vấn đề. Ta giả sử trường hợp sau để thấy rõ vấn đề, giả sử tập dữ liệu của chúng ta có 1000 đối tượng, trong đó mô hình dự đoán chính xác 100 đối tượng và bỏ qua 900 đối tượng còn lại.

- Xét giá trị Precision, mô hình của chúng ta, chỉ dự đoán những trường hợp đúng và bỏ qua những trường hợp khác. Lúc này, FP của chúng ta sẽ bằng 0, $\text{Precision} = TP / TP$ do đó mô hình sẽ được xem là dự đoán chính xác 100%, mặc dù đã bỏ qua rất nhiều đối tượng khác cần được dự đoán.
- Tiếp theo ta xét giá trị Recall, mặc dù Precision cho thấy rằng mô hình dự đoán chính xác 100%. Lúc này có 900 đối tượng đã bị bỏ qua, $FN = 900$. $\text{Recall} = 100 / (100 + 900) = 10\%$. 90% đối tượng bị dự đoán sai. Mô hình của chúng ta hoàn toàn rất tệ trong trường hợp này.

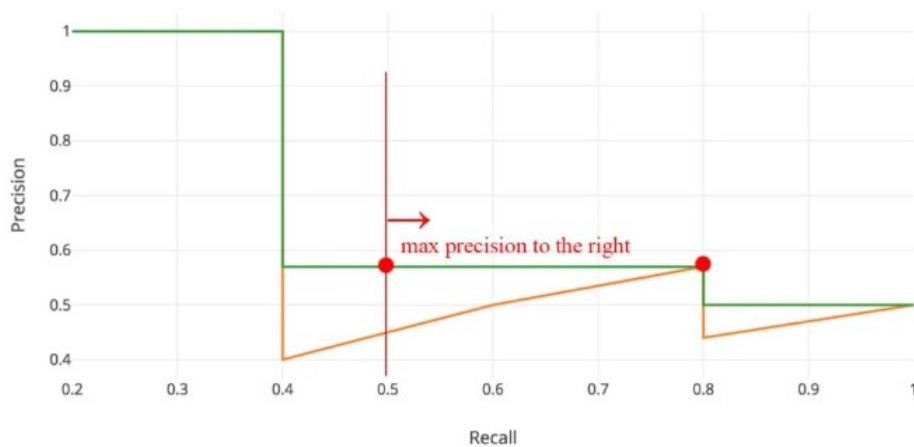
Ngoài ra, Precision và Recall sẽ không cố định mà chịu sự biến đổi theo ngưỡng xác suất được lựa chọn.

Sự đánh đổi giữa Precision và Recall khiến cho kết quả của mô hình thay đổi : precision cao, recall thấp hoặc precision thấp, recall cao. Khi đó rất khó để lựa chọn đâu là một mô hình tốt vì không biết rằng đánh giá trên precision hay recall sẽ phù hợp hơn. Chính vì vậy việc kết hợp

cả Precision và Recall là AP sẽ giúp mô hình được đánh giá một cách trực quan hơn.

4.1.4 Công thức tính AP

Giá trị AP [9] [11] [12] là giá trị phía dưới đường biểu diễn mối quan hệ precision – recall. Để đơn giản việc tính đường zig-zag này, ta xấp xỉ chúng như sau: Tại mỗi recall level, ta thay giá trị precision bằng giá trị precision lớn nhất tại recall level đó.



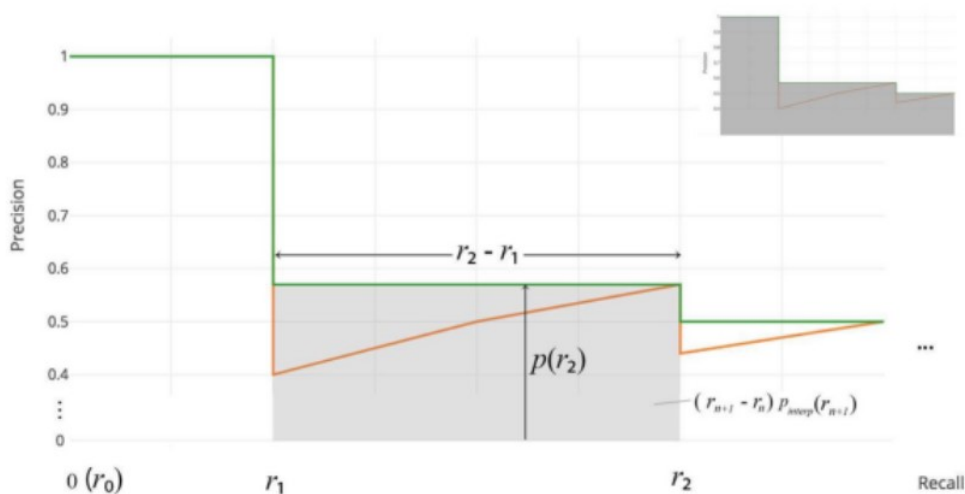
Hình 37: Ví dụ về xấp xỉ Precision và Recall

- Chia đường biểu diễn precision – recall (sau khi đã xấp xỉ) thành 11 đoạn bằng nhau theo recall (0, 0.1, 0.2, . . . , 1). Khi đó, AP được tính bằng công thức:

$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0.1) + \dots + AP_r(1.0))$$

Hình 38: Công thức tính Average Percision

Ngoài ra, thay bằng tính bằng interpolated AP(cách trên), ta có thể tính luôn phần diện tích phía dưới đường biểu diễn precision-recall bằng cách tính tổng các hình chữ nhật xấp xỉ.



$$AP = \sum (r_{n+1} - r_n) p_{interp}(r_{n+1})$$

$$p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r})$$

Hình 39: AUC Area Under the Curve

4.1.5 Accuracy Full Sequence & Accuracy Per Character

Tính toán chỉ số AFS & APC [15] dựa vào chỉ số dự đoán đúng đối với nhãn-full_sequence/ký tự-per_char trong từng nhãn single_label_accuracy (SLA).

- Đối với Accuracy Full Sequence: với mỗi label được dự đoán đúng SLA sẽ được trả về 1, ngược lại sẽ trả về 0.
- Đối với Accuracy Per Character: với từng nhãn, SLA được tính theo công thức sau:

$$SLA = \frac{C}{n}$$

Hình 40: SLA đối với trường hợp ký tự.

Trong đó:



- SLA: độ chính xác trên từng nhãn.
- C: Số ký tự dự đoán đúng trong một nhãn.
- n: Số ký tự có trong nhãn tương ứng trên.

Sau khi có được giá trị SLA của từng nhãn thì độ chính xác của cả AFS & APC sẽ được tính bởi công thức sau:

$$ALA = \sum_{i=0}^N \frac{SLA_i}{N}$$

Hình 41: Average Label Accuracy.

Trong đó:

- ALA: (Average Label Accuracy) độ chính xác trung bình của tất cả SLA.
- SLA: độ chính xác trên từng nhãn.
- N: Tất cả các nhãn.

4.1.6 Character Error Rate

Tính toán chỉ số CER [13] [14] dựa trên kỹ thuật khoảng cách Levenshtein bằng cách đếm số lượng tối thiểu các hoạt động cấp ký tự cần thiết để chuyển đổi văn bản tham chiếu đầu vào thành tệp đầu ra OCR.

CER được tính theo công thức sau:

$$CER = \frac{S + D + I}{N}$$

Hình 42: Character Error Rate

Trong đó:



- S = Số lần thay thế
- D = Số lần xóa
- I = Số lần chèn
- N = Tổng số ký tự trong văn bản tham chiếu. Mẫu số N có thể được tính theo công thức:
$$N = S + D + C \text{ (trong đó } C = \text{Số ký tự đúng)}$$

Kết quả thu được đại diện cho phần trăm ký tự trong văn bản tham chiếu được dự đoán sai. Giá trị CER càng thấp (với 0 là đúng tuyệt đối), hiệu suất của model càng tốt.

4.2 Kết quả và đánh giá mô hình:

1. Tham số của quá trình train.

```
1115: 4.046449, 4.850825 avg loss, 0.001000 rate, 4.765887 seconds, 71360 images, 22.588736 hours left
loaded: 6.262369 seconds - performance bottleneck on CPU or Disk HDD/SSD
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.533896), count: 17, class_loss = 5.813572, iou_loss = 31.332184, total_loss = 37.145756
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.565322), count: 10, class_loss = 2.941202, iou_loss = 4.000922, total_loss = 7.022124
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.000000), count: 1, class_loss = 0.002267, iou_loss = 0.000000, total_loss = 0.002267
total_bbox = 764874, rewritten_bbox = 0.007844 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.517810), count: 23, class_loss = 7.378582, iou_loss = 26.042336, total_loss = 33.420918
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.530386), count: 22, class_loss = 7.394115, iou_loss = 6.414536, total_loss = 13.808651
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.564851), count: 2, class_loss = 0.871452, iou_loss = 0.253065, total_loss = 1.124517
total_bbox = 764921, rewritten_bbox = 0.007844 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.403671), count: 17, class_loss = 4.773602, iou_loss = 16.698882, total_loss = 21.471684
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.541473), count: 7, class_loss = 2.469327, iou_loss = 2.009311, total_loss = 4.478638
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.000000), count: 1, class_loss = 0.002726, iou_loss = 0.000000, total_loss = 0.002726
total_bbox = 764945, rewritten_bbox = 0.007844 %
```

Hình 43: Thông số quá trình train

Trong đó:

- 115: Số vòng train đến hiện tại.
- 4.046449, 4.850825 avg loss: cái này nghĩa là loss của vòng hiện tại là 4.046449 còn loss trung bình cho đến vòng hiện tại là 4.850825.

2. Đánh giá kết quả mô hình YOLO:

Module	AP@0.5-0.9
YOLOv4	0.779

AP@0.5-0.9 là Average Precision của các AP khi có threshold từ 0.5 đến 0.95 (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95). Có thể thấy với 10 threshold, mAP của mô hình YOLO đạt gần 78%, tương đối cao. Điều đó chứng tỏ cứ 100 hình thì mô hình YOLO sẽ có thể nhận diện đúng 78 hình.

3. Đánh giá kết quả mô hình VietOCR:



Module	acc full seq	acc per char	gpu time
vgg_transformer	0.7666	0.9228	20.90 @ RTX 2080 Ti
vgg_seq2seq	0.7646	0.9201	9.56 @ RTX 2080 Ti

4. Đánh giá toàn bộ pipeline:

Module	Character Error Rate				
	SELLER	ADDRESS	TIMESTAMP	TOTAL_COST	average CER
vgg_transformer	0.41	0.674	0.724	0.734	0.635
vgg_seq2seq	0.425	0.688	0.739	0.82	0.668

Có thể thấy trường **SELLER** là trường thông tin có Loss nhỏ nhất trong các trường. Điều này cũng dễ hiểu vì trường SELLER thường nằm ở vị trí cố định và ít có biến thể khác. Còn những trường khác thì CER loss vẫn còn đang cao vì những trường này thường nằm ở nhiều vị trí trong các hóa đơn khác nhau và có nhiều biến thể. Ví dụ như trường ADDRESS có label là "36, phía bắc cầu thị Nai, Nhơn Hội, TP Quy Nhơn" thì kết quả trả về sẽ có thể là "36 phía bắc cầu thị Nai" hoặc "Nhơn Hội, TP Quy Nhơn" ... Cho nên khá khó đánh giá được chính xác cho những trường hợp này. Về mặt kết quả, nhận thấy rằng average CER của chúng tôi vẫn còn khá khiêm tốn với 2 con số **0.635**, **0.668**. Trong khi các đội top cuộc thi hiện có kết quả là **0.21** [1]. Do đó chúng tôi sẽ phải cần phải điều chỉnh, tham khảo lại mô hình phương pháp lần dữ liệu để có thể đạt được kết quả tốt hơn trong tương lai.

5 Hướng phát triển

- Thực hiện khảo sát phương pháp của các top team tham gia cuộc thi để nâng cao độ chính xác của các mô hình.
- Thực hiện tiền xử lý với các trường hợp khó hơn như (Hóa đơn bị nghiêng nặng vd: 180,-90 độ ..., hóa đơn bị nhàu , mờ chữ).
- Gán nhãn và thu thập thêm dữ liệu để train các model, đặc biệt là VietOCR để cho ra kết quả tổng thể tốt hơn.



- Nhận thấy sản phẩm của chúng tôi chỉ mới dưới dạng demo đồ án. Trong tương lai chúng tôi muốn nâng cao tốc độ xử lý và làm sản phẩm hoàn thiện (web hoặc app mobile).



Tài liệu

- [1] Mobile-Captured Image Document Recognition for Vietnamese Receipts (MC-OCR) - Legacy [Link](#)
- [2] [YOLO Series] Train YOLO v4 train trên COLAB chi tiết và đầy đủ (A-Z) [Link](#)
- [3] Yolo v4, v3 and v2 for Windows and Linux [Link](#)
- [4] YOLOv4 Object Detection Algorithm with Efficient Channel Attention Mechanism [Link](#)
- [5] PaddleOCR by PaddlePaddle [Link](#)
- [6] VietOCR by pbcquoc [Link](#)
- [7] PICK: Processing Key Information Extraction from Documents using Improved Graph Learning-Convolutional Networks [Link](#)
- [8] PICK-PyTorch by wenwenyu [Link](#)
- [9] Series YOLOv4: 3 Đánh giá model bằng mAP -Object detection [Link](#)
- [10] Intersection over Union (IoU) for object detection [Link](#)
- [11] mAP (mean Average Precision) [Link](#)
- [12] Object Detection Metrics With Worked Example [Link](#)
- [13] CHAR ERROR RATE [Link](#)
- [14] Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER) [Link](#)
- [15] Accuracy Full Sequence (AFS) & Accuracy Per Character (APC) [Link](#)