Lợi ích của sử dụng Colab (là một dịch vụ cung cấp môi trường Jupyter Notebook hoàn toàn trực tuyến. Nó cho phép người dùng tạo, chia sẻ và chỉnh sửa các tệp notebook một cách dễ dàng mà không cần cài đặt bất kỳ phần mềm nào) :

* Sử dụng Jupyter Notebooks trực tuyến - Google Colab cho phép tạo và chạy các Jupyter Notebooks trực tuyến mà không cần cài đặt môi trường phát triển phức tạp trên máy tính cá nhân.
* Khả năng chia sẻ và cộng tác - Người dùng có thể chia sẻ notebook với những người khác để cùng làm việc trên cùng một notebook, tạo điều kiện thuận lợi cho việc học tập và làm việc nhóm.
* **Dùng GPU và TPU miễn phí - Google Colab cung cấp truy cập miễn phí đến GPU và TPU, đặc biệt hữu ích cho các tác vụ tính toán nặng về mặt số học, đặc biệt là trong lĩnh vực học máy và deep learning. Việc sử dụng các card GPU hoặc TPU có sẵn giúp tăng tốc độ xử lý và huấn luyện mô hình so với việc sử dụng CPU thông thường.**
* Lưu trữ dữ liệu trên Google Drive và tích hợp Google Cloud - Người dùng có thể truy cập và lưu trữ dữ liệu trực tiếp từ Google Drive, tạo điều kiện thuận lợi cho việc làm việc với tập tin dữ liệu lớn.

**3. Tạo các khung hình chữ nhật bao lấy những hàng có xuất hiện chữ cái – ký tự.**

Sử dụng PyTorch cho trình phát hiện văn bản CRAFT giúp phát hiện vùng văn bản một cách hiệu quả bằng cách phát hiện từng vùng ký tự và mối quan hệ giữa các ký tự.

**Thư viện: “craft-text-detector”.**

**Cài thư viện trên colab: “!pip install craft-text-detector”**

**Sử dụng:**

from craft\_text\_detector import Craft

#image\_path = r"CCCD23\_output"

# set image path and export folder directory

#image = image\_path + ".jpg" # can be filepath, PIL image or numpy array

output\_dir = 'outputs/'

# create a craft instance

craft = Craft(output\_dir=output\_dir, crop\_type="poly", cuda=False)

# apply craft text detection and export detected regions to output directory

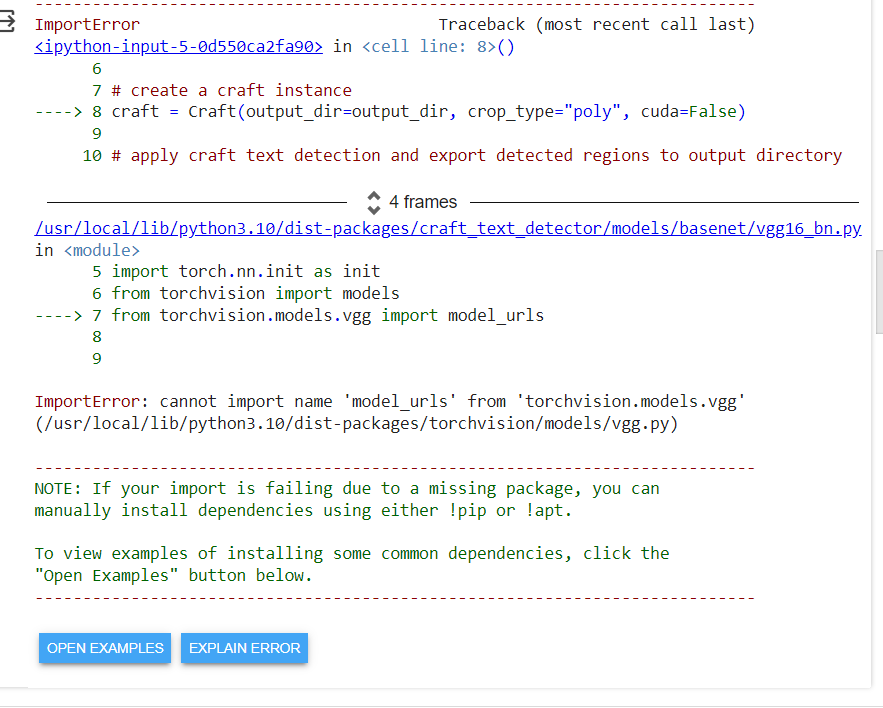
prediction\_result = craft.detect\_text(img)

# unload models from ram/gpu

craft.unload\_craftnet\_model()

craft.unload\_refinenet\_model()

**Nếu chạy mà bị lỗi này**



**Nhấp vào** 

**Nó sẽ hiện ra file thư viện này ở phía bên phải**



**Xóa dòng**

from torchvision.models.vgg import model\_urls

**Chèn ở chỗ vừa xóa:**

all = [

'VGG', 'vgg11', 'vgg11\_bn', 'vgg13', 'vgg13\_bn', 'vgg16', 'vgg16\_bn',

'vgg19\_bn', 'vgg19',

]

model\_urls = {

'vgg11': 'https://download.pytorch.org/models/vgg11-bbd30ac9.pth',

'vgg13': 'https://download.pytorch.org/models/vgg13-c768596a.pth',

'vgg16': 'https://download.pytorch.org/models/vgg16-397923af.pth',

'vgg19': 'https://download.pytorch.org/models/vgg19-dcbb9e9d.pth',

'vgg11\_bn': 'https://download.pytorch.org/models/vgg11\_bn-6002323d.pth',

'vgg13\_bn': 'https://download.pytorch.org/models/vgg13\_bn-abd245e5.pth',

'vgg16\_bn': 'https://download.pytorch.org/models/vgg16\_bn-6c64b313.pth',

'vgg19\_bn': 'https://download.pytorch.org/models/vgg19\_bn-c79401a0.pth',

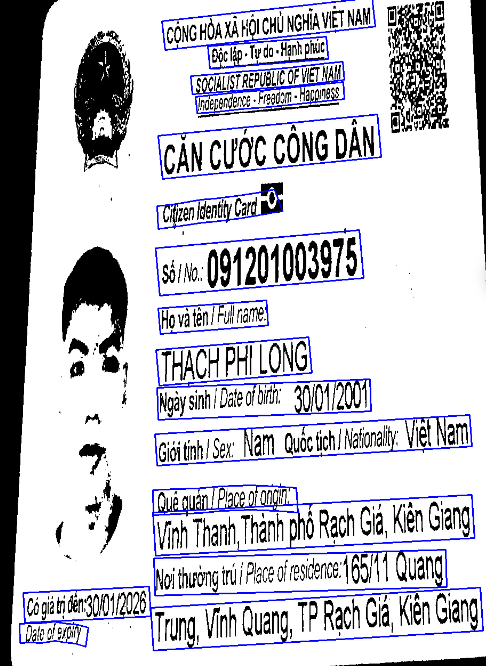
}

**Sau đó chạy lại.**

**Ảnh đầu vào**



**Ảnh đầu ra nằm trong folder “outputs”**



**4. Sử dụng thư viện vietocr để nhận dạng chữ tiếng việt**

VietOCR Là giải pháp nhận dạng ký tự quang học (OCR) tập trung vào cải thiện độ chính xác cho việc nhận dạng ký tự Tiếng Việt.

Phát triển dựa trên lõi OCR Tesseract (Tesseract là công cụ OCR mã nguồn mở được tài trợ bởi Google, được xem là một trong nhưng công cụ OCR tốt nhất hiện nay).

**Thư viện: “vietocr”.**

**Cài thư viện trên colab: “!pip install --quiet vietocr”**

**Sử dụng:**

**Sử dụng model có sẵn đã được training:**

import matplotlib.pyplot as plt

from PIL import Image

from vietocr.tool.predictor import Predictor

from vietocr.tool.config import Cfg

config = Cfg.load\_config\_from\_name('vgg\_transformer')

# config['weights'] = './weights/transformerocr.pth'

config['cnn']['pretrained']=False

config['device'] = 'cuda:0'

detector = Predictor(config)

**Truy cập vào folder chứa các khung hình chữ nhật được lấy ra ở phần 1.**

import os

count\_data = 0

final\_data = []

dir\_path = f'./outputs/image\_crops'

count = 0

for path in os.listdir(dir\_path):

    # check if current path is a file

    if os.path.isfile(os.path.join(dir\_path, path)):

        count += 1

**Phát hiện chữ trên mỗi khung hình chữ nhật:**

for i in range(count):

  img = f'outputs/image\_crops/crop\_{i}.png'

  img = Image.open(img)

  s = detector.predict(img)

  print(s)

**Dữ liệu thu thập được:**

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

SOCIALISTREPUBLIC OF VIET NAM

Independence - Freedom - Happiness

CÀN CƯỚC CÔNG DÂN

Citizen Identity Carding

091201003975

Họ và tên/Full name:

THACH PHI LONG

Ngày sinh/Date of binh: 30/01/2001

Giới tính / Sex Nam Quốc tịch/Nationality: Việt Nam

Quê quán/Place of ongin:

Vĩnh Thanh, Thành phố Rạch Giá, Kiên Giang

Noi thường trú / Place of residence: 165/11 Quang

Có giá trị đến:30/01/2026

Trung, Vĩnh Quang, TP Rạch Giá, Kiên Giang

Dale of expity

Date,ofexpiry

**Để thu thập được dữ liệu cần thiết, ta cần sử dụng một số hàm để xử lý dữ liệu thô**

def test\_case(string, list):

  for i in list:

    list1 = []

    list1 = string.split(i)

    if(list1[0] != string):

      return list1

  return None

def format\_data(count, string):

  if string[-12:].isnumeric() and count == 0:

    return string[-12:]

  elif string.isupper() and count == 1:

    return string

  elif string[-4:].isnumeric() and count == 2:

    return string[-10:]

  elif count == 3 and "sex" in string.lower():

    if string.find('sex') == -1:

      index = string.find('Sex')

    else:

      index = string.find('sex')

    gender = string[index + 4 : index + 8]

    country = string[-8:]

    return [gender, country]

  elif count == 4:

    if "lace" in string or "of" in string or "ori" in string:

      return None

    else:

      return string

  elif count == 5 and len(string) > 30:

      if(test\_case(string, [":", "dence", "ce"]) != None):

        list1 = test\_case(string, [":", "dence", "ce"])

        return list1[-1]

  elif count == 6 and len(string) > 13:

    if string[-4:].isnumeric() or "of" in string or "ex" in string:

      return None

    else:

      return string

  return None

**Sau đó ta sẽ chạy lệnh dưới đây để format dữ liệu thành dạng mảng chứa các dữ liệu cần thiết**

try:

  for i in range(count):

    img = f'outputs/image\_crops/crop\_{i}.png'

    img = Image.open(img)

    s = detector.predict(img)

    if (format\_data(count\_data,s) != None):

      final\_data.append(format\_data(count\_data,s))

      count\_data += 1

  final\_data[5 : 7] = [' '.join(final\_data[5 : 7])]

  temp = final\_data[3]

  final\_data.pop(3)

  final\_data.insert(3,temp[0])

  final\_data.insert(4,temp[1])

  print(final\_data)

except:

  print("Please try with the Image that has more Sensitivite, more Brightness, Please no shadow in the Image")

**Kết quả:**

['091201003975', 'THACH PHI LONG', '30/01/2001', 'Nam ', 'Việt Nam', 'Vĩnh Thanh, Thành phố Rạch Giá, Kiên Giang', ' 165/11 Quang Trung, Vĩnh Quang, TP Rạch Giá, Kiên Giang']

**Cuối cùng ta sẽ lưu vào database:**

**Truy xuất vào database(Excel) và lưu trữ dữ liệu vừa thu thập được ở bước trên:**

from openpyxl import  load\_workbook

path = "CCCD\_data.xlsx"

def query\_data\_excel(list, path):

    wb = load\_workbook(path)

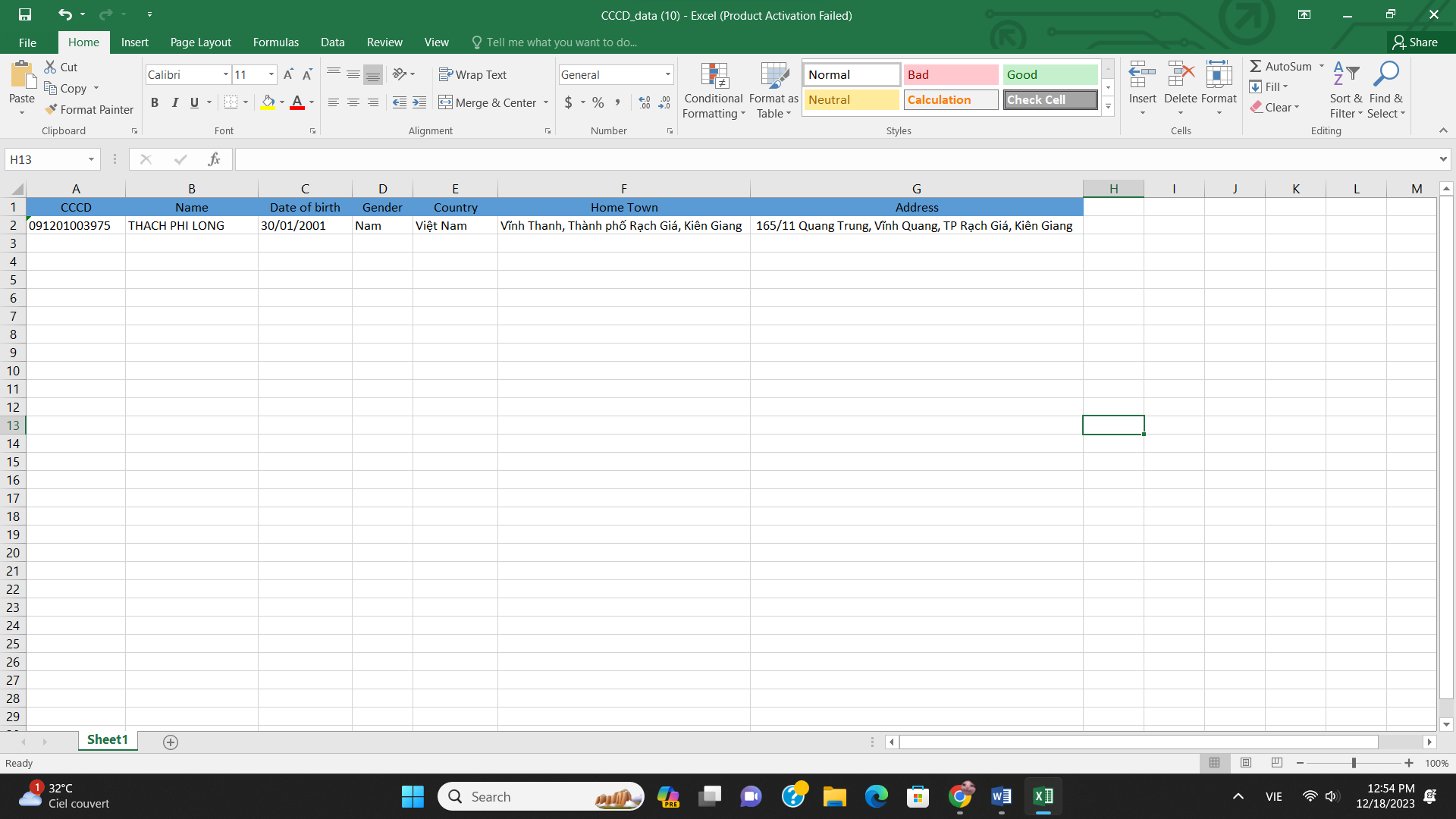
    ws = wb.active

    #Lay du lieu trong database(excel)

    ws.append(list)

    wb.save(path)

query\_data\_excel(final\_data, path)

**Kết quả:** 

**Lưu ý: Muốn sử dụng một ảnh khác để test hãy thay thế đường dẫn file và thêm file vào folder “train”**

