Xây dựng mô hình mạng neural tích chập tự động giải captcha.

2. Nội dung

2.1. Tổng quan

2.1.1. Captcha

Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) là một dạng của phép thử Turing, trong đó người được kiểm tra (con người/máy tính) được cung cấp một hoặc một số hình ảnh của một tập các ký tự đã qua xử lý kỹ thuật số (làm biến dạng, đổi phông, thay đổi bố cục,….) nhằm tạo ra hình ảnh bộ ký tự mới mà máy tính không thể nhận diện, nhưng một người dùng với nhận thức và giác quan bình thường có thể nhận diện.

Phép thử Turing là một bài kiểm tra được đề xuất bởi Alan Turing vào năm 1950, với mục đích đánh giá liệu một cỗ máy có khả năng suy nghĩ giống với một con người hay không.

2.1.2. Captcha và sự phát triển của trí tuệ nhân tạo.

Mục đích của ngành khoa học nghiên cứu trí tuệ nhân tạo là tạo ra những máy tính có khả năng suy nghĩ giống con người. Phương pháp phổ biến để đánh giá một cỗ máy có khả năng suy nghĩ giống con người hay không là yêu cầu máy tính làm những bài kiểm tra liên quan đến khả năng ra quyết định, một máy tính được coi là có khả năng suy nghĩ giống con người nếu những quyết định mà nó đưa ra trùng khớp với những quyết định mà một con người bình thường đưa ra khi đối mặt với bài kiểm tra tương tự.

Nhận diện captcha là một trong những bài kiểm tra phân biệt người/máy được sử dụng phổ biến nhất trong thực tế. Tính phổ biến của captcha cho thấy bài kiểm tra này là đáng tin cậy, có khả năng phân biệt giữa người và máy với độ chính xác cao. Vậy nếu một trí tuệ nhân tạo có khả năng vượt qua một bài kiểm tra nhận diện captcha phức tạp, có thể khẳng định trong một giới hạn nào đó rằng trí tuệ nhân tạo này đã phần nào đạt được khả năng suy nghĩ giống với một con người (về phương diện nhận diện ký tự).

Mục đích của mini-project này là xây dựng một trí tuệ nhân tạo sử dụng công nghệ mạng neural tích chập có khả năng giải mã một đoạn captcha 4 ký tự, trong đó các ký tự là các chữ số và các chữ cái thuộc bảng chữ cái latin, không xét đến các ký tự 0-O,1-I bởi sự tương đồng của các ký tự này dễ gây nhầm ngay cả đối với con người.

2.2. Cơ sở lý thuyết

2.2.1. Mạng neural tích chập

Mạng neural tích chập (CNN) là một loại mạng nerural chuyên biệt để xử lý dữ liệu có cấu hình lưới(dữ liệu chuỗi thời gian, dữ liệu hình ảnh,…). Mạng neural tích chập sử dụng một phép toán tuyến tính đặc biệt gọi là phép tích chập.

Một các tổng quát, phép tính tích chập là hợp của 2 hàm số với tham số thực



Trong đó, hàm thứ nhất, hàm x đóng vai trò dữ liệu đầu vào (input), còn hàm thứ hai w là nhân (kernel).

Trong các ứng dụng học máy, đầu vào thường là các mảng nhiều chiều chứa dữ liệu, nhân là các mảng nhiều chiều chứa tham số. Các mảng nhiều chiều này được gọi chung là tensor.

Ta có thể thực hiện phép tính tích chập theo nhiều trục tọa độ một cách đồng thời. Ví dụ, khi ta có dữ liệu đầu vào là những hình ảnh 2 chiều, ta cũng cần có kernel 2 chiều tương ứng:



Do phép tích chập có tính giao hoán, ta cũng có thể viết biểu thức trên dưới dạng tương đương sau:



Cái khái niệm cơ bản của một mạng neural tích chập:

Feature (đặc điểm): CNNs so sánh hình ảnh theo từng mảnh. Các mảnh mà nó tìm được gọi là các feature. Bằng cách tìm ở mức thô các feature khớp nhau ở cùng vị trí trong hai hình ảnh, CNNs nhìn ra sự tương đồng tốt hơn nhiều so với việc khớp toàn bộ bức ảnh. Mỗi feature giống như một hình ảnh mini – một mảng hai chiều nhỏ. Các feature khớp với các khía cạnh chung của các bức ảnh.

Convoluton (Tích chập): quá trình CNN quan sát từng feature của một bức ảnh, sử dụng một filter (bộ lọc với kích thước cố đinh), đồng thời thu lấy những thông tin về ý nghĩa của feature bằng cách nhân ma trận điểm ảnh thuộc feature với một ma trận trọng số nào đó, và “lọc” bỏ những phần thông tin không cần thiết bằng cách đưa kết quả qua một hàm kích hoạt nào đó (vd: relu)

Tương tác thưa (kết nối thưa/ trọng số thưa): trong các lớp của mạng neural truyền thống, người ta sử dụng phép nhân ma trận giữa ma trận đầu vào và một ma trận trọng số để thu được ma trận đầu ra tương ứng. Điều này có nghĩa mỗi phần tử của ma trận đầu ra có mối liên hệ với mọi phần tử của ma trận đầu vào. Tuy nhiên đối với mạng neural tích chập, tính chất tương tác thưa cho phép sử dụng ma trận trọng số có kích thước nhỏ hơn rất nhiều so với kích thước của ma trận đầu vào. Ví dụ: dữ liệu đầu vào có thể là một ma trận điểm ảnh kích thức hàng triệu pixel, trong khi ma trận trọng số có thể chỉ cần kích thước hàng chục hoặc hàng trăm pixel. Điều này đem lại 2 lợi ích: tiết kiệm không gian bộ nhớ lưu trữ tham số và cho phép phân tích dữ liệu đầu vào ở mức độ chi tiết cao, giúp phát hiện các đặc tính cục bộ.

Chia sẻ tham số: một phần tử của ma trận trọng số được sử dụng bởi nhiều hàm trong một mô hình. Đối với mang neural truyền thống, mỗi phần tử của ma trận trọng số được sử dụng duy nhất một lần. Trái lại, trong mạng neural tích chập, một phần tử của ma trận trọng số được nhân với hầu như tất cả các phần tử của input, trừ một số phần tử biên tùy theo từng trường hợp cụ thể.

Pooling: hàm pooling có tác dụng giảm kích thước dữ liệu những vẫn giữ lại những đặc tính quan trọng bằng cách thay thế một nhóm dữ liệu bởi một dữ liệu có tính đại diện. Ví dụ: max pooling 3x3 thay thế 9 phần tử thuộc một ô vuông 3x3 bằng phần tử lớn nhất trong 9 phần tử đó.

Sau khi pooling, một hình ảnh sẽ có khoảng một phần tư số điểm ảnh so với lúc bắt đầu. Vì nó giữ các giá trị lớn nhất từ mỗi cửa sổ, nó sẽ bảo toàn tính khớp của mỗi feature bên trong cửa sổ. Nghĩa là nó không quan tâm quá nhiều về vị trí chính xác nơi feature khớp, miễn là nó khớp ở chỗ nào đó trong cửa sổ. Kết quả là CNNs có thể tìm xem liệu một feature có nằm trong hình ảnh mà không cần lo nó nằm ở đâu. Điều này giúp giải quyết vấn đề của máy tính là quá trực nghĩa.

2.2.2. Kỹ thuật tiền xử lý dữ liệu

2.2.2.1. Grayscale

Grayscale là một không gian màu sắc trong đó chứa các sắc độ khác nhau của màu xám, với sắc độ tối nhất tương ứng với màu đen và sắc độ sáng nhất tương ứng với màu trắng. Trong những trường hợp mà màu sắc của ảnh không đem lại thông tin hữu ích phục vụ quá trình huấn luyện, ảnh được chuyển về dạng grayscale để lược bỏ thông tin không cần thiết(tín hiệu nhiễu – noise) mà vẫn giữ lại thông tin hữu ích (hình dáng, kích thước của vật thể trong ảnh….)

2.2.2.2. Threshold

Threshold là kỹ thuật phân vùng hình ảnh đơn giản nhất. Từ một hình ảnh định dạng grayscale, threshold có thể tạo ra một hình ảnh nhị phân.

Với các phương pháp threshold cơ bản, nếu giá trị của một pixel lớn hơn một hằng số threshold, pixel đó sẽ được gán giá trị tương ứng với màu trắng, ngược lại pixel đó sẽ được gán giá trị tương ứng với màu đen.

Threshold thích ứng: việc chọn ra một hằng số threshold cố định và áp dụng để đánh giá toàn bộ bức ảnh trong nhiều trường hợp không phải là giải pháp tối ưu. Dưới một số điều kiện ánh sáng, sẽ có những phần của bức ảnh có mức độ sáng lớn(hoặc nhỏ) hơn phần còn lại. Những khu vực của bức ảnh có độ sáng khác nhau cần được đánh giá thông qua những giá trị threshold khác nhau phù hợp. Thuật toán threshold thích ứng tính toán giá trị threshold phù hợp với từng phân vùng của bức ảnh, từ đó cho kết quả threshold tốt hơn.

2.2.2.3. Contours

Contours là một đường cong nối tất cả các điểm liên tục có cùng màu sắc hoặc mật độ. Contours được sử dụng cho mục đích nhận diện vật thể.

Việc xác định contours đạt độ chính xác cao với những hình ảnh nhị phân. Vì vậy trước khi xác định contours cần thực hiện threshold.

Trong OpenCV, contours được xác định là đường viền bao quanh các vật thể trắng trên nền đen. Vì vậy, sau bước threshold cần thu được hình ảnh với vật thể có màu trắng và nền màu đen.

2.2.2.4. Bounding Rectangle

Bounding Rectangle của một tập hợp điểm là hình chữ nhật bao quanh tập hợp điểm đó. Trong OpenCV có 2 loạt Bounding Rectangle: Bounding Rectangle thường và Bounding Rectangle xoay.

Mỗi Bounding Rectangle được đại diện bởi 4 tham số: (x,y,w,h) trong đó x,y là tọa độ của điểm ở góc trên bên trái hình chữ nhật; w,h là chiều dài và chiều rộng của hình chữ nhật.

2.3. Các thư viện được sử dụng.

2.3.1. os

os là một module cho phép sử dụng các hàm tương tác với hệ điều hành.

os.path cho phép thao tác với các đường dẫn.

Phương thức được sử dụng trong project:

Os.path.join(): kết nối một hoặc nhiều thành phần của đường dẫn một cách phù hợp. Phương thức này nối các phần của đường dẫn với đúng một dấu ngăn cách (‘/’) theo sau mỗi phần ngoại trừ phần đuôi của đường dẫn.

os.path.basename(path): trả lại tên file/folder cuối cùng mà đường dẫn path chỉ tới,loại bỏ những folder trung gian trong đường dẫn.

os.path.splitext(path): phương thức tách pathname thành 2 phần: root và ext. Trong đó ext là extension của path còn root là phần còn lại.

Tham số:

Path: một đối tượng dạng đường dẫn cho biết đường dẫn tới một file cụ thể, đối tượng có thể thuộc kiểu string hoặc kiểu bytes.

os.path.exists(path): phương thức kiểm tra liệu một đường dẫn cụ thể có tồn tại hay không.

os.makedirs(path,mode=Oo777,exist\_ok=False): phương thức tạo đường dẫn, nếu những đường dẫn cấp trung gian chưa tồn tại thì tạo cả đường dẫn cấp trung gian.

2.3.2. glob

glob là một module cho phép tìm kiếm tất cả các đường dẫn tuân thủ một khuân dạng cụ thể (theo các quy tắc được sử dụng bởi Unix shell, kết quả của tìm kiếm được trả lại theo thứ tự ngẫu nhiên).

Phương thức được sử dụng trong project:

glob.glob(pathname,recursive=False): trả về một danh sách các đường dẫn khớp với pathname(pathname phải là một xâu ký tự chứa đặc tả đường dẫn). pathname có thể là một đường dẫn tuyệt đối hoặc tương đối.

Nếu recursive=true, ký tự “\*\*” có thể tương ứng với tên file bất kỳ hoặc các directory, hoặc các đường link dẫn tới các directory.

2.3.3. Imutils

Imutils là một thư viện chứa tập các hàm xử lý hình ảnh cơ bản như chuyển đổi, xoay, thiết lập kích thước, hiển thị hình ảnh Matplotlib, sắp xếp contours, xác định góc.

Phương thức được sử dụng trong project:

Imutils.paths.list\_image(basePath,contains=None): trả lại đường dẫn tới các file nằm trong folder tương ứng với đường dẫn basePath, nếu contains là một string khác None thì sẽ trả về đường dẫn tới những file mà tên file có chứ đoạn string đó.

Imutils.resize(image,width=None,height=None): Thay đổi kích thước hình ảnh.

Image: hình ảnhgốc có kích thước h x w, cần thay đổi kích thước

Width và height: Nếu width=height=None thì trả về hình ảnh gốc; nếu width=None thì hình ảnh mới có kích thước mới width\_new=(height/h)\*w; height\_new=height; các trường hợp còn lại width\_new=width, height\_new=(weight/w)\*h.

2.3.4. OpenCV

OpenCV là một thư viện phần mềm thị giác máy tính và học máy mã nguồn mở. OpenCV được xây dựng nhằm cung cấp cơ sở hạ tầng cho các ứng dụng thị giác máy tính và để tăng tốc việc sử dụng học máy trong các sản phẩm thương mại.

Phương thức được sử dụng trong project:

cv2.imread(path,flag):tải một hình ảnh từ một file cụ thể. Nếu không đọc được hình ảnh(do không tìm được file, không có quyền truy cập thích hợp) thì phương thức trả về một ma trận rỗng.

cv2.cvtColor(src,code[,dst[,dstCn]]): phương thức chuyển đổi không gian màu của một bức ảnh.

Tham số:

src: bức ảnh sẽ được chuyển đổi không gian màu sắc;

code:mã chuyển đổi không gian màu sắc;

dst:hình ảnh output, có kích thước và chiều sâu giống với hình ảnh src, là một tham số không bắt buộc;

dstCn:số channel của hình ảnh dst, là một tham số không bắt buộc.

cv2.copyMakeBorder(src,top,bottom, left,right,borderType,value): phương thức tạo đường biên xung quanh bức ảnh, tương tự như khung ảnh.

Tham số:

src:bức ảnh nguồn top:độ rộng của đường biên phía trên, tính theo số lượng pixel. bottom:độ rộng của đường biên phía dưới, tính theo số lượng pixel.

left:độ rộng của đường biên phía bên trái, tính theo số lượng pixel.

right:độ rộng của đường biên phía bên phải, tính theo số lượng pixel.

boderType: miêu tả loại đường biên sẽ được thêm vào (vd: cv2.BORDER\_CONSTANT,cv2.BORDER\_REFLECT,…)

value: một tham số không bắt buộc, miêu tả màu sắc của đường biên nếu borderType= cv2.BORDER\_CONSTANT

cv2.threshold(source, thresholdValue, maxVal, thresholdingTechnique): phương thức thực hiện kỹ thuật threshold. Trong kỹ thuật threshold, giá trị của mỗi pixel được so sánh với giá trị threshold. Nếu giá trị của pixel nhỏ hơn giá trị threshold thì giá trị pixel được gán bằng 0, ngược lại được gán giá trị maximum(thường là 255). Đây là một kỹ thuật phân mảng phổ biến, dùng để tách biệt vật thể và phông nền trong một bức ảnh.

Tham số:

Source: nhập hình ảnh dưới dạng array.

thresholdValue: giá trị threshold.

maxVal: giá trị lớn nhất có thể được gán cho một pixel.

ThresholdingTechnique: Kiểu threshold sẽ được áp dụng.

cv2.findContours(src,mode,method): hàm trích xuất đường viền khối (contours) từ hình ảnh. Hàm hoạt động tốt nhất với hình ảnh nhị phân, nên trước tiên ta cần áp dụng các kỹ thuật threshold.

Định nghĩa contour: một đường cong nối tất cả các điểm liên tiếp (dọc theo đường biên) có cùng màu sắc và độ đậm. Đây là một khái niệm hữu ích cho việc phân tích hình dáng và nhận dạng vật thể.

Tham số:

Src: hình ảnh gốc.

Mode: chế độ trích xuất contours (vd: cv2.RETR\_EXTERNAL, cv2.RETR\_TREE )

Method: phương thước xấp xỉ contours (vd: cv2.CHAIN\_APPROX\_NONE, cv2.CHAIN\_APPROX\_SIMPLE )

cv2.boundingRect(points): phương thức trả về một tam giác vuông bao quanh tập hợp điểm points.

Tham số:

points: một tập hợp điểm, trong project là đường contours.

output: (x,y,w,h) trong đó (x,y) là tọa độ đỉnh phía trên bên trái của tam giác vuông, w là độ dài cạnh đáy, h là chiều cao của tam giác.

cv2.imwrite(filename,img): phương thức giúp lưu một hình ảnh vào bất kỳ thiết bị lưu trữ nào.

Tham số:

filename: một xâu ký tự cho biết tên của file mà hình ảnh sẽ được lưu vào. Filename phải bao gồm định dạng hình ảnh (vd: .jpg,.png,…)

image: hình ảnh cần được lưu.

2.3.5. Numpy:

Các phương thức được sử dụng trong project:

Numpy.expand\_dims:

2.3.6. Pickle

Pickle là một module sử dụng giao thức nhị phân để chuyển hóa các cấu trúc đối tượng trong Python thành dạng chuỗi (bytes stream) hoặc ngược lại

Các phương thức dùng trong project:

pickle.dump(obj, file, protocol=None, \*, fix\_imports=True, buffer\_callback=None): viết dạng biểu diễn pickle (dạng chuỗi) của một đối tượng obj vào thư mục file.

Pickle.load(file, \*, fix\_imports=True, encoding="ASCII", errors="strict", buffers=None): đọc biểu diễn dạng chuỗi nhị phân của một đối tượng từ tập tin file, trả về đối tượng đã được xây dựng lại từ biểu diễn đó.

2.3.7. Sklearn

Các phương thức sử dụng trong project:

Sklearn.model\_selection.train\_test\_split(arrays,test\_size,train\_size,random\_state,shuffle,stratify): chia mảng hoặc ma trận (tương ứng tham số arrays) thành các tập con để huấn luyện hoặc kiểm thử.

Tham số:

arrays: có thể là list, mảng numpy, pandas dataframe, ma trận thưa scipy, phải có cùng kích thước shape[0]

test\_size: có thể là kiểu float hoặc int, nếu là float thì nằm trong đoạn [0.0,1.0], cho biết tỷ lệ số phần tử của arrays thuộc tập test, nếu là int cho biết số lượng mẫu test, nếu là None thì dựa vào train\_size để tính, nếu train\_size cũng là None thì được đặt giá trị mặc định 0.25

train\_size: tương tự test\_size, với giá trị mặc định 0.75

2.3.8. Keras

Là một API được thiết kế cho người làm việc trong lĩnh vực deep learning, cung cấp các giao diện lập trình ứng dụng đơn giản và ổn định, giúp giảm thiểu tối đa thao tác cho người dùng, cung cấp các cảnh báo lỗi rõ ràng và dễ thực thi.

Các lớp/ phương thức sử dụng trong project:

Keras.Model(): một lớp (class) có chức năng nhóm các lớp (layers) để tạo thành một đối tượng với các đặc tính phục vụ cho việc huấn luyện mô hình và suy diễn.

Các tham số:

Inputs: một đối tượng của lớp keras.Input hoặc một list các đối tượng của lớp keras.Input

Outputs: đầu ra của mô hình, tùy thuộc bài toán

Name: dạng xâu ký tự, tên của mô hình.

Keras.models.Sequential(): một lớp (class) có chức năng tập hợp một ngăn xếp tuyến tính gồm các layers để tạo thành một đối tượng keras.Model

Các phương thức:

Sequential.add(layer): thêm layer vào đỉnh ngăn xếp.

Sequential.pop(): loại bỏ layer cuối cùng thêm vào ngăn xếp.

Keras.models.load\_model(file\_path): trả về model đã được huấn luyện và lưu trong tập tin có đường dẫn file\_path

Keras.Layers:

Keras.layers.convolutional: trả về một layer tích chập

VD: Conv1D layer

Conv2D layer

Conv3D layer

SeparableConv1D layer

SeparableConv2D layer

DepthwiseConv2D layer

Conv2DTranspose layer

Conv3DTranspose layer

Keras.layers.convolutional.Conv2D:

Các thuộc tính:

Filter: số nguyên dương, cho biết số chiều của không gian đầu ra

Kernel\_size: một số nguyên dương hoặc một tuple/list 2 số nguyên dương, cho biết kích thước của một filter

Strides: một số nguyên dương hoặc một tuple/list 2 số nguyên dương, cho biết độ dài bước nhảy của filter dọc theo chiều cao và chiều rộng của input

Padding: nhận giá trị “valid” hoặc “same”.

Data\_format: “channels\_last” hoặc “channel\_first”. Cho biết thứ tự của các chiều của input.

Activation: hàm kích hoạt sẽ được sử dụng

Use\_bias: kiểu Boolean, cho biết layer có sử dụng vector bias hay không

kernel\_regularizer: hàm chuẩn hóa áp dụng cho ma trận trọng số

bias\_regularizer: hàm chuẩn hóa áp dụng cho vector bias

activity\_regularizer: hàm chuẩn hóa áp dụng cho output

kernel\_constraint: Hàm ràng buộc áp dụng cho ma trận trọng số

bias\_constraint: Hàm ràng buộc áp dụng cho vector bias

Keras.layers.Maxpooling2D:

pool\_size: một số nguyên dương hoặc một tuple 2 số nguyên dương, do biết kích thước của cửa số pooling

strides: một số nguyên dương hoặc một tuple 2 số nguyên dương, cho biết độ dài bước nhảy của cửa sổ pooling.

padding: “valid” hoặc “same”. Nếu là “valid” thì không dùng padding; nếu là “same” thì sử sụng padding sao cho với stride=1 thì kích thước output bằng kích thước input

data\_format: “channel\_last” hoặc “channel\_first”

Keras.layers.core:

Dense: trả về một layer mạng neural kêt nối đầy đủ

Các thuộc tính:

Units: số nguyên dương, cho biết số chiều của không gian output

Activation: cho biết hàm kích hoạt sẽ sử dụng; nếu không được đặc tả thì mặc định sử dụng hàm a(x)=x;

Use\_bias: kiểu Boolean, cho biết layer có dùng vector bias hay không.

Kernel\_initializer: khởi tạo vector trọng số

Bias\_initializer: khởi tạo vector bias

Kernel\_regularizer: hàm chuẩn hóa áp dụng lên vector trọng số

Bias\_regularizer: hàm chuẩn hóa áp dụng lên vector bias

Activity\_regularizer: hàm chuẩn hóa áp dụng lên vector đầu ra của layer

Kernel\_constraint: hàm ràng buộc áp dụng lên vector trọng số

Bias\_constraint: hàm ràng buộc áp dụng lên vector bias

2.4. Cấu trúc chương trình

2.4.1. Các file/folder lưu trữ dữ liệu

generated\_captcha\_images:

1 folder chứa các file .png

Mỗi file .png có tên file là đoạn text 4 ký tự, là dạng text của hình ảnh mã captcha mà file đó lưu.

VD: file 2A2X.png có nội dung là hình ảnh sau:



extracted\_letter\_images:

Một folder chứ 32 folder con.

Mỗi folder được đặt tên bởi 1 ký tự (chữ số/chữ cái latinh ngoại trừ các ký tự 1,I,O,0).

Trong mỗi folder chứ các file .png.

Mỗi file .png chứa ảnh của ký tự được dùng để đặt tên cho folder chứ nó dưới những phông dạng khác nhau.

Mỗi file .png được đặt tên dưới dạng dãy text 6 ký tự, là số thứ tự của file trong folder.

VD: nội dung file 000001.png và 000007.png thuộc folder 7:





captcha\_model.hdf5:

hdf5 ( Hierarchical data format 5) là một khuôn dạng tệp tin được sử dụng để lưu trữ một lượng lớn dữ liệu.

hdf5 cung cấp file dữ liệu lớn với cấu trúc tương đối đơn giản, chỉ sử dụng 2 loại đối tượng:

dataset: một mảng nhiều chiều, các phần tử trong mảng đồng nhất về kiểu.

group: một cấu trúc có thể chứa trong nó các dataset hoặc các group khác.

Nhờ 2 kiểu đối tượng trên, khuôn dạng hdf5 cho phép tổ chức dữ liệu trong file một cách có thứ bậc, tương tự như cấu trúc của hệ thống tệp tin( các tài nguyên trong file .hdf5 có thể được truy xuất bằng cú pháp tương tự đường dẫn file: path/to/resource). Hdf5 còn cung cấp hệ thống kiểu dữ liệu cải tiến.

Trong project, file captcha\_model.hdf5 được sử dụng để lưu trữ mô hình học được sau quá trình huấn luyện. Câu lệnh lưu mô hình vào file được keras cung cấp:

FILE\_NAME=”D:\Python\deep\_learning\solving\_captchas\_code\_examples\captcha\_model.hdf5”

model.save(FILE\_NAME)

model\_labels.dat:

Phần mở rộng .DAT đại diện cho một tập tin có chứa dữ liệu thô. Điều đó có nghĩa là nó không được kết hợp với bất kỳ một chương trình hoặc ứng dụng cụ thể nào.

Tệp tin model\_labels.dat chứa 32 nhãn (label) sẽ được sử dụng trong bước phân loại để nhận diện ký tự.

2.4.2. Các file chứa source code python (.py)

extract\_single\_letters\_from\_captcha.py:

Tệp tin này chứa mã nguồn của module tiền xử lý dữ liệu đầu vào, từ những hình ảnh captcha 4 ký tự thu về ảnh của từng ký tự độc lập.

Tóm tắt cơ chế hoạt động của extract\_single\_letters\_from\_captcha.py:

Nhận vào đường dẫn tới folder chứa tất cả ảnh captcha thuộc tập dữ liệu huấn luyện.

Duyệt qua từng bức ảnh, thực hiện tiền xử lý:

Sử dụng OpenCV, đọc dữ liệu ảnh, chuyển ảnh thành dạng grayscale, tạo đường biên, threshold, xác định contours.

Dựa vào đường viền (contours) và hình chữ nhật bao (bounding rectangle) để thực hiện xác định vật thể trong bức ảnh, ở đây vật thể là những ký tự đôc lập.

Sử dụng OpenCV, lưu lại từng ký tự vào tệp tin .png tương ứng thuộc folder extracted\_letter\_images.

Các thư viện sử dụng:

import os

import os.path

from cv2 import cv2

import glob

import imutils

train\_model.py:

Module này nhận dữ liệu đầu vào là tập các hình ảnh ký tự đơn lẻ (thuộc folder extracted\_letter\_images), trả về một mô hình mạng neural tích chập. Mô hình này có thể phân loại một hình ảnh ký tự vào lớp tương ứng với ký tự mà hình ảnh đó biểu diễn ( vd: nhận vào hình ảnh ký tự 7, gán cho bức ảnh label 7).

Hàm resize\_to\_fit(image,width,height):

Thay đổi kích thước của ảnh để phù hợp với các hàm huấn luyện mô hình.

Tóm tắt cơ chế hoạt động của train\_model.py:

Đoc tất cả ảnh từ folder extracted\_letter\_images;

Với mỗi ảnh đầu vào, chuyển ảnh về dạng grayscale, dùng hàm resize\_to\_fit đưa ảnh về kích thước 20x20, thêm chiều thứ 3 (tiền xử lý ảnh để đáp ứng yêu cầu của keras), lưu vào data[];

Dựa vào tên file chưa ảnh để xác định ký tự trong ảnh, lưu vào labels[];

Phân chia tập huấn luyện và tập kiểm thử (dùng cross-validation);

Xây dụng mô hình mạng neural với 2 lớp tích chập xen kẽ với 2 lớp MaxPooling, 1 layer ẩn sử dụng hàm kích hoạt relu và layer kết nối đầu đủ sử dụng hàm kích hoạt softmax;

Huấn luyện mô hình, lưu lại mô hình thu được vào file captcha\_model.hdf5

solve\_captchas\_with\_model.py:

Giải mã 1 captcha 4 ký tự bất kỳ sử dụng mô hình mạng neural được lưu tron g tệp captcha\_model.hdf5.

Tóm tắt cơ chế hoạt động của solve\_captchas\_with\_model.py:

Nhận hình ảnh chứa captcha 4 ký tự cần nhận diện, thực hiện tiền xử lý ảnh như ở extract\_single\_letters\_from\_captcha.py để thu được 4 ký tự độc lập;

Nhận diện từng ký tự dựa vào mô hình mạng neural ở tệp captcha\_model.hdf5;

Trả về dự đoán.