TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ
Xử lý bài toán Nhận dạng MNist trên PYTHON

GVHD: Từ Lãng Phiêu
SV: Trương Chấn Đông - 3120410133
Mai Xuân Hiếu - 3121410193
Lê Anh Duy - 3121410117

TP. HỒ CHÍ MINH, THÁNG 4/2024

Mục lục

1 Phần giới thiệu

1.1 Giới thiệu về Mnist

- Nhận diện chữ số MNIST là bài toán phân loại, nơi mô hình học máy hoặc mạng nơ-ron được huấn luyện để phân loại mỗi hình ảnh vào một trong các lớp từ 0 đến 9. Bài toán này không chỉ giúp người mới học làm quen với các kỹ thuật cơ bản trong học máy và deep learning, mà còn là nền tảng để phát triển các ứng dụng nhận diện hình ảnh phức tạp hơn trong thực tế.

1.2 Mục đích đề tài

- Mục đích của đề tài là xây dựng một ứng dụng nhận diện chữ số viết tay từ 0 đến 9 dựa trên bộ dữ liệu MNIST. Ứng dụng này có thể nhận diện và phân loại các chữ số được vẽ trên một giao diện đồ họa hoặc từ một hình ảnh được tải lên. Mục tiêu chính là thực hiện một ứng dụng thực tế và hữu ích, giúp người dùng dễ dàng và nhanh chóng nhận diện chữ số viết tay thông qua công nghệ máy tính. Đồng thời, đề tài cũng nhằm mục đích giới thiệu và áp dụng các phương pháp và công nghệ trong lĩnh vực học máy và thị giác máy tính vào một ứng dụng cụ thể.

1.3 Phạm vi của báo cáo

1.3.1 Phạm vi nghiên cứu:

- Đề tài tập trung vào việc xây dựng một ứng dụng nhận diện chữ số viết tay từ 0 đến 9, sử dụng bộ dữ liệu MNIST và các kỹ thuật học máy như Support Vector Machine (SVM) và Histogram of Oriented Gradients (HOG). Đồng thời, phạm vi nghiên cứu cũng bao gồm việc tích hợp các công nghệ web như Flask để tạo ra một giao diện trực tuyến cho việc nhận diện chữ số.

1.3.2 Phạm vi ứng dụng:

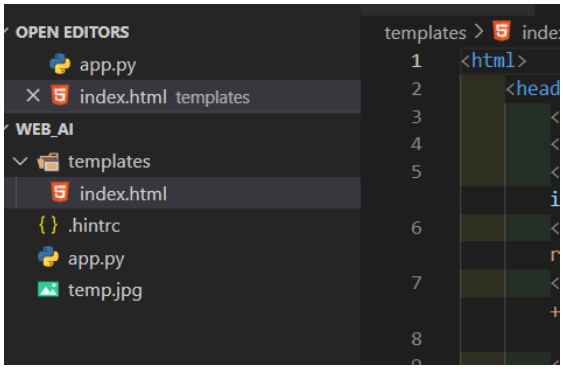
- Ứng dụng nhận diện chữ số viết tay từ 0 đến 9 nhằm mục đích cung cấp một công cụ trực tuyến giúp người dùng nhận diện chữ số từ ảnh. Phạm vi tập trung vào việc xử lý ảnh đầu vào, sử dụng phương pháp HOG (Histogram of Oriented Gradients) để trích xuất đặc trưng và mô hình học máy (Linear SVM) để dự đoán chữ số. Ứng dụng có thể áp dụng trong nhiều tình huống thực tế như tự động đọc số điện thoại từ danh bạ ảnh, ghi số từ bảng điều khiển, hoặc tích hợp vào các hệ thống tự động hoá và ứng dụng web khác.

2 Phần lý thuyết

2.1 Giới thiệu về công nghệ sử dụng và thuật toán

2.1.1 Ngôn ngữ và công nghệ web

Sử dụng thư viện Flask để tạo ra 1 trang web nhanh chóng. Cấu hình các router đường dẫn tạo ra trong miền. 1 file HTML được sử dụng để hiển thị chương trình, và sử dụng ngôn ngữ javascript để tạo các hiệu ứng động.



Hình 1: Caption 1

Hình 2: Caption 2

2.1.2 Các thư viện sử dụng

- **TensorFlow:** TensorFlow cho phép các lập trình viên tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 sê-ri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học, và mỗi kết nối hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là 'tensor'. TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python.
- **Numpy:** Numpy sử dụng để sử dụng các câu tính toán phức tạp, vẽ đồ thị 3D và phân đoạn hình ảnh.
- **Sklearn data và dataset metrix:** Sklearn data được sử dụng để tạo dữ liệu từ mnist số, sau đó sử dụng metrix để phân bố dữ liệu.

2.2 Thuật toán HOG (Histrogram of oriented gradient)

- Thuật toán HOG (Histogram of Oriented Gradients) là một phương pháp phổ biến trong xử lý ảnh và nhận dạng đối tượng. Được giới thiệu bởi Navneet Dalal và Bill Triggs vào năm 2005, HOG đã trở thành một công cụ mạnh mẽ trong nhiều ứng dụng nhận dạng đối tượng khác nhau.

2.2.1 Nguyên lý hoạt động

- **Tính toán gradient:** Đầu tiên, ảnh được chia thành các ô (cell) nhỏ. Gradient của từng điểm ảnh trong mỗi ô được tính bằng cách sử dụng các phép toán gradient như Sobel hoặc Prewitt.

- **Tính toán hướng và biểu đồ hướng gradient:** Sau khi tính toán gradient, hướng của gradient được xác định và chia thành một số khoảng (bins) dựa trên góc của gradient. Thông thường, số lượng bins là 9 để biểu diễn các góc từ 0 đến 180 độ.

- **Tính toán histogram gradient:** Đối với mỗi ô, một histogram gradient được tạo ra bằng cách tính toán tần suất xuất hiện của các góc gradient trong ô đó.

- **Tính toán cell normalization:** Các histogram gradient được chuẩn hóa để giảm ảnh hưởng của ánh sáng và cải thiện tính toàn vẹn của đặc trưng.

- **Tính toán khối (block) gradient:** Các ô được nhóm lại thành các khối, và các histogram gradient của các ô trong mỗi khối được kết hợp để tạo ra đặc trưng cuối cùng.

- **Tính toán vector đặc trưng HOG:** Tất cả các đặc trưng HOG của các khối được sắp xếp thành một vector duy nhất, tạo ra biểu diễn cuối cùng của ảnh.

2.2.2 Ứng dụng

- **Nhận dạng đối tượng:** HOG thường được sử dụng để nhận dạng đối tượng trong ảnh, bao gồm nhận dạng con người, xe hơi, và các đối tượng khác.

- **Nhận dạng chữ viết:** Trong một số ứng dụng, HOG cũng được sử dụng để nhận dạng chữ viết tay hoặc chữ số.

- **TPhát hiện đối tượng:** Ngoài việc nhận dạng đối tượng, HOG cũng có thể được sử dụng để phát hiện đối tượng trong ảnh, bao gồm việc phát hiện biên và đường viền.

3 Mô hình nhận dạng chữ số viết tay

- Trong phần này, chúng ta sẽ đi sâu vào lý thuyết về cách xây dựng một mô hình nhận dạng chữ số sử dụng các thư viện OpenCV, scikit-image, scikit-learn và Flask.

3.1 Dataset MNIST:

- Tập dữ liệu MNIST là một tập dữ liệu lớn chứa các chữ số viết tay từ 0 đến 9. Mỗi hình ảnh có kích thước 28x28 pixel và được gắn nhãn với số tương ứng. Tập dữ liệu này thường được sử dụng làm bài toán thử nghiệm đầu tiên khi làm việc với mạng neural và các thuật toán nhận dạng chữ viết tay.

Hình 3: Dataset Mnist

3.2 Trích xuất đặc trưng với HOG (Histogram of Oriented Gradients)

- HOG là một kỹ thuật phổ biến được sử dụng để trích xuất đặc trưng từ hình ảnh. Kỹ thuật này dựa trên việc tính toán các hướng gradient của các pixel trong ảnh và biểu diễn chúng dưới dạng histogram.
- Trong OpenCV và scikit-image, chúng ta có thể sử dụng các hàm như hog() để trích xuất đặc trưng HOG từ hình ảnh.

3.3 Mô hình phân loại LinearSVC (Linear Support Vector Classifier):

- LinearSVC là một mô hình học máy dựa trên hỗ trợ tuyến tính, thường được sử dụng cho các bài toán phân loại. Mô hình này tìm ra đường biên tuyến tính để phân chia các lớp dữ liệu.
- Trong scikit-learn, chúng ta có thể sử dụng lớp LinearSVC để tạo và huấn luyện mô hình phân loại LinearSVC.

3.4 Tiến trình xây dựng mô hình nhận dạng chữ số:

Bước 1: Chuẩn bị dữ liệu
- Sử dụng tập dữ liệu MNIST để huấn luyện và kiểm tra mô hình. Tập dữ liệu này chứa các hình ảnh chữ số viết tay từ 0 đến 9.
Bước 2: Trích xuất đặc trưng
- Sử dụng HOG để trích xuất đặc trưng từ các hình ảnh trong tập dữ liệu MNIST.
Bước 3: Huấn luyện mô hình
- Sử dụng LinearSVC để huấn luyện mô hình phân loại dựa trên các đặc trưng HOG đã trích xuất.
Bước 4: Tạo ứng dụng Flask
- Flask cung cấp API để nhận diện chữ số từ ảnh được tải lên.

3.5 Tích hợp vào ứng dụng web với Flask:

- Sử dụng Flask để xây dựng một ứng dụng web đơn giản cho phép người dùng viết chữ số trên giao diện.
- Khi người dùng hoàn thành việc viết, ảnh của chữ số được chụp và gửi đến máy chủ.
- Máy chủ sử dụng mô hình LinearSVC đã được huấn luyện để nhận diện chữ số từ hình ảnh và trả về kết quả cho người dùng.

4 Phần lập trình

4.1 Giải thích code trong chương trình

Khai báo thư viện trong chương trình:

```
from flask import Flask, render_template, request , jsonify
port tensorflow as tf
import base64


import cv2
import numpy as np
from skimage.feature import hog
from sklearn.svm import LinearSVC
from keras.datasets import mnist
from sklearn.metrics import accuracy_score
```

Hình 4

Sau khi khai báo thư viện cần thiết cho chương trình, bước tiếp theo là tạo data từ mnist:

```
(X_train,y_train),(X_test,y_test) = mnist.load_data()
#cho x_train
X_train_feature = []
for i in range(len(X_train)):
    feature = hog(X_train[i],orientations=9,pixels_per_cell=(14,14),cells_per_block=(1,1),block_norm="L2")
    X_train_feature.append(feature)
X_train_feature = np.array(X_train_feature,dtype = np.float32)
```

Hình 5

Tách dữ liệu thành hai tập train và test từ bộ dữ liệu mnist được cung cấp bởi thư viện sklearn. Sau khi tách xong, chúng tôi tiếp tục tách nhãn thành một tệp riêng, nhãn này được sử dụng để dự đoán ($x_{train\_feature}$) và làm tương tự cho tập test ($x_{test\_feature}$). Sau đó, chúng tôi tiến hành khai báo model LinearSVC và cài đặt các thông số cho nó.

```
#cho x_test
X_test_feature = []
for i in range(len(X_test)):
    feature = hog(X_test[i],orientations=9,pixels_per_cell=(14,14),cells_per_block=(1,1),block_norm="L2")
    X_test_feature.append(feature)
X_test_feature = np.array(X_test_feature,dtype=np.float32)
model = LinearSVC(C=10)
model.fit(X_train_feature,y_train)
y_pre = model.predict(X_test_feature)
print(accuracy_score(y_test,y_pre))
```

Hình 6

Khai báo model LinearSVC với thông số C là 10, sau đó ta sẽ train model đó với $x_{train\_feature}$ và y_{train}. Sau đó tạo thành 1 file y-pre từ file feature rồi tính độ chính xác. Tạo trang web và đường dẫn tên miền bằng Flask:

```python
app = Flask(__name__)



@app.route('/')
def index():
    return render_template("index.html")

@app.route('/recognize' , methods = ['POST'])
def recognize():
    if request.method == 'POST':
        print('adsd')
        data = request.get_json()
        imageBase64 = data['image']
        imgBytes = base64.b64decode(imageBase64)
```

Hình 7

Tạo 2 trang có tên miền khác nhau những cũng truy cấp 1 file html , trang nhận dạng kết quả là trang method bằng post. Trang nhận diện post có 1 file image lấy từ trang giao diện khi người dùng vẽ. Hình ảnh sẽ được truyền cho file test rồi model sẽ nhận diện ảnh đó rồi đưa ra kết quả

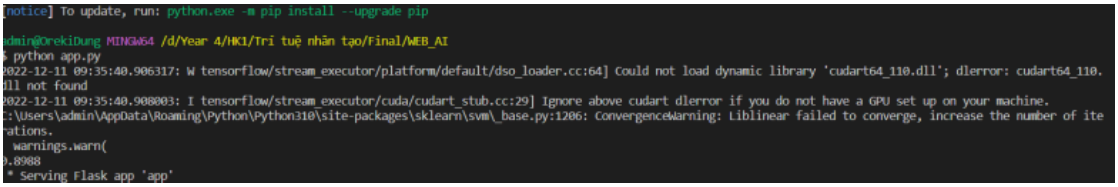
```python
    image = cv2.imread("temp.jpg")
    im_gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    im_blur = cv2.GaussianBlur(im_gray,(5,5),0)
    im,thre = cv2.threshold(im_blur,90,255,cv2.THRESH_BINARY_INV)
    contours,hierachy = cv2.findContours(thre,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    rects = [cv2.boundingRect(cnt) for cnt in contours]
    for i in contours:
        (x,y,w,h) = cv2.boundingRect(i)
        cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),3)
        roi = thre[y:y+h,x:x+w]
        roi = np.pad(roi,(20,20),'constant',constant_values=(0,0))
        roi = cv2.resize(roi, (28, 28), interpolation=cv2.INTER_AREA)
        roi = cv2.dilate(roi, (3, 3))

        # Calculate the HOG features
        roi_hog_fd = hog(roi, orientations=9, pixels_per_cell=(14, 14), cells_per_block=(1, 1),block_norm="L2")
        nbr = model.predict(np.array([roi_hog_fd], np.float32))
        cv2.putText(image, str(int(nbr[0])), (x, y),cv2.FONT_HERSHEY_DUPLEX, 2, (0, 255, 255), 3)
        cv2.imshow("image",image)
```

Hình 8

4.2 Các bước thực hiện chương trình

- B1: Chạy chương trình bằng python app.py trong file :
+ Chương trình sẽ lấy file html để hiện thị , lấy port là 5000 được cấu hình bằng flask.
+ Nếu chưa tải Flask thì nên tải Flask trước bằng câu lệnh : pip3 install flask



Hình 9

B2: Hình ảnh giao diện chương trình sẽ hiển thị , người dùng sẽ ấn vào clear để xóa khoảng đen trong hình ảnh.

Hình 10

B3: Sau khi ấn clear , người dùng vẽ các chữ số từ 1 đến 9 trong ô vuông và nhấn nút màu xanh để chương trình nhận diện:

Hình 11

- Người dùng có thể thay đổi cỡ chữ , tô dậm chữ và thay đổi màu sắc của nét vẽ

4.2.1 Kết quả chương trình

- Dự đoán số 3 :

Hình 12

- Vẽ số 9

Hình 13

5 Nhìn nhận và hướng phát triển

5.1 Ưu điểm và nhược điểm (HOG + SVM)

5.1.1 Ưu điểm

- **Hiệu quả với dữ liệu đơn giản:** Các phương pháp như HOG kết hợp với SVM rất hiệu quả khi nhận diện chữ số trên các tập dữ liệu đơn giản và sạch, chẳng hạn như bộ dữ liệu MNIST.
- **Hiểu và giải thích được:** Các đặc trưng HOG dễ hiểu và giải thích, giúp cho việc chẩn đoán và điều chỉnh mô hình trở nên dễ dàng hơn.
- **Tài nguyên tính toán ít hơn:** So với các phương pháp học sâu, các phương pháp truyền thống thường yêu cầu ít tài nguyên tính toán và có thể chạy nhanh hơn trên phần cứng thông thường.
- **Đơn giản để triển khai:** Với các thư viện như scikit-learn và OpenCV, việc triển khai các phương pháp này tương đối đơn giản và nhanh chóng.

5.1.2 Nhược điểm

- **Khả năng tổng quát hóa kém:**Các phương pháp truyền thống có thể gặp khó khăn khi nhận diện chữ số trên các tập dữ liệu phức tạp hoặc khi chữ số có nhiều biến dạng.
- **Cần điều chỉnh thủ công:**Việc lựa chọn và điều chỉnh các tham số như kích thước ô (cell size), số lượng hướng gradient (orientations) và kích thước khối (block size) có thể mất nhiều thời gian và yêu cầu kinh nghiệm.
- **Hiệu suất thấp hơn trên dữ liệu lớn:**Trên các tập dữ liệu lớn và đa dạng, các phương pháp truyền thống thường có hiệu suất kém hơn so với các phương pháp học sâu.

5.2 Hướng phát triển

- Ứng dụng nhận diện chữ số có tiềm năng phát triển mạnh mẽ trong nhiều lĩnh vực thực tế. Trong giáo dục, nó có thể tự động hóa việc chấm điểm và hỗ trợ học tập. Trong y tế, nó giúp số hóa hồ sơ bệnh nhân và hỗ trợ chẩn đoán. Trong giao thông, hệ thống giám sát biển số xe và phát hiện vi phạm giao thông có thể được triển khai. Trong tài chính, nó cải thiện xử lý giao dịch trên máy ATM và số hóa tài liệu. Trên thiết bị di động, ứng dụng có thể chụp ảnh và nhận diện chữ số trực tiếp. Tích hợp vào thực tế ảo và tăng cường sẽ cung cấp trải nghiệm tương tác tiên tiến. Cải thiện giao diện người dùng và phản hồi tức thì sẽ nâng cao trải nghiệm người dùng, làm cho ứng dụng dễ sử dụng và hiệu quả hơn.

6 TÀI LIỆU THAM KHẢO

Phân nhóm các thuật toán Machine Learning). Được truy cập tại :
https://thorpham.github.io/blog/Nh%E1%BA%ADn-d%E1%BA%A1ngch%E1%BB%AF-s%E1%BB%91vi%E1%BA%BFttayfbclid=IwAR2qYQdeY4IGc589cOwoxNfbMUlfBOUuIM7vxflorC17w_DiQKFtL5p5Q#:~:text=C%C3%B3%20r%E1%BA%A5t%20nhi%E1%BB%81u%20thu%E1%BA%ADt%20to%C3%A1n,chi%20ph%C3%AD%20t%C3%ADnh%20to%C3%A1n%20th%C3%A2p&utm_source=zalo&utm_medium=zalo&utm_campaign=zalo.
Thuật toán HOG. Được truy cập tại :
https://phamdinhkhanh.github.io/2019/11/22/HOG.html