Báo cáo: Xây dựng và sử dụng mô hình nhận dạng chữ số viết tay

Dự án này bao gồm hai tệp Python một thư mục chứa hình ảnh và một file lưu mô hình đã được huấn luyện. Mặc dù dự án này có vẻ ngắn gọn, nhưng nó đã bao gồm tất cả các bước cần thiết để xây dựng và huấn luyện một mô hình học sâu để nhận dạng chữ số viết tay, cũng như triển khai mô hình trong một ứng dụng web

1. train model.ipynb: Tệp này chứa mã để tạo tập dữ liệu từ các hình ảnh chữ số viết tay được lưu trong thư mục 'imgs', xử lý tập dữ liệu, kết hợp tập dữ liệu với tập dữ liệu MNIST, xây dựng mô hình học sâu, huấn luyện mô hình và cuối cùng là lưu mô hình đã được huấn luyện.

2. app.py: Tệp này chứa mã để xây dựng một ứng dụng web sử dụng Streamlit. Trong ứng dụng này, người dùng có thể vẽ một chữ số và mô hình sẽ dự đoán chữ số đó. Người dùng cũng có thể tải lên một hình ảnh và mô hình sẽ dự đoán chữ số trong hình ảnh đó. Hình ảnh vẽ hoặc tải lên có thể được lưu lại với tên tệp duy nhất.

3. Thư mục 'imgs': Thư mục này chứa các hình ảnh chữ số viết tay. Mỗi hình ảnh được chuyển đổi thành mảng numpy và nhãn của hình ảnh được xác định dựa trên tên tệp.

I. Xây dựng và huấn luyện mô hình (train model.ipynb)

Đây là một đoạn mã Python sử dụng thư viện Keras để xây dựng và huấn luyện một mô hình nhận dạng chữ số viết tay. Dưới đây là giải thích chi tiết :

Dự án này là một ứng dụng nhận dạng chữ số viết tay sử dụng mô hình học sâu. Dưới đây là quy trình tổng quan:

1. **Tạo và xử lý tập dữ liệu**: Tạo tập dữ liệu từ các hình ảnh chữ số viết tay trong thư mục 'imgs' và xử lý chúng để chuẩn bị cho việc huấn luyện mô hình.

```
def create_dataset(img_folder):
    img_data_array = []
    labels = []
    for filename in os.listdir(img_folder):
        if filename.endswith(".png"):
            img = Image.open(os.path.join(img_folder, filename))
            img_array = np.array(img)
            img_data_array.append(img_array)
            label = int(filename[3])
            labels.append(label)
    return img_data_array, labels

img_data, labels = create_dataset('imgs')
img_data = np.array(img_data)
labels = np.array(labels)
img_data = img_data.reshape(len(img_data), 784)
img_data = img_data.astype('float32') / 255
labels = np_utils.to_categorical(labels, 10)
```

2. **Tải và xử lý tập dữ liệu MNIST**: Tải tập dữ liệu MNIST, xử lý và kết hợp với tập dữ liệu đã tạo ở bước 1

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_train = X_train.astype('float32')
X_train /= 255
Y_train = np_utils.to_categorical(y_train, 10)
X_train = np.concatenate((X_train, img_data))
Y_train = np.concatenate((Y_train, labels))
```

3. **Xây dựng mô hình**: Xây dựng một mô hình học sâu tuần tự với hai lớp Dense và hai lớp Dropout.

```
model_dense = Sequential([
    Dense(512, input_shape=(784,), activation='relu'),
    Dropout(0.2),
    Dense(512, activation='relu'),
    Dropout(0.2),
    Dense(10, activation='softmax')
])
```

4. **Huấn luyện mô hình**: Huấn luyện mô hình với tập dữ liệu đã được kết hợp. Kích thước batch là 128 và số epoch là 11.

```
model_dense.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model_dense.fit(X_train, Y_train, batch_size=128, epochs=11, verbose=1)
```

5. **Lưu mô hình**: Lưu mô hình đã được huấn luyện vào tệp ‘dense_model.h5’ để sử dụng sau này. Đoạn mã sau minh họa quá trình này:

```
model_dense.save('dense_model.h5')
```

II. Sử dụng mô hình để dự đoán (app.py)

1. Import các thư viện cần thiết

```
import os
import numpy as np
import matplotlib.pyplot as plt
import random
from PIL import Image
from keras.models import load_model
from tensorflow.python.keras.utils import np_utils
from streamlit_drawable_canvas import st_canvas
import streamlit as st
import time
```

2. model = load_model('dense_model.h5'): Tải mô hình đã được huấn luyện từ tệp ‘dense_model.h5’.

```
model = load_model('dense_model.h5')
```

3. def process_image(image):: Định nghĩa hàm process_image để xử lý hình ảnh trước khi đưa vào mô hình. Các bước xử lý bao gồm thay đổi kích thước hình ảnh, chuyển đổi hình ảnh thành ảnh xám, chuyển đổi hình ảnh thành mảng numpy, thay đổi hình dạng của mảng, chuyển đổi kiểu dữ liệu của mảng và chuẩn hóa giá trị pixel.

```
def process_image(image):
    image = image.resize((28, 28))
    image = image.convert('L')
    image = np.array(image)
    image = image.reshape(1, 784)
    image = image.astype('float32')
    image /= 255
    return image
```

4. with col1: Bắt đầu khối mã cho cột 1. Trong cột này, người dùng có thể vẽ một chữ số và sau đó nhấn nút ‘Predict’ để dự đoán chữ số đã vẽ. Nếu người dùng muốn lưu hình ảnh đã vẽ, họ có thể nhấn nút ‘Save’. Hình ảnh sẽ được lưu ở folder “imgs” với tên tệp duy nhất bao gồm tiền tố được chọn và dấu thời gian hiện tại. Số lượng hình ảnh đã lưu cũng sẽ được hiển thị.

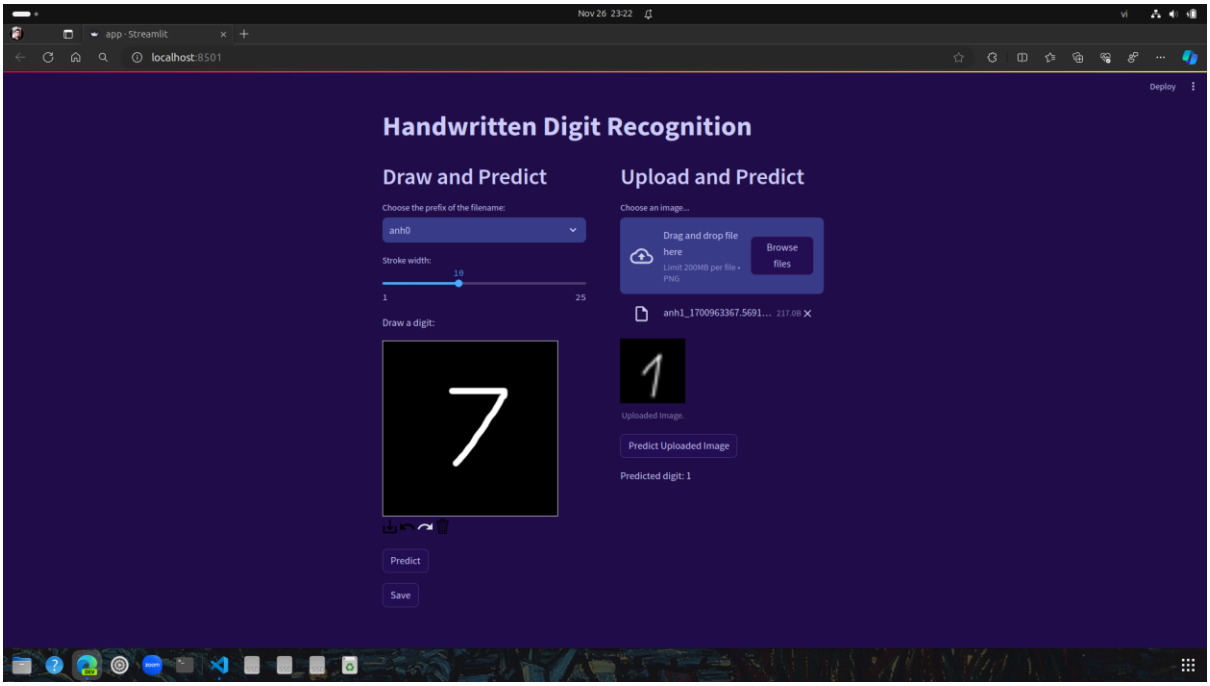
```
with col1:
    st.header("Draw and Predict")
    prefix = st.selectbox("Choose the prefix of the filename:", ["anh0", "anh1", "anh2",
"anh3", "anh4", "anh5", "anh6", "anh7", "anh8", "anh9"])

    stroke_width = st.slider("Stroke width: ", min_value=1, max_value=25, value=10)

    st.write("Draw a digit:")
    canvas_result = st_canvas(
        fill_color="rgba(255, 165, 0, 0.3)",
        stroke_width=stroke_width,
        stroke_color='#ffffff',
        background_color="#000000",
        width=280,
        height=280,
        drawing_mode="freedraw",
        key="canvas",
    )

    if canvas_result.image_data is not None:
        img = Image.fromarray(np.uint8(canvas_result.image_data)).convert('L')
        img = img.resize((28,28))
        img_arr = np.array(img)
        if np.any(img_arr):
            if st.button('Predict'):
                processed_image = process_image(img)
                prediction = model.predict(processed_image).argmax()
                st.write(f"Predicted digit: {prediction}")
            if st.button('Save'):
                timestamp = time.time()
                img.save(f'imgs/{prefix}_{timestamp}.png')
                files = os.listdir('imgs')
                images = [file for file in files if file.startswith(prefix)]
                image_counter = len(images)
                st.write(f"Images saved: {image_counter}")
```

5. with col_space:: Bắt đầu khối mã cho cột không gian. Cột này không chứa nội dung và chỉ dùng để tạo không gian giữa hai cột khác.
6. with col2: Bắt đầu khối mã cho cột 2. Trong cột này, người dùng có thể tải lên một hình ảnh chứa một chữ số và sau đó nhấn nút ‘Predict Uploaded Image’ để dự đoán chữ số trong hình ảnh đã tải lên.

```python
with col2:
    st.header("Upload and Predict")
    uploaded_file = st.file_uploader("Choose an image...", type="png")

    if uploaded_file is not None:
        img = Image.open(uploaded_file)
        img = img.resize((28,28))
        img_arr = np.array(img)

        st.image(img, caption='Uploaded Image.')

        if np.any(img_arr):
            if st.button('Predict Uploaded Image'):
                processed_image = process_image(img)
                prediction = model.predict(processed_image).argmax()
                st.write(f"Predicted digit: {prediction}")
```

Kết hợp lại tạo thành một hệ thống hoàn chỉnh để huấn luyện một mô hình phân loại chữ số viết tay và sau đó sử dụng mô hình đó để dự đoán chữ số từ hình ảnh do người dùng cung cấp, dù là thông qua vẽ trực tiếp trên ứng dụng hoặc tải lên một hình ảnh. Điều này cho phép người dùng tương tác trực tiếp với mô hình và thấy kết quả của nó ngay lập tức.

III. Dao diện của ứng dụng



Người thực hiện: Nguyễn Thanh Tùng

Lớp Python 2301

Cảm ơn Mentor Trần Tuấn Anh đã chỉ dạy hướng dẫn giúp em có thể hoàn thành dự án này. Tuy còn nhiều thiếu xót, rất mong nhận được lời nhận xét đến từ anh.