

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN



Nhận diện và giải trò chơi Sudoku qua ảnh chụp

Thị giác máy tính
Giảng viên: TS. Hà Mạnh Toàn

Sinh viên thực hiện: Trần Anh Minh 21000242

Hà Nội - 2024

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Nhận diện và giải trò chơi Sudoku through qua ảnh chụp

Thị giác máy tính
Giảng viên: TS. Hà Mạnh Toàn

Sinh viên thực hiện: Trần Anh Minh 21000242

Hà Nội - 2024

MỤC LỤC

Lời nói đầu	3
1 Giới thiệu bài toán	4
1.1 Giới thiệu trò chơi Sudoku	4
1.2 Giới thiệu bài toán	4
1.3 Cấu trúc báo cáo	5
2 Tiền xử lý ảnh đầu vào	6
2.1 Tổng quan tiền xử lý ảnh	6
2.2 Chuyển đổi sang thang độ xám	6
2.3 Làm mịn ảnh bằng Gaussian Blur	7
2.4 Phân ngưỡng động	8
3 Xác định đường biên và ma trận Sudoku	10
3.1 Khái niệm đường biên	10
3.2 Phương pháp phát hiện đường biên	10
3.3 Xác định ma trận Sudoku từ đường biên tứ giác lớn nhất	12
3.4 Biến đổi phối cảnh	13
4 Mô hình học máy Residual Network 18 nhận diện ký tự	15
4.1 Giới thiệu về ResNet-18	15
4.2 Cấu trúc và cách thức hoạt động	15
4.3 Hiệu suất của mô hình ResNet-18	16
5 Giải bài toán Sudoku bằng thuật toán quay lui	18
5.1 Thuật toán quay lui	18
6 Một số kết luận	20
6.1 Về khả năng nhận diện bảng Sudoku từ ảnh	20
6.2 Về nhận dạng chữ số	20
6.3 Về giải bài toán Sudoku	20
6.4 Một số giải pháp phát triển	20
Tài liệu tham khảo	22

MỤC LỤC HÌNH ẢNH

Hình 1.1 Câu đố Sudoku và lời giải của nó.	4
Hình 1.2 Lời giải Sudoku hiển thị trực tiếp trên ảnh	5
Hình 2.1 Ảnh 3 màu RGB và ảnh xám	7
Hình 2.2 Bộ lọc Gaussian Blur kích thước 15×15 , $\sigma = 1$	7
Hình 2.3 Ảnh xám trước và sau khi áp dụng bộ lọc Gaussian Blur	8
Hình 2.4 Ảnh xám trước và sau khi phân ngưỡng	9
Hình 3.1 Các đường biên tìm được trên ảnh nhị phân biểu diễn lên ảnh gốc .	11
Hình 3.2 Đường biên lớn nhất biểu diễn lên ảnh gốc	13
Hình 3.3 Ảnh trước và sau khi áp dụng phép biến đổi phối cảnh	14
Hình 3.4 Các hộp số của bảng Sudoku	14
Hình 4.1 Cấu trúc của mạng ResNet-18.	15
Hình 4.2 Các chữ số với font khác nhau trong tập dữ liệu	16
Hình 4.3 Ma trận nhầm lẫn trên tập dữ liệu kiểm thử	16
Hình 4.4 Nhận diện chữ số có trong ma trận Sudoku sử dụng ResNet-18 . .	17
Hình 5.1 So sánh đầu vào và đầu ra của thuật toán.	19
Hình 5.2 Lời giải sudoku hiển thị trực tiếp trên ảnh	19

LỜI NÓI ĐẦU

Với sự phát triển vượt bậc của công nghệ và khoa học dữ liệu, thị giác máy tính đã trở thành một công cụ quan trọng trong việc giải quyết các bài toán thực tiễn. Những ứng dụng của thị giác máy tính ngày càng phong phú và đa dạng, từ nhận diện khuôn mặt, phát hiện vật thể đến trích xuất và xử lý thông tin từ hình ảnh. Xuất phát từ ý tưởng ứng dụng công nghệ để giải quyết các bài toán tự động, tôi đã lựa chọn đề tài "Nhận diện và giải trò chơi Sudoku thông qua ảnh chụp". Thông qua đó để tìm hiểu sâu hơn về các phương pháp và công nghệ hiện đại trong xử lý ảnh và thị giác máy tính.

Tôi xin gửi lời cảm ơn chân thành đến TS. Hà Mạnh Toàn, người đã tận tình hướng dẫn, định hướng trong quá trình nghiên cứu và hoàn thành báo cáo này.

Do nhiều yếu tố hạn chế, báo cáo không tránh khỏi những thiếu sót. Rất mong nhận được các ý kiến đóng góp để hoàn thiện hơn trong tương lai.

Xin chân thành cảm ơn!

Hà Nội, Ngày 2 tháng 3 năm 2025

CHƯƠNG 1

GIỚI THIỆU BÀI TOÁN

1.1 Giới thiệu trò chơi Sudoku

Sudoku là một loại trò chơi câu đố logic tăng trí tuệ, yêu cầu người giải đố tìm ra câu trả lời và đặt các con số tự nhiên (từ 1 đến 9) vào một ma trận có kích thước là 9x9, được chia thành 9 khối con 3x3. Ban đầu, một vài con số sẽ được điền vào các ô vuông. Để hoàn thành bảng giải đố, người chơi cần đặt các con số vào ô vuông sao cho mỗi cột, mỗi hàng và mỗi khối con đều chứa tất cả số từ 1 đến 9. Đây là một tựa game logic thú vị và được hưởng ứng đến từ rất nhiều nước trên thế giới, câu hỏi đặt ra là liệu có cách nào để khi có một đề Sudoku bất kỳ, ta chỉ cần chụp ảnh lại và sẽ có thể được giải câu đố đó một cách tự động hay không?

	8			5	1	7	9	
		2		6		8	4	
9		3			6			
2	7			8	5		3	
4				5	8	1	2	
		8		4	2			7
8				3			1	
3	5	4		1		9		
9	6			2	4	7		

(a) Câu đố Sudoku

6	8	2	4	3	5	1	7	9
7	1	5	2	9	6	3	8	4
9	4	3	8	7	1	6	2	5
2	7	1	6	8	9	5	4	3
4	6	9	3	5	7	8	1	2
5	3	8	1	4	2	9	6	7
8	2	7	9	6	3	4	5	1
3	5	4	7	1	8	2	9	6
1	9	6	5	2	4	7	3	8

(b) Lời giải

Hình 1.1: Câu đố Sudoku và lời giải của nó.

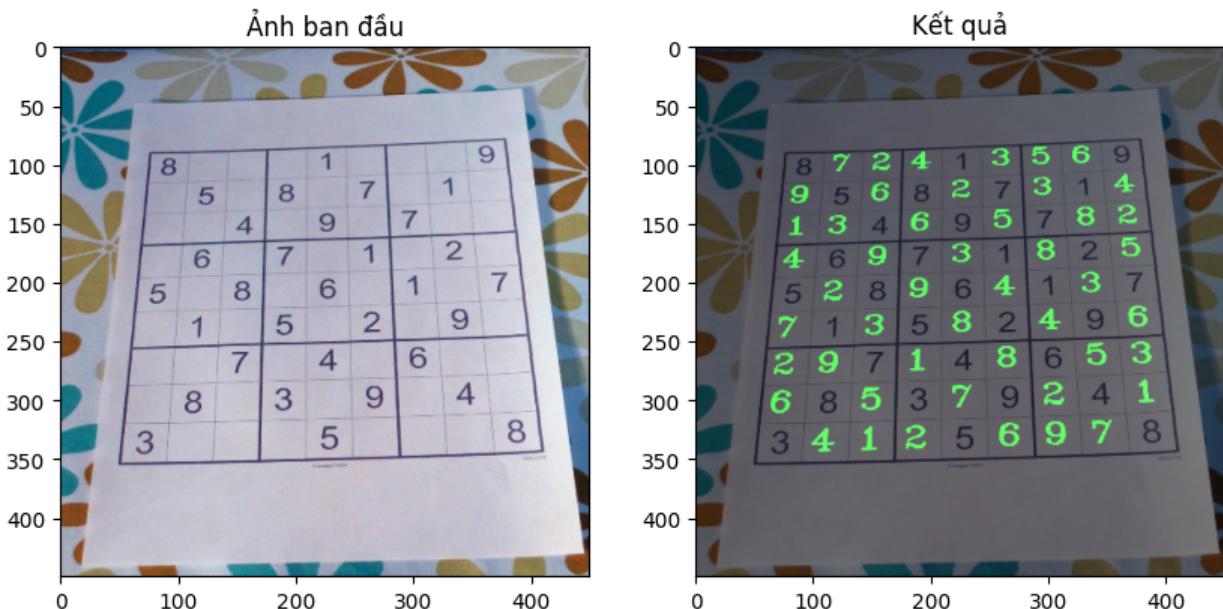
1.2 Giới thiệu bài toán

Trong thời đại công nghệ phát triển mạnh mẽ, thị giác máy tính và học máy đã trở thành những công cụ hữu ích và đầy tiềm năng trong việc giải quyết nhiều bài toán thực tế. Ta hoàn toàn có thể sử dụng những ứng dụng đó trong việc giải trò chơi Sudoku một cách tự động. Bởi luật chơi tuy tưởng chừng đơn giản, nhưng để giải nhanh và chính xác vẫn là một thách thức nếu giải Sudoku thủ công.

Trong báo cáo này, bài toán được giải quyết bằng cách tận dụng sức mạnh của thị giác máy tính và học máy để nhận diện và trích xuất lưới Sudoku cùng các con số trong ô từ hình ảnh chụp bằng điện thoại di động hoặc máy tính. Sau khi các chữ số được nhận dạng, chúng được đưa vào ma trận và lời giải được tìm bằng phương pháp quay lui, đảm bảo tính chính xác và hiệu quả.

Cụ thể, sau khi xây dựng ta có thể thu được đầu vào và đầu ra như sau:

- Đầu vào: Ảnh đề câu đố sudoku
- Đầu ra: Ảnh lời giải của câu đố đầu vào



Hình 1.2: Lời giải Sudoku hiển thị trực tiếp trên ảnh

Trong đó đầu vào có thể lấy bằng cách chèn các file ảnh hoặc chụp trực tiếp thông qua camera của máy tính.

1.3 Cấu trúc báo cáo

Báo cáo được chia thành các nội dung chính tương ứng với các quá trình xây dựng mô hình như sau:

- Tiền xử lý ảnh đầu vào.
- Xác định đường viền và ma trận câu đố Sudoku.
- Xây dựng mô hình học máy Resnet-18 để nhận diện các chữ số trong bảng trò chơi Sudoku.
- Thuật toán quay lui giải Sudoku.
- Kết quả thực nghiệm.
- Kết luận.

CHƯƠNG 2

TIỀN XỬ LÝ ẢNH ĐẦU VÀO

2.1 Tổng quan tiền xử lý ảnh

Để đảm bảo tính nhất quán về kích thước và định dạng, ảnh được resize về kích thước chuẩn 450×450 . Kích thước này được chọn vì nó đảm bảo tỷ lệ hợp lý để xử lý các ô vuông Sudoku, đồng thời giúp giảm kích thước dữ liệu mà không làm mất quá nhiều thông tin quan trọng.

Tiền xử lý ảnh bao gồm ba bước chính:

- 1) **Chuyển đổi sang thang độ xám:** Loại bỏ thông tin màu không cần thiết để tập trung vào độ sáng và các đặc trưng cơ bản.
- 2) **Làm mịn ảnh bằng Gaussian Blur:** Loại bỏ nhiễu cục bộ trong ảnh và làm mịn các vùng không cần thiết.
- 3) **Phân ngưỡng động (Adaptive threshold):** Chuyển ảnh sang dạng nhị phân, làm nổi bật các đường viền và số trong ô Sudoku.

2.2 Chuyển đổi sang thang độ xám

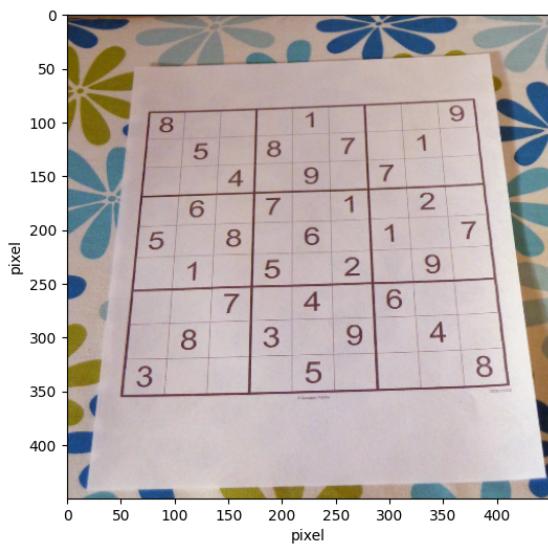
Việc chuyển đổi ảnh từ không gian màu (RGB) sang thang độ xám giúp giảm bớt thông tin dư thừa và tập trung vào các đặc điểm chính của ảnh. Đồng thời, nó làm giảm kích thước dữ liệu xử lý từ 3 kênh (R, G, B) xuống 1 kênh thang xám, từ đó cải thiện hiệu suất xử lý. Thang độ xám chỉ lưu trữ độ sáng của mỗi điểm ảnh, điều này phù hợp với bài toán nhận dạng các đường viền và số trong ô Sudoku.

Công thức chuyển đổi từ màu (RGB) sang thang độ xám:

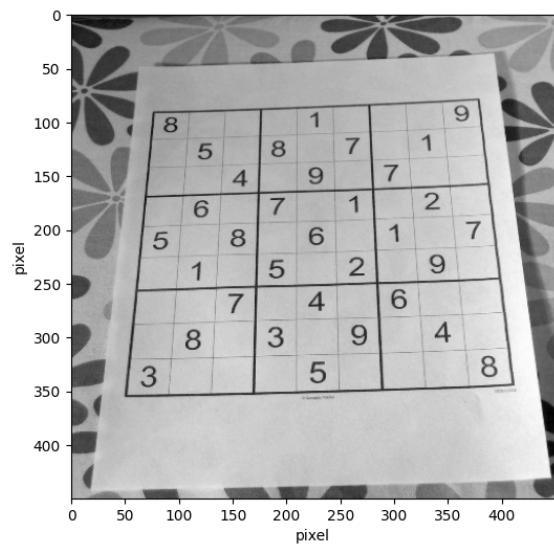
$$I(x, y) = 0.2989 \cdot R(x, y) + 0.5870 \cdot G(x, y) + 0.1140 \cdot B(x, y)$$

trong đó:

- $I(x, y)$ là giá trị độ sáng của điểm ảnh tại vị trí (x, y) trong ảnh thang độ xám.
- $R(x, y)$, $G(x, y)$, và $B(x, y)$ là giá trị màu đỏ, xanh lá và xanh dương tại vị trí (x, y) trong ảnh RGB.



(a) Ảnh ban đầu đã điều chỉnh kích thước



(b) Ảnh xám

Hình 2.1: Ảnh 3 màu RGB và ảnh xám

2.3 Làm mịn ảnh bằng Gaussian Blur

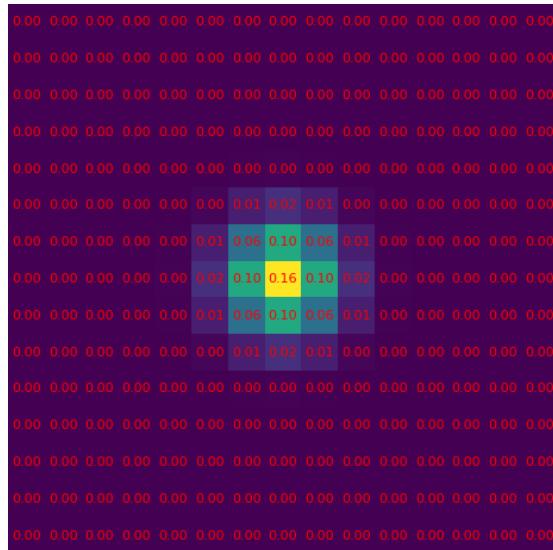
Bộ lọc Gaussian Blur được sử dụng để giảm nhiễu cục bộ và làm mịn ảnh, giúp làm nổi bật các đường biên. Giá trị bộ lọc Gaussian được tính toán dựa trên công thức:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

trong đó:

- x, y : tọa độ điểm trong bộ lọc so với tâm chính giữa của nó;
- σ : độ lệch chuẩn, điều khiển mức độ làm mịn.

Mỗi phần tử trong bộ lọc được chuẩn hóa để tổng giá trị bằng 1, đảm bảo độ sáng tổng thể không thay đổi.

Hình 2.2: Bộ lọc Gaussian Blur kích thước 15×15 , $\sigma = 1$

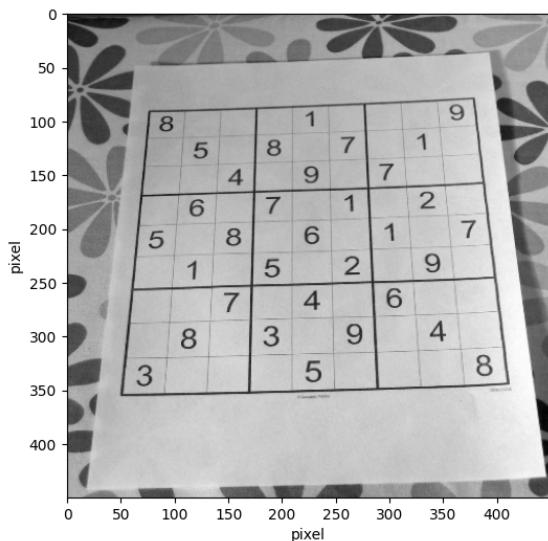
Khi áp dụng bộ lọc lên ảnh, giá trị mới tại mỗi điểm ảnh được tính là tổng trọng số của các điểm lân cận, nhân với giá trị bộ lọc tương ứng:

$$I'(i, j) = \sum_{x,y} G(x, y) \cdot I(i + x, j + y)$$

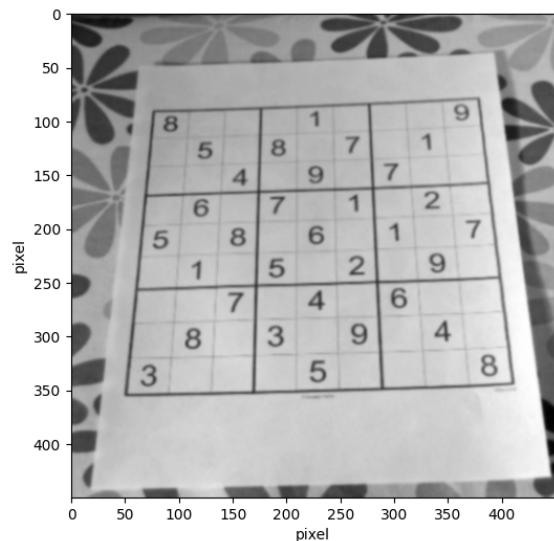
trong đó $I(i, j)$ là giá trị điểm ảnh gốc, $I'(i, j)$ là giá trị sau khi làm mịn.

Đối với kích cỡ ảnh là 450×450 , kích thước bộ lọc ta chọn là $(15, 15)$, đủ lớn để làm mịn rõ rệt các vùng nhiễu nhỏ trong ảnh. Độ lệch chuẩn σ được đặt là 1, giúp giữ lại nhiều chi tiết lớn hơn mà vẫn giảm thiểu nhiễu.

Nhiều trong ảnh có thể kể đến các đường viền mờ hoặc đốm sáng làm giảm hiệu quả phân ngưỡng. Gaussian Blur giúp làm mịn các vùng nhiễu mà không làm mất đi các đặc điểm lớn như đường kẻ trong Sudoku.



(a) Ảnh xám chưa làm mịn



(b) Ảnh làm mịn bằng Gaussian Blur

Hình 2.3: Ảnh xám trước và sau khi áp dụng bộ lọc Gaussian Blur

2.4 Phân ngưỡng động

Phân ngưỡng động (adaptive thresholding) có tác dụng chuyển ảnh từ dạng thang độ xám sang ảnh nhị phân, làm nổi bật rõ ràng các đường viền và số.

Mỗi điểm ảnh (x, y) sẽ có một ngưỡng riêng được xác định bằng công thức:

$$T(x, y) = \frac{1}{W} \sum_{(i,j) \in W} I(i, j) - C,$$

trong đó:

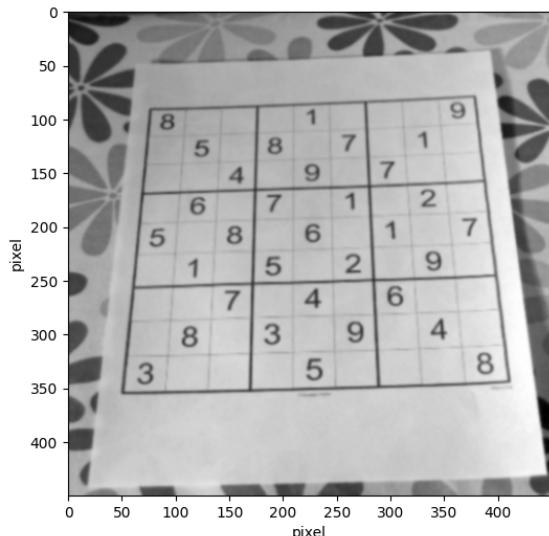
- W là cửa sổ kích thước $m \times m$ bao quanh điểm (x, y)
- $C = 2$ là giá trị hằng số trừ đi để điều chỉnh ngưỡng.

Điểm ảnh được phân loại như sau:

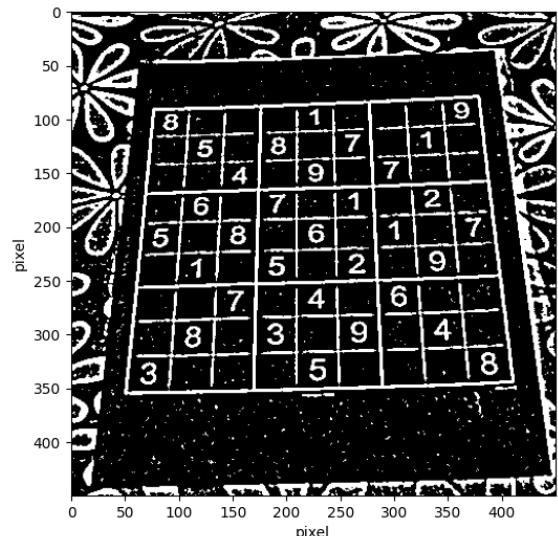
$$I'(x, y) = \begin{cases} 255, & \text{nếu } I(x, y) \leq T(x, y), \\ 0, & \text{nếu } I(x, y) > T(x, y). \end{cases}$$

Ánh sáng không đồng đều trong ảnh Sudoku thường gây khó khăn khi áp dụng phân ngưỡng toàn cục. Phân ngưỡng động đảm bảo rằng các ô, số và đường kẻ vẫn được làm nổi bật bất chấp sự thay đổi cục bộ của ánh sáng.

Đối với kích thước ảnh là 450×450 cửa sổ kích thước 11×11 được chọn để xử lý các thay đổi cục bộ nhưng không quá lớn để làm mất chi tiết quan trọng. Hằng số $C = 2$ giúp điều chỉnh phù hợp với độ sáng trung bình trong các cửa sổ.



(a) Ánh làm mịn bằng Gaussian Blur



(b) Ánh sau khi được phân ngưỡng

Hình 2.4: Ánh xám trước và sau khi phân ngưỡng

CHƯƠNG 3

XÁC ĐỊNH ĐƯỜNG BIÊN VÀ MA TRẬN SUDOKU

Đường biên (contour) là một công cụ được sử dụng để xác định và phân tích các đối tượng trong ảnh. Ta cần xác định đường biên của ảnh để có thể lấy ra ma trận Sudoku từ ảnh nhị phân.

3.1 Khái niệm đường biên

Đường biên (contour) là tập hợp các điểm liên tiếp nhau tạo thành ranh giới bao quanh một đối tượng trong ảnh. Để tìm được các đường biên, ảnh cần được chuyển sang dạng nhị phân, trong đó các vật thể và nền được phân tách rõ ràng. Đường biên được xác định dựa trên các pixel biên của đối tượng.

3.2 Phương pháp phát hiện đường biên

Fương pháp phát hiện đường biên dựa trên việc quét ảnh nhị phân và sử dụng thuật toán Suzuki và Abe để dò theo các pixel biên của đối tượng. Hàm `cv2.findContours` trong OpenCV là một công cụ tiêu chuẩn để thực hiện nhiệm vụ này. Hàm này hoạt động dựa trên các bước sau:

- 1) **Phát hiện biên của đối tượng:** Hàm tìm kiếm các pixel biên trong ảnh nhị phân dựa trên sự thay đổi cường độ giữa các pixel láng giềng. Một pixel được coi là biên nếu gradient của nó thỏa mãn điều kiện sau:

$$\nabla I(x, y) > T,$$

với $\nabla I(x, y)$ là gradient tại điểm (x, y) và T là ngưỡng xác định biên.

Trong thực tế, gradient của ảnh không được tính trực tiếp mà thường sử dụng các bộ lọc Sobel để xấp xỉ đạo hàm riêng tại mỗi điểm.

- Đạo hàm theo các trục x và y được ước lượng bằng tích chập ảnh với các bộ lọc Sobel tương ứng:

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

- Gradient tổng hợp được tính từ hai thành phần:

$$\nabla I(x, y) = \sqrt{G_x^2 + G_y^2},$$

trong đó:

$$G_x(x, y) = \sum_{i=-1}^1 \sum_{j=-1}^1 K_x(i, j) \cdot I(x + i, y + j).$$

$$G_y(x, y) = \sum_{i=-1}^1 \sum_{j=-1}^1 K_y(i, j) \cdot I(x + i, y + j).$$

- 2) **Dò theo đường biên:** Thuật toán quét ảnh để phát hiện các vùng có giá trị pixel khác 0. Từ một điểm biên, nó dò theo hướng chiều kim đồng hồ để tìm toàn bộ đường biên bao quanh đối tượng.
- 3) **Xấp xỉ hình dạng đường biên:** Để giảm bớt số lượng điểm lưu trữ, ta sử dụng phương pháp xấp xỉ cv2.CHAIN_APPROX_SIMPLE loại bỏ các điểm thẳng hàng và chỉ lưu giữ các điểm góc.

Điều kiện loại bỏ điểm thẳng hàng:

$$(x_3 - x_2)(y_2 - y_1) = (y_3 - y_2)(x_2 - x_1),$$

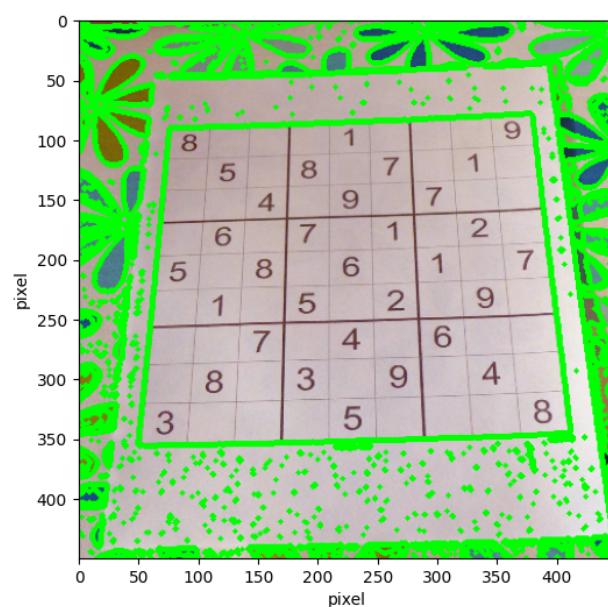
với (x_1, y_1) , (x_2, y_2) , và (x_3, y_3) là ba điểm liên tiếp.

- 4) **Kết quả của hàm cv2.findContours bao gồm:**

- **Danh sách đường biên (contours):** Mỗi đường biên là một tập hợp các điểm (x, y) liên tiếp xác định ranh giới của đối tượng:

$$\text{Contour}_i = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}.$$

- **Cấu trúc phân cấp (hierarchy):** Thông tin về mối quan hệ giữa các đường biên, bao gồm các đường biên cha, con hoặc đồng cấp.



Hình 3.1: Các đường biên tìm được trên ảnh nhị phân biểu diễn lên ảnh gốc

3.3 Xác định ma trận Sudoku từ đường biên tứ giác lớn nhất

Sau khi tìm được các đường biên (contour) từ ảnh nhị phân, bước tiếp theo là xác định đường biên đại diện cho ma trận Sudoku. Đây là đường biên có diện tích lớn nhất và có hình dạng gần giống một hình chữ nhật.

- Diện tích của một đường biên được tính bằng công thức Shoelace:

$$\text{area} = \sum_{i=1}^n \frac{1}{2} |x_i y_{i+1} - y_i x_{i+1}|,$$

với $(x_{n+1}, y_{n+1}) = (x_1, y_1)$.

- Độ dài chu vi của đường biên được tính bằng:

$$\text{peri} = \sum_{i=1}^n \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}.$$

Sử dụng độ dài chu vi, ta áp dụng thuật toán xấp xỉ hình đa giác *Douglas-Peucker* để kiểm tra xem đường biên có thể được xấp xỉ bằng một hình tứ giác (tức là có 4 đỉnh) hay không. Thuật toán này giảm số lượng điểm của đường biên bằng cách duy trì sai số tối đa giữa mỗi điểm của đường biên gốc và đường biên xấp xỉ không vượt quá một ngưỡng ε . Đồng thời, thuật toán cũng tối thiểu hóa tổng sai số giữa các điểm nhằm đảm bảo rằng đường biên xấp xỉ có hình dạng gần nhất với đường biên gốc.

$$\text{approx} = \arg \min \sum_{i=1}^n d(p_i, C),$$

với $d(p_i, C)$ là khoảng cách từ điểm p_i (thuộc đường biên gốc) đến đường biên được xấp xỉ C , sao cho:

$$d(p_i, C) \leq \varepsilon.$$

Chọn ε được chọn tỷ lệ thuận với chu vi đường biên: $\varepsilon = \alpha \cdot \text{peri}$ (ví dụ: $\alpha = 0.02$).

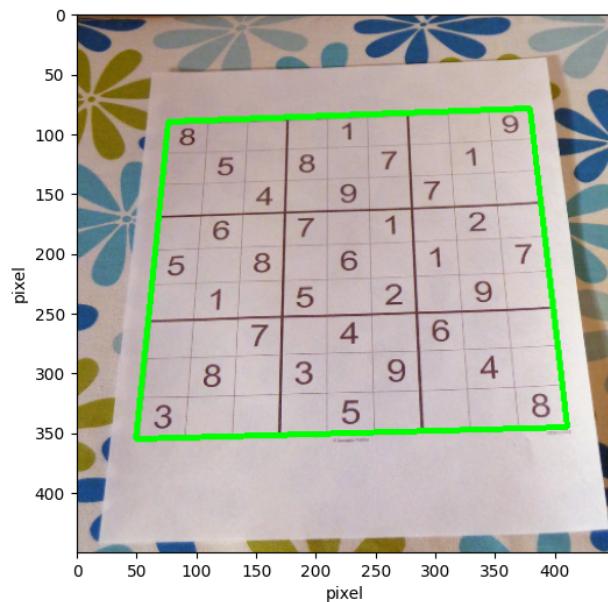
Đường biên lớn nhất được chọn sẽ có bốn đỉnh (P_1, P_2, P_3, P_4) . Tuy nhiên, các đỉnh này có thể không theo thứ tự phù hợp cho các bước xử lý tiếp theo. Do đó, cần thực hiện sắp xếp lại thứ tự các đỉnh như sau:

- Tính tổng tọa độ $(x + y)$ cho mỗi đỉnh. Đỉnh có tổng nhỏ nhất là góc trên bên trái, và đỉnh có tổng lớn nhất là góc dưới bên phải:

$$\text{top-left} = \arg \min(x + y), \quad \text{bottom-right} = \arg \max(x + y).$$

- Tính hiệu tọa độ $(x - y)$ cho mỗi đỉnh. Đỉnh có hiệu nhỏ nhất là góc trên bên phải, và đỉnh có hiệu lớn nhất là góc dưới bên trái:

$$\text{top-right} = \arg \min(x - y), \quad \text{bottom-left} = \arg \max(x - y).$$



Hình 3.2: Đường biên lớn nhất biểu diễn lên ảnh gốc

3.4 Biến đổi phối cảnh

Mục tiêu của bước này là biến đổi bức ảnh của bảng Sudoku sao cho các tọa độ của 4 đỉnh của bảng (các góc của tứ giác đường biên) được biến đổi thành các tọa độ của 4 góc của hình vuông trong không gian ảnh.

Đầu tiên, ta chọn 4 điểm trên ảnh gốc là các đỉnh của bảng Sudoku. Gọi các tọa độ này là:

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$$

Ta cần một phép biến đổi tuyến tính này có thể được mô tả bởi một ma trận H , chuyển đổi tọa độ cũ thành các tọa độ mới:

$$(x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3), (x'_4, y'_4)$$

Tức là $\forall i = 1, 2, 3, 4$ ta có:

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = H \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

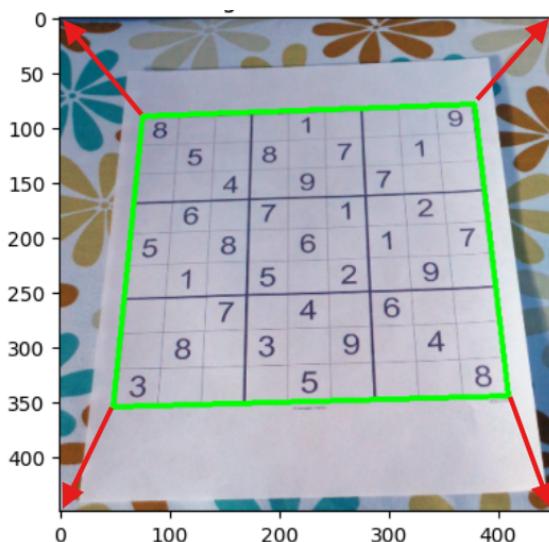
Để tính ma trận biến đổi H , ta có thể thiết lập một hệ phương trình từ các cặp tọa độ $(x_i, y_i) \rightarrow (x'_i, y'_i)$ như sau:

$$\begin{aligned} x'_1(a_{31}x_1 + a_{32}y_1 + 1) &= a_{11}x_1 + a_{12}y_1 + a_{13} \\ y'_1(a_{31}x_1 + a_{32}y_1 + 1) &= a_{21}x_1 + a_{22}y_1 + a_{23} \\ (\text{do } a_{31}x_1 + a_{32}y_1 + 1 = 1) \end{aligned}$$

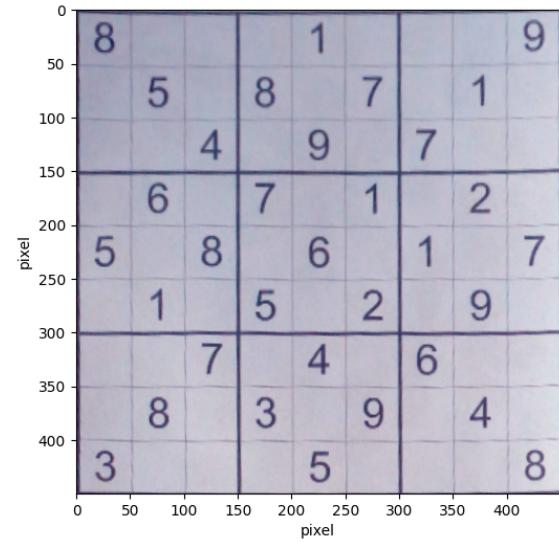
Làm tương tự cho các điểm còn lại $(x_2, y_2), (x_3, y_3), (x_4, y_4)$, ta thu được hệ phương trình tuyến tính như sau:

$$A \cdot a = b \iff \underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2 x_2 & -x'_2 y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2 x_2 & -y'_2 y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3 x_3 & -x'_3 y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3 x_3 & -y'_3 y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4 x_4 & -x'_4 y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4 x_4 & -y'_4 y_4 \end{bmatrix}}_A \cdot \underbrace{\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix}}_a = \underbrace{\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix}}_b \iff a = A^{-1} \cdot b$$

Sau khi giải được a gồm các hệ số của ma trận biến đổi H , ta có thể sử dụng ma trận này để áp dụng lên các điểm trong miền bị giới hạn bởi đường biên, từ đó thu được bức ảnh đã được biến đổi thành hình vuông.



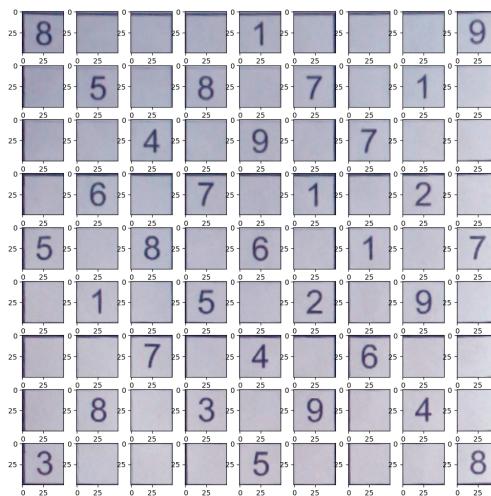
(a) Ảnh trước phép biến đổi tuyến tính



(b) Ảnh sau phép biến đổi tuyến tính

Hình 3.3: Ảnh trước và sau khi áp dụng phép biến đổi phôi cảnh

Lúc này chỉ cần chia ảnh thành 81 ô vuông bằng nhau kích thước 50×50 là ta có thể lấy được các ô của ma trận Sudoku.



Hình 3.4: Các hộp số của bảng Sudoku

CHƯƠNG 4

MÔ HÌNH HỌC MÁY RESIDUAL NETWORK 18 NHẬN DIỆN KÝ TỰ

Trong chương này, chúng ta sẽ trình bày cách sử dụng mô hình Residual Network 18 (ResNet-18) để nhận diện các ký tự số trong bảng Sudoku. ResNet-18 là một trong những mạng học sâu (deep learning) phổ biến, nổi bật với khả năng xử lý các vấn đề học sâu hiệu quả nhờ vào kiến trúc residual blocks, giúp giảm thiểu hiện tượng suy giảm gradient (vanishing gradient) trong quá trình huấn luyện.

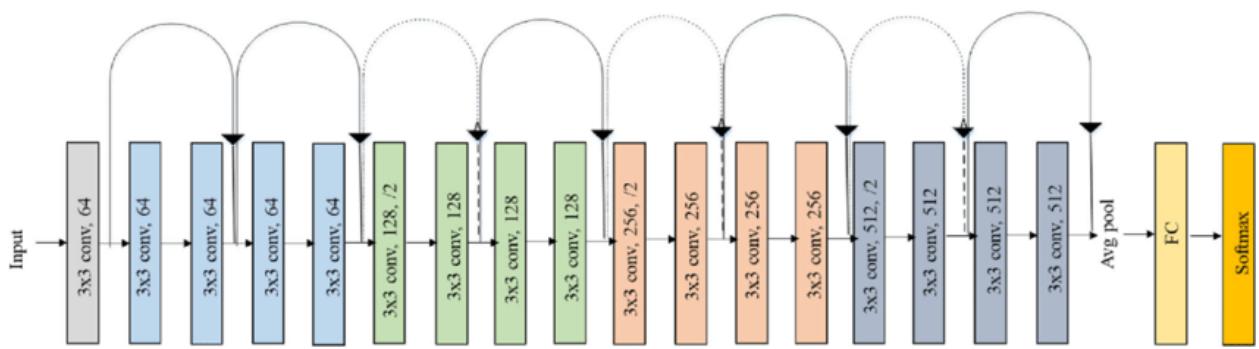
4.1 Giới thiệu về ResNet-18

ResNet-18 là một biến thể của mạng Residual Network, với 18 lớp. Mô hình này sử dụng các residual blocks, trong đó mỗi block có thể có một kết nối nhảy (skip connection) qua một lớp, giúp tăng độ sâu của mạng mà không làm mất đi hiệu quả trong việc huấn luyện. Một số đặc điểm chính của ResNet-18 bao gồm:

- Kiến trúc mạng sâu với 18 lớp.
- Sử dụng skip connections để giảm thiểu vấn đề giảm gradient.

4.2 Cấu trúc và cách thức hoạt động

ResNet-18 bao gồm các lớp convolutional cơ bản, các lớp residual block, và lớp fully connected (FC) ở cuối mạng. Dưới đây là mô hình tổng quát của ResNet-18:



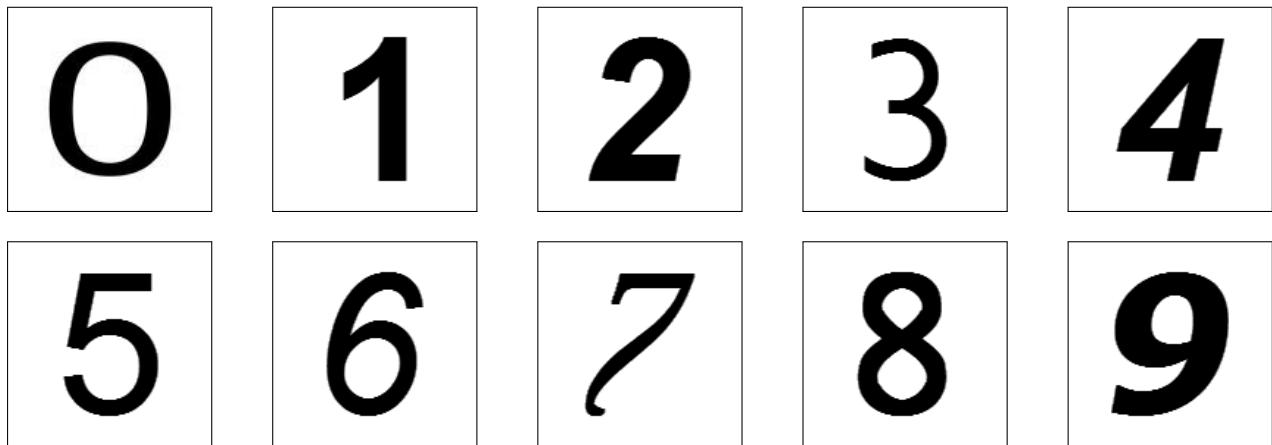
Hình 4.1: Cấu trúc của mạng ResNet-18.

Trong hình minh họa, các skip connection được biểu diễn bằng các mũi tên nối giữa đầu vào và đầu ra của các khối residual. Skip connection cho phép đầu vào x của một khối được truyền trực tiếp và cộng với đầu ra $F(x, W)$ của khối đó. Cách thiết

kế này giúp bảo toàn thông tin từ đầu vào, đồng thời giảm nguy cơ mất gradient khi truyền ngược, nhờ cung cấp một đường dẫn trực tiếp cho gradient qua nhiều lớp. Hơn nữa, các khối residual học phần dư giữa đầu vào và đầu ra mong muốn, giúp mạng hội tụ nhanh hơn và duy trì hiệu suất cao ngay cả khi mạng trở nên rất sâu.

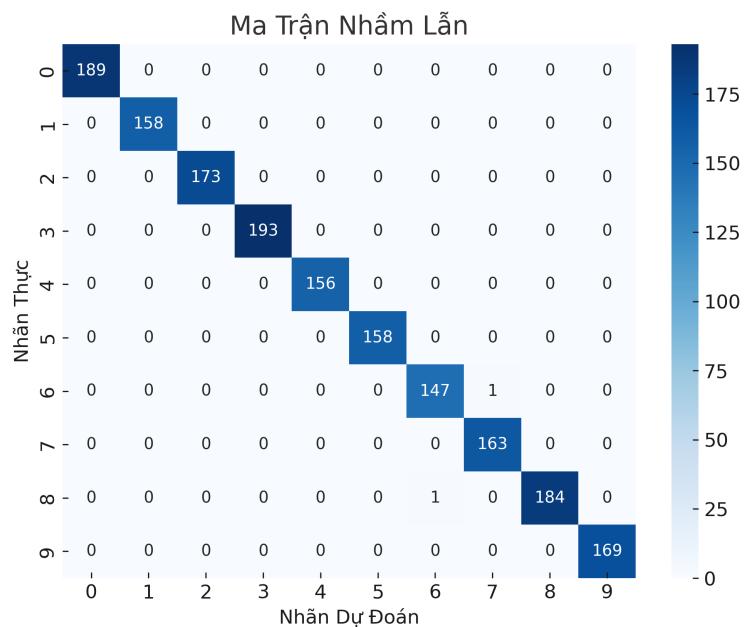
4.3 Hiệu suất của mô hình ResNet-18

Mô hình ResNet-18 được huấn luyện trên một bộ dữ liệu gồm 8460 ký tự số từ 0 đến 9, trong đó 80% dữ liệu được sử dụng để huấn luyện và 20% dành cho kiểm thử. Các chữ số trong bộ dữ liệu thuộc nhiều font khác nhau và đã được chuẩn hóa về kích thước 128×128 .



Hình 4.2: Các chữ số với font khác nhau trong tập dữ liệu

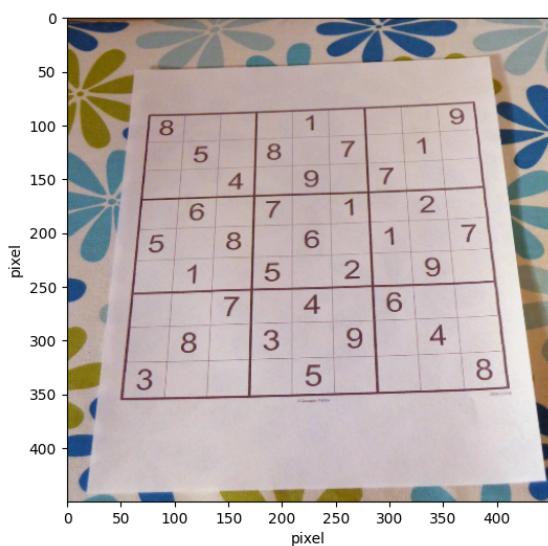
Mô hình ResNet-18 đạt hiệu suất 99.88% trên tập kiểm thử. Kết quả này cho thấy mô hình có khả năng tổng quát hóa tốt trên dữ liệu kiểm thử và không có dấu hiệu overfitting.



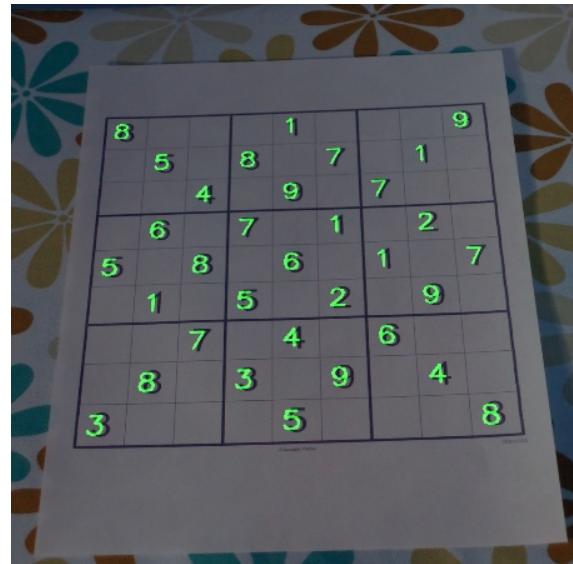
Hình 4.3: Ma trận nhầm lẩn trên tập dữ liệu kiểm thử

Lưu ý: Do trong quá trình huấn luyện, mô hình chỉ được cung cấp dữ liệu của các chữ số từ 0 đến 9 mà không có dữ liệu về ô trống. Khi kiểm thử, mô hình cho hiệu quả khá tốt, vì vậy ta đặt ngưỡng 0.8: nếu không có phần tử nào trong vector đầu ra vượt quá 0.8, ô đó được coi là ô trống.

Cuối cùng, với một ảnh bảng Sudoku ta thu được kết quả nhận diện như sau:



(a) Ảnh ban đầu



(b) Các chữ số được nhận diện

Hình 4.4: Nhận diện chữ số có trong ma trận Sudoku sử dụng ResNet-18

CHƯƠNG 5

GIẢI BÀI TOÁN SUDOKU BẰNG THUẬT TOÁN QUAY LUI

5.1 Thuật toán quay lui

Thuật toán quay lui là một phương pháp giải quyết các bài toán mang tính tổ hợp, ta xây dựng nghiệm dần từng bước và quay lui nếu gặp phải mâu thuẫn. Sau khi nhận diện được ma trận Sudoku ta triển khai thuật toán quay lui giải Sudoku theo các bước sau:

- 1) Tìm một ô trống trong bảng Sudoku.
- 2) Thủ điền các số từ 1 đến 9 vào ô đó, kiểm tra xem số vừa điền có hợp lệ không.
- 3) Nếu số hợp lệ, tiếp tục điền các ô trống khác bằng cách đệ quy. Mỗi một lớp đệ quy chịu trách nhiệm điền và xóa một ô trong bảng.
- 4) Nếu không thể điền số nào hợp lệ tại lớp đệ quy ta quay lui về lớp đệ quy trước, tức là xóa số vừa điền và thử số khác.
- 5) Thuật toán dừng lại khi bảng Sudoku được hoàn thành (không còn ô trống nào).

Mã giả mô tả giải Sudoku bằng thuật toán quay lui:

```
function Solve(grid):  
    pos = FindEmpty(grid)  
    if pos is None:  
        return True # Bang da hoan thanh  
  
    for num in range(1, 10):  
        if Valid(grid, num, pos):  
            Assign(grid, pos, num) # Dien so vao o trong  
  
            if Solve(grid):  
                return True  
  
            Remove(grid, pos) # Quay lui neu sai  
  
    return False # Khong tim duoc nghiem
```

- **Hàm Valid:** Kiểm tra tính hợp lệ của số tại một vị trí bằng cách duyệt qua hàng, cột và ô 3×3 tương ứng. Điều này đảm bảo không vi phạm các ràng buộc của bài toán Sudoku.

- **Hàm FindEmpty:** Tìm ô trống đầu tiên trong bảng. Nếu không còn ô trống, bài toán đã được giải.
- **Hàm Solve:** Dựa trên đệ quy, thuật toán thử điền số và quay lui nếu cần thiết, đảm bảo duyệt hết các trường hợp khả thi.

Đầu vào hàm **Solve** sẽ là một mảng hai chiều kích thước 9×9 , trong đó các ô trống được biểu diễn bằng giá trị 0. Đầu ra của hàm **Solve** là một ma trận 9×9 đã được điền đầy đủ các chữ số từ 1 đến 9, thỏa mãn các luật của Sudoku: mỗi hàng, mỗi cột và mỗi vùng 3×3 đều chứa đủ các chữ số từ 1 đến 9 mà không trùng lặp.

$$\begin{bmatrix} 8 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 9 \\ 0 & 5 & 0 & 8 & 0 & 7 & 0 & 1 & 0 \\ 0 & 0 & 4 & 0 & 9 & 0 & 7 & 0 & 0 \\ 0 & 6 & 0 & 7 & 0 & 1 & 0 & 2 & 0 \\ 5 & 0 & 8 & 0 & 6 & 0 & 1 & 0 & 7 \\ 0 & 1 & 0 & 5 & 0 & 2 & 0 & 9 & 0 \\ 0 & 0 & 7 & 0 & 4 & 0 & 6 & 0 & 0 \\ 0 & 8 & 0 & 3 & 0 & 9 & 0 & 4 & 0 \\ 3 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 8 \end{bmatrix}$$

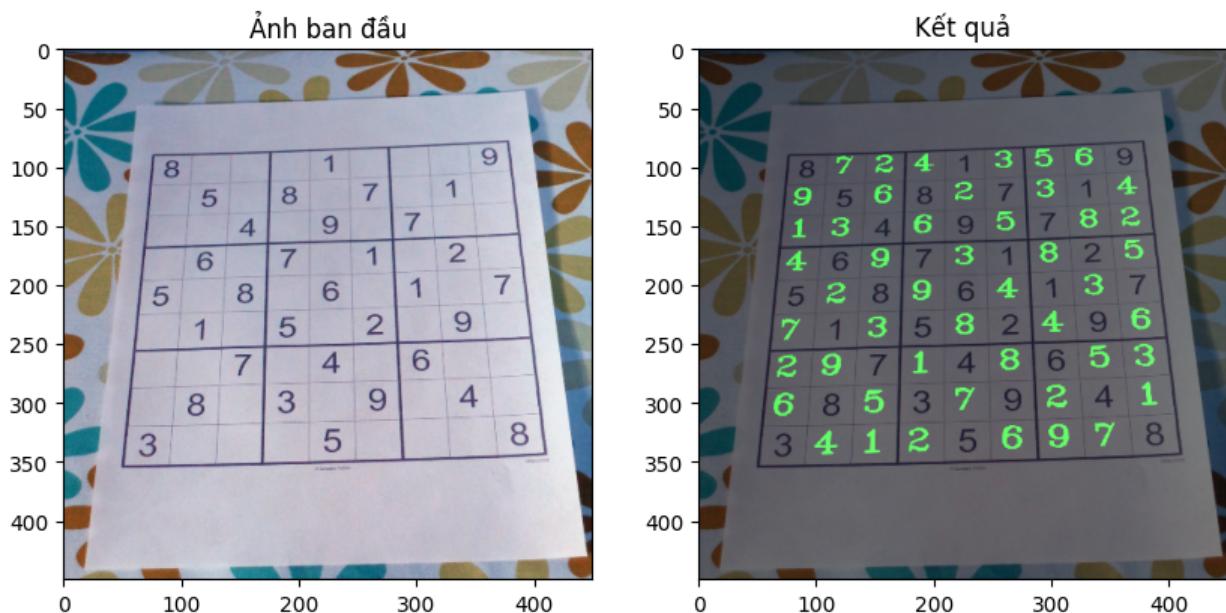
(a) Đầu vào thuật toán.

$$\begin{bmatrix} 8 & 7 & 2 & 4 & 1 & 3 & 5 & 6 & 9 \\ 9 & 5 & 6 & 8 & 2 & 7 & 3 & 1 & 4 \\ 1 & 3 & 4 & 6 & 9 & 5 & 7 & 8 & 2 \\ 4 & 6 & 9 & 7 & 3 & 1 & 8 & 2 & 5 \\ 5 & 2 & 8 & 9 & 6 & 4 & 1 & 3 & 7 \\ 7 & 1 & 3 & 5 & 8 & 2 & 4 & 9 & 6 \\ 2 & 9 & 7 & 1 & 4 & 8 & 6 & 5 & 3 \\ 6 & 8 & 5 & 3 & 7 & 9 & 2 & 4 & 1 \\ 3 & 4 & 1 & 2 & 5 & 6 & 9 & 7 & 8 \end{bmatrix}$$

(b) Đầu ra thuật toán.

Hình 5.1: So sánh đầu vào và đầu ra của thuật toán.

Sau khi giải được câu đố Sudoku, ta biểu diễn trực tiếp lời giải lên hình ban đầu để thuận tiện quan sát.



Hình 5.2: Lời giải sudoku hiển thị trực tiếp trên ảnh

CHƯƠNG 6

MỘT SỐ KẾT LUẬN

6.1 Về khả năng nhận diện bảng Sudoku từ ảnh

- Chương trình có thể phát hiện chính xác đường biên của bảng Sudoku từ các ảnh chụp trong điều kiện ánh sáng và chất lượng ảnh tốt.
- Độ chính xác có thể giảm trong các trường hợp ảnh bị nhiễu, bị xoay, hoặc méo (perspective distortion).
- Do phép biến đổi phôi hình là một phép biến đổi tuyến tính, chương trình hoạt động hiệu quả nhất khi dữ liệu đầu vào là ảnh phẳng với các cạnh của bảng Sudoku không bị cong. Nếu ảnh bị cong, chương trình sẽ gặp khó khăn trong việc chia bảng thành các ô vuông để đưa vào mô hình ResNet-18 nhận diện.
- Tốc độ phản hồi gần như là ngay lập tức.

6.2 Về nhận dạng chữ số

- Mô hình nhận dạng chữ số được train trên nhiều phông chữ nên hoạt động tốt trên các chữ số rõ ràng (như chữ số in trên giấy, app trên điện thoại).
- Chương trình sẽ từ chối nhận diện nếu đầu vào là ảnh không phải câu đố Sudoku.
- Tốc độ phản hồi và độ chính xác là rất cao khi đầu vào là dữ liệu ảnh phù hợp.

6.3 Về giải bài toán Sudoku

- **Độ chính xác của thuật toán giải Sudoku:**

- Với đầu vào chính xác (các chữ số được nhận dạng đúng), thuật toán giải Sudoku có thể đạt độ chính xác 100% trong việc tìm ra lời giải.

- **Tốc độ giải bài toán:**

- Thuật toán giải quay lui được cài đặt đầy đủ các nhánh cận, cho phép giải quyết các bảng Sudoku tiêu chuẩn (9x9) với tốc độ nhanh.

6.4 Một số giải pháp phát triển

- Hệ thống cần được cải thiện để xử lý ảnh trong môi trường thực tế như: bảng Sudoku bị mờ quá nhiều, xử lý vết mực che lấp nhiều nội dung.

- Để triển khai thực tế, cần tích hợp thêm các bước kiểm tra và sửa lỗi nhận dạng tự động hoặc cho người dùng tự sửa đổi một cách thủ công khi phát hiện ra lỗi nhận diện.
- Sử dụng các thuật toán phát hiện biên nâng cao (như *Canny Edge Detection*) kết hợp với mạng học sâu để phát hiện chính xác bảng Sudoku ngay cả trong trường hợp ảnh bị méo.
- Xây dựng hệ thống web hoặc app di động với giao diện thân thiện hơn với người dùng.
- Chương trình hoàn toàn có thể phát triển để giải Sudoku với các kích cỡ khác nhau, nhưng yêu cầu người sử dụng phải nhập kích cỡ trước.

TÀI LIỆU THAM KHẢO

- 1) Nguyen, A., Tran, B., & Le, D. (2020). *Image Processing Techniques for Sudoku Grid Extraction*. Journal of Machine Vision, 18(4), 45–58.
- 2) Zhang, X., & Li, Y. (2023). *Grid Alignment and Warping Techniques for Sudoku Extraction from Scanned Images*. Proceedings of the International Conference on Image Processing (ICIP), pp. 876–885.
- 3) He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778.
- 4) OpenCV Documentation (2024). *Comprehensive Guide to Image Processing in Python*. Available at: <https://docs.opencv.org>.