Xử Lý Bài Toán Sudoku bằng CNN

Trương Thị Cẩm Ly* and Quách Trọng Nghĩa*

*Trương Đại học Công nghiệp Thành phố Hồ Chí Minh

Thị Giác Máy Tính
Ngày 15 tháng 10 năm 2022

Nội dung

1. Giới thiệu nội dung
2. Nghiên cứu liên quan
3. Bộ dữ liệu
4. Mô hình
5. Kết quả
6. Hướng phát triển

Giới thiệu nội dung

- Các trò chơi logic như Sudoku đã được chứng minh là giúp trì hoãn các chứng rối loạn thần kinh và Sa sút trí tuệ
- Mục tiêu của trò chơi là điền các chữ số vào một lưới 9×9 sao cho mỗi cột, mỗi hàng, và mỗi phần trong số chín lưới con 3×3 cấu tạo nên lưới chính (cũng gọi là "hộp", "khối", hoặc "vùng") đều chứa tất cả các chữ số từ 1 tới 9.
- Câu đố đã được hoàn thành một phần, người chơi phải giải tiếp bằng việc điền số. Mỗi câu đố được thiết lập tốt có một cách làm duy nhất

Nội dung

1 Giới thiệu nội dung

2 Nghiên cứu liên quan

3 Bộ dữ liệu

4 Mô hình

5 Kết quả

6 Hướng phát triển

Nghiên cứu liên quan

- Tìm kiếm thuật toán: Các thuật toán phổ biến nhất cho việc xử lý sudoku là tree search, heuristic search và quay lui.
- Convolution Neural Networks: phương pháp tiếp cận máy học, để giảm bớt chi phí tính toán.
- LSMT

Nội dung

1 Giới thiệu nội dung

2 Nghiên cứu liên quan

3 Bộ dữ liệu

4 Mô hình

5 Kết quả

6 Hướng phát triển

Bộ dữ liệu

- 1 Million Sudoku Games của Kaggle
- data có 2 cột là Quiz và Solution

Quiz	000009010306180700000654000740000 030002090600010000058000345000003 068407080900000
Solution	425739816396182745871654329748526 9 315328916746194732589673451821532 68 497284917563

Hình: Dataset Examples

Nội dung

1 Giới thiệu nội dung

2 Nghiên cứu liên quan

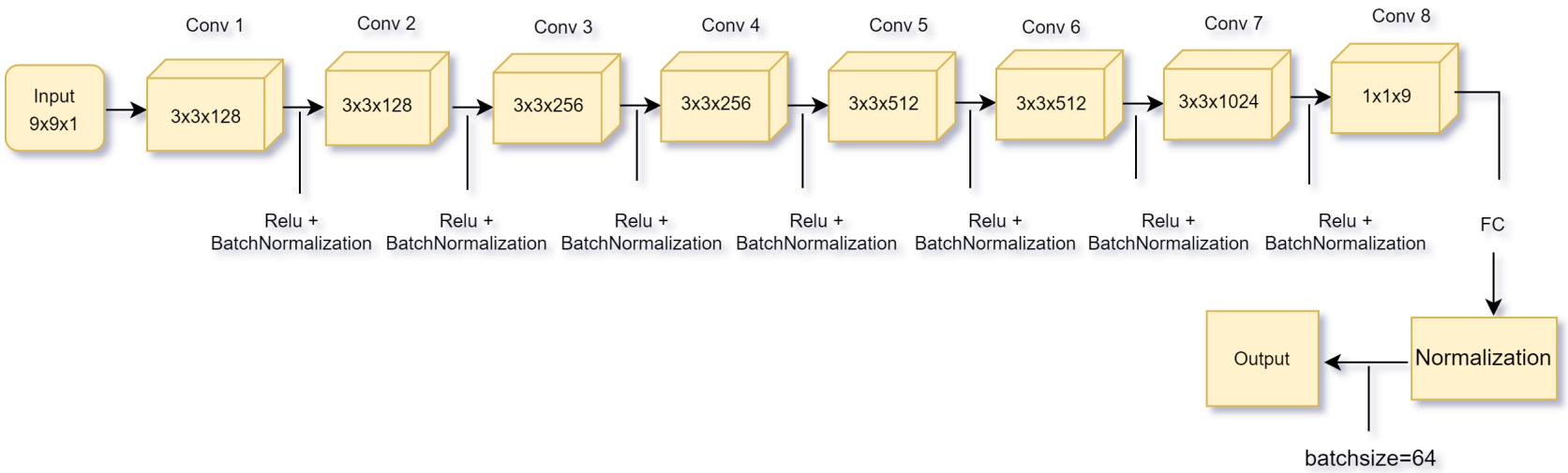
3 Bộ dữ liệu

4 Mô hình

5 Kết quả

6 Hướng phát triển

Kiến trúc mô hình



Hình: Kiến trúc mô hình

Batch Normalization

Chuẩn hóa dữ liệu input ở các layer bất kì về phân phối phân phối chuẩn với trung bình xấp xỉ 0 và phương sai xấp xỉ 1

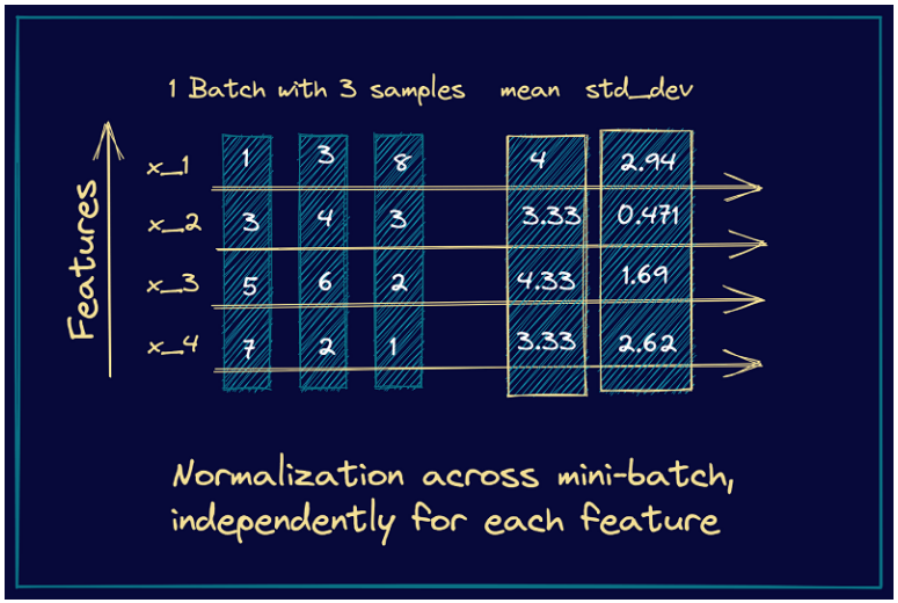
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i //mini - batchmean$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \quad //mini - batchvariance$$

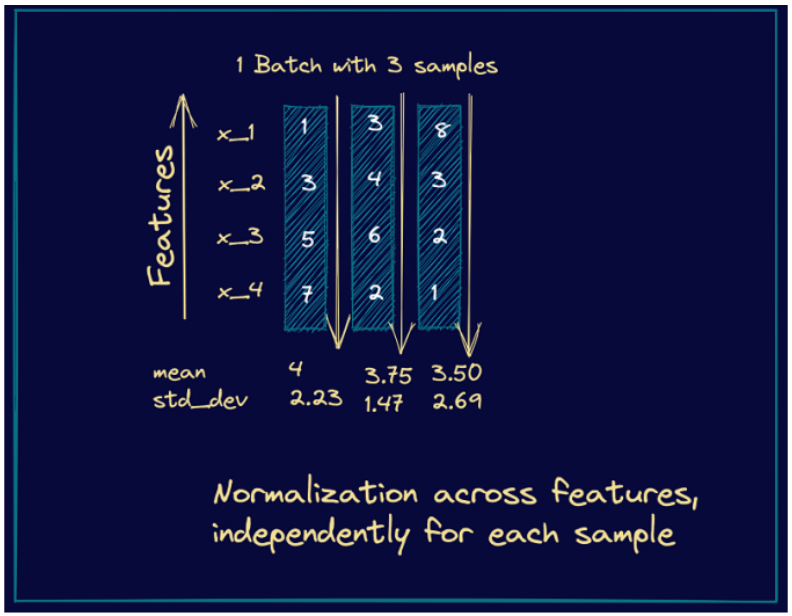
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} //normalize$$

Layer Normalization

Layer Norm (LN) sử dụng chung một giá trị trung bình và phương sai trong 1 hidden layer. Vì vậy, LN không phụ thuộc vào mini-batch, nên có thể huấn luyện với batch-size $= 1$ mà không gặp vấn đề gì.



(a) $BatchNormalization$



(b) $LayerNormalization$

Hình: Sự khác nhau giữa LayerNorm và BatchNorm

Activation

Vì đầu ra mong muốn không có số âm nên hàm kích hoạt relu sẽ được sử dụng ở các lớp Convolutions. Công thức Activation Relu:

$$\text{Relu} = max(x, 0)$$

Activation Softmax được sử dụng để tính xác suất các nhãn. Công thức Activation Softmax:

$$P(C_k|x) = y_k(x) = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}}$$

Trong đó:

- x đại diện cho kết quả đầu ra từ lớp trước
- $a_k = W_k^T x + b_k$ với W_k và b_k đại diện cho trọng số của lớp softmax và bias của lớp K.

Nội dung

1 Giới thiệu nội dung

2 Nghiên cứu liên quan

3 Bộ dữ liệu

4 Mô hình

5 Kết quả

6 Hướng phát triển

Chỉ số đánh giá

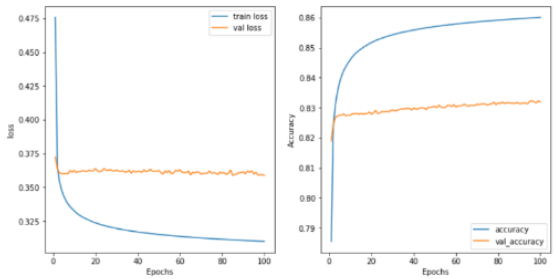
- Accuracy: tính toán tỉ lệ phần trăm các ô được gán nhãn chính xác. Công thức tính Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Loss: Sử dụng hàm Sparse Categorical cross-entropy để chỉ ra mô hình dự đoán sai bao nhiêu phần trăm (tính theo cell) so với thực tế.

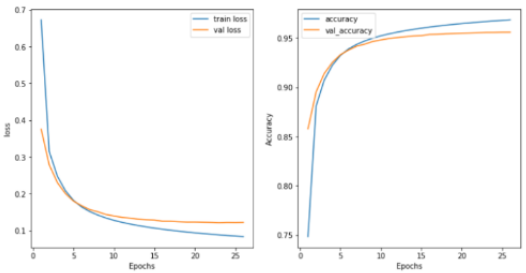
$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_1})]$$

Kết quả



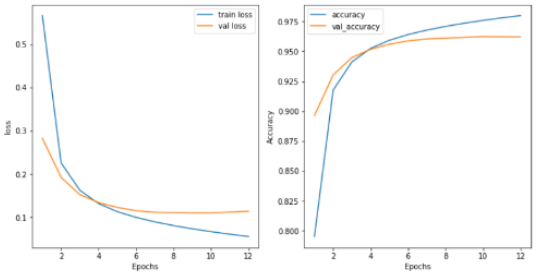
Hình 6: Độ chính xác và sai số của mô hình baseline

(a) Train-Loss của mô hình baseline



Hình 7: Độ chính xác và sai số của mô hình 8 lớp Conv và batchsize=32

(b) Train-Loss của mô hình 8Conv, batchsize=32



Hình 8: Độ chính xác và sai số của mô hình 8 lớp Conv và batchsize=64

(c) Train-Loss của mô hình 8Conv, batchsize=64

Kết quả

Architecture	Epoch	Time	Train Accuracy	Test Accuracy	Train Loss	Test Lost
3 Layer CNN, batchsize=32	100	5h	0.8503	0.8364	0.3315	0.8364
8 Layer CNN, batchsize=32	26	6h	0.9685	0.9559	0.098	0.1213
8 Layer CNN, Batchsize=64 + LayerNormalization	12	5h	0.98	0.962	0.0555	0.1138

Hình: So sánh mô hình

Kết quả

```
array([[0, 4, 6, 5, 0, 0, 1, 0, 0],
       [7, 0, 2, 3, 0, 0, 0, 0, 5],
       [0, 1, 0, 0, 8, 0, 2, 0, 0],
       [0, 8, 0, 0, 3, 0, 7, 4, 0],
       [3, 0, 9, 1, 6, 0, 0, 0, 0],
       [0, 5, 0, 0, 7, 0, 0, 6, 0],
       [0, 0, 0, 9, 2, 0, 8, 0, 7],
       [0, 3, 0, 0, 0, 6, 0, 5, 2],
       [1, 0, 0, 4, 0, 8, 9, 0, 0]])
```

(a) Đầu vào

```
array([[8, 4, 6, 5, 9, 2, 1, 7, 3],
       [7, 9, 2, 3, 4, 1, 6, 8, 5],
       [5, 1, 3, 6, 8, 7, 2, 9, 4],
       [6, 8, 1, 2, 3, 5, 7, 4, 9],
       [3, 7, 9, 1, 6, 4, 5, 2, 8],
       [2, 5, 4, 8, 7, 9, 3, 6, 1],
       [4, 6, 5, 9, 2, 3, 8, 1, 7],
       [9, 3, 8, 7, 1, 6, 4, 5, 2],
       [1, 2, 7, 4, 5, 8, 9, 3, 6]])
```

(b) Đầu ra mong muốn

```
array([[8, 4, 6, 5, 9, 2, 1, 7, 3],
       [7, 9, 2, 3, 4, 1, 6, 8, 5],
       [5, 1, 3, 6, 8, 7, 2, 9, 4],
       [6, 8, 1, 2, 3, 5, 7, 4, 9],
       [3, 7, 9, 1, 6, 4, 5, 2, 8],
       [2, 5, 4, 8, 7, 9, 3, 6, 1],
       [4, 6, 5, 9, 2, 3, 8, 1, 7],
       [9, 3, 8, 7, 1, 6, 4, 5, 2],
       [1, 2, 7, 4, 5, 8, 9, 3, 6]])
```

(c) Đầu ra của mô hình

```
array([[1, 5, 0, 0, 0, 7, 0, 0, 3],
       [0, 9, 4, 0, 1, 0, 0, 5, 6],
       [0, 0, 2, 9, 0, 0, 8, 0, 0],
       [8, 0, 0, 0, 6, 0, 0, 0, 2],
       [4, 0, 0, 1, 0, 3, 0, 9, 7],
       [5, 0, 9, 0, 2, 0, 0, 6, 0],
       [7, 6, 0, 4, 9, 0, 0, 8, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 2, 5, 0, 3, 0, 4]])
```

(a) Đầu vào

```
array([[1, 5, 8, 6, 4, 7, 9, 2, 3],
       [3, 9, 4, 8, 1, 2, 7, 5, 6],
       [6, 7, 2, 9, 3, 5, 8, 4, 1],
       [8, 1, 7, 5, 6, 9, 4, 3, 2],
       [4, 2, 6, 1, 8, 3, 5, 9, 7],
       [5, 3, 9, 7, 2, 4, 1, 6, 8],
       [7, 6, 3, 4, 9, 1, 2, 8, 5],
       [2, 4, 5, 3, 7, 8, 6, 1, 9],
       [9, 8, 1, 2, 5, 6, 3, 7, 4]])
```

(b) Đầu ra mong muốn

```
array([[1, 5, 8, 6, 4, 7, 9, 2, 3],
       [3, 9, 4, 8, 1, 2, 7, 5, 6],
       [6, 7, 2, 9, 4, 5, 8, 4, 1],
       [8, 1, 7, 5, 6, 9, 4, 3, 2],
       [4, 2, 6, 1, 8, 3, 5, 9, 7],
       [5, 3, 9, 7, 2, 4, 4, 6, 8],
       [7, 6, 3, 4, 9, 1, 2, 8, 5],
       [2, 4, 5, 8, 7, 8, 6, 1, 9],
       [9, 8, 1, 2, 5, 6, 3, 7, 4]])
```

(c) Đầu ra của mô hình

Hình: Kết quả mô hình

Nội dung

1 Giới thiệu nội dung

2 Nghiên cứu liên quan

3 Bộ dữ liệu

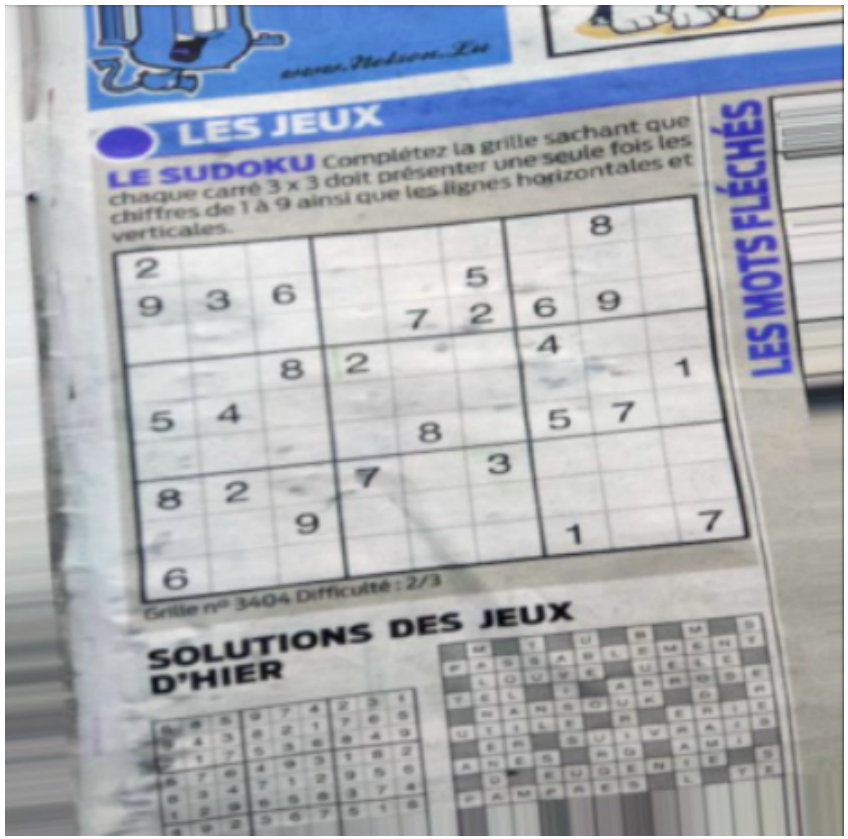
4 Mô hình

5 Kết quả

6 Hướng phát triển

Dữ liệu Sudoku

Đọc một ảnh chụp hình Sudoku bất kì mà người chơi muốn giải.
Tập dữ liệu ảnh Sudoku được lấy từ Kaggle với 2100 ảnh:



Huấn luyện mô hình phân loại chữ số

Dữ liệu gộp từ 2 tập MNIST và Chars74K.

(a) Data MNIST

(b) Data Chars74K

Mô hình sử dụng: CNN
Kết quả mô hình:

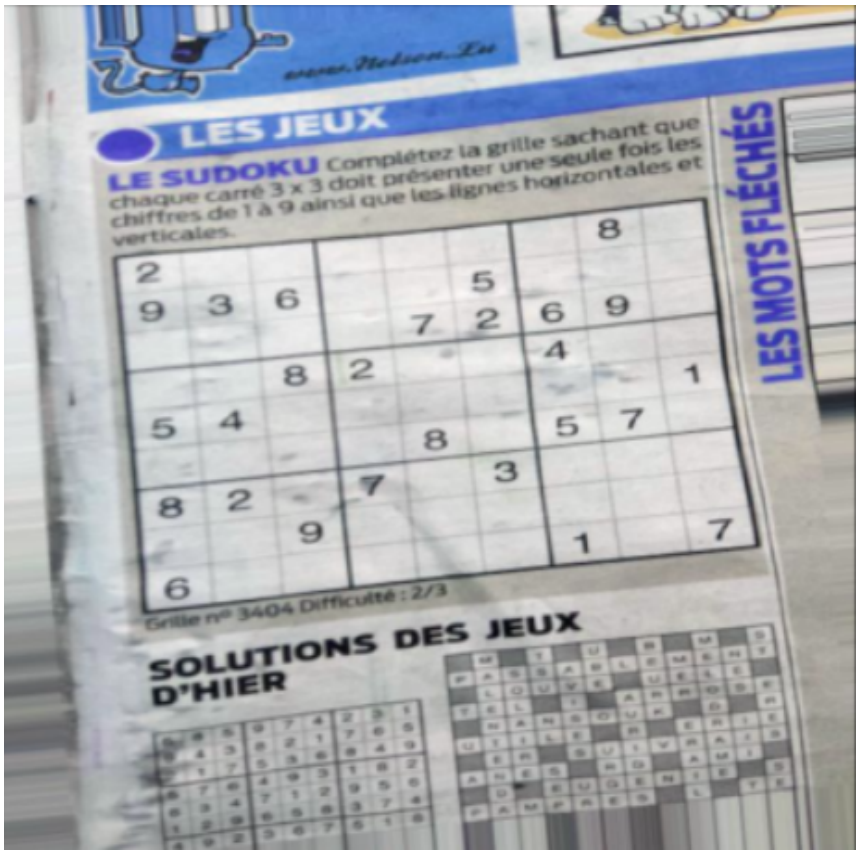
Train Accuracy	Validation Accuracy	Train Loss	Validation Loss	Test Accuracy	Test Loss
0.9931	0.9923	0.024	0.029	0.989	0.036

Detect Ảnh

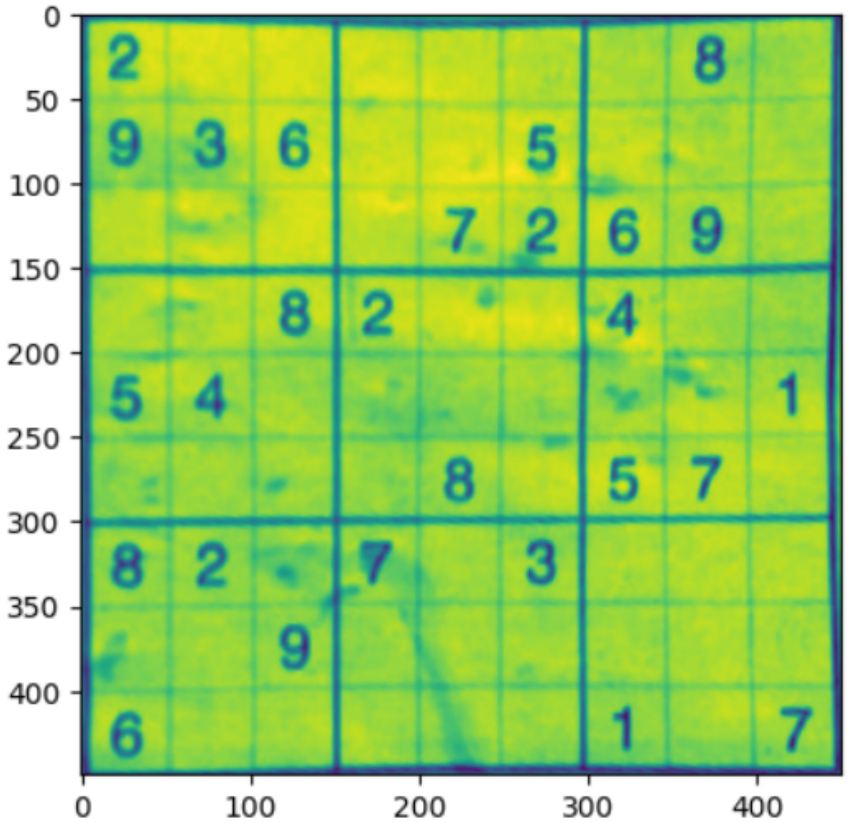
- Sử dụng một số hàm có săn của OpenCV để cắt xén ma trận sudoku trong ảnh.
  Ban đầu dùng **findContours** - trả về các tọa độ điểm biên của đối tượng. Dựa trên các điểm biên đó để tính chu vi và diện tích của các đối tượng. Đối tượng nào có chu vi được bao đóng lại và có diện tích lớn nhất thì đó chính là hình Sudoku ở trong hình.
- Tiếp đến sử dụng hàm **approxPolyDP** để bóp méo về dạng sudoku thẳng đứng, resize về kích thước 450x450.

Detect Ảnh

Sau khi phát hiện được hình sudoku, ta được các ảnh có dạng:



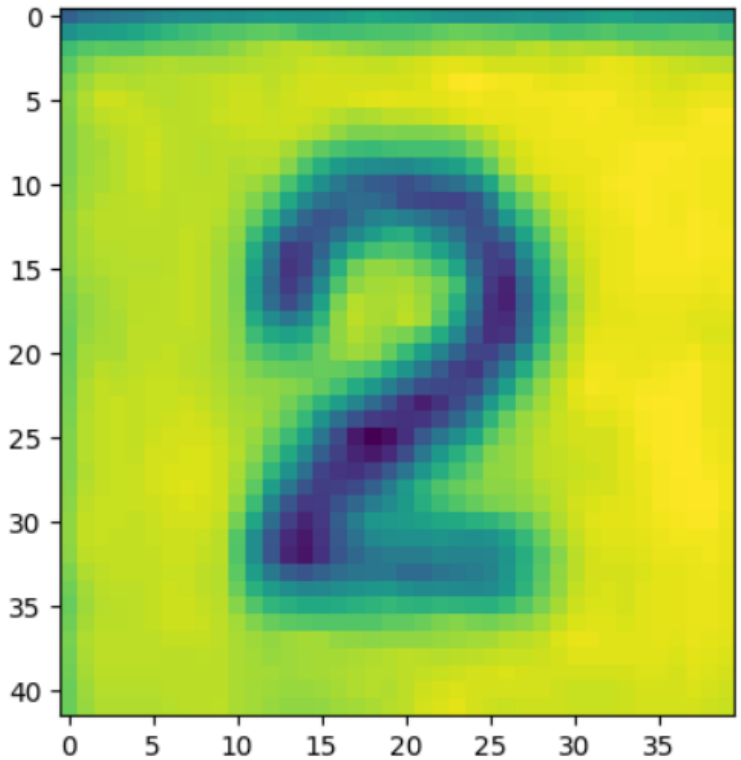
(a) Ảnh đầu vào

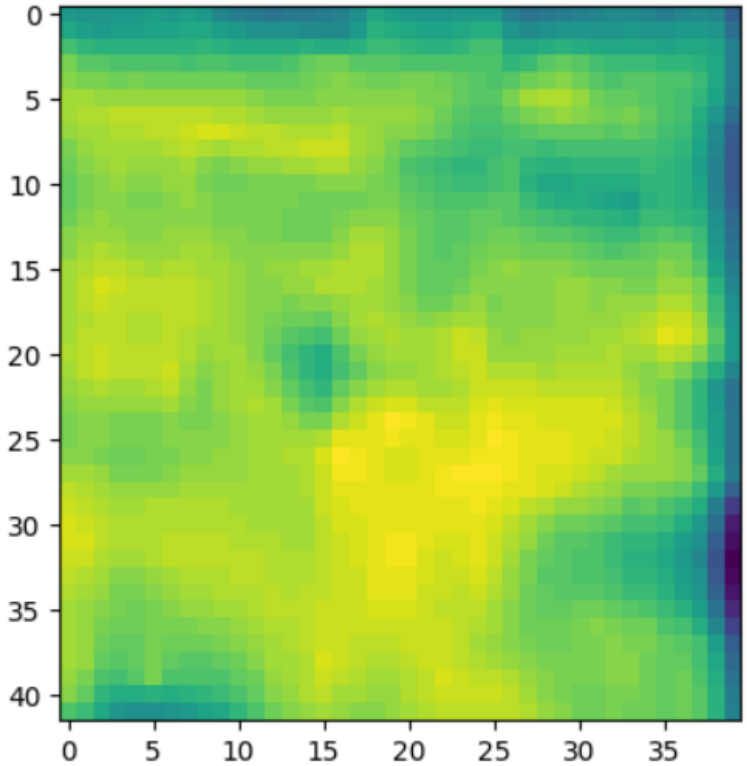


(b) Sau khi detect

Detect Số

Để dự đoán được các số có trong hình sudoku, tiến hành cắt các cell ra và đưa vào mô hình phân loại chữ số:





Do ảnh chưa mịn và có độ phân giải khá thấp nên khi đưa vào mô hình phân loại chữ số sẽ có sai số

Cảm ơn thầy và các bạn đã lắng nghe!