Sudoku Recognition and Solving:

Project & Presentation by

Rupali Vyas
BT17GCS091

Vista Vincent
BT17GCS150

Yash Kejriwal
BT17GCS123

Introduction

Building a program that recognizes the **Sudoku Board** with the image as an Input and solves it using backtracking.

Related Work

- Handwritten Digit Recognition Using Convolutional Neural Networks-Akm Ashiquzzaman

- Digit Classification using Convolutional Neural Network- Mrs. Neha Sharma , Kartikay Sharma , Shourya Bhatnagar , Deeksha Sharma, Danish Vij

Breaking Down The Project

1. Preprocessing Image

Applying necessary filters to make the grid clearer

Input Image

Binary Image

Steps involved in preprocessing image :

- Converting input image to grayscale image.
- Using Gaussian Blur technique on grayscale image to reduce noise.
- Thresholding the Gaussian Blur image to convert it into a binary image.

2. Finding Largest Polygon

To identify the grid

Corner of the largest polygon highlighted.

Joining the corners to get the largest polygon.

To find the largest polygon the below step is used :

- Contour Detection : This step involves finding the largest contour and selecting the 4 extreme most pts that will be the 4 corners. This can be done by the function **cv2.findContours** that will find all the contours in the image and the largest one can be found out by selecting the largest one by area. Once the largest polygon has been found the we can easily find the 4 corners joining them to form the largest polygon.

3. Cropping and Warping of Image

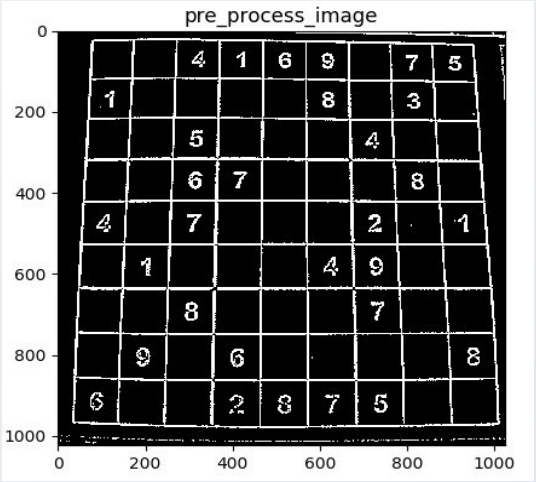
To get a better view of the grid

Once we find the coordinates of the 4 corners, we use these points to crop and warp the image to get a better perspective, which enables us to extract sub grids with precision.

We achieve this using the function cv2.warpPerspective(). warpPerspective takes 3 arguments: source image, transformation matrix, and the output image size..

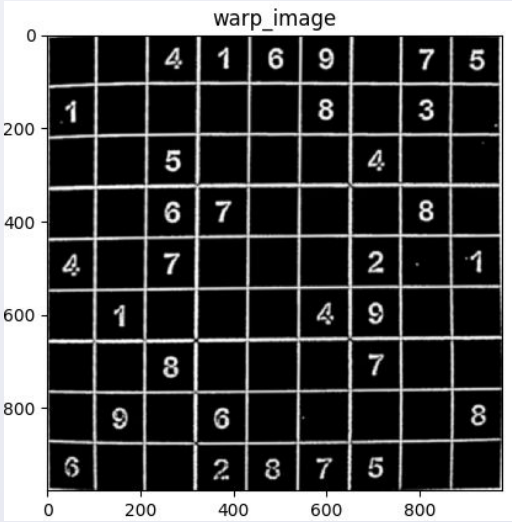
To find the transformation matrix *'m'*:

- Append the coordinates of the 4 corners in an array, *"src"*
- Find the longest side, *'S'*
- Describe a square with side 'S', and append the coordinates into another array, *"dst"*. This will be our desired perspective.
- Use cv2.getPerspectiveTransform(src,dst) to find the transformation matrix that will skew the image to fit a square.



After finding the transformation matrix, we call cv2.warpPerspective() function with the required parameters to get the desired warped image.

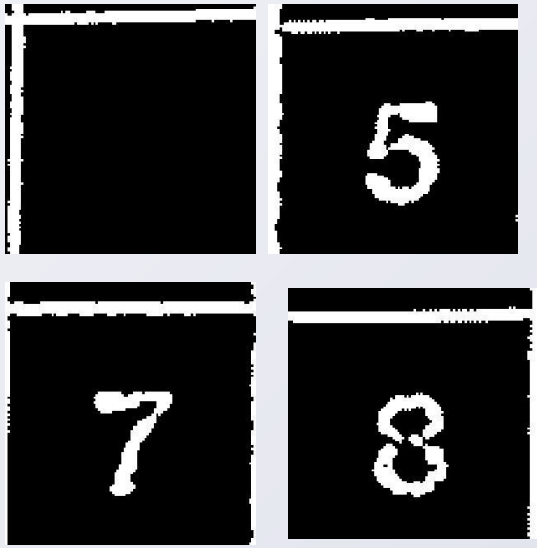
We then apply a gaussian blur to denoise the image and get it ready to extract the internal subgrids containing the numbers



4. Extracting Individual Cells of the grid

For digit recognition

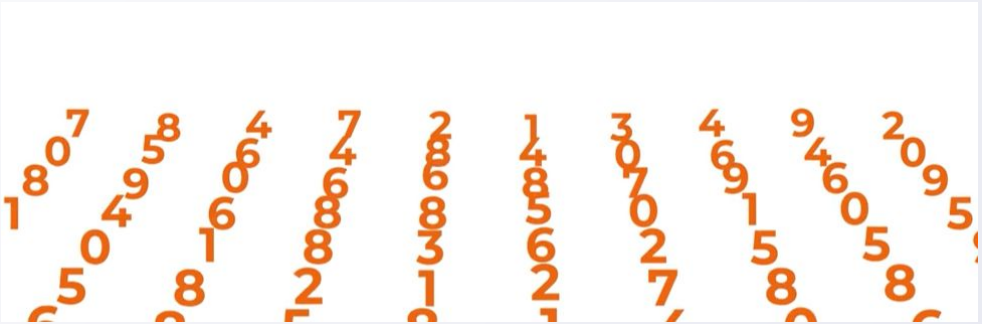
```
pts=hei/9
pts2=wid/9

rows = []
col=[]
hei_ptr=0
wid_ptr=0
for m in range(10):
    rows.append(hei_ptr)
    hei_ptr+=pts
for j in range(10):
    col.append(wid_ptr)
    wid_ptr+=pts2
nos=[]          #array of numbers
for row in rows[:-1]:
    for c in col[:-1]:
        cropped_new= wrped[int(row):int(row+pts), int(c):int(c+pts2)]
        nos.append(cropped_new)
```



5. Convolutional Neural Network

For digit classification

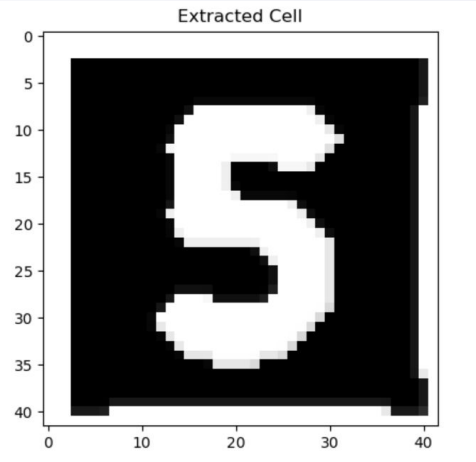
Raw Data (NN)

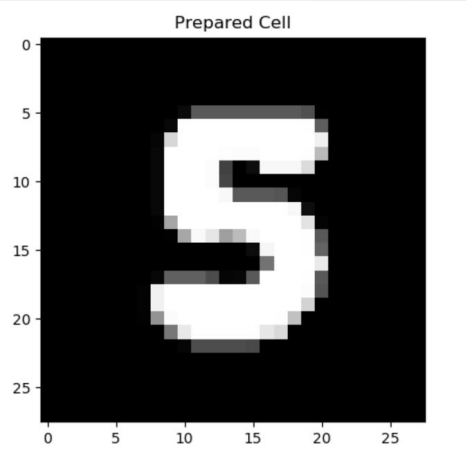


Spatial Data (CNN)

6. Preparing the image for the model

For digit classification






```
def prepare(no):
    IMG_SIZE = 28
    img_array = no.copy()
    new_img= img_array[7:(img_array.shape[0]-5), 7:(img_array.shape[1]-7) ]
    resized_digit = cv2.cv2.resize(new_img, (18,18))
    padded_digit = np.pad(resized_digit, ((5,5),(5,5)), "constant", constant_values=0)
    return padded_digit.reshape(-1,IMG_SIZE,IMG_SIZE,1)
```

7. Decoding and Storing Prediction

For obtaining the final array (2D)

```
blank = np.mean(prepare(nos[2]))
model = tf.keras.models.load_model("neuralnet12.model")
for no in nos:
    if np.mean(prepare(no))>blank:
        prediction = model.predict([prepare(no)])
        temp.append(np.argmax(prediction))
    else:
        temp.append(0)
```

8. Backtracking

Finds solution for the 2D list

There were separate functions to check box, row and column for validation of the move

```
for num in range(1, 10):
        if(is_safe(arr, row, col, num)):
                arr[row][col]= num
                if(solve_sudoku(arr)):
                    return True
                arr[row][col] = 0
return False
```

Code Execution

Complications & Improvements

Complication #1

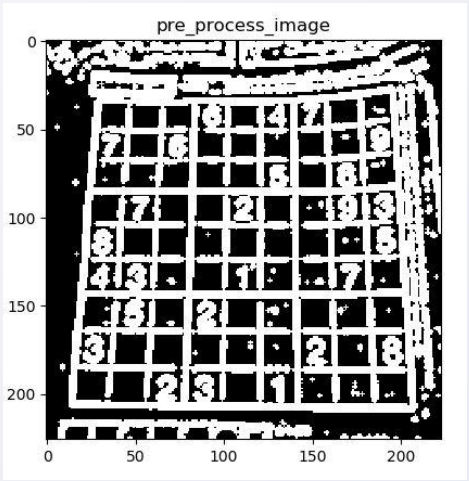
PROBLEMS WITH NOISY IMAGE

- Noisy Images makes the process of grid extraction difficult to achieve.
- In the project we apply **Gaussian Blur** with a Filter size of (5x5) in order to reduce the noise.
- The problem with blurring the images is that after we apply the thresholding to the blurred image it makes the grid lines wider and the digits difficult to recognize.
- This makes the grid extraction and prediction more difficult..

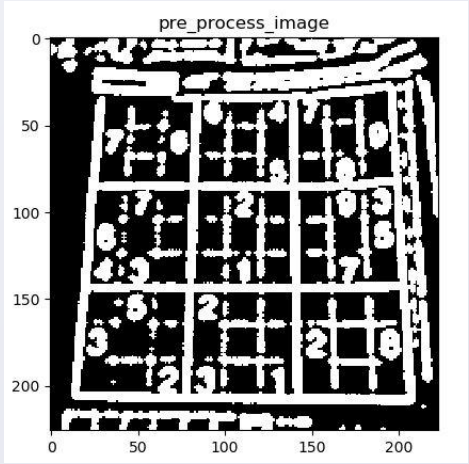
Example:

Note: As the filter size increase the noise decreases but the image quality is compromised

ORIGINAL IMAGE



BLURRED (3X3 FILTER) & THRESHOLDED



BLURRED (9X9 FILTER) & THRESHOLDED

Complication #2

PROBLEMS WITH CNN MODEL PREDICTIONS

- Even under ideal conditions the current model is failing to identify certain numbers.
- In our case, for some cases:
  - 1 is being identified as 8 or 2

Thank You